

Numerical Analysis of Infinite Markov Processes

by P.J.B. King

Computing Laboratory,  
University of Newcastle upon Tyne

Ph.D. Thesis

August 1979

## Acknowledgements

This thesis owes a great deal to the enthusiasm and interest of my supervisor, Dr. I. Mitrani. He was a constant source of ideas and improvements.

Other members of the Computing Laboratory, both staff and students, made many helpful suggestions.

The research was supported financially by the Science Research Council, and latterly by the University of Newcastle upon Tyne.

# Numerical Analysis of Infinite Markov Processes

## Contents

Abstract

0 Introduction

1 Markov processes: Definitions and Notation

2 Literature Survey

3 Algorithms

4 Practical Considerations

5 Examples

6 Conclusions

References

## Numerical Analysis of Infinite Markov Processes

### Abstract

The estimation of the steady state probability distribution of discrete state Markov processes with an infinite state space is the subject of this thesis. The measurement and analysis of complex queueing systems is the motive for this investigation, since a classical approach to analysing queueing systems is to imbed the model in a Markov process. The literature pertaining to the numerical solution of Markov processes is surveyed, with the object of finding a method which will be applicable to a wide class of processes with a minimum of prior analysis.

A general method of numbering discrete states in an infinite domain is developed and used to map the discrete state spaces of Markov processes into the positive integers, for the purpose of applying standard numerical techniques. A theoretical result which has not been previously employed, is implemented and compared with two other algorithms which were developed for use with finite state space Markov processes. Some problems with no closed form analytic solution are studied numerically and steady state and marginal distributions are found.

## List of Tables

Table 3.1 Fillin of sample sparse matrices

Table 5.1 Performance of algorithms on M/M/1  $\rho=0.3$

Table 5.2 Performance of algorithms on M/M/1  $\rho=0.3$

Table 5.3 Performance of algorithms on M/M/1  $\rho=0.3$

Table 5.4 Performance of algorithms on M/M/1  $\rho=0.3$

Table 5.5 Performance of algorithms on M/M/1  $\rho=0.9$

Table 5.6 Performance of algorithms on non-preemptive priority systems

Table 5.7 Performance of algorithms on non-preemptive priority systems

Table 5.8 Marginal probabilities for non-preemptive priority systems

Table 5.9 Marginal probabilities for non-preemptive priority systems

Table 5.10 Marginal probabilities for non-preemptive priority systems

Table 5.11 Marginal probabilities for non-preemptive priority systems

Table 5.12 Response times of distributed computer system

## List of Figures

- Figure 3.1 Iterations of Stewart's and Brandwajn's methods
- Figure 5.1 Marginal probabilities for non-preemptive priority;  
linear graph.
- Figure 5.2 Marginal probabilities for non-preemptive priority;  
logarithmic graph.
- Figure 5.3 Response times of distributed computer system

## 0 Introduction

Modern day computer installations are increasingly complex systems, whose performance is difficult to evaluate. The prediction of the effects of changes to the system on its performance is likewise a difficult and time consuming operation. There are four well understood and useful means of investigating such computer systems; (1) benchmarks; (2) monitors; (3) simulation; and (4) analytical modelling. Benchmarks and monitors are means of studying existing systems. Simulation and analytical modelling are applied to probabilistic models of systems which may or may not exist.

Benchmarks usually take the form of a 'typical' workload for the system being investigated. The performance of the system is measured as it executes this benchmark, and can be compared with the performance of other systems executing the same benchmark. Benchmarks can only be used on existing systems; they have no predictive power and if the effect of proposed changes to a system is to be investigated, then the changes must first be made before running the benchmark.

Monitors are means of observing the activity of existing systems. Like benchmarks, they have no power of prediction and any changes in the system must be made before they can be evaluated. Monitors can be hardware devices, which record or count state changes in the electronics of the computer, or software routines which are called at strategic

points in the system to record pertinent data. Hardware monitors have the advantage that they do not interfere with the system and the disadvantage that it is difficult to correlate their results with the software of the system. Software monitors do interfere with the system, although typically by only a small amount, but their data is easily associated with particular pieces of software in the system.

Simulation and mathematical analysis are both means of studying probabilistic models of systems. In order to construct such a probabilistic model, we decide on the most important components in the system and describe their behaviour by probability distributions. Simulation consists of exercising this model repeatedly to give different realisations of the system being modelled and hence a sample of the model's performance. The simulation itself can be driven by random numbers drawn from the distributions which are thought to represent the activities of the system. This is sometimes called Monte-Carlo simulation. The alternative approach is to use a record of the actual activities of the system over a period of time as input to the model. This is trace driven simulation. In either case, extensive validation of the model must be carried out to prove that the model faithfully reflects the real system. Prediction of the effect of changes to the system can be done by making the changes in the model and re-running the simulation.

When using mathematical analysis to study probabilistic models, it is usually necessary to simplify the system in order to make the mathematics tractable. Generally, the study of such models involves the



solution of a large set of simultaneous equations. If closed form solutions can be found then it is a very cheap and accurate approach. Numerical methods of solving these equations take the middle ground between simulation and closed form solutions.

The models which have closed form solutions which are suitable for calculation tend to be rather simple. Other systems such as G/G/1, whilst they have general solutions, are not suitable for calculation purposes without extensive analysis. Although the class of systems for which solutions are known is now quite large, there remain simple systems whose general solution is either unknown or is only known in a computationally impractical form. For example, in systems with priority queues, either pre-emptive or non pre-emptive, although the mean number in each priority class can be easily found, and has a simply calculated formula, the distribution of the number of customers in each priority class is known only in terms of various relations that the generating function must satisfy. It is not possible to find a simple closed form for this generating function and so calculate the state probabilities from these relations. Numerical solutions can be found to a larger class of systems. Fewer simplifying assumptions need to be made in order to solve the equations. Simulation models can be arbitrarily complex, at the expense of their computer run time.

Closed form solutions give a functional form to the solution. The effect of changes in parameters can be predicted and calculated easily. Numerical methods give only the solution for a single set of parameters, but they are cheap and accurate. Simulation models also give solutions

for only a single set of parameters, but their accuracy is proportional to their running time. Calculating a numerical solution is always cheaper than performing simulation experiments to the same accuracy. The situation is rather akin to the problem of calculating definite integrals. The function can be integrated in the classical manner and a general form found for the integral. This form can then be used to calculate the value for many sets of parameters. Alternatively, Gaussian quadrature can be used to evaluate the integral for a single set of parameters. If the integrand is particularly complex, then Monte-Carlo methods can be used to evaluate the result for a single set of parameters. A large sample must be used to ensure accuracy.

A numerical solution may be possible for more complex systems than those which are soluble by purely analytic methods. Clearly the systems which can be solved numerically are restricted, in that they must engender a set of equations whose solution can be calculated more or less easily. For example, the G/G/1 system, although it has a general solution [8], is not readily solved numerically, since the 'solution' involves taking the  $n$ -fold convolution of infinite series for all  $n$ . Ponstein [38] and Neuts and his colleagues [36,31,32,24] have attacked the problem of the single server system with arbitrary distributions of both service times and inter-arrival intervals. Both authors take the approach of making the problem one of discrete time. That is, changes of state occur only at times  $t=0,1,2,\dots$ . Ponstein analyses an infinite set of equations which model the system and demonstrates a numerical method, based on polynomial root finding in the complex field, which finds the solution of the equations. Neuts, as well as considering

discrete time steps, restricts the state space of the problem to be finite too, arguing that continuous time and unbounded state spaces are analytic conveniences which are not needed in the age of the electronic computer. While this argument is not without validity, the simplicity of analysis which made continuous time and unbounded state spaces attractive in the pre-computer era, also applies to numerical analysis of solving such systems.

A simpler approach to solving complex, generally distributed queueing systems is to extend the description of the states of the system so that the enlarged system is a Markov process and to find the steady state distribution of that Markov process. This was the basis of Erlang's classical method of stages and is the approach that we shall follow. We investigate various proposed algorithms, with a view to finding a method, or small number of methods, which is applicable to a large class of systems with a minimum of methods for the systems, being needed to ensure the convergence and accuracy of the solution.

Chapter 1 of this thesis contains a definition of Markov processes and of the terms that we use to classify and describe them. The notational conventions that are used are also introduced.

The second chapter is a survey of the literature concerning the numerical solution of Markov processes. Both the finite state space and the infinite state space problem are examined.

The following chapter describes, in some detail, the algorithms which are to be investigated, with emphasis on their computer

implementation. Some costs of the algorithms are estimated and compared, and their performance on a simple system is analysed.

Chapter 4 considers some further practical problems involved in the computer application of theoretical methods for solving Markov processes.

Chapter 5 compares the performance of the algorithms experimentally. A system with known solution is solved using all the algorithms. Marginal probabilities for a system with no closed form probability distribution are found. A distributed computing system is modelled to confirm heuristic arguments about its performance.

## 1 Markov processes

In this chapter, we define Markov processes and chains and the terms that we use to classify them. The notational conventions used throughout the thesis are given.

### 1.1 Definition

Let the set of possible states of the system being studied be  $X$ , and let  $X(t)$  be the state of the system at time  $t$ .  $\{X(t), t \geq 0\}$  is a Markov process if the probability of the system being in a particular state at time  $t + \Delta t$ ,  $X(t + \Delta t)$ , depends only on the state of the system at time  $t$ ,  $X(t)$ , and not on any previous history of the system. We shall only be concerned with systems where  $\{X\}$  is a discrete, denumerable set of states. For the time being, without loss of generality, we consider  $\{X\} = N$ , the set of natural numbers (positive integers). If state changes only occur at times  $t = 0, 1, 2, \dots$  or if the passage of time is of no interest, then we can denote the states of the system as  $X_i$  at 'time'  $i$ . In this case, the process is called a Markov chain.

Markov chains are characterised by their transition probability matrices,  $P$ , where  $p_{ij}$  is the probability that the system will be in state  $j$  at instant  $k+1$ , given that it is in state  $i$  at instant  $k$ . Markov processes can be described mathematically in two, essentially

equivalent, ways. We define  $s_{ij}(t)$  as the probability that a process in state  $i$  at time 0, is in state  $j$  at time  $t$ . We shall denote the matrix  $(s_{ij}(t))$  by  $S(t)$ . The matrix of instantaneous transition rates,  $Q$ , is given by the derivative of  $S$  with respect to  $t$ , at  $t=0$ . We shall only be interested in processes where the transition rates, or the transition probabilities in the case of Markov chains, are constant. These are called temporarily homogeneous processes.

In this case, it is well known that

$$S(t) = \exp \{Qt\} \quad (1.1)$$

where  $\exp\{.\}$  is the (matrix) exponential power series. (See for example [8] P 46.)

The states of the system can also be classified. If there exists a sequence of states  $k=i_0, i_1, \dots, i_n=j$  such that  $q_{i_t i_{t+1}} \neq 0$  for  $t=0, 1, 2, \dots, n-1$ , then we write  $k \rightarrow j$  and say that there is a path from  $k$  to  $j$ . It is important to realise that this is not a reflexive relationship. The existence of a path from  $k$  to  $j$  has no implications about the existence of a path from  $j$  to  $k$ . If  $k \rightarrow j$  and  $j \rightarrow k$  then  $j$  and  $k$  are said to communicate. The states of the system can be arranged into subsets or classes within which all the states communicate. Classes which have no paths leading to states outside the class are called essential. Classes which do have paths to states outside the class are inessential. The states which belong to inessential classes are transient. States which are members of essential classes are either all recurrent or all transient. Different classes may be recurrent or transient. Once the system has entered an essential class, it can only

occupy states in that class, and in no other classes, subsequently. If returns to a state of a chain can only be made at times  $d, 2d, 3d, \dots$  etc. for some  $d > 1$ , then the state is periodic with period  $d$ . Other states are aperiodic. All states in the same class have the same period. If all the states of a chain are members of the same essential class, then the chain is said to be irreducible.

As an example, consider the 5 state Markov chain defined by P

$$P = \begin{matrix} & \begin{matrix} 0.5 & 0.4 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0.3 \\ 0.1 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0.7 & 0 & 0.7 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{matrix} \end{matrix}$$

There are three classes of states. The states  $\{1,2\}$  form an essential, aperiodic class;  $\{4,5\}$  form an essential class with period 2;  $\{3\}$  is an inessential class.

We can also define

$$\pi_i(t) = \text{Prob system in state } i \text{ at time } t$$

and the (row) vector  $\underline{\pi}(t)$ , which represents the probability distribution of the states at time  $t$ .

If we assume that a chain is in state  $i$  at time  $t=0$ , then the probability distribution of being in a particular state at time  $t=1, 2, 3, \dots$  etc. is given by  $\underline{e}_i P, \underline{e}_i P^2$  etc. , where  $\underline{e}_i$  is the unit row vector with 1 in the  $i$ th component and 0 in all others. If the chain is irreducible, or if there is only a single essential class of states, then the powers of the matrix  $P, P^k$ , tend to a matrix which has

identical rows. Thus  $\underline{e}_i P^k$  will tend to a vector which is independent of the starting state,  $i$ , and is called the steady state distribution. We denote the steady state distribution by  $\underline{\pi}$ . A vector  $\underline{x}$  which satisfies  $\underline{x} = \underline{x}P$ , is a stationary distribution. Clearly,  $\underline{\pi}$  is a stationary distribution. If the process is honest, it is the unique stationary distribution. Under similar conditions, as  $t \rightarrow \infty$  the matrix of transition functions,  $S(t)$ , tends to a matrix with identical rows, each row equal to  $\underline{\pi}$ .

We shall consider that we know only the transition rates between the states of the system and that they are independent of time. This implies that we know the rate matrix  $Q$ , and use this to define the process.

Define  $Q = (q_{ij})$  as the matrix of transition rates from state  $i$  to state  $j$ , without defining  $q_{ii}$  for the moment, then we have

$$\begin{aligned} \pi_i(t+\Delta t) &= \pi_i(t) (1 - \sum_{k \neq i} q_{ik} \Delta t) \\ &\quad + \sum_{k \neq i} q_{ki} \pi_k(t) \Delta t + o(\Delta t^2) \end{aligned}$$

Define  $q_{ii} = - \sum_{k \neq i} q_{ik}$  (as it must if  $Q$  is conservative) then

$$\pi_i(t+\Delta t) = \pi_i(t) + \sum_k \pi_k(t) \cdot q_{ki} \Delta t + o(\Delta t^2)$$

Let  $\underline{\pi}(t) =$  row vector of  $\{\pi_i(t)\}$  then

$$\underline{\pi}(t+\Delta t) = \underline{\pi}(t) + \underline{\pi}(t)Q\Delta t + o(\Delta t^2) \quad (1.2)$$

Hence, rearranging and taking the limit as  $\Delta t \rightarrow 0$

$$\dot{\underline{\pi}}(t) = \underline{\pi}(t)Q \quad (1.3)$$



but in steady state  $\dot{\underline{\pi}}(t) = \underline{0}$  and  $\underline{\pi}(t) = \underline{\pi}$ , hence

$$\underline{\pi}Q = \underline{0} \quad (1.4)$$

where  $\underline{\pi}$  is the steady state probability vector. Equations (1.4) are called the balance equations of the Markov process. Equations (1.3) are the forward Chapman-Kolmogorov equations.

An alternative approach comes directly from equation (1.2). Since, when steady state is reached

$$\begin{aligned} \underline{\pi}(t+\Delta t) &= \underline{\pi}(t) = \underline{\pi} \\ \underline{\pi} &= \underline{\pi} + \underline{\pi}Q\Delta t + o(\Delta t^2) \\ \underline{\pi} &= \underline{\pi}P \end{aligned} \quad (1.5)$$

where  $P = I+Q\Delta t$  and we neglect terms of order  $\Delta t^2$ . We are free to choose  $\Delta t$  subject to the limitation that it ought to be small enough to make the probability of more than one state change in any time interval  $(t, t+\Delta t) = o(\Delta t^2)$ .  $P$  is the so called jump chain matrix of the process, and can be interpreted as the probability transition matrix of an equivalent Markov chain, with the restriction on  $\Delta t$  being that

$$\sum_j p_{ij} = 1 \text{ for all } i, \text{ or } P \text{ is stochastic.}$$

In this case, it can be shown that  $P$  will have at least one eigenvalue of unit modulus, and in fact one eigenvalue equal to 1. (Consider  $\underline{e}$  the column vector with all components equal to 1, clearly it is a right eigenvector.)  $\underline{\pi}$  is thus the left eigenvector corresponding to the unit eigenvalue. It can be shown also using Gerschgorin's Theorem that all eigenvalues of  $P$  are less than or equal to 1 in modulus. This has important consequences for the convergence of some algorithms

Nothing so far stated has excluded processes with an infinite state space. A little care is needed to define infinite matrices and the various operations on them, but the same results hold in general. We shall be mainly interested in such systems and in the effects of truncating the state space.

## 1.2 Notation

Except for a few sections, we have attempted to be consistent in our notation throughout this thesis. Where it is possible we have used the most popular convention from the literature of Markov processes, but on occasion we have had to deviate in order to preserve our own consistency. For example, the matrix of transition functions,  $S$ , is most often denoted by  $P$  in the literature, but we use that for the transition probability matrix of a Markov chain.

Capital Roman letters are used to denote matrices, with individual elements denoted by the same letter in lower case, subscripted. Vectors are lower case letters, Greek or Roman, underlined, and their components are the same letter with an appropriate subscript. All vectors are considered as row vectors. Column vectors and transposed matrices are designated by a prime ' .  $\underline{e}_j$  represents the unit vector, with all components equal to 0 except the  $j$ th.  $\underline{e}$  is the vector with all components equal to 1.  $I$  represents the identity matrix. We denote the  $n \times n$  truncation of the infinite matrix  $A$  by  $(n)A$ . Conventionally this is a "north west corner" truncation, that is

$$\begin{aligned} (n) a_{ij} &= a_{ij} \text{ if } i \leq n \text{ and } j \leq n \\ &= 0 \text{ otherwise.} \end{aligned}$$

Summations are taken over all possible values of the index, unless otherwise indicated.

More specialised conventions are described below.  $Q$  is the (possibly infinite) matrix of instantaneous transition rates. It is conservative.  $P$  is a probability transition matrix and as such is non-negative and stochastic.  $\underline{\pi}$  is the steady state probability vector.  $A$  is a general non-symmetric matrix and  $Z$  is its inverse.  $L$  is a lower triangular matrix,  $U$  is an upper triangular matrix, and  $D$  is a diagonal matrix.

## 2 Literature Survey

In this chapter we survey the literature concerning numerical methods for finding the steady state distributions of Markov chains and processes. For a process with a finite state space, this involves either of two classical problems of linear algebra, the calculation of eigenvectors or the solution of simultaneous linear equations. Various optimisations and computational improvements can be found because we have additional knowledge about the structure of the matrices involved.

### 2.1 Infinite State Spaces

Various authors have considered the problem of approximating infinite matrices by their finite truncations. These developments have nearly always been motivated by our problem, namely solution of Markov processes, but very little numerical evidence has been presented.

Kemeny [28] considers the problem of approximating the transition matrix for an infinite Markov chain. His approach is to find representations for  $P$  in terms of matrices  $C$ ,  $D$ , and  $E$ , such that

$$C = E^{-1}$$

$$P = CDE$$

and hence

$$P^n = CD^nE$$

Many such representations are known for finite dimensional  $P$ , but when infinite state spaces are involved more care is needed. For special structures of  $P$ , slowly spreading Markov chains, Kemeny shows that such a representation is possible and  $D^n$  is easily calculated.

Jensen and Kendall [26] consider those systems with bounded generators, that is  $q_{ij} > M$  for some constant  $M < 0$  and for all  $i$ . This includes many interesting systems, but not all. For example, the  $M/M/\infty$  system has  $q_{ij} = -(\lambda + (i-1)\mu)$  when the states are numbered  $1, 2, \dots$ , and hence does not have a bounded generator. They recommend using a matrix squaring procedure on the matrix  $I+Qt$  when the process is known to be aperiodic. They state that if the state space is infinite then the matrix will have to be truncated, but they do not produce any numerical evidence that they actually attempted to solve any processes. By a simple argument, they show upper and lower bounds for elements of  $(n)S$ , but do not discuss how these bounds relate to  $S$  (the infinite matrix).

Using the Perron-Frobenius theorem for non-negative matrices Seneta [41,42,43] showed that by considering  $P$ , the transition probability matrix for an infinite Markov chain (which is non-negative), the steady state probabilities,  $\pi_j$ , could be approximated using the  $n \times n$  truncation of  $I-P$ . Tweedie [49] extended this result by proving that the same approximation was valid when  $Q$  was used in place of  $I-P$ .

He also showed that less restrictive conditions on  $Q$  were needed for his result. He also developed another approximation formula that converges to give elements of the matrix  $S(\infty)$ , which correspond to states in different communicating classes of the state space.

Seneta is the only author to have considered algorithmic and numerical aspects of the infinite case in any detail. Golub and Seneta [18] consider the special case of a system in which all the elements of one column of the matrix are greater than some constant, which is itself strictly greater than zero. That implies that there is always one state of the system which is reachable in a single transition from all other states in the system. This is rather unrealistic for many real life situations.

Defining the row vector  $\underline{y}$  by

$$y_j = d(j) \text{ where } \inf_i p_{ij} > d(j) > 0 \\ = 0 \text{ otherwise,}$$

they show that

$$\underline{\pi}(I - P + \underline{e}' \cdot \underline{y}) = \underline{y}$$

in the infinite case, and that the solution of the truncated set of equations

$${}^{(n)}\underline{\pi}({}^{(n)}I - {}^{(n)}P + \underline{e}' \cdot {}^{(n)}\underline{y}) = {}^{(n)}\underline{y} \\ {}^{(n)}\pi_i \rightarrow \pi_i \text{ from below as } n \rightarrow \infty.$$

By constructing certain special cases of  $P$ , they can produce arbitrarily slow convergence to the true solution. Subsequently, in [19] stochastic matrices of a different form are investigated. It is shown that if the truncations of  $P$  can be made stochastic in such a way that  $n-1$  of the

equations

$${}^{(n)}\underline{\pi}({}^{(n)}P)^* = {}^{(n)}\underline{\pi} \quad (2.1)$$

where  ${}^{(n)}P^*$  is the stochasticised version of  ${}^{(n)}P$ , are identical to  $n-1$  of the first  $n$  of equations (1.5), then the solutions to (2.1),  ${}^{(n)}\underline{\pi}$ , tend to  $\underline{\pi}$  from above as  $n$  tends to  $\infty$ . As before, special cases of  $P$  can be constructed with arbitrarily slow convergence.

In a later report [1] arbitrary stochastic matrices are numerically investigated, comparing the approach of solving the eigenvector problem implied by equations (1.4) and the limits given by Seneta. Several algorithms are tested for each approach. To use the limits given by Seneta's work they calculate  $(I-P)^{-1}$  using Gaussian elimination, Jacobi, Gauss-Seidel, and SOR iteration methods and an unusual non-stationary iterative method due to Fraser et al. [16]. Of these methods they show that Gauss-Seidel is at least as good as Jacobi or SOR iteration. The non-stationary method, however, outperforms Gauss-Seidel (by several orders of magnitude in some cases). Nevertheless, they recommend Gaussian elimination to find the limits given by Seneta. Taking the eigenvector approach of equations (1.4) they consider both general eigenvector algorithms, such as the power method, and also more specialised algorithms developed expressly for non-negative matrices, such as Yamamoto's [51] and that of Hall and Porsching [23]. Despite their specialised nature these algorithms are easily outperformed by inverse iteration. The power method was not actually tested since it is known to have poor convergence in many cases. Comparing their two recommended methods they find that inverse iteration is, in general, faster than Gaussian elimination. However the method based on Seneta's

results does give bounds on the accuracy of the estimates so they recommend a judicious use of both methods. Inverse iteration to discover the approximate order of truncation necessary, followed by Gaussian elimination to give upper and lower bounds to the estimates. This work is also reported, perhaps more accessibly, in [2].

Their test matrices are all full and the truncations tested are of various orders  $\leq 35$ . In practice, when modelling real world systems the transition rate matrices are very sparse, since transitions tend to only be made to neighbouring states of which there are few, but the state spaces will be much larger.

## 2.2 Finite State Spaces

Turning now to finite state space problems, many authors have dealt with the calculation of the stationary distribution. Paige et. al [37] review eight algorithms that have been used in the past. They recommend solving

$$\underline{\pi}(I + P + \underline{e}' \cdot \underline{u}) = \underline{u}$$

where  $\underline{u}$  is a row vector such that  $\underline{u} \cdot \underline{e}' \neq 0$ .

On the basis of numerically testing 60 different stochastic matrices of orders 8, 10, and 40, they suggest that the best choice for  $\underline{u}$  is a row of  $P$ . Another good choice for  $\underline{u}$  is a unit vector although this does not give such good computational results in practice.



Equation (1.1) has been used by several authors to find estimates for  $S(t)$ . It is convenient to rearrange this formula for computation. If we define  $\beta = -1/\sup(q_{ij})$  then

$$S(t) = \exp\{-t/\beta\} \exp \{(I+Q\beta)t/\beta\}$$

This rearrangement has the advantage that all the partial sums are of constant sign. Grassmann [20] has used this approach to find transient probabilities for queueing networks. He only considers finite state space problems and approximates steady state by allowing  $t \rightarrow \infty$ , stopping when successive estimates are close enough. Kerridge [29] has also worked with a slightly different rearrangement of (1.1), and gives some examples of applying his approach to problems with a small, finite state space.

Wallace [50] was the first to use numerical technique for solving 'real' queueing systems. The method used was essentially the power method, although Jacobi iteration was also available, and as such convergence was proportional to the sub dominant eigenvalue. A very efficient sparse matrix code enabled them to perform multiplications in  $o(\text{non-zeroes})$  multiplications. The Recursive Queue Analyser (RQA) as the system was called, has been successfully used on many problems up to 5000 states in size. This is almost 100 times the number of states that any other author has reported solving, with any method. A later paper [25] describes the use of the RQA in an integrated package for designing computer systems.

Stewart, in his thesis [44] demonstrates the difficulties of applying the power method to decomposable or nearly-decomposable

systems. In these cases the sub-dominant eigenvalue is close to 1 and convergence will be very slow. He develops a simultaneous iteration technique which converges on the  $m$  dominant eigenvalues and the corresponding eigenvectors simultaneously. Its rate of convergence is governed by the ratio of the first and the  $m+1^{\text{st}}$  eigenvalue. In a later survey [46], he compares various iterative methods with his simultaneous iteration method. When one has no estimate for  $\underline{\pi}$ , the favoured approach is to solve

$$\underline{\pi}P = \underline{x}$$

for arbitrary  $\underline{x}$ . Now  $P$  is singular, so Gaussian elimination will fail when a zero pivot is encountered. However if this zero pivot is replaced by machine epsilon and the calculation continued, the solution vector, while being a very inaccurate solution to the equations, is such that the errors in each element are of the same order and it is a very good approximation to  $\underline{\pi}$ . If solutions are required to a closely related system, then this solution can be used as a first approximation, and simultaneous iteration used to find the solution of the new system. In another paper, Stewart [47] presents a special purpose method very similar to row-Gaussian elimination on the matrix  $Q'$ , solving the homogenous equations  $Q'\underline{\pi}' = \underline{0}'$ , with the last equation replaced by the normalising condition.

Gaver and Humfeld [17] have used modified forms of Gauss-Seidel iteration to solve the balance equations (1.4). They suggest performing an iteration of the Gauss-Seidel method, and then replacing  $\pi_1$  by  $1 - \sum \pi_i$

or 0, whichever is larger. The idea is to preserve the sum of the probabilities as 1. They also claim to have a proof that ordinary Gauss-Seidel iteration will converge, although the balance equations are singular. The 'solution' arrived at by this means will not, other than by chance, satisfy the normalisation condition on probability distributions, which has to be imposed by normalisation.

Brandwajn [6] has developed two iterative methods for solving Markov processes that arise from multi-dimensional state space problems. The first method is based on the equivalence and decomposition method for solving queueing networks. A probabilistically equivalent network is defined and solved analytically. The resulting solution is used in an iterative procedure to find the solution to the original problem. Unfortunately, the convergence of this method is not guaranteed. However, if it does converge the method works better the more nearly completely the system is decomposable. This is in contrast to most methods which have great difficulty in solving nearly completely decomposable systems. The rate of convergence depends heavily on achieving the correct decomposition into the equivalent system. If the system is decomposed with respect to the "wrong" variable, then convergence can be extremely slow, if attained. His second method is a much more local attack on the balance equations (1.4). By careful choice of the relaxation factor, the invariance of  $\sum \pi_i$  is preserved between iterations of an under-relaxed Gauss-Seidel method. The convergence of this method can also be proved.

Shat and Raju [4] have studied the effects of truncation on finite

state space Markov processes. Working from properties of the first passage time they develop a procedure for estimating a suitable size of truncation, in advance, which will give good estimates of the first passage time for a state. Unfortunately their procedure will not generalise to infinite state spaces for several reasons. The key result that is used to calculate first passage times only holds for finite state Markov processes. Even allowing for this the method depends on a particular restriction on the structure of  $P$ , which while it holds for single queue systems, would impose a very strange state numbering on other systems.

Mitrani and Hine [35] have used a novel generating function approach to provide approximations for a general two dimensional birth-death process. They assume that the transition rates out of state  $i, j$  are independent of  $j$ , for  $j > J$ . Their method proceeds as follows. Assume that  $\pi_{ij} = 0$  for all  $i > I$ . This gives us a set of  $I+1$  equations which relate the  $I+1$  generating functions  $G_i(z)$ . These can be solved symbolically, and the  $\pi_{ij}$  evaluated. The process is then repeated for larger values of  $I$ , until the probabilities stop changing. The process normally converges very fast.

### 3 Algorithms

After careful consideration of the literature, it was decided that 3 methods for calculating the steady state probability distribution were both general enough in the class of problems to which they applied, and offered enough advantages over similar methods to warrant further investigation.

Tweedie's results [49] are the only theoretical approaches which are directly related to the infinite state space case. They give bounds on the ratios of elements of the steady state probability vector using cofactors of elements of the finite truncations of  $Q$ , the instantaneous transition rate matrix.

Iterative methods start with an estimate for  $\underline{\pi}$  and successively improve on the estimate. We shall consider Stewart's simultaneous iteration method for calculating eigenvectors applied to the equation  $\underline{\pi}P = \underline{\pi}$ . Since we are only interested in the dominant eigenvector of  $P$ , von-Mises power method could be used, but it is known to have poor convergence properties in many cases.

The other iterative method that we shall examine is Brandwajn's method. It is an attempt to solve the global balance equations (1.4), which are homogenous, by a novel relaxation method. The relaxation factor is different for each equation and ensures that the invariance of

$\sum \pi_j$  is preserved between iterations.

We shall now discuss the algorithms in more detail, with particular attention to the problems of their implementation on a computer. The computational requirements of each algorithm in terms of both space and time are compared. The final section attempts to analyse the performance of two of the algorithms when applied to the M/M/1 system.

### 3.1 Tweedie's Method

R.L. Tweedie has extended Seneta's work [41,42,43] on finite truncations of an infinite matrix in [48]. Many of the restrictions on the structure of the matrix, that Seneta found necessary, are lifted. A later extension [49], applies specifically to Markov processes and is the theoretical basis for our direct method of estimating  $\underline{\pi}$ . Given  $Q$ , he shows that

$$\frac{\text{cof}(i,j)}{\text{cof}(j,j)} \uparrow \frac{\pi_i}{\pi_j} \downarrow \frac{\text{cof}(i,i)}{\text{cof}(j,i)} \quad (3.1)$$

as  $n \rightarrow \infty$ , where  $\text{cof}(i,j)$  is the cofactor of the  $i,j$  entry in the  $n \times n$  north west corner truncation of  $Q$ . These approximations form bounds on the possible values of  $\underline{\pi}$ , and are valid when  $i$  and  $j$  belong to the same essential class of states. The obvious way to utilise these results is to calculate the cofactors of the elements of a single row and column of  $Q$ , say the first, and of the diagonal elements. From these values, the ratios of the probabilities of various states to the probability of state 1 can be found with both upper and lower bounds. Seneta proved the

same results for the cofactors of the truncations of  $I-P$ , which is a first approximation to  $-Q$ . Tweedie also presents an approximation which converges to the ratio between elements of  $S(\infty)$ . When cast in terms of  $\pi$ , this approximation is

$$\frac{\text{cof}(i,j)\text{cof}(j,i)}{\text{cof}(j,j)^2} \rightarrow \frac{\pi_i}{\pi_j} \quad (3.2)$$

This estimate converges under less severe restrictions than the bounds, but since we will have only one essential class, this freedom will be of no account. All states that we are interested in will belong to the only essential class.

Although the theory develops these approximations in terms of cofactors of  $(n)Q$ , this is not a practical way to calculate them. Cofactors of elements of  $n \times n$  matrices are the determinants of  $(n-1) \times (n-1)$  matrices, and calculation of a determinant is as costly as solving simultaneous linear equations of the same order. However, cofactors are intimately related to inverse matrices. In fact,  $z_{ij} = \text{cof}(j,i)/D$ , where  $D$  is the determinant of the matrix and  $z_{ij}$  is the  $i,j$  element of the inverse. Thus the ratio of cofactors of a matrix is equal to the ratio of corresponding elements of the transpose of the matrix's inverse. Instead of calculating  $3n$  cofactors of an  $n \times n$  matrix, we need to find  $3n$  elements of the inverse matrix ( $3n-2$  actually, since the diagonal element  $i,i$  is in row  $i$  and column  $i$  too).

Although the results hold for any Markov process, for the processes that we are interested in, the matrix,  $Q$ , will be sparse; that is most of its entries are zero. Typically from any state the system will only

be able to make transitions into a small number of other states. For example, in the M/M/1 system, the Q matrix is tridiagonal. Transitions are made from state  $i$  to state  $i+1$ , representing an arrival, and to state  $i-1$ , representing a departure from the system. The diagonal of the matrix is also non-zero since the system is conservative.

We shall only consider direct methods for finding the inverse of Q. Although iterative methods such as the Gauss-Seidel method are common for sparse matrices, we do not use them here. The convergence of such methods is guaranteed for the matrices in which we are interested by the diagonal dominance of Q, but the inversion of a matrix in this manner is equivalent to solving  $n$  sets of linear equations. Solution of one set of such equations by an iterative method is of no assistance as far as solving the same equations with a different right hand side. Also the convergence would probably be very slow for reasons discussed in the section on Stewart's method. The conjugate gradient method of solving linear equations has been gaining in popularity recently, but it is also iterative in nature, and has the added disadvantage of dealing with symmetric matrices only. It can be modified to deal with unsymmetric matrices, at the expense of doubling the number of operations per iteration and the condition number of the matrix, but the equivalence of inversion to solving  $n$  sets of linear equations remains. Theory predicts that with exact arithmetic the conjugate gradient method will converge in exactly  $n$  iterations, and its utility for sparse matrices depends on its convergence to an acceptable approximation in considerably less than  $n$  iterations. In practice, when applied to this problem approximately  $2n$  iterations were used for each equation.



Sparse matrix codes attempt to take advantage of the zero and non-zero elements in the matrix such that operations on such matrices involve the non-zero entries only. Although a special purpose sparse matrix code could have been written, which could have taken account of the known structure of  $Q$ , a general purpose set of sparse matrix handling routines was used for this work [10]. Special purpose routines have been used by Stewart [47] to solve the global balance equations (1.4), with the last equation replaced by the normalising condition. The routines always used the diagonal element as pivot. Although the diagonal dominance of  $Q$  ensures accuracy, this will give rise to excessive fill-in, that is zero elements becoming non-zero. From our point of view, the main drawback to writing special purpose code to manipulate sparse versions of  $Q$  is that we have very little knowledge of its structure, other than that it is sparse. Different systems give rise to  $Q$  matrices with radically different patterns and sizes of non-zero element. The choice of representation for the system will also affect the positioning of the non-zero elements in  $Q$ .

The routines used in this work are widely available and perform the operation of solving the equations

$$\underline{Ax}' = \underline{b}' \quad (3.3)$$

This they do by forming the L/U factorisation of  $A$ , and providing routines which will operate on vectors using this factorisation of  $A$ .  $L$  is a lower triangular matrix with unit elements on the diagonal.  $U$  is upper triangular. The routines can operate on vectors using  $A$ ,  $A^{-1}$ , and

their transposes. As is common with sparse matrix codes, the routines choose pivots for their factorisation based not only on the size of the elements which remain to be eliminated, but also on the number of zero elements which will become non-zero if a particular element is chosen. The element which causes the minimum fill-in will be chosen as pivot, subject to the additional constraint that it must not be less than some user supplied sparsity factor times the largest remaining element in that particular row or column. Altering the value of this parameter does not affect the accuracy or sparsity of the resulting form of the inverse much. If no account is taken of size when pivots are chosen, but only of the amount of fill-in they will cause, an inaccurate solution may result. The accuracy of the decomposition can be monitored using a standard technique [10]. Once again only if no account is taken of size when choosing pivots does this perturbation factor become large.

To calculate the various bounds and approximations given by Tweedie we need, assuming that the ratios with the first state will be used, the first row, the first column and the diagonal of the inverse. The first row and column are easily computed, but to find the diagonal by conventional means requires calculating the complete inverse matrix (albeit row by row). This can be an expensive business. However, Erisman and Tinney have presented an algorithm [14] which will calculate a subset of the elements of the inverse of a sparse matrix. Those elements that correspond to non-zero elements in the transpose of the L/U factorisation of the original matrix, which the diagonal elements of A clearly do, are members of this subset. The algorithm assumes that any non-zero elements which become zero as a result of the factorisation

remain stored as if they were non-zero; that is, their values become zero but they remain as elements in the representation of the matrix. All sparse matrix routines known to us operate in this way. Any element of the inverse which corresponds to a non-zero element in the transpose of the L/U factorisation can be calculated. Again considering the matrix A, and denoting its inverse by Z, we factorise A into LDU, where L is unit lower triangular, U is unit upper triangular, and D is a diagonal matrix. This factorisation is easily constructed from the previous one, by dividing elements of U by the diagonal elements. It is easily seen that

$$Z = D^{-1}L^{-1} + (I - U)Z \quad (3.4)$$

$$Z = U^{-1}D^{-1} + Z(I - L) \quad (3.5)$$

Note that  $(I - U)$  and  $(I - L)$  are strictly upper and lower triangular, respectively, and have a zero diagonal.  $D^{-1}$  is easily calculated and is also diagonal.  $U^{-1}$  and  $L^{-1}$  are upper and lower triangular, respectively, and have unit diagonals. Thus we use (3.4) to calculate elements of Z above the diagonal

$$z_{ij} = - \sum_{k=i+1}^n u_{ik} z_{kj} \quad (3.6)$$

and (3.5) for elements below the diagonal.

$$z_{ij} = - \sum_{k=j+1}^n z_{ik} l_{kj} \quad (3.7)$$

$U^{-1}$  and  $L^{-1}$  are not required. Elements on the diagonal of Z can be calculated from either

$$z_{ij} = 1/d_{ii} - \sum_{k=i+1}^n u_{ik} z_{ki} \quad (3.8)$$

or

$$z_{ii} = 1/d_{ii} - \sum_{k=i+1}^n z_{ik} l_{ki} \quad (3.9)$$

Normally we choose whichever formula involves fewer non-zero elements. Although the formulae involve  $Z$ , it can be shown that any element of  $Z$ ,  $z_{ij}$  (say), which corresponds to a non-zero in  $(LDU)'$ , can be calculated from the formulae, since the calculation involves only other elements of  $Z$ ,  $z_{st}$  (say), in the same subset and such that  $s \geq i$  and  $t \geq j$ . Knowing that we need the diagonal of  $Z$ , we can find from (3.8) or (3.9) which off-diagonal elements of  $Z$  are needed. Only those that will be multiplied by non-zero elements of  $U$  or  $L$  need be calculated. These elements of  $Z$  may in their turn require other elements. Eventually, we can find the complete set of elements that are needed and calculate them, starting with  $z_{nn}$  and working backwards.

It is interesting to note that this algorithm gives us 3 methods for calculating an element of the inverse's diagonal. (1) as an element of a row; (2) as an element of a column; and (3) using the Erisman/Tinney algorithm. Both (1) and (2) are provided by the sparse matrix routines. Tests of the Erisman/Tinney algorithm indicate that, although a rigorous error analysis is difficult, the value that it calculates for a diagonal element seldom differs from the values generated by (1) or (2) by any more than methods (1) and (2) already differ from each other. Comparison of the speeds of the two methods for calculating the diagonal of the inverse indicate that for the vast majority of cases the Erisman/Tinney algorithm is much faster. The naive method of calculating each row or column individually is only better in

those cases where the sparse factorisation of  $Q$  has a large number of non-zeroes in each row or column. Duff and Reid [13] have observed this phenomenon and attribute it to the method used to store the sparse representation of the matrix. Other representations of the matrix might well avoid this problem. For example, Duff [12] has written a set of sparse matrix routines which outperform those used in this work by a factor of 3 for typical test data, but they were not available at the time that this work was carried out. Another direct method, the AQ algorithm, has recently been developed by Borland [5] for solving linear equations. This method factors the matrix  $A$  into  $A=LQ$ , where  $L$  is a lower triangular matrix as before, and  $Q$  is an orthogonal matrix. Solutions to  $\underline{Ax}'=\underline{b}'$  are then found using

$$\underline{y}' = L^{-1}\underline{b}' \quad (3.10)$$

$$\underline{x}' = Q'\underline{y}' \quad (3.11)$$

(Recall that  $Q^{-1}=Q'$  for an orthogonal matrix.) Borland claims that there are great savings to be made using this algorithm, since advantage can be taken of sparsity in  $\underline{b}$  as well as in  $A$ .  $\underline{b}$  is very sparse when we are finding the inverse, since it will be the unit vector. Another advantage is that the  $LQ$  factorisation often has fewer non-zero elements than the  $LU$  factorisation of the same matrix. (Note that the  $L$  matrices are not the same in these factorisations.) Unfortunately, the method has only recently been published and in the absence of tried and tested subroutines to perform it, there was not time to do any tests with the algorithm.

Theory predicts that all elements of  $Q^{-1}$  will be of the same sign, since their ratios approximate the ratios of probabilities, which are all non-negative. In numerical practice this is not always the case. For example, the  $Q$  matrix corresponding to the  $M/M/1$  system with arrival rate 0.1 and service rate 1 is tridiagonal. Its inverse is easily calculated, and is positive. The sparse matrix routines, however, give a negative estimate when the truncation is larger than 20 states. If the truncated matrix is inverted by a standard subroutine, taking no account of its sparsity, it is reported to be singular.

To allow for these difficulties, a fairly generous policy is followed to give an estimate for  $\pi_j$ . If either, or both, of the estimates given by the less severe approximations (3.2) falls within the interval defined by the upper and lower bounds (3.1), then it, or their mean, is used for  $\pi_j$ . If the less severe approximations lie outside the interval given by the bounds, then the mid-point of the interval is used. If the upper bound, or the lower, is negative, the positive one is used. If all the approximations are negative, then zero is used as an estimate.

### 3.2 Stewart's Method

Stewart has developed a simultaneous iteration method for finding partial eigensolutions of matrices. The method calculates the subset of the eigenvectors corresponding to the dominant subset of eigenvalues. Both right-hand and left-hand eigenvectors are found, but in our case,

we shall only need the dominant left eigenvector of  $P$ , the jump chain matrix of the process. A variant of the algorithm, lopsided simultaneous iteration, converges to either the left-hand or the right-hand eigenvectors.

When using this method, or the power method, we need to construct  $P$ , the jump chain matrix, from  $Q$ . Clearly,  $P = I + Q\Delta t$  gives the probability that a transition will be made between states  $i$  and  $j$  in the time interval  $(t, t + \Delta t)$ , given that we are in state  $i$  at time  $t$ . Since the system is Markovian, and thus memoryless, the only restriction that we need to place on  $\Delta t$  being that  $P$  must be stochastic. If  $P$  is stochastic, then  $\underline{e}'$ , the column vector with each entry 1, is a right eigenvector and  $\underline{\pi}$  is a left eigenvector. Recalling that  $\underline{\pi}Q = \underline{0}$ , the factor  $\Delta t$  will not affect the eigenvector provided that  $P$  remains stochastic.

Let  $R$  denote the absolute value of the maximum modulus diagonal element of  $Q$ . Since  $Q$  is constructed to have zero row sums and all the off-diagonals are non-negative, the diagonal must be negative. If  $0 < \Delta t \leq 1/R$  then the row sums of  $P$  will be 1, and the elements of  $P$  will be non-negative, which is the definition of a stochastic matrix. By Gerschgorin's theorem, no eigenvalue of  $P$  can be greater than 1, which is trivially an eigenvalue. If  $\Delta t < 1/R$  then all the eigenvalues of unit modulus must be equal to 1, since the eigenvalues are contained in the union of the circles, centre  $p_{ii}$ , radius  $1 - p_{ii}$ , in the complex plane. The circumferences of these circles all meet at the point 1 on the real axis. If  $\Delta t = 1/R$  then the unit circle centred at the origin contains

all the eigenvalues, but there can now be complex eigenvalues of unit modulus. In the presence of multiple eigenvalues of the same modulus the power method converges to a vector which is a linear combination of the corresponding eigenvectors. Stewart's method [27] will converge correctly even in the presence of several equal modulus eigenvalues, although we must use more eigenvector estimates than the matrix has equal modulus eigenvalues. It is also important to choose  $\underline{\pi}$  as the eigenvector corresponding to 1, and not to some complex eigenvalue. If the matrix  $P$  is irreducible, then the unit modulus eigenvalue will be unique. Normally our processes will give rise to an irreducible  $P$ , but not always. Seneta [43] proves that matrices corresponding to processes with a single essential class of states, have a unique unit eigenvalue and corresponding (left) eigenvector  $\underline{\pi}$ . We shall always construct  $Q$  for our processes, such that there is only a single essential class of states. Thus convergence of the power method must be to the eigenvalue 1, and the eigenvector  $\underline{\pi}$ , at a rate which depends on the ratio of the dominant eigenvalue (1) to the subdominant eigenvalue. Stewart's method will also converge and its rate of convergence depends on the ratio of the dominant eigenvalue to the maximum modulus eigenvalue whose corresponding eigenvector is not being found.

The choice of  $\Delta t$  should be made so that the rate of convergence to the dominant eigenvalue is as fast as possible. Wallace [50] recommends that  $\Delta t$  should be chosen as large as possible, and since he was using the power method, used  $\Delta t = 0.99/R$ . When using Stewart's method, we can choose  $\Delta t = 1/R$ , and still converge on the correct eigenvector. Numerical experiments with an M/M/1 system indicate that a small  $\Delta t$  will



tend to cluster the eigenvalues close to 1. For example, when the arrival rate was 0.8, the service rate 1, and  $\Delta t = 0.1/1.8$ , then the eigenvalues all fell in the range (0.8,1.0). When  $\Delta t$  increased, so did the interval containing the eigenvalues. In this case the eigenvalues were relatively evenly distributed along the interval, but M/M/1 is very well behaved anyway.

Stewart's method consists of choosing a set of estimates,  $U$ , for the  $m$  dominant eigenvectors of the matrix,  $A$  (say), for which we wish to find the eigensolution. We shall consider right eigenvectors here, but there is no loss of generality, since to find left eigenvectors we apply the same algorithm to  $A'$ . The first column of  $U$  is our estimate for the dominant eigenvector, the second column for the sub-dominant eigenvector, and so on. We then perform the following operations until the estimates of the eigenvectors converge.

$$V = AU \quad (1)$$

$$G = U'U \quad (2)$$

$$H = U'V \quad (3)$$

$$GB = H \quad (4)$$

$$E = \text{Eigenvectors of } B. \quad (5)$$

$$W = VE \quad (6)$$

$$\text{Normalise } W \text{ and test for convergence.} \quad (7)$$

$$U = W \quad (8)$$

B is a matrix which abstracts the relationship between the eigenvectors, and it is called the interaction matrix. If this interaction analysis is not performed, then the columns of U would all converge on the dominant eigenvector! Stewart has identified various problems which can occur with this simultaneous iteration method. He has also developed several optimisations of the general algorithm stated above. First, the initial estimate of  $\underline{\pi}$  might be orthogonal to  $\underline{\pi}$ . In this case, convergence would not be to  $\underline{\pi}$ , but to some other eigenvector. This is extremely unlikely, and has never been observed, but can be overcome by replacing the trial eigenvector corresponding to the least significant eigenvalue by a random vector at each iteration. G is symmetric and positive-definite, by construction, so that the equation solution implied by step (4) can be accurately performed without pivoting, using Choleski's algorithm. Occasionally, the interaction matrix, B, is defective. This can happen because A has a defective set of dominant eigenvectors, or by chance in the course of the iterations. If B is defective, then the eigenvector estimates in E will be almost parallel, and in the following iteration the equation solution at step (4) may fail. In this case, one can either omit the interaction analysis for that iteration, or follow Stewart's suggested solution, which is to modify G by adding machine epsilon to the diagonal and re-solve step (4). Another improvement is to perform several 'power' type iterations for each interaction analysis. This involves replacing step (1) with  $V=A^n U$ , where n is the number of 'power' type iterations to perform. The optimum number can be calculated from the rate at which the trial eigenvectors are changing. The calculation of the complete set

of eigenvectors of the interaction matrix,  $B$ , can be carried out by any method. We use the QR algorithm. It is important that the eigenvectors of  $B$  are stored in  $E$  ordered according to the magnitude of the corresponding eigenvalues of  $B$ . That is, the first column of  $E$  should contain the dominant eigenvector of  $B$ , and so on. Precautions must also be taken to ensure consistent ordering of these eigenvalues from one iteration to the next. For example, if the eigenvalues of  $B$  contain  $5$  and the conjugate pair  $3 \pm 4i$ , they must always be sorted such that  $5 > 3 \pm 4i$  or  $3 \pm 4i < 5$  consistently. In fact, for our purposes, we must ensure that we sort  $1$  as the largest eigenvalue, and any unit modulus complex eigenvalues as subdominant. The eigenvalues of  $B$  are the best approximation available to the eigenvalues of  $A$ .

### 3.3 Brandwajn's Method

This method is an iterative procedure for solving the global balance equations (1.4). It is specifically designed for solving Markov processes, since it features an unusual relaxation step designed to maintain the invariance of  $\sum \pi_i$  from one iteration to the next. The method is generalised from one presented in [6]. The paper presents another method also, based on the equivalence and decomposition approach to solving queueing networks, which uses the concepts of conditional probability. Although the method we use compares unfavourably with the other developed in that paper, this method is always convergent, and does not depend on the ordering of states.

As with Stewart's method we choose a first approximation to  $\underline{\pi}$  and denote it by  $\underline{\pi}(0)$ .  $\underline{\pi}(t)$  denotes the estimate of  $\underline{\pi}$  after  $t$  iterations. For each  $i$ , from 1 to  $n$ , in that order, we perform the following calculation.

$$\begin{aligned} \pi_i(t+1) = & \left[ \pi_i(t) \left( 1 - \Omega \sum_{j=1}^{i-1} q_{ij} \right) \right. \\ & \left. + \Omega \left( \sum_{j=1}^{i-1} \pi_j(t+1) q_{ij} + \sum_{j=i+1}^n \pi_j(t) q_{ij} \right) \right] \\ & \div \left[ 1 + \Omega \sum_{j=i+1}^n q_{ij} \right] \end{aligned} \quad (3.12)$$

$\Omega$  is an arbitrary constant. If  $\Omega$  is positive and  $\Omega < 1/\max \sum q_{ij}$  then we can easily see that no  $\pi_i(t+1)$  will be negative, assuming that all  $\pi_j(t)$  were non-negative.

To show the invariance of  $\sum \pi_i$ , rearrange the equations (3.12) to give

$$\begin{aligned} \pi_i(t+1) - \pi_i(t) = & -\Omega \left( \pi_i(t) \sum_{j=1}^{i-1} q_{ij} + \pi_i(t+1) \sum_{j=i+1}^n q_{ij} \right) \\ & + \Omega \left( \sum_{j=1}^{i-1} \pi_j(t+1) q_{ij} + \sum_{j=i+1}^n \pi_j(t) q_{ij} \right) \end{aligned} \quad (3.13)$$

When we sum all these equations together the terms weighted by  $\Omega$  add to give zero, thus

$$\sum_{i=1}^n \pi_i(t+1) = \sum_{i=1}^n \pi_i(t) \quad (3.14)$$

Defining  $d_i(t) = \pi_i(t) - \pi_i(t-1)$ , we clearly have  $\sum d_i(t) = 0$ , for all  $t$ . By rearranging (3.13) subtracting to give  $d_i(t+1)$  in terms of  $d_i(t)$ , we can show that

$$\sum_{i=1}^n |d_i(t+1)| < \sum_{i=1}^n |d_i(t)| \quad (3.15)$$

which proves that the method converges.

Attempting to analyse the rate of convergence of the method is difficult. Although superficially it is akin to successive over relaxation, the relaxation factor is different for each of the equations.

### 3.4 Analysis of Algorithms

Attempting to analyse the performance of the algorithms a priori, rather than experimentally, is difficult. The efficiency of the direct method based on the ratios of cofactors of the truncation will depend heavily on the size and pattern of the non-zero elements. The iterative algorithms of Stewart and Brandwajn will not be affected by the sparsity or otherwise of  $Q$ , but the proximity of the first estimate for  $\underline{x}$  to the final solution will critically affect the number of iterations taken.

For the purpose of analysing the algorithms, we shall assume that the system being modelled has been truncated at state  $n$ , giving rise to a  $Q$  matrix containing  $x$  non-zero elements. Note that the diagonal elements will always be non-zero. Also note that  $x$  will be of order  $n$ , rather than of order  $n^2$ . When using Stewart's algorithm, we assume that  $m$  eigenvalues and corresponding eigenvectors are being found. For the iterative methods, we only give a measure of the number of operations per iteration.

In terms of space utilisation, all the algorithms require some representation of  $Q$ . The representation we use is that expected by our sparse matrix handling subroutines. The non-zero elements are stored in an array in column order, and in row order within each column. A corresponding array holds the row index of the elements, and a smaller array holds the index of the first element in each column. This requires  $x$  locations to store real numbers and  $x+n+1$  locations to store integers. Although designed to be used by the Gaussian elimination routines, this representation is as good as any other for Stewart's or Brandwajn's methods, both of which basically require the ability to post-multiply a vector by the sparse matrix. We shall compare the different algorithms vis-a-vis their extra space requirement and the number of floating point operations, both additions and multiplications, that they involve. We shall also assume the use of  $2n$  locations to store the latest estimate for  $\pi$  and the previous estimate.

The direct method of inversion of  $Q$  uses an unpredictable amount of extra space and of time. The pattern of the non-zero elements and, to a lesser extent, their size affects the performance dramatically. Although our  $Q$  matrices are far from random, either in size or position of the non-zeroes, the only practical approach to analysing the algorithm must assume that they are random matrices, in which all off-diagonal elements have equal probability of being non-zero. This probability is independent of the presence or absence of other non-zero elements in the same row or column. As was pointed out previously, we do not know enough about the general case to make any other assumptions. Also, although the subroutines choose the pivots to factorise  $Q$  depending not only on their

size but also to minimise the number of zeroes that become non-zero, the only practical analysis assumes that the diagonal elements are chosen as pivots. Thus the analysis should give an upper bound on the space used by the method.

This analysis of sparse Gaussian elimination was developed by Duff [11] and is extended here to encompass the Erisman/Tinney algorithm. We assume that all the diagonal elements are non-zero and that all the off-diagonal elements have equal probability,  $p$ , of being non-zero. In our case  $p = (x-n)/n(n-1)$ . As the elimination proceeds, we are working on smaller and smaller matrices, in which the probability of the elements being non-zero becomes larger. After  $i$  variables have been eliminated we are considering a matrix of size  $(n-i) \times (n-i)$ , and we denote the probability that an off-diagonal element of such a matrix is non-zero by  $p_{i+1}$ . Thus  $p_1 = p = (x-n)/n(n-1)$ . The obvious first approach to the problem of finding  $p_i$  for  $i=2, \dots, n-1$ , argues as follows. An element will be non-zero at stage  $i$  if it was non-zero at stage  $i-1$ , (probability  $p_{i-1}$ ), or if it was zero at that stage (probability  $1-p_{i-1}$ ) and the elements at the head of its row and column were both non-zero (probability  $p_{i-1}^2$ ). Hence we have

$$p_i = p_{i-1} + (1-p_{i-1})p_{i-1}^2 \quad (3.16)$$

Duff shows that this is an upper bound, even for the case of diagonal pivoting, and proves that the correct formulae are given by

$$p_i = 1 - \sum_{k=1}^{i-1} \binom{i-1}{k-1} h_k (1-p_1)^{k(i+1-k)} \quad (3.17)$$

where

$$h_k = 1 - \sum_{j=1}^{k-1} \binom{k-1}{j-1} h_j (1-p_1)^{j(k-j)} \quad (3.18)$$

In either case, or if some better estimate of  $p_i$  could be calculated using some knowledge of the structure of  $Q$ , or of the pivotal strategy involved, the values of  $p_i$  give the probabilities of a non-zero element in row  $i$  of  $U$  or in column  $i$  of  $L$ , after factorisation. Thus the expected number of non-zeroes in row  $i$  of  $U$  or column  $i$  of  $L$  is  $(n-i)p_i$  and the total number of non-zero elements in the factorised form of  $Q$  is

$$n + 2 \sum_{i=1}^{n-1} (n-i)p_i \quad (3.19)$$

The number of operations performed on each element will give a measure of the time that the algorithm will take. The elements of the original matrix can be divided into two classes as far as the number of operations on them are concerned. The elements on the diagonal will be operated on each time the elements at the ends of its row and column are non-zero, which occurs at stage  $i$  with probability  $p_i^2$ . Thus the expected number of operations on the element in the  $i,i$  position is  $\sum p_k^2$ . For off-diagonal elements Duff shows that the expected number of operations on an element in the  $i,j$  position is given by

$$\min\{i,j-1\} \sum_{k=2}^{\infty} (p_k - p_{k-1}) / (1 - p_{k-1}) \quad (3.20)$$

This counts the number of operations, in fact they are additions, performed on these elements. Each operation on an element consists of multiplying two other (non-zero) elements together, and adding the



result to the element being operated on. Hence, the expected total number of additions to factorise the matrix is

$$\sum_{i=2}^{n-1} \left\{ \sum_{k=1}^{i-1} p_k^2 + 2(n-i) \sum_{k=2}^i (p_k - p_{k-1}) / (1 - p_{k-1}) \right\} \quad (3.21)$$

and a similar number of multiplications are performed. Note that this takes no account of the housekeeping operations needed to keep track of which element is which in the sparse matrix, but just the operations on the non-zero elements themselves. Also, since the number of operations performed is highly dependent on the fillin of zero elements, we can expect that these formulae will in fact overestimate the operations to be performed.

The Erisman/Tinney algorithm that we use to calculate the diagonal elements of the inverse can also be analysed using Duff's result. In row  $i$  of  $U$ , or column  $i$  of  $L$ , there are  $(n-i)p_i$  non-zero elements. Calculation of an arbitrary diagonal element of the inverse,  $Z_{ij}$ , requires the calculation of the  $(n-i)p_i$  elements in the appropriate row or column of  $Z$  which correspond to the non-zero elements in column  $i$  of  $L$  or row  $i$  of  $U$ . Having calculated them, we require  $(n-i)p_i$  operations to calculate the diagonal element. Again each operation involves a multiplication and an addition. Each element of the appropriate row or column itself requires the calculation of  $(n-i-1)p_i$  other elements of  $Z$ . Thus to calculate the diagonal element and all the elements in the  $i$ th row and column of  $Z$  which can be found using this method, we need  $(n-i)p_i(1+2(n-i-1)p_i)$  operations. Hence the total number of operations required to calculate the complete subset of elements of the inverse is

$$\sum_{i=1}^{n-1} (n-i)p_i [1+2(n-i-1)p_i] \quad (3.22)$$

This is a gross overestimate of the number of operations that will be performed to calculate only the diagonal elements of the inverse. As we have noted already, the estimate we have for  $p_i$  is an overestimate, or even an upper bound if we calculate it crudely. Even if we know the  $p_i$  more exactly, this estimate assumes that we calculate the complete subset of elements of the inverse that the algorithm provides. This is not the case. We calculate only the subset of elements that are needed to calculate the diagonal. Consider the element  $z_{11}$ . We can calculate this using either

$$z_{11} = 1/d_{11} - \sum_{k=2}^n u_{1k} z_{k1} \quad (3.23)$$

or

$$z_{11} = 1/d_{11} - \sum_{k=2}^n z_{1k} l_{k1} \quad (3.24)$$

Normally, if there are fewer non-zeroes in row 1 of U than there are in column 1 of L, we calculate using the first formula. In this case, the elements of the first row of the inverse will never be needed. The picture is not so clear for later elements on the diagonal, where the needed elements will depend on the pattern of non-zeroes. In practice, about 75% of the possible elements of the inverse need to be calculated in order to find all the diagonal elements.

The extra space required by the routines can be calculated using Duff's result. We denote the number of non-zeroes in the L/U

factorisation by  $f(x)$ , which we calculated above. The factorisation routines require  $f(x)-x$  extra locations to store real numbers, the new non-zeroes. The inversion routine also needs  $f(x)$  real storage locations for the elements of the inverse. To store pivoting information, row and column numbers, and similar housekeeping data, an extra  $14n+f(x)-x$  integers are needed. A further  $3n$  real locations store the first row, first column, and leading diagonal of the inverse. The grand total is  $2f(x)-x+3n$  real locations, and  $14n+f(x)-x$  integer locations.

Other sparse matrix routines which code the structure of the non-zeroes in a different way might have different space overheads. For example, Duff's routines [12] require  $2f(x)-x+4n$  extra real locations and  $15n+f(x)-x$  integer locations, to factorise the matrix. The Erisman/Tinney algorithm encoded in a suitable form for these subroutines would need at least another  $f(x)$  real locations.

As a worst, in some sense, case example of the errors that can arise by treating our matrices as random consider the M/M/1 system. It gives rise to a tri-diagonal Q matrix. The truncation of size  $n$  contains  $3n-2$  non-zero elements and no zero elements are filled in during the elimination so the L/U factorisation contains  $3n-2$  non-zeroes. Consideration of the Erisman/Tinney algorithm applied to this system shows that the elements of the inverse's diagonal can be calculated by the following recurrence

$$z_{ii} = 1/d_{ii} - u_{ii+1}z_{i+1i} \quad (3.25)$$

$$z_{i+1i} = -z_{i+1i+1}l_{i+1i} \quad (3.26)$$

(We could have used the alternative formula for the diagonal elements, but the same number of elements of  $Z$  is needed.) Only  $2n-1$  elements of the inverse are calculated.

As remarked above, the estimates of  $p_i$  are expected to be too large for several reasons. To give a more realistic example of the extent of the overestimate we give in Table 3.1 the actual number of non-zeros in the L/U factorisation of the  $Q$  matrix for several different systems, each truncated at several different states. For comparison, the number of non-zeroes predicted by (3.19), both using the naive estimates (3.16) and Duff's exact results (3.17),(3.18). Although superficially simple, Duff's results are very difficult to calculate with in practice. The summation in (3.18) is very close to 1, and the cancellation error which occurs when it is subtracted from 1 dominates the calculation for any reasonable size of  $n$ . Even the use of quadruple precision real numbers (about 32 decimal places) only delays the onset of the problem. Most of the numbers in Table 3.1 were calculated using exact rational arithmetic. For interest, we also tabulate the number of elements of the inverse that were needed to calculate the inverse's diagonal using the Erisman/Tinney algorithm, and  $p$ , the probability that an off diagonal element is non-zero.

Model 1 is an M/M/1 system. As noted above, the  $Q$  matrix is tri-diagonal, and no fill-in of non-zero elements occurs in practice. Model 2 is a two class non-preemptive priority system. There are about 4

non-zeroes in each row. Two examples at each truncation size are given, to illustrate the effect of different sizes of element on the factorisation. In each case, the pattern of non-zeroes is the same. Model 3 represents a 2 processor network in which each processor can either be working or broken. In either state, the processors transfer customers to the other node, although at different rates. There are approximately 9 non-zero elements in each row. The fourth model is of two parallel M/M/1 queues. Arriving customers join a queue at random and remain in that queue until served. There are 5 non-zeroes in each row of Q.

Table 3.1

Model	n	nnz	p	number of non zeroes			
				Experiment	Duff	Naive	Inverse
1	15	43	0.1333	43	67	78	29
	20	58	0.1	58	99	125	39
	100	298	0.02	298	1024	2479	199
2	42	125	0.0482	163	285	477	107
	42	125	0.0482	172	285	477	117
	110	365	0.0212	629	1714	3895	472
	110	365	0.0212	579	1714	3895	375
3	84	532	0.0642	1500	3506	4631	1267
	112	732	0.0409	2450	6209	8338	*12544
	220	1524	0.0271	7228	23975	33055	*48400
4	120	540	0.0294	1318	4320	7252	1025

The entries marked \* represent the full inverse matrix. In these cases, the Erisman/Tinney algorithm was significantly slower than the naive method of finding the inverse.

The table shows the drawback of Stewart's special purpose routines [47] which use the diagonal elements as pivots. They give rise to excessive fillin, limiting the size of system that can be solved.

It is worth noting that the conjugate gradient method takes  $5n+2x$  multiplications and additions per iteration. To solve a single set of equations needs  $(5n+2x) \cdot i+3x$  additions and multiplications, where  $i$  is the number of iterations taken. Since we have to solve  $n$  sets of equations, and seldom take fewer than  $2n$  iterations for each set, the conjugate gradient method uses of the order of  $n^3$  additions and multiplications. Its extra space requirements are quite small, only 2 extra vectors both containing  $n$  real locations.

Stewart's method can be simply analysed as far as a single iteration is concerned. The number of iterations needed to reach an acceptable solution is highly problem dependent. Each iteration consists of multiplication of the  $n \times m$  matrix of the eigenvector estimates by the sparse matrix, followed by an eigenvector interaction analysis and normalisation of the estimates. A vector can be multiplied by the sparse matrix with  $x$  floating point multiplications and  $x$  floating point additions. Thus the multiplication takes  $m$  times this number of operations. The interaction analysis involves 3 multiplications of an  $n \times n$  matrix by an  $n \times m$  matrix, taking  $3nm^2$  multiplications and additions; the solution of  $m$  linear equations in  $m$  variables with  $m$  right-hand sides, using Choleski's method which takes  $7/6m^3 - 1/2m^2 + 1/3m$  multiplications and  $7/6m^3 + m^2 - 13/6m$  additions; a complete set of eigenvectors for an  $n \times n$  matrix by the QR algorithm, taking  $O(m^3)$

multiplications and additions. If we assume that each eigenvalue of the interaction matrix takes 10 iterations to be found, then the QR algorithm uses  $25m^3 + 57m^2 - 123m$  multiplications and  $25m^3 + 52m^2 - 160m$  additions. The normalisation consists of dividing the elements of each eigenvector in order to make the largest element 1, and uses  $mn$  multiplications.

No extra storage is needed for integers, but  $3m^2$  real locations are used by the interaction analysis phase, and an extra  $(m-2)n$  real locations are needed to store the sub-dominant eigenvectors.

Brandwajn's method is the most thrifty of the three algorithms analysed. As with Stewart's method we can only find the number of operations per iteration. Each iteration takes  $x+2n$  multiplications and  $x+n$  additions to form the new estimate of  $\Sigma\pi_i$ . Before the iterations start, there is a once and for all overhead of  $x$  additions and  $2n$  multiplications involved in calculating  $\Omega$  and the row sums. The only extra storage is  $2n$  real locations used to hold the sums of the sub-diagonal and super-diagonal elements in each row of  $Q$ .

We can use these operation counts to estimate when the iterative methods of Stewart and Brandwajn are preferable to the direct method. We calculate  $k(x,n)$ , the maximum number of iterations each method is allowed to take and still be more effective than the direct method. Clearly,  $k$  is a function of the number of non-zeroes,  $x$ , and the size of the truncation,  $n$ . In order to evaluate  $k$  for various values of  $x$  and  $n$ , we make the following two assumptions. First, we consider that  $x$  is a linear function of  $n$ . This is a reasonable assumption, often borne out

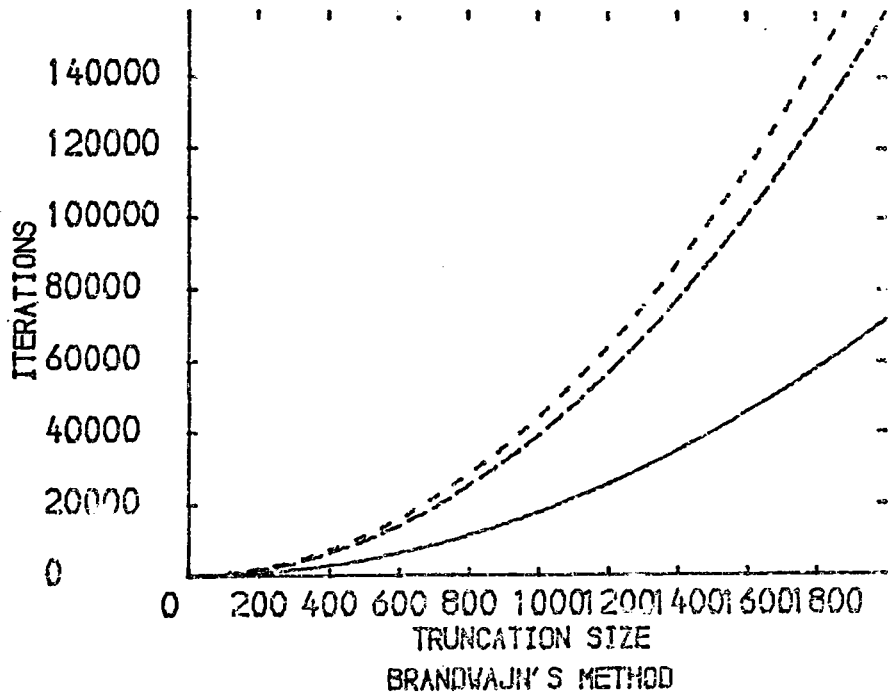
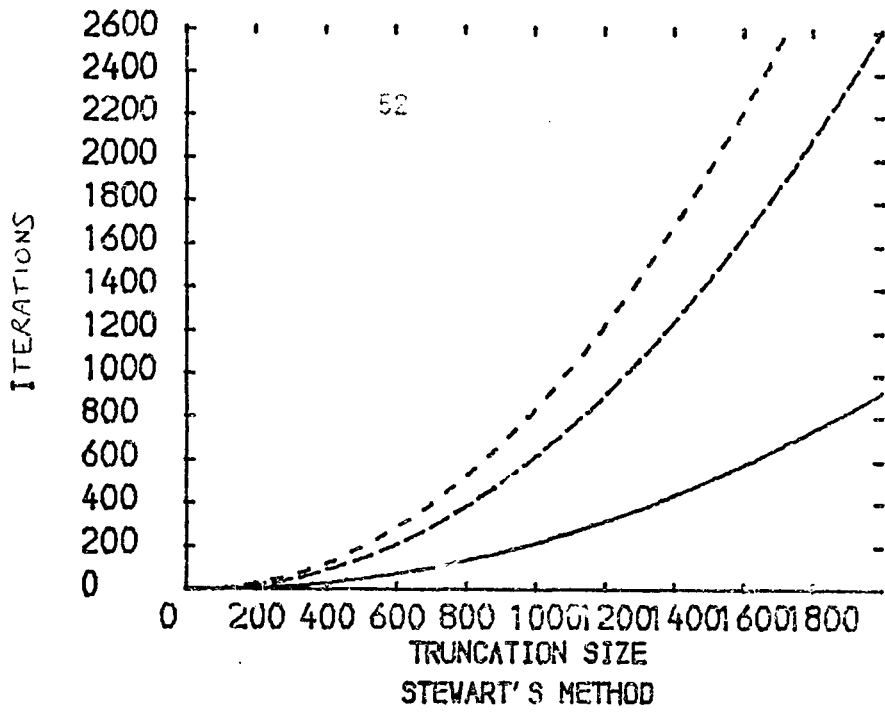


in practice. Since, as we remarked earlier, Duff's exact results are very hard to evaluate accurately, the naive estimates (3.16) were used to calculate  $k(x,n)$ . This means that the function  $k$  is biased against the direct method of solution. We also assume that  $m$ , the number of eigenvector estimates in Stewart's method, is calculated according to the following formula. If  $n < 30$  then  $m=3$ ; if  $n \geq 100$  then  $m=10$  else  $m=n/10$ . Floating point additions are assumed to take 1 unit of time, and multiplications take  $f$  units. All calculations are made with  $f=1.2$ .  $k(x,n)$  is graphed for a selection of values of  $x$  and  $n$  in Figure 3.1.

### 3.5 Theoretical Application

In this section, we apply Tweedie's method to the general one dimensional birth-death process and show that the upper bound that he derives is attained. When applied to the M/M/1 system, the errors involved in using the lower bound or the less stringent approximation can be found. An attempt is also made to analyse the performance of Stewart's method on the M/M/1 system.

The one dimensional birth-death process is a conservative Markov process, with its states indexed by the positive integers. (Some authors use the non-negative integers, but there is no loss in generality involved in ignoring 0). Transitions occur from state  $i$  to state  $i+1$  at rate  $\lambda_i$  and to state  $i-1$  at rate  $\mu_i$ . Its behaviour has been extensively investigated, but the result that we shall need is



NONZEROS PER ROW

- 3 —————
- 4.5 - - - - -
- 6 - . - . - .

FIGURE 3.1

$$\pi_m = \pi_1 \prod_{t=2}^m \frac{\lambda_{t-1}}{\mu_t} \quad (3.27)$$

where  $\pi_1$  can be found from the normalising condition. (See, for example, [40, Pp 83-87].) Since Tweedie's method only gives us the ratio between probabilities anyway, this normalising factor is of no account. The  $Q$  matrix of the birth-death process is particularly simple, being tri-diagonal.

$$\begin{array}{cccccc} -\lambda_1 & \lambda_1 & 0 & 0 & \dots \\ \mu_2 & -(\lambda_2 + \mu_2) & \lambda_2 & 0 & \dots \\ 0 & \mu_3 & -(\lambda_3 + \mu_3) & \lambda_3 & \dots \\ \cdot & \cdot & \cdot & \cdot & \dots \end{array}$$

To use Tweedie's method we need the ratios of cofactors of  ${}_{(n)}Q$ , or equivalently, the ratios of elements of its inverse matrix. If we let  $\theta = {}_{(n)}Q^{-1}$  then we can calculate  $\theta$  in the following manner. Take two matrices,  $A$  and  $B$ , and initialise them  $A := {}_{(n)}Q$  and  $B := I$ . We then perform exactly the same elementary row operations on  $A$  and  $B$ , with the object of reducing  $A$  to  $I$ . This is equivalent to multiplying  ${}_{(n)}Q$  by  $\theta$ , so that  $B$  will contain the value of  $\theta$ . The easiest way to reduce  $A$  to  $I$  is as follows. First we reduce  $A$  to a unit upper triangular matrix. Assume that columns 1 to  $i-2$  are already reduced; that is they contain only zeroes below the diagonal. We add  $-\mu_i$  times row  $i-1$  to row  $i$ , and thus make the element  $a_{i,i-1}$  zero. Next, we divide row  $i$  by  $a_{i,i}$  to make the diagonal element unity. When this has been done for  $i=1,2,3,\dots,n$ ,  $A$  is reduced to a matrix with unit diagonal,  $-1$  in all elements  $a_{i,i-1}$ , and zero everywhere else. This is easily reduced to the identity matrix by adding row  $i$  to row  $i-1$  for  $i=n,n-1,\dots,3,2$ . Performing the same

operations on I, in parallel, gives the values of  $\theta$ . They are easily shown to be

$$\theta_{ij} = \frac{1}{\lambda_j} \sum_{k=j}^n \prod_{t=j+1}^k \mu_t / \lambda_t \quad (i \leq j) \quad (3.28)$$

$$\theta_{ij} = \frac{1}{\lambda_j} \sum_{k=i}^n \prod_{t=j+1}^k \mu_t / \lambda_t \quad (j \leq i) \quad (3.29)$$

Tweedie's theorems give the upper bound on the ratio of  $\pi_m / \pi_1$  as  $\theta_{mm} / \theta_{m1}$ . On substituting the values for  $\theta$  found above, we get

$$\prod_{t=2}^m \frac{\lambda_{t-1}}{\mu_t}$$

which is the correct answer. If the values for  $\theta$  are substituted in the other formulae, then the lower bound is not exact. Rather than pursue the general one-dimensional birth-death process, we turn to a specific case which is well known, the M/M/1 queue.

The M/M/1 queue is a special case of the one dimensional birth-death process, with  $\lambda_i = \lambda$ , the arrival rate, and  $\mu_i = \mu$ , the service rate, for all  $i$ . It is very well understood and both steady state and transient probability distributions are known. The steady state distribution of the M/M/1 queue is given by  $\pi_i = \rho^{i-1} (1-\rho)$  where the states are numbered so that state  $i$  represents the state of the system in which there are  $i-2$  waiting customers and 1 customer being served, and where  $\rho = \lambda/\mu$  is the traffic intensity.  $\pi_1$  represents the idle system. For the M/M/1 queue with arrival rate  $\lambda$  and service rate  $\mu$ , the transition rate matrix  $Q$  is given by

$$\begin{array}{cccc}
 Q = & -\lambda & \lambda & 0 & 0 \dots \\
 & \mu & -(\lambda+\mu) & \lambda & 0 \dots \\
 & 0 & \mu & -(\lambda+\mu) & \lambda \dots
 \end{array}$$

If we substitute the constants  $\lambda$  and  $\mu$  in the equations (3.28) and (3.29) above we find that, after a little algebra, replacing  $\lambda/\mu$  by  $\rho$  and casting out common factors (since we are only interested in ratios between elements), the elements of  $\theta$  are

$$\begin{aligned}
 \theta_{ij} &= \rho^{j-1}(1-\rho)^{n-j+1} && (i \leq j) \\
 &\rho^{j-1}(1-\rho)^{n-i+1} && (j \leq i)
 \end{aligned}$$

Let us now apply these formulae to the problem of estimating  $\pi_k$ . In this case,  $\pi_1$  is known from other considerations. (Little's theorem gives the probability of the idle state for any single server queueing system as  $1-\rho$ .) Anyway we need to choose an arbitrary state to normalise the probabilities with and state 1 is as good as any. Having chosen one state we can not use the formulae to estimate  $\pi_k/\pi_1$ . The bounds in equation (3.1) give us an upper and lower bound by taking  $j=1$  and  $i=k$ . Equation (3.2) gives us two approximations; one taking  $j=1$  and  $i=k$ , and one taking  $j=k$  and  $i=1$ . Only one set of bounds are found because taking  $j=k$  and  $i=1$  in (3.1) gives the same bounds.

Substitution of the formula for  $\theta_{ij}$  in the appropriate formulae gives the following estimates for  $\pi_k$ .

$$\begin{aligned} \text{lower} = lb_k &= \frac{\theta_{1k}}{\theta_{11}} = \rho^{k-1} \frac{(1 - \rho^{n-k+1})}{(1 - \rho^n)} \\ &\rightarrow \rho^{k-1} \text{ as } n \rightarrow \infty. \end{aligned}$$

$$\begin{aligned} \text{upper} = ub_k &= \frac{\theta_{kk}}{\theta_{k1}} = \rho^{k-1} \frac{(1 - \rho^{n-k+1})}{(1 - \rho^{n-k+1})} \\ &= \rho^{k-1} \text{ for all } n. \end{aligned}$$

$$\begin{aligned} \text{approx} = ap1_k &= \frac{\theta_{k1}\theta_{1k}}{\theta_{11}^2} = \rho^{k-1} \frac{(1 - \rho^{n-k+1})}{(1 - \rho^n)^2} \\ &\rightarrow \rho^{k-1} \text{ as } n \rightarrow \infty. \end{aligned}$$

$$\begin{aligned} \text{approx} = ap2_k &= \frac{\theta_{kk}^2}{\theta_{k1}\theta_{1k}} = \frac{(\rho^{k-1} (1 - \rho^{n-k+1}))^2}{\rho^{k-1} (1 - \rho^{n-k+1})^2} \\ &= \rho^{k-1} \text{ for all } n. \end{aligned}$$

In this case  $ub_k$  and  $ap2_k$  can be disregarded since they give the correct answer regardless of any truncation. Note that even if we do not know  $\pi_1$  the error will be of the same magnitude for all  $k$ . If  $\pi_1$  is found from the normalising condition, its value will be the same as if the M/M/1/n queueing system was solved. Exact formulae can be calculated for the relative errors in  $lb_k$  and  $ap1_k$ , but they are not given here since no particularly elegant form results. They have the property, however, that the error in any particular probability,  $\pi_k$ , depends not only on  $n$ , the size of the truncation, but also on  $k$ , the state being approximated. It is also interesting to note that we have  $ap1_k < lb_k$ . This is explained by the fact that the approximations converge under less strict pre-conditions on the states. The more stringent conditions

under which the bounds converge are of no practical hinderance to us. In processes with a more irregular structure than M/M/1 the two approximations often both lie outside the interval given by the lower and upper bounds, generally one on either side of the interval. These two properties, the exact answer being given by one of the two bounds and one of the approximations lying outside the lower bound to upper bound interval, seem to partly be functions of the state chosen to normalise against. If the ratios  $\pi_k/\pi_n$  are calculated, then the lower bound and the other approximation give the correct answer and the upper bound is less than the first approximation. This property is not governed by the sizes of the probabilities, which might be conjectured, since in this case  $\pi_1$  is the largest, and  $\pi_n$  the smallest probability. The Q matrix corresponding to the M/M/2 queueing system is tridiagonal, and the system is a one-dimensional birth-death process. Thus, by our first investigation, equations (3.28) and (3.29) will apply, and the upper bound will give the correct answer. By choice of the values of  $\lambda$  and  $\mu$ , it is possible to have  $\pi_2$  as the largest probability though.

We now turn to Stewart's method of simultaneous iteration to find the dominant left eigenvector of P. We shall attempt to analyse its performance on the M/M/1 system. In von-Mises power method for finding the dominant eigenvector, the rate of convergence is proportional to  $|\Omega_1/\Omega_2|$  where  $\{\Omega_i\}$  are the eigen values of P, in order of magnitude. ( $\Omega$  is chosen rather than the more conventional  $\lambda$  to avoid confusion with the arrival rate in the queue.) In Stewart's simultaneous iteration method [27], the rate of convergence can be shown to be proportional to  $|\Omega_1/\Omega_{m+1}|$  when m trial vectors are used. The m trial vectors converge to

the eigenvectors corresponding to  $\alpha_1$  to  $\alpha_m$ . Thus the eigenvalues of  $P$  will give us an idea of how well the methods will converge. As previously, we define  $P$  in terms of  $l$  and  $m$ , where  $l = \lambda/\alpha$  and  $m = \mu/\alpha$  with  $\alpha > \lambda + \mu$ . Hence,  $P$  is

$$P = \begin{matrix} 1-l & l & 0 & 0 & \cdot \\ m & 1-(l+m) & l & 0 & \cdot \\ 0 & m & 1-(l+m) & l & \cdot \\ 0 & 0 & m & 1-(l+m) & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{matrix}$$

where all row sums are = 1 except the last. This is close, in some sense, to two matrices whose eigensolution is known. The first is a standard tridiagonal matrix  $T$  defined by

$$T = \begin{matrix} 1-(l+m) & l & 0 & 0 & \cdot \\ m & 1-(l+m) & l & 0 & \cdot \\ 0 & m & 1-(l+m) & l & \cdot \\ 0 & 0 & m & 1-(l+m) & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{matrix}$$

in which all row sums are 1, except the first and the last. The other matrix is the stochastic matrix,  $S$ , the probability transition matrix of an  $M/M/1/n$  queueing system, that is one in which any customers arriving when there are  $n$  customers already in the system are lost.  $S$  equals  $P$  everywhere except  $s_{nn}$ , which equals  $1-m$ , in order to preserve the stochasticity of the matrix. The following analysis of the eigenvalues of  $S$  and  $T$  can be found in Courtois [9].



Let us deal with  $T$ , first. The eigenvalues of  $T$  are, by definition, the roots of the equation

$$D_n(\Omega(T)) = \det ( T - \Omega(T).I ) = 0 \quad (3.30)$$

Expanding this determinant along its last row to give the Sturm sequence, we find the following difference equation.

$$D_i(\Omega(T)) = (1-l-m)D_{i-1}(\Omega(T)) - l m D_{i-2}(\Omega(T)) \quad (3.31)$$

with the following boundary conditions

$$D_0(\Omega(T)) = 1$$

$$D_1(\Omega(T)) = 1 - l - m - \Omega(T)$$

Solving the difference equation, we have the result

$$\Omega(T) = 1 - (l+m) - \sqrt{4lm} \cos k\pi/(n+1)$$

for  $k=1,2,\dots,n$ .

Turning to  $S$ , we first remark that it has a unit eigenvalue, since  $\underline{e}^1$  is a right eigenvector. The eigenvalues are the roots of

$$D_n^*(\Omega(S)) = \det ( S - \Omega(S).I ) = 0 \quad (3.32)$$

Using elementary properties of determinants we can evaluate  $D_n^*$  in the following manner. Add each column to the last column. Since the row sums of  $S$  are 1, the last column will now have each element equal to  $1 - \Omega(S)$ . We extract this as a common factor. Now subtract row  $i+1$  from row  $i$  for  $i=1,2,\dots,n-1$ . We add column  $i+1$  to column  $i$  for  $i=2,3,\dots,n-2$  and finally expand by the last column to find that

$$D_n^*(\Omega(S)) = [ 1 - \Omega(S) ] \cdot D_{n-1}(\Omega(S))$$

So that the eigenvalues of  $S$  are

$$1 \text{ and } 1 - (l+m) - \sqrt{4lm} \cos k\pi/n$$

for  $k=1,2,\dots,n-1$ .

Attempting to find the eigensolution of P in a similar manner, we unfortunately generate a difference equation which does not have a simple analytic solution. Since they are so similar one might hope that the eigenvalues of T and S would enable us to make deductions about the eigenvalues of P. Except for the dominant eigenvalue this appears not to be so. Simple numerical examples will demolish any reasonable hypotheses about orderings such as

$$|\Omega_i(S)| > |\Omega_i(P)|$$

$$|\Omega_i(P)| > |\Omega_i(T)|$$

Both these conjectures seem reasonable, but can be shown by direct numerical calculation of some examples to be false for all  $i \neq 1$ .

The only invariant relation appears to be

$$1 = \Omega_1(S) > \Omega_1(P) > \Omega_1(T) > 0$$

$$1 > \Omega_1 > 1 - (1+m) \sqrt{4.1.m \cos(\pi/(n+1))}$$

This relation can be deduced from the Perron-Frobenius theorem for non-negative matrices. If A is a non-negative matrix with dominant eigenvalue  $\lambda$ , which necessarily has multiplicity 1, and B is a non-negative matrix such that  $B \leq A$ , elementwise, then any eigenvalue of B,  $\beta$ , (say), will satisfy  $\beta \leq \lambda$ . Further,  $\beta = \lambda$  implies that  $B = A$ . Now,  $S > P > T$  elementwise, so the relationship given above holds between their dominant eigenvalues.

Having failed to find the rate of convergence of Stewart's method, even when applied to the M/M/1 system we now investigate if there are any simple bounds on the eigenvalues. For simplicity, we shall only consider the sub-dominant eigenvalue, since it seems unlikely that we

will be able to bound other eigenvalues with any more ease. Various authors have calculated bounds on the values of eigenvalues of matrices. Bauer's [3] are the "best" for our purposes and when applied to the sub-dominant eigenvalue  $\omega_2$  give

$$|\omega_2| \leq 0.5 \min \left\{ \max_{1 \leq j, k \leq n} \left| \frac{\sum_{i=1}^n \psi_i \left( \frac{a_{ij}}{\psi_j} - \frac{a_{ik}}{\psi_k} \right)}{\sum_{i=1}^n \psi_i} \right|, \max_{1 \leq j, k \leq n} \left| \frac{\sum_{i=1}^n \phi_i \left( \frac{a_{ji}}{\phi_j} - \frac{a_{ki}}{\phi_k} \right)}{\sum_{i=1}^n \phi_i} \right| \right\} \quad (3.33)$$

where  $a_{ij}$  is the matrix and  $\psi$  is the dominant right eigenvector and  $\phi$  is the dominant left eigenvector. Apart from the impracticality of calculating both the right and left eigenvectors corresponding to the dominant eigenvalue in order to find a bound on the rate of convergence of algorithms to calculate the dominant left eigenvector, the formula has another drawback. If there are more than 2 zero elements in each row and column (as there normally will be in our systems) this bound reduces to the dominant eigenvalue!

Lynn and Timlake [33] have developed a bound for the sub-dominant eigenvalue of non-negative matrices.

$$|\omega_2| \leq \sqrt{1 - a^\beta \frac{1 - a^{\beta-1}(1-a^n)}{1 - a^{\beta-1}(1-a)}} \quad (3.34)$$

where  $a = \min \{ a_{ij} \mid a_{ij} > 0 \}$  and

$\beta$  is the index of primitivity of the matrix, that is

$$\beta = \min \{ t \mid A^t > 0 \}$$

$$\beta \leq n-1$$

Although this bound is much easier to compute than Bauer's it is in some sense less effective. On all our test problems, which were matrices  $S$ ,  $T$

and  $P$ , as above with different values for  $\lambda, \mu$ , and  $\alpha$ , the bound returned the value 1 even for matrices whose dominant eigenvalue was less than 1. Thus we are forced to conclude that there is essentially no means of estimating the convergence rate of Stewart's method a priori.

#### 4 Practical Considerations

The theoretical methods presented in the previous chapter for finding the steady state probability distribution of a Markov process seem to fulfill our criteria of general applicability and do not need more than minimal prior analysis of the system. However, before they can be used in a general purpose computer program there are several practical problems to overcome.

First among these problems is that theoretical results in Markov processes and chains are developed in terms of matrices which have as their index sets the set of states of the system. Computers and most human beings prefer to use the positive integers as an index set when numerical calculation is required.

Secondly, there are, in general, an infinite number of states to be considered. Related to that problem is the fact that although they purport to calculate steady state probabilities, all the methods only calculate the ratios between such probabilities. In the finite case this presents little problem because of the additional condition  $\sum \pi_j = 1$ , but when the state space is infinite this property is more difficult to apply.

Thirdly, although the steady state distribution is of interest, it is more often the case that moments or marginal distributions are

required. With this in mind we may be able to find different convergence criteria for these cases.

#### 4.1 State Numbering

Theoretical developments in Markov processes and chains are often presented in terms of matrices of transition rates or transition probabilities. Entries in these matrices are indexed by pairs of elements from the state space of the Markov process. These elements of the state space can be quite arbitrary, even though we have restricted ourselves to solving discrete state space problems. In order to easily solve arbitrary processes, we need to be able to construct matrices indexed by some standard index set, from specifications which are given in terms of the state space of the original Markov process. Many processes give rise to some natural index set, and other processes often map easily into such a set, but these mappings are obviously adapted to the problem in question, and require too much prior analysis to be of general utility.

We seek a mapping from arbitrary state spaces into some standard index set. The natural choice for our standard index set is  $N$ , the set of natural numbers (positive integers). This set is beloved of FORTRAN programs and hence of general purpose numerical software. In order to be able to map efficiently from the original state space into  $N$ , it is necessary to place some restrictions on the representations of states in the original state space. We shall assume that the states can be

represented as fixed, finite dimensional vectors of integers. The actual components may be either bounded or unbounded, but they will, in general, be non-negative. For example, the possible states of an  $n$  class pre-emptive priority queueing system can be represented by a vector of non-negative integers,  $I=(i_1, i_2, \dots, i_n)$  where  $i_1$  represents the number of customers from class 1 present,  $i_2$  the number from class 2, and so on. A non pre-emptive system of  $n$  classes could be represented by  $I=(i_0, i_1, i_2, \dots, i_n)$  here  $i_0$  is an integer in the range 1 to  $n$ , representing the class of customer currently receiving service, with some arbitrary value when the system is empty. We know of no discrete Markov processes of practical significance whose states cannot be represented in this manner.

We now seek a mapping between  $S$ , as restricted above, and  $N$ , the natural numbers that should possess the following properties. First, it must be easy to calculate, since each non-zero transition rate or probability that we add to the matrix will involve the calculation of two indices. Secondly, it must have an easily calculable inverse mapping. There is no point in being able to calculate the steady state probabilities in terms of  $N$  if we cannot interpret this steady state in terms of  $S$ . More graphically, finding that state 5 has probability 0.999 is of no use unless we can show that state 5 corresponds to  $(1,3,2)$  (say) in our index set  $S$ .

Thirdly, the mapping should be surjective (onto). It should map our index set into  $N$  without leaving any gaps. States with no transitions might cause difficulties to general purpose numerical subroutines.

Finally, and of lesser importance, it should map "close", in some sense, states into "close" integers. This property is desirable for heuristic reasons, since it might tend to keep the matrix in a banded form.

If the state space is finite then the problem essentially disappears. There are several mappings developed by computer scientists in conjunction with array subscripting which fulfill our purpose admirably. At worst, a table could be maintained giving every state in both representations, although maintenance and searching of this table could become a major problem if the state space was large. This tabular method of state transformation was used by Stewart in the MARCA package [45].

If the state space is infinite other approaches are needed. Clearly, the list or table method could be used, perhaps with some sort of hashing function, but a more general method is called for. Let us, for the time being, restrict the problem to that of finding a mapping in the case where  $S = Z^{+n}$ , that is, all of the components of  $i = (i_1, \dots, i_n) \in S$  are non-negative integers, with no upper bound. The case in which some of the components are bounded will be dealt with later.

Let us further restrict ourselves and take as an example the mapping from  $Z^{+2} \rightarrow Z^+$ . This has the advantage that we can draw the states indexed by  $Z^{+2}$  in a conventional manner and perhaps gain some insight from the diagram.



. . . .  
 0,3 1,3 2,3 3,3

. . . .  
 0,2 1,2 2,2 3,2

. . . .  
 0,1 1,1 2,1 3,1

. . . .  
 0,0 1,0 2,0 3,0

The problem is essentially to label the points in the above figure.

There are two obvious number schemes.

The first is to number the states as below

. . . .  
 8 7 6 11

. . . .  
 3 2 5 10

. . . .  
 0 1 4 9

That is starting at (0,0) as state 0 and travelling round the perimeters of successively larger squares. This mapping can be easily calculated by

$(i,j) \rightarrow$  if  $i \geq j$  then  $i^2+j$   
else  $j^2+2j-i$  fi

and with slightly more trouble inverted by

$t \rightarrow (i,j)$

where  $s := \lfloor \sqrt{t} \rfloor$

$k := t - s^2$

if  $k \leq i$  then  $i := s; j := k$

else  $i := 2s - k; j := s$  fi

The obvious alternative method is to number the states along the "diagonals".

.	.	.	.
5	8	12	
.	.	.	.
2	4	7	11
.	.	.	.
0	1	3	6

This is calculated as

$(i,j) \rightarrow (i+j)(i+j+1)/2$

The inverse is given by

$t \rightarrow (i, j)$

using the following algorithm.

```

k := max {p | p(p+1)/2 ≤ t}
j := t - k(k+1)/2
i := k - j

```

Turning to the more general  $n$ -tuple case we can attempt to generalise the algorithms. The first "box" mapping could be extended analogously to number states over the boundaries of successively nested hypercubes, but it becomes increasingly hard to visualise and calculate. The second mapping, along the "diagonals", can however be fairly simply generalised to operate on  $n$ -tuples. The key observation is that the term  $(i+j)(i+j+1)/2$  counts the number of lattice points inside and on the boundary of the triangular area defined by  $x \geq 0$ ,  $y \geq 0$ , and  $x+y < i+j$ . The second term represents the number of integer lattice points on the line  $x+y=i+j$  such that  $y < j$ . Note that this second term is totally independent of one component of the original state description.

Extending this numbering scheme to 3-dimensional space is relatively easily visualised. First, we count the number of integer lattice points contained within the tetrahedron  $x \geq 0$ ,  $y \geq 0$ ,  $z \geq 0$ , and  $x+y+z < i+j+k$ , and then add the two dimensional value given by two of the components. That is,

$$(i, j, k) \rightarrow \alpha(i+j+k) + (j+k)(j+k+1)/2 + k$$

where  $\alpha(t)$  is the number of lattice points in the tetrahedron  $x+y+z < t$ .

To find  $\alpha(t)$  we note that the number of lattice points is just the sum of the number of lattice points lying on each of the planes  $x+y+z=i$

for  $i=0,1,2,\dots,t$ . The number of points on the plane  $x+y+z=i$  is just  $i(i+1)/2$  since the plane is triangular in shape.

$$\text{Thus } \alpha(t) = \sum_{i=1}^t i(i+1)/2 = t(t+1)(t+2)/6$$

Hence the mapping is

$$(i,j,k) \rightarrow (i+j+k)(i+j+k+1)(i+j+k+2)/6 \\ + (j+k)(j+k+1)/2 + k$$

Let us proceed at once to the case of  $k$ -dimensional spaces. We again count the integer lattice points contained in the simplex defined by the origin, and the points  $\underline{x}$ .  $\underline{x} = \{\text{cons. } \underline{e} \mid \underline{e} \text{ is the unit vector and } \text{cons} = \sum i \text{ where } I = (i_1, i_2, \dots, i_k) \text{ is the state we are trying to map.}\}$

The full mapping is

$$(i_1, i_2, \dots, i_k) \rightarrow \frac{\sum_{n=1}^k \prod_{j=0}^{n-1} (\sum_{r=1}^n (i_r) + j)}{n!}$$

Each term of the form

$$\sum_{r=1}^n i_r + j$$

represents the number of integer lattice points inside or on the boundary of the simplex defined by the points  $\underline{x}$  in  $R^n$ . This is because the number of lattice points in the body  $\sum x_r < p$  is just the sum of the number of lattice points lying on the planes

$$\sum_{r=1}^n x_r = t \text{ for } t=1,2,\dots,p.$$

We must prove that

$$\sum_{t=1}^p \prod_{j=0}^{n-1} (t+j) = \prod_{j=0}^n (p+j)$$

This is clearly so for  $n=1$  since the equation reduces to

$$\sum_{t=1}^p t = p(p+1)/2 \quad \text{a well known result!}$$

For arbitrary  $n$  the result is certainly true for  $p=1$  since

$$\prod_{j=0}^{n-1} (j+1) = n!/n! = 1$$

and

$$\prod_{j=0}^n (j+1) = \frac{(n+1)!}{(n+1)!} = 1$$

Assume the formula is true for some particular  $p=s$  (say), that is

$$\sum_{t=1}^s \prod_{j=0}^{n-1} (t+j) = \prod_{j=0}^n (s+j)$$

then

$$\begin{aligned} \sum_{t=1}^{s+1} \prod_{j=0}^{n-1} (t+j) &= \prod_{j=0}^n (s+j) + \prod_{j=1}^n (s+j) \\ &= \prod_{j=1}^n (s+j) \times (s/(n+1)+1) \end{aligned}$$

$$= \prod_{j=1}^n \frac{(s+j)(s+n+1)}{(n+1)!}$$

$$= \prod_{j=0}^n \frac{(s+j+1)}{(n+1)!}$$

Hence the formula holds for  $p=s+1$ , but it holds for  $p=1$  and hence for all finite  $p$ .

The inverse mapping from some integer  $z$  to the  $k$ -tuple  $I=(i_1, i_2, \dots, i_k)$  can be stated algorithmically as

```

x := z ;
for n:=k step -1 until 2 do
  begin
    q := max {i |  $\prod (i+j)/n!$  ≤ x} ;
    w(n):=q;
    x:= x -  $\prod(q+j)/n!$  ;
  end;
  i(1):=w(1);
  for n:=2 until k do i(n):=w(n)-w(n-1):

```

This algorithm can easily be implemented. The only difficulty is that of calculating  $q$  efficiently. The naive approach, searching the integers  $0, 1, 2, \dots$  etc. leads to a search of length  $q$ , where each step involves 1 multiplication and 1 division (of integers). The calculation of the initial value of the product term  $\prod(q+j)$  comes essentially free, since if  $q=0$  it is also zero, and if  $q=1$  it equals  $n!$  which will be have been calculated anyway.

A little thought will lead us to a much shorter search, since

$$\begin{aligned} \prod_{j=0}^{n-1} (q+j+1) &> x \\ \frac{\prod_{j=0}^{n-1} (q+j+1)}{n!} &> x \\ \Rightarrow (q+n)^n &> x \cdot n! \\ \Rightarrow q+n &> \sqrt[n]{x \cdot n!} \\ \Rightarrow q &> \sqrt[n]{x \cdot n!} - n \\ \Rightarrow q &\geq \lfloor \sqrt[n]{x \cdot n!} - n + 1 \rfloor \end{aligned}$$

However  $\frac{q^n}{n!} < x$

$$\Rightarrow q \leq \sqrt[n]{x \cdot n!}$$

Hence starting our search at  $q = \lfloor \sqrt[n]{x \cdot n!} - n + 1 \rfloor$  will result in a search of length (at most)  $n$ , again taking 1 multiplication and division for each iteration. However, in this case, we also have to calculate an initial value for the product term  $\prod_{j=0}^{n-1} (q+j+1)$  which involves a further  $n$  multiplications, and we need to calculate an  $n$ -th root. In practice, these additional calculations appear to balance out the theoretically longer search involved in the naive algorithm.

This mapping performs a transformation, and its inverse, between  $\mathbb{Z}^{+n}$  and the non-negative integers. It is simple to define the mapping from the non-negative integers to the positive integers as the action of adding 1. This is easily inverted too!

Returning to the more general problem, in which our state space  $S$  can be represented by  $k$ -tuples of integers, some of which are bounded, and the remainder are unbounded non-negative integers. Without loss of generality, we can assume that  $i_1$  to  $i_m$  are bounded and  $i_{m+1}$  to  $i_k$  are

unbounded. That is ,

$$S = \{(i_1, \dots, i_k) \mid i_j \in Z^+ \text{ and } i_1 \leq L_1 \text{ and } i_2 \leq L_2 \text{ and } \dots \text{ and } i_m \leq L_m\}$$

Let us lump together all states in the above representation which have the same values for  $i_{m+1}$  to  $i_k$ . To this lumped representation, we can apply the transformation developed above. But each state in the lumped representation corresponds to

$$e = \prod(L_j + 1)$$

states in the original model. We can easily develop a transformation  $A$  (say) which will uniquely map  $(i_1, \dots, i_m) \rightarrow a$  where  $0 \leq a < e$ . An array mapping function will do. If the transformation developed above is denoted by  $\Delta$ , then the full transformation from  $S \rightarrow Z^{+n}$  is given by

$$(i_1, \dots, i_k) \rightarrow A(i_1, \dots, i_m) + e \cdot \Delta(i_{m+1}, \dots, i_k)$$

This transformation can be inverted by dividing the state's representation by  $e$ . The integer part of the quotient can then be used to find the unbounded part of the state representation, and the remainder to find the bounded part.

This numbering scheme also has a heuristic advantage. When we calculate marginal distributions and conditional probabilities, we are more likely to be interested in the boundary conditions than in arbitrary states. For example, in an  $n$  class priority system, we are much more likely to want to calculate the marginal distribution of class  $i$  customers given that class  $j$  is empty than the marginal distribution given that class  $j$  has 6 customers. The numbering scheme that we have developed concentrates on states with low indices and thus pays more attention to the boundaries.



This representation of  $S$ , and its mapping into  $N$ , produces a problem of its own. As we increase the number of states being considered from  $n_1$  to  $n_2$  (say), we have to consider all the states which correspond to those integers. It is possible that not all states so generated will correspond to states which the system can possibly enter. For example, in a 2 class non-preemptive priority system, the states can be represented by triples of integers,  $(i_0, i_1, i_2)$ , where  $i_0$  represents the class of the customer in service and  $i_k$  represents the number of customers in class  $k$ . Thus  $(2,4,3)$  represents the system with 4 class 1 customers, and 3 class 2 customers, one of whom is being served. In this representation,  $(1,0,2)$  corresponds to 7 in  $N$ , and as such will be generated although the system being modelled could never be in that state, since the class 2 customers would be served. Similarly the state  $(2,5,0)$  will be generated, but would never be entered by the system. We shall classify states as valid or invalid, according to whether or not the system being modelled can ever enter them. We must have a procedure for dealing with invalid states as they arise. The obvious solution would be never to generate them, or to give such states no transitions at all. In order not to generate invalid states, we would need a numbering scheme which was specific to a particular problem. Allowing invalid states to be generated but giving them no transitions is also impractical, since zero rows in the  $Q$  matrix would cause problems for general purpose numerical software. We have considerable freedom in constructing  $Q$  and provided that we do not interfere with the relationship of the valid states to each other, we can do almost anything with the transitions among invalid states. The solution that we

have adopted is to give the invalid states transitions which ensure that they are transient states in the Markov process defined by  $Q$ . The valid states correspond to the recurrent states in the process; these are the only states with non-zero stationary probabilities.

This is easily done by ensuring that no valid state makes a transition to an invalid state. The invalid states can make transitions to each other or to any valid states. Since we construct  $Q$  by rows in the natural order of  $N$ , the first invalid state must make a transition to a valid state. This is because we do not, in general, know the index of the next invalid state. In fact, to ensure the convergence of the algorithms, we must have at least one transition from an invalid state to a valid one, so this is no hardship. Including such a transition ensures that the invalid states are transient, since with probability 1 the system will enter a valid state, and thereafter it can never enter an invalid state.

There are three essentially different ways of defining the transitions out of an invalid state. Since we wish to keep the number of non-zeroes in  $Q$  to a minimum, it seems sensible to restrict an invalid state to a single transition, and as pointed out above it should be to an earlier state in  $N$ . This ensures that all rows of  $Q$  corresponding to invalid states have only two non-zeroes. If transitions were allowed to higher numbered states, there would be no guarantee that the state to which a transition was made from a particular invalid state would be part of a truncation of  $Q$  which contained that invalid state. This could give problems to Brandwajn's method.

The first possible construction of  $Q$  is to give each invalid state a transition to the previous invalid state, and the first invalid state a transition to some arbitrary state, say 1. In this scheme, all the invalid states form a single inessential class of states and are transient. A second option is to give each invalid state a single transition to a single arbitrary, but valid, state, say 1. This choice makes each invalid state an inessential class with a single member. A third choice for the transitions out of invalid states is to give each state a single transition to the previous valid state. Once again each invalid state forms its own, individual inessential class. As well as choosing the states to which an invalid state may make transitions, we may also choose the rate at which transitions are made from invalid states.

The bounds given by Tweedie's result hold so long as the two states whose ratio of probabilities is to be estimated belong to the same essential class of states, and there is at most a single finite essential class. The only effect that a different choice of construction for  $Q$  will have is to possibly alter the sparsity structure of  $Q$  and hence, the amount of fill-in generated by calculation of  $Q^{-1}$ . The rate at which transitions are made out of invalid states will have a much smaller effect than the actual position of the non-zeroes.

Heuristically, it seems likely that the third option, a transition to the previous valid state, will be best since it will keep the non-zeroes close to the diagonal of  $Q$ , and minimise the fill-in. The option of linking all the invalid states into a single inessential class is only marginally worse as far as Tweedie's method is concerned, because it

seems likely that the previous invalid state could be an arbitrary distance from the diagonal. The previous valid state is an arbitrary distance from the diagonal too, but it seems reasonable that there will be many more valid than invalid states. The choice of a single state as the target for all the transitions from invalid states will tend to make the corresponding column of  $Q$  have a rather high proportion of non-zeroes and consequently create excessive fill-in. The values of  $Q^{-1}$  corresponding to valid states are not affected by these choices, and experiment confirms the heuristic reasoning above which prefers the third option.

If Stewart's method is to be used, either of the choices which give rise to many small inessential classes will be equivalent as far as convergence of the probabilities of the invalid states to zero is concerned. Consider an invalid state,  $s$ , say. It makes a single transition to a valid state, and there are no transitions which lead into  $s$ . If the current estimate of  $\pi_s$  is  $\beta$  then the effect of post-multiplication by  $P$  is to make the new estimate  $\beta p_{ss}$ . Thus the estimate converges to zero geometrically. This suggests that we should choose the rate of transition out of  $s$ , and any other invalid states, to be as large as possible. This will make  $p_{ss}$  small and give fast convergence to zero for the probabilities of invalid states. If the transitions out of invalid states are all into a particular valid state, then the error by not having the probabilities of invalid states identically zero will be concentrated in that state. Although the effect of these transient states dies away very quickly, it seems sensible to share the error among as many states as possible. If the invalid states

form a single inessential class, then the rate at which  $\pi_s$  tends to zero will be modified. Assume that  $t$  is the index of the next invalid state and hence  $p_{ts}$  is non-zero. Clearly,  $\pi_s(n+1) = \pi_s(n)p_{ss} + \pi_t(n)p_{ts}$ . There will be a 'last' invalid state, at least when we consider a finite truncation, which will converge to zero geometrically. Thus all the invalid states will tend to zero, but possibly more slowly than if many inessential classes are used.

The ultimate convergence of Brandwajn's method is also not affected by the choice of any of these options. We shall only consider the case where invalid states form individual inessential classes of a single state. That is, either of the second or third options described above is used. In this case, an invalid state,  $s$  (say), makes a single transition to a state  $t$ ;  $t < s$ . The iterations of Brandwajn's method which affect state  $s$  have the form

$$\pi_s(i+1) = \pi_s(i) (1 - \omega q_{st})$$

Recall that there is only a single transition out of  $s$ , and no transitions with  $s$  as destination. That implies that column  $s$  of  $Q$  is zero, except for  $q_{ss}$ . Whatever the value of  $\pi_s(0)$ , it will converge to zero, geometrically fast, if  $\omega < 1/q_{st}$ . The condition for the convergence of the method under normal conditions is  $\omega < 1/\max\{\sum q_{ij}\}$  and since  $\max\{\sum q_{ij}\} \geq q_{st}$ , the inclusion of invalid states imposes no extra constraints on  $\omega$ . The first option, of linking all the invalid states into a single inessential class, can also be shown to not affect the ultimate convergence of the method. As with Stewart's method, the choice between these options must be made on the grounds of their affect on the speed of convergence.

Since we can test the validity of a state, we could force the first estimate of  $\underline{\pi}$  to have  $\pi_s = 0$  for all invalid states,  $s$ . This is not done for two reasons. The testing of the validity of states might be an arbitrarily complex operation and hence we should only perform such checking when it is unavoidable. Even if such checking were cheap, there is no guarantee that Stewart's method will not introduce non-zero elements into  $\underline{\pi}$  at the invalid states. The analysis above shows that the ordinary post-multiplication by  $P$  will not, but the interaction analysis phase includes sub-dominant eigenvectors for which the elements corresponding to invalid states need not be identically zero.

## 4.2 Denumerable State Spaces

Theoretically, any method for calculating the steady state probability vector of a denumerably infinite state space Markov process must involve an infinite number of probabilities (or their ratios), in practice this is not such a great problem. Since  $\sum \pi_i = 1$ , and  $\pi_i \geq 0$  for all  $i$ , even the ratios of probabilities must have a finite sum. This means that all but a finite number of these probabilities will be less than some arbitrary, but positive,  $x$ . For example, all but the first  $n$  states of an M/M/1 system have probabilities which are less than  $\rho^n(1-\rho)$ , where  $\rho$  is the traffic intensity. Whilst not all systems have this conveniently regular behaviour, it is none the less true that however we choose to number the states,  $\pi_i \rightarrow 0$  as  $i \rightarrow \infty$ . Hence trivially, there exists an integer  $k(x)$  such that  $\pi_i < x$ , for all  $i > k(x)$ . Computers represent their numbers finitely, so there is a smallest

non-zero representable number. On the IBM 360/370 series computers, this number is equal to  $5^{-79}$ . The existence of a smallest representable number implies that all but a finite number of states will have zero probability as far as the computer is concerned. If  $\rho$ , the traffic intensity, is 0.99 for an M/M/1 system, then only 18,000 (!) states have non zero machine representations. Although ideally we would like to consider all states with non-zero machine probability, the foregoing argument demonstrates the impracticality of such a course.

A more important parameter of computers as far as numerical methods are concerned, is  $\epsilon$ , machine epsilon.  $\epsilon$  is the smallest positive number such that  $1+\epsilon > 1$ . On IBM 360/370 computers its value is about  $2^{-16}$  (for double precision real numbers). When we are calculating marginal distributions, we will have to sum the probabilities which belong to the subset of the state space whose marginal probability we are trying to calculate. If we find probabilities that are less than  $\epsilon \cdot p$ , where  $p$  is the largest probability in the subset, then we can ignore them. Although in a general Markov process,  $\pi_i \rightarrow 0$  as  $i \rightarrow \infty$ , we have no means of knowing that the smallness of  $\pi_k$  in any way implies the smallness of  $\pi_{k+1}$ . This problem has no general solution, but it is reasonable to assume that those states which have the highest probabilities will correspond to relatively small indices, whatever our state numbering.

The problem that all methods suffer from, namely that they calculate not probabilities as such, but the ratios between probabilities, can be approached in two ways. Since we know that  $\sum \pi_i = 1$  (by definition), and we assume that we have included all the important

states (ones with large probability) in our state space, we can merely sum the 'probabilities' that we calculated, and divide through to normalise them. An alternative approach is possible for a fairly large class of systems. In many cases, we can deduce the true probability of some state from other considerations. For example, the probability of an empty system for any single server queueing system, is equal to  $1-\rho$ , where  $\rho$  is the traffic intensity. This is easily proved using Little's Theorem. Knowing the true probability of some state, and having calculated its ratio to all the other probabilities in the system, we can easily calculate all the true probabilities.

The problem remains, however, at what state should we truncate the system in order to get "good", in some measurable sense, approximations. If we are interested in the probability of a particular state and are using Tweedie's method then we will get upper and lower bounds on our approximation. Seneta et al. [1] prove that inversion of  $Q$  by Gaussian elimination is a well-conditioned problem. Reid [39] has shown that, even when pivots are chosen to maintain sparsity, Wilkinson's error analysis can be used, and the perturbations in the original matrix satisfy

$$|e_{ij}| \leq (3.01) \epsilon M m_{ij} \quad (4.1)$$

where  $M$  is the maximum value of any element at any stage of the elimination and  $m_{ij}$  is the number of multiplications performed on the  $ij$  element. Thus we can calculate the bounds for a particular size of truncation, and repeat the calculation for larger and larger truncations until they are acceptable. We can estimate  $m_{ij}$  in (4.1) using Duff's results. As remarked before, this will presumably be an overestimate,



since we choose pivots to minimise fillin, and hence the number of operations to be performed, whereas Duff assumes that the diagonal elements are used as pivots. Even so, the bound on the errors given above is very generous. The sparse matrix routines record the growth of errors in the course of the elimination as a maximum possible relative perturbation of the elements of A. That is, they estimate  $\max\{|e_{ij}/a_{ij}|\}$ . From the above, we would expect this to be of order  $(3.01)\epsilon \cdot \max\{m_{ij}\}$ . In practice it is usually about 2 or 3 times machine epsilon.

When using the other methods, the position is less clear because of their iterative nature. The method used to ensure that a large enough truncation is being used is rather ad hoc. A size for truncation is chosen, and the stationary probability found for this size. A larger truncation size is then generated, and the stationary distribution found again. This procedure is repeated until the stationary distribution calculated at successive truncation sizes is the same, to within our error limits. An alternative termination criterion was suggested by Seneta, namely when the dominant eigenvalue of P was close enough to 1. By the Perron-Frobenius theorem, it will never equal 1 exactly, but in practice it becomes almost equal to 1 for very small sizes of truncation.

### 4.3 Moments and Marginal Distributions

If moments or marginal distributions are being sought, then Tweedie's method does not directly help us. We could, obviously, demand that all probabilities making up the distribution were approximated to the appropriate accuracy. This however seems unnecessarily harsh, and is probably unobtainable in practice. (If we truncate at state  $n$ , we will never have the probability of state  $n$  very accurately.) Since we are working on a computer we only have finite accuracy, and we can treat as zero all probabilities which are less than  $\epsilon \cdot p$ , where  $p$  is the largest probability in our marginal distribution. This is not strictly true, since if we add numbers in order of increasing magnitude we minimise this truncation error. To find a marginal distribution we find the probabilities of the constituent states until a sufficient number of them are less than  $x \cdot p$  where  $p$  is the largest constituent probability, and  $x$  is an accuracy factor we choose.  $\epsilon$  can be chosen, but in practice reasonable estimates can be obtained with much larger values of  $x$ .

When finding moments the same considerations apply, except one. We could stop on finding one state with a small enough probability, but that lays us open to errors of the following kind. Consider, for example, the system consisting of two totally independent M/M/1 queues. Let the state  $(i,j)$  represent  $i$  customers in the first queue and  $j$  in the second. If the traffic intensities are say 0.2 and 0.9 respectively, then the probability of state  $(i,j)$  is  $0.08 \cdot (0.2)^i \cdot (0.9)^j$ . Thus it is possible for the state  $(m,0)$  to have a very small probability while the state  $(0,m)$  still has a sizeable one. If we merely waited until some

arbitrary state had reached the required smallness we might choose a state such as  $(m,0)$  while there is still a substantial contribution to be made by states of the form  $(0,m)$ . The heuristic solution adopted is to consider all states on the diagonal  $(i,j)$  such that  $i+j=m$  and only to cease expansion when all the states on such a diagonal satisfy our convergence criterion. The convergence criterion is satisfied by state  $(i,j)$  when its probability is  $p$ , if

$$i \cdot p < a \cdot P \quad \text{and} \quad j \cdot p < a \cdot P$$

where  $a$  is the requested accuracy and  $P$  is the probability of the state in the marginal distribution whose probability is largest. Similar heuristics can be applied in higher dimensional state spaces.

We could also go on adding states to the marginal distribution so long as this adding procedure is having some effect on the marginal distribution or the moments. This is essentially a heuristic version of the above. We could adopt the same approach to estimating single state probabilities as well, continuing to expand the truncation so long as the sum of the "probabilities" was changing. In practice, these heuristic variants appear not to be as reliable as the criteria proposed in the previous paragraph.

#### 4.4 Program structure

This section gives an outline of the structure of the program used to compare and evaluate the different methods. If individual programs had been written to perform each algorithm, there would have been a

large amount of common coding. For example, the part of the coding which manipulates  $Q$  and sets up truncations would be common to all the programs. Rather than have the problems of dealing with several versions of the same piece of program, a single program was written which called subprograms to perform the different algorithms. An outline of the structure is shown in Figure 4.1. This approach has both advantages and disadvantages. The main disadvantage is that the program is more complex than would be necessary for a single algorithm. An advantage, apart from the simpler maintenance mentioned above, is that one can change algorithms easily. For example, if one is using the direct method, and the size of truncation becomes too large for the Gaussian elimination subroutines, the program will use Stewart's method which needs less working storage. The estimate for  $\pi$  which has already been found by Tweedie's method can be used as the first approximation to start off the iterations.

Since the three methods use different amounts of core storage in different manners, a language which supports dynamic allocation of storage is almost essential. The main part of the program, is written in Simula 67, although almost any language of the Algol60 family could have been used. For most of the numerical methods subroutines, for example the QR algorithm which is needed by Stewart's method, the N.A.G. library was used. A few similar routines, for example, the Erisman/Tinney algorithm were coded by the author in Fortran. In order to make use of the program the representation of states as vectors of integers must be decided upon. Also one must decide on which states are valid and which invalid. A short piece of initialising code and two subroutines are then

```

do forever
  read request;
  work out truncation size;
  while request not satisfied do
    expand Q by rows, until all rows
      in the truncation are present ;
    set up truncation of Q
    if Stewarts then convert Q to P (truncated form);
    if Stewarts or Brandwajns then
      read old estimate for  $\pi$  (if any) ;
    case method of
      Tweedie's;
      Stewart's;
      Brandwain's;
    -----
    test  $\pi$  for satisfaction of request;
    if not satisfactory then increase truncation size;
  -----
  write out results ;
  -----

```

Figure 4.1.

written. The initialisation program must set up various parameters for the main solution program, such as the number of integers in a state representation, the number of any bounded components in the state representation, and their bounds. The two subroutines are used to construct Q. As it expands its representation of Q, the main program, for each state to be added calls one of the subroutines to check whether or not the state is valid. The invalid states are automatically linked to the previous valid state, as described above. For valid states, the other subroutine is called to return all the states to which transitions can be made from this state and the rates at which these transitions occur. The subroutines work on the external numbering of states as vectors of integers, and need have no knowledge of the internal state numbering scheme. This is equivalent to constructing a row of Q. It is conceptually slightly simpler to construct Q in this fashion.

Constructing it by columns would require the subroutine to return the states that could have made transitions into this state. Although theoretically equivalent, it would be much harder to deal with invalid states using such a construction. Another alternative would be for the user to provide a subroutine which was given two states as parameters, and returned the transition rate between them. While this is perhaps conceptually more elegant than the method adopted, it would be in practice very inefficient. As the truncation was increased in size from state  $n$  to state  $n+1$  (say), the subroutine would need to be called  $2n+1$  times and most of the calls would return the transition rate as 0.

Since the Gaussian elimination routines destroy the representation of the matrix that they factorise, the representation of  $Q$  is held on an indexed sequential file. Each record of the file contains the non-zero elements of a column of  $Q$ . Although this involves a large number of indexed operations on the file as  $Q$  is expanded, when a truncation is set up only the first  $n$  records need to be read, sequentially. If the records of the file contained rows, then the program would need to check that only those elements within the current truncation were included. Another advantage of using a file to save  $Q$  is that if the model is to be examined again, the representation can be re-used, without having to be re-generated. A similar advantage is achieved by saving  $\underline{\pi}$  on a file. If a closely related system is to be investigated, then the solution to the original system can be used as a first estimate for the iterative methods of solution.

The program will deal with requests for three different types of

calculation. We shall illustrate them using a two class preemptive priority M/M/1 system. Requests can then be in

- 1)  $(i,j)$  is a request to calculate the probability of  $i$  customers in the first queue and  $j$  in the second. Depending on the method being used, the program will continue expanding the truncation size in use until either the bounds given by Tweedie's results are closer than the error limit being used, or until the values of the probability at successive truncation sizes differs by less than the error limit. In both cases, the error is treated as a relative error. That is, the difference between the upper and lower bound divided by the lower bound must be less than the error limit.
- 2)  $(*,*)$  causes the program to calculate the mean number of customers in each queue. Optionally, higher moments of the queue lengths can be calculated.
- 3)  $(*,j)$  will calculate the marginal probability that there are  $j$  customers in the second queue. The mean length of the first queue, conditional on there being  $j$  class 2 customers is also calculated.

The initial truncation size is calculated by adding 2 to each component of the state description. Thus, if the probability of state  $(2,3)$  is requested, the state space is truncated at  $(4,5)$  initially, and the system solved. If the 'answer' is not satisfactory, the size of the truncation is increased by 25%, and the system resolved. Note that the method used to test the iterative methods for convergence implies that we will always have to increase the truncation size at least once. When

moments or marginal probabilities are requested, the initial truncation is chosen by setting any variable subscripts in the state description equal to 2. The initial truncation when the request is  $(*,6)$  is thus at state  $(2,6)$ .



## 5 Examples

In order to evaluate the various methods, and compare their effectiveness, several systems with known steady state distributions were solved using all the algorithms.

The obvious problem to try and solve is the M/M/1 system. It is well understood and has a particularly simple structure. Other related systems which fall into the class of simple birth-death processes are the M/M/k system, a single queue with Poisson arrivals and exponential service requests, but with k servers; the M/1/∞ system, in which each arrival receives service immediately; and the discouraged arrival M/M/1 system, in which arriving customers enter the system with a probability that depends on the length of the queue. Since these are all simple birth-death processes, from our analysis in Chapter 3, their solutions will be given exactly by the upper bound in Tweedie's method. To test the program and the algorithms on systems with more complex state spaces but which also have known solutions, artificial examples can be constructed from simple birth-death processes in parallel. For example, two parallel M/M/1 queues can be described by a pair of integers (i,j), representing the state of i customers in the first queue and j in the second. If the queues are totally independent, then customers will arrive in queue k at rate  $\lambda_k$  and the server will satisfy their requests at rate  $\mu_k$ . In this case, the probability of a particular state (i,j) is

given by  $\rho_1^i (1-\rho_1) \times \rho_2^j (1-\rho_2)$ .

Turning to more complicated systems, which may or may not have closed form solutions the program was tested against many published numerical results. The  $M/E_k/1$  queue, the system with Poisson arrivals and Erlang  $k$  service requests to a single server, has a known mean [40, P 166]. Grassman has investigated it numerically [21]. The system consisting of two parallel queues, in which arrivals join the shorter queue, has an analytic solution if the servers at the head of each queue serve at the same rate. If they serve at different rates, then the only results available are due to Grassman [22]. He has used a numerical method to find the transient solutions to a finite state space version of this problem. A system which models a network of unreliable computers was also modeled. Theoretical results are known for a special case of this model [34]. All these systems have been solved using the algorithms above, and the results agree closely.

In this chapter, we shall present comparisons between the three algorithm's performance on some systems with known solutions, and also find numerical solutions for two systems which have no known analytical solution. All comparisons were run on an IBM 370/168 and timings are given in seconds of CPU time. For all tests the programs error limit was set to  $5^{-3}$ . The non-preemptive priority  $M/M/1$  system has known means, but the distribution of the numbers of customers in each priority class is unknown. The other unsolved system that we shall study is supposed to model a system of multiple micro-processors. Arriving jobs belong to various classes and the processors are individually dedicated to jobs of

a particular class.

## 5.1 M/M/1 Systems

We first compare the performance of the methods on solving the M/M/1 system. This is well understood and has a very simple structure. We are even able to apply the truncation methods of Tweedie symbolically. We were unable to estimate the rate of convergence of Stewart's algorithm very accurately. Table 5.1 shows the results of finding the probability that 10 customers are present in an M/M/1 system with traffic intensity,  $\rho=0.3$ . The true probability is  $4.133343 \times 10^{-6}$

Table 5.1.

Method	Truncation	Value	Time
Tweedie	15	$4.1234 \times 10^{-6}$	1.0
Stewart	20	$4.1334 \times 10^{-6}$	1.8
Brandwajn	20	$4.1323 \times 10^{-6}$	1.1
Stewart(r)	27	$4.1390 \times 10^{-6}$	3.5
Brandwajn(r)	20	$4.1360 \times 10^{-6}$	1.3

The allowed error was  $5 \times 10^{-3}$ . It will be seen that all the methods gave the correct answer to within 0.5%. The direct method of Tweedie used a truncation of only 15 states, whereas both Stewart's and Brandwajn's methods needed 20 states. The iterative methods labelled (r) were initialised with a random vector, instead of the unit vector. This did not improve the accuracy, or even the rate of convergence.

The program can also be used to find moments and conditional probabilities. The results of finding the mean number of customers present by all three methods are given in table 5.2.

Table 5.2.

Method	Truncation	Value	Time
Tweedie	20	0.42857	2.4
Stewart	20	0.42857	2.5
Brandwajn	20	0.42857	1.7
Stewart(r)	27	0.42857	4.3
Brandwajn(r)	20	0.42857	1.8

Since the true result is 0.42857 all methods reached the correct answer. No further expansion of the truncation was necessary for any of the methods, except Tweedie's.

If the system is arbitrarily truncated at 100 states all the methods find the correct probability of there being 10 customers in the system. The times taken are given in Table 5.3. The different times taken by the iterative methods are reflections of the time that they take to find a stationary distribution, starting with the 'correct' answer found previously. In the rows marked (r), the extra elements in the first estimate were given random values. The other rows represent the time taken when these extra elements were initialised to 0.

Table 5.3.

Method	Time
Tweedie	2.8
Stewart	4.6
Brandwajn	3.9
Stewart(r)	14.8
Brandwajn(r)	7.5

The following tables ( 5.4 and 5.5) give the same results for an M/M/1 system with traffic intensity,  $\rho=0.9$  . The probability that there are 10 customers is given by  $3.4867 \cdot 10^{-2}$ , and the mean number of customers is 9.

Table 5.4.

Method	Truncation	Value	Time
Tweedie	74	$3.48544 \cdot 10^{-2}$	5.4
Stewart	74	$3.48544 \cdot 10^{-2}$	13.8
Brandwajn	94	$3.47951 \cdot 10^{-2}$	11.4
Stewart(r)	74	$3.48547 \cdot 10^{-2}$	14.4
Brandwajn(r)	150	$3.75539 \cdot 10^{-2}$	34.1

All the methods, except Brandwajn(r), give answers which are well inside the error bound asked of the program. Tweedie's method is significantly faster than Stewart's method, and gives results which are equally accurate. The only poor performance comes from Brandwajn's method when a random vector is used to initialise the estimate. The 'answer' returned has an error of about 7%. The time taken to find this 'answer' is also extremely long.

Table 5.5.

Method	Truncation	Value	Time
Tweedie	119	8.998	11.6
Stewart	94	8.889	16.9
Brandwajn	119	8.937	16.2
Stewart(r)	150	9.000	83.3
Brandwain(r)	>700	*****	>200

Once again, Tweedie's method wins. It not only achieves an accurate result, but it also does it faster than the other methods. This despite taking a larger truncation than Stewart's method. Brandwajn's method, using a random first estimate, had not returned an answer after 200 seconds of CPU time had been used, and had expanded the truncation to over 700 states. It can be seen that using a random vector, instead of the unit vector, as first approximation has no particular advantage in terms of the accuracy achieved and a definite disadvantage in terms of the time taken to reach the solution.

On the basis of these results, we might conclude that Tweedie's method was the only one which reasonably approximates the correct solution in a reasonable time. However, we have already remarked on the simple structure of the M/M/1 system, especially favourable to sparse Gaussian elimination. We shall also compare the methods on the non-preemptive priority system in the following section.

## 5.2 Non-preemptive Priority System

In a non-preemptive priority M/M/1 system, the customers belong to various priority classes. They arrive in a Poisson stream and join the rear of a queue corresponding to their priority class. When the single server finishes the service of a customer, the first customer in the highest priority non-empty queue is chosen for service.

Any state of the system can be described by a vector of  $n+1$  integers if there are  $n$  priority classes. The first component takes on values between 1 and  $n$ , representing the class to which the customer in service belongs. (Different classes may have different service requirements.) The other  $n$  integers represent the number of customers in the various priority classes, including the customer being served. Let us, for the purposes of illustration, consider a 2 class system in which customers of class 1 take priority over class 2 customers. The state  $(1,2,4)$  represents the system when there is 1 class 1 customer being served, one class 1 customer waiting, and 4 class 2 customers. From any state in this system, there will be three transitions. Two of the transitions correspond to the event of a customer arriving, one transition for each class of arrival. These transitions occur at the respective arrival rates and when the system is in state  $(1,2,4)$  have as their destinations the states  $(1,3,4)$  and  $(1,2,5)$ . The other corresponds to the customer coming to the end of his service, and occurs at the appropriate rate. The destination state of this transition is the state with the number of customers in the corresponding priority class reduced by 1, in this case it is  $(1,1,4)$ . If there are no customers left in the

priority class, or there are customers of a higher priority waiting, then the integer representing the class of customer being served will be different too. For example, the end of service of the customer in state  $(1,1,4)$  is represented by a transition to state  $(2,0,4)$ . As noted in the previous chapter, states such as  $(1,0,4)$  and  $(2,1,0)$  will be generated, although the system being modelled could never enter them. These states are marked as invalid. The other problem is the idle system, when there are no customers of either priority class present. Clearly, we could use either  $(1,0,0)$  or  $(2,0,0)$  to represent this state. Arbitrarily, we choose to use  $(1,0,0)$  and mark  $(2,0,0)$  as invalid. [In fact, we could allow both  $(1,0,0)$  and  $(2,0,0)$  to be valid, representing the idle state entered by a class 1 customer leaving or a class 2 customer leaving, respectively. The only problem that this would cause is that the probability of the idle system, easily found using Little's theorem, would correspond to the sum of the probabilities of states  $(1,0,0)$  and  $(2,0,0)$ .]

We shall study the 2 class non-preemptive priority system. Class 1 jobs have priority over class 2 jobs. Class 1 jobs have exponentially distributed processing times with mean 0.5 seconds, while class 2 jobs have exponentially distributed times with mean 1 second. Table 5.6 shows the value of the mean number of customers in each class, both as predicted by theory and as calculated using our three algorithms. We assume that the arrival rate of class 1 jobs is 0.5 per second, and that the arrival rate of class 2 jobs is 0.1, 0.2, 0.3, 0.4, and 0.5 per second.

Table 5.7 gives the equivalent results for a system in which



class 2 jobs arrive at rate 0.1 per second, and class 1 jobs arrive at rates 0.25, 0.5, 0.75, 1.0, and 1.25 per second.

Table 5.6.

Rate	Method	Truncation	Class 1	Class 2	Time
0.1	Theory	$\infty$	0.40000	0.14615	$\infty$
0.1	Tweedie	72	0.40009	0.15598	5.6
0.1	Stewart	72	0.39596	0.14373	6.4
0.1	Brandwajn	72	0.39858	0.14532	7.3
0.2	Theory	$\infty$	0.46667	0.37117	$\infty$
0.2	Tweedie	110	0.47997	0.37117	8.2
0.2	Stewart	72	0.45332	0.33703	6.4
0.2	Brandwajn	110	0.46555	0.35565	11.9
0.3	Theory	$\infty$	0.53333	0.67778	$\infty$
0.3	Tweedie	156	0.55851	0.69784	12.3
0.3	Stewart	110	0.52232	0.63704	11.8
0.3	Brandwajn	110	0.52938	0.66291	11.8
0.4	Theory	$\infty$	0.60000	1.20000	$\infty$
0.4	Tweedie	272	0.63392	1.21990	25.6
0.4	Stewart	156	0.58830	1.09757	19.0
0.4	Brandwajn	210	0.59835	1.18394	26.1
0.5	Theory	$\infty$	0.66667	2.16667	$\infty$
0.5	Tweedie	600	0.70845	2.18370	78
0.5	Stewart	342	0.66184	2.02322	72
0.5	Brandwajn	600	0.66762	2.20701	102

Table 5.7.

Rate	Method	Truncation	Class 1	Class 2	Time
0.25	Theory	$\infty$	0.17143	0.12396	$\infty$
0.25	Tweedie	42	0.17370	0.12692	1.9
0.25	Stewart	42	0.16988	0.12244	1.8
0.25	Brandwajn	42	0.17089	0.12396	2.2
0.5	Theory	$\infty$	0.40000	0.14615	$\infty$
0.5	Tweedie	72	0.40689	0.15598	5.6
0.5	Stewart	72	0.39596	0.14373	6.4
0.5	Brandwajn	72	0.39858	0.14532	7.3
0.75	Theory	$\infty$	0.72000	0.18762	$\infty$
0.75	Tweedie	156	0.72940	0.20015	12.8
0.75	Stewart	110	0.70890	0.18195	12.0
0.75	Brandwajn	156	0.71902	0.18713	18.0
1.0	Theory	$\infty$	1.20000	0.27500	$\infty$
1.0	Tweedie	272	1.22140	0.30433	26.2
1.0	Stewart	210	1.19065	0.26945	31.3
1.0	Brandwajn	210	1.19521	0.27257	26.0
1.25	Theory	$\infty$	2.00000	0.50000	$\infty$
1.25	Tweedie	600	2.03999	0.55488	80.2
1.25	Stewart	462	1.98028	0.48436	149.0
1.25	Brandwajn	462	1.99415	0.49666	80.4

It can be seen that Tweedie's method is at least as efficient as the other two. In most cases, it takes less time to reach a solution and in those cases for which Stewart's method was faster, Tweedie's method has considered a larger truncation. It will be noticed that Tweedie's method consistently overestimates the value of the moments. The reason for this can be found by examining the behaviour of the upper and lower bounds on the probability of a particular state as the truncation size increases. In this case, unlike M/M/1, the lower bound is a much better estimate than the upper bound. It both starts closer to the correct value and converges faster than the upper bound. Since the estimate used by the program is the mean of the upper and lower bounds, the estimate will be an overestimate. Since it is possible to increase the speed of the factorisation by a factor of 2, by using better sparse matrix subroutines, we recommend Tweedie's method for all problems in which the size of truncation needed can be successfully dealt with. It has the added advantage that it provides upper and lower bounds on the accuracy of its estimates, whereas the other methods have no way of doing this.

The program was used to calculate the marginal distributions for both classes of customer, with their arrival rates 0.3 and 0.5 jobs per second. The 4 different systems thus modelled have their marginal distributions tabulated in tables 5.8-5.11, and displayed graphically in figure 5.1.

Table 5.8.

Class 1 arrival rate=0.3 service rate=2.0

Class 2 arrival rate=0.3 service rate=1.0

Probability of n customers in class.

n	Class 1	Class 2
0	0.7808	0.6251
1	0.1704	0.2252
2	$0.3784 \cdot 10^{-1}$	$0.7858 \cdot 10^{-1}$
3	$0.8512 \cdot 10^{-2}$	$0.2798 \cdot 10^{-1}$
4	$0.1931 \cdot 10^{-2}$	$0.1016 \cdot 10^{-1}$
5	$0.4407 \cdot 10^{-3}$	$0.3750 \cdot 10^{-2}$
6	$0.1009 \cdot 10^{-3}$	$0.1403 \cdot 10^{-2}$

Table 5.9.

Class 1 arrival rate=0.5 service rate=2.0

Class 2 arrival rate=0.3 service rate=1.0

Probability of n customers in class.

n	Class 1	Class 2
0	0.5500	0.6047
1	0.2292	0.2342
2	$0.7514 \cdot 10^{-1}$	$0.9316 \cdot 10^{-1}$
3	$0.2728 \cdot 10^{-1}$	$0.3852 \cdot 10^{-1}$
4	$0.9291 \cdot 10^{-2}$	$0.1646 \cdot 10^{-1}$
5	$0.3145 \cdot 10^{-2}$	$0.7208 \cdot 10^{-2}$
6	$0.1061 \cdot 10^{-2}$	$0.3206 \cdot 10^{-2}$

Table 5.10.

Class 1 arrival rate=0.3 service rate=2.0

Class 2 arrival rate=0.5 service rate=1.0

Probability of n customers in class.

n	Class 1	Class 2
0	0.7346	0.4138
1	0.1989	0.2386
2	$0.5032^{-1}$	0.1391
3	$0.1228^{-1}$	$0.8232^{-1}$
4	$0.2932^{-2}$	$0.4937^{-1}$
5	$0.6916^{-3}$	$0.2988^{-1}$
6	$0.1618^{-3}$	$0.1819^{-1}$

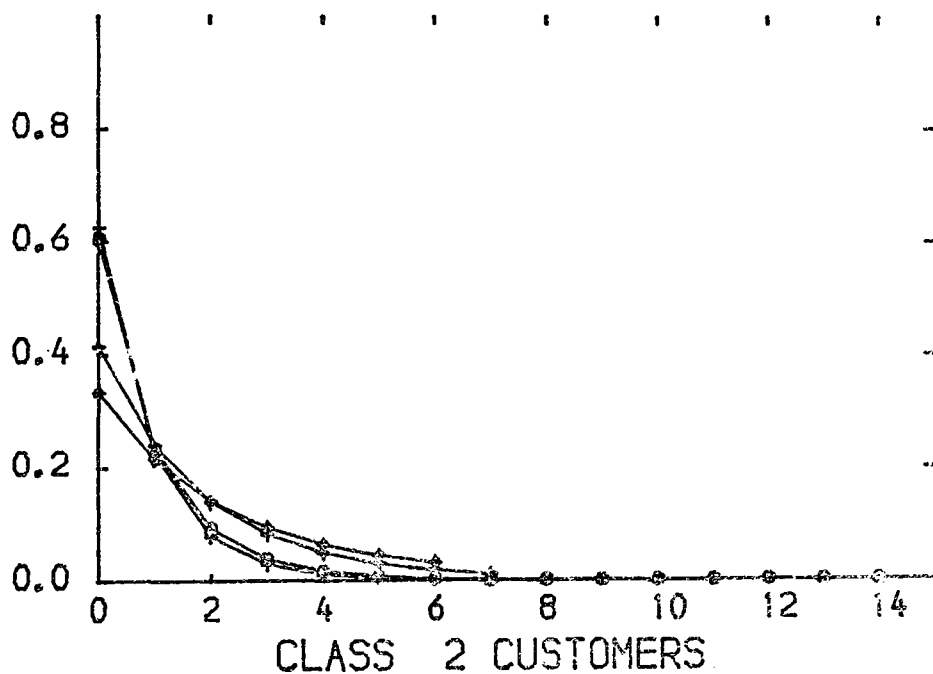
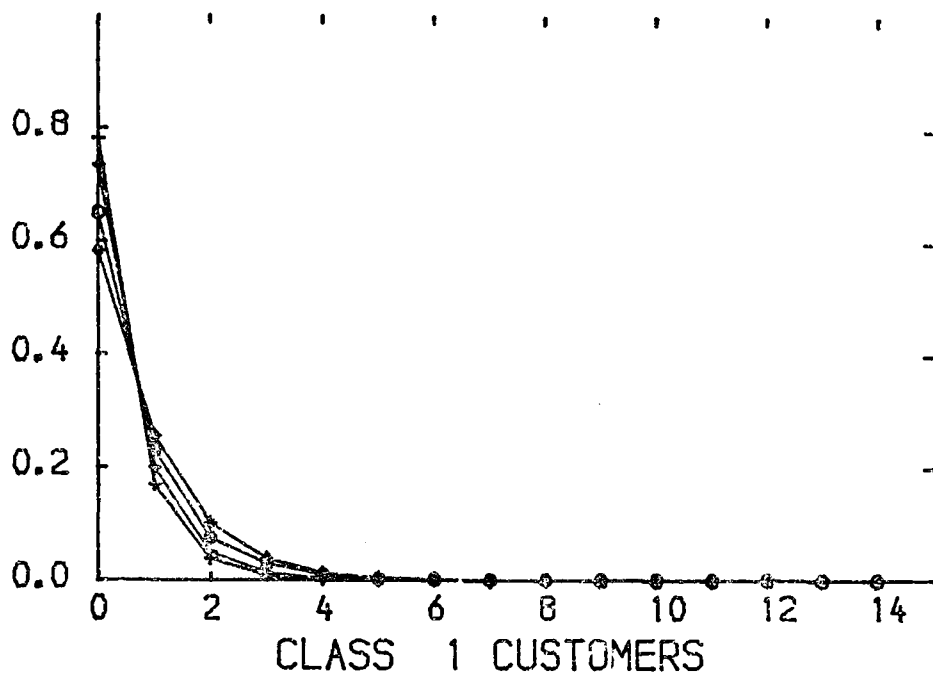
Table 5.11.

Class 1 arrival rate=0.5 service rate=2.0

Class 2 arrival rate=0.5 service rate=1.0

Probability of n customers in class.

n	Class 1	Class 2
0	0.5833	0.3333
1	0.2569	0.2141
2	0.1012	0.1405
3	$0.3767^{-1}$	$0.9460^{-1}$
4	$0.1354^{-1}$	$0.6497^{-1}$
5	$0.4756^{-2}$	$0.4517^{-1}$
6	$0.1647^{-2}$	$0.3164^{-1}$



LINEAR

5.8=+ 5.9=0 5.10=+ 5.11=+

MARGINAL PROBABILITIES

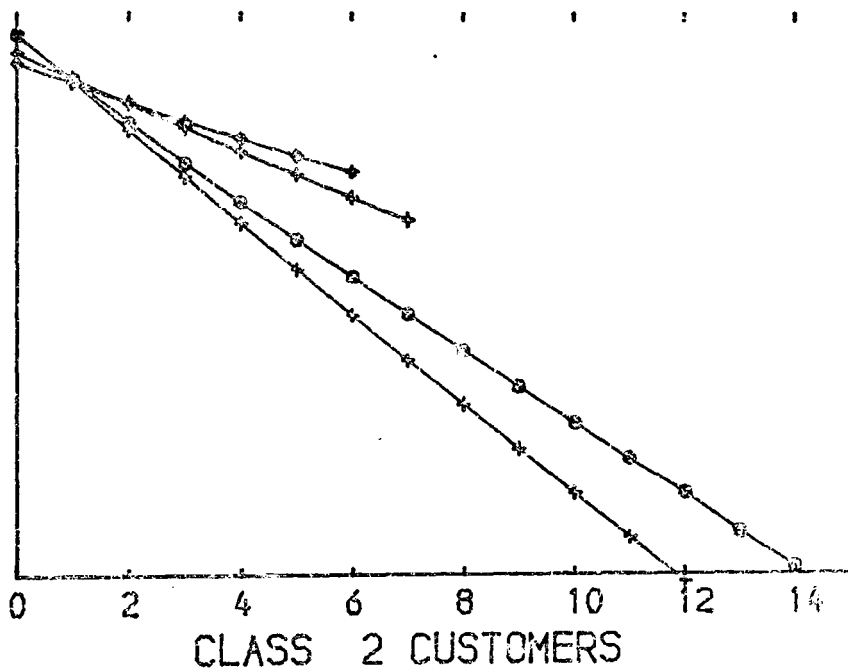
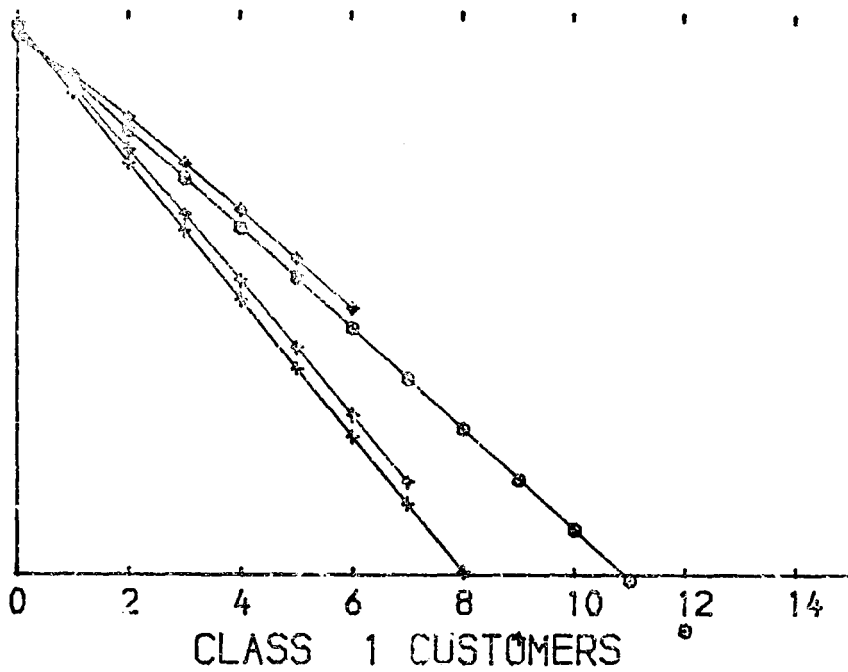
FIGURE 5. 1

Examination of the graphical output suggests that the marginal distribution of the numbers in each priority queue is geometric. If the marginal probabilities are plotted on a logarithmic scale as in figure 5.2 this hypothesis is supported. The correlation between the logarithms of the marginal probabilities and the number in the queue is very high, greater than 0.99 in magnitude in all cases. From this result, it seems that a geometric distribution is a very good approximation to the marginal distribution of the number of customers in each priority class.

### 5.3 Multiple-Microprocessor System

The model of a distributed microprocessor system which we shall investigate is as follows. There are  $n$  identical microprocessors, each capable of processing  $c$  instructions per second. Jobs arrive from outside the system in two classes. Those in class  $i$  arrive in a Poisson stream at rate  $n \cdot \lambda_i$ , with exponentially distributed length, mean  $1/\mu_i$  instructions. Thus the expected execution time of a class  $i$  job is  $1/c\mu_i$  seconds. The processors are also divided into two sets.  $k$  processors are dedicated to class 1 jobs and the remaining  $n-k$  to class 2 jobs. A processor will process jobs from the other class, rather than be unnecessarily idle, but an arriving job of the class to which the processor is dedicated will preemptively seize the processor. Since the jobs have exponentially distributed length there is no need to distinguish between the cases of a job having to restart when it has been preempted and a job continuing at the point of preemption. This model was postulated as a discrete means of approximating the





LOG

5.8=+ 5.9=○ 5.10=+ 5.11=\*

MARGINAL PROBABILITIES

FIGURE 5. 2

performance of Kleinrock's discriminatory processor sharing scheme [30]. Class 1 jobs receive proportion  $k/n$  of the available processor power and class 2 jobs the remainder. Kleinrock's original analysis of such systems has been shown to be erroneous, and the expected response time of jobs in the various classes of such a system have been given by Fayolle, Iasnogorodski, and Mitrani [15]. For a two class system such as ours, the expected response times are

$$W_1 = [1 + \mu_1 \rho_2 (g_2 - g_1) / D] / \mu_1 (1 - \rho) \quad (5.1)$$

$$W_2 = [1 + \mu_2 \rho_1 (g_1 - g_2) / D] / \mu_2 (1 - \rho) \quad (5.2)$$

where

$$D = \mu_1 g_1 (1 - \rho_1) + \mu_2 g_2 (1 - \rho_2)$$

$g_i$  is the proportion of the processor allocated to class  $i$ ,  $\mu_i$  the service rate for class  $i$  jobs,  $\rho_i$  the traffic intensity for class  $i$ , and  $\rho$  is the total traffic intensity.

The states of the system can be represented by a pair of integers, each integer representing the number of jobs in the appropriate priority class, including those being served. Because of the preemptive nature of the dedication of processors and the memoryless property of the exponential distribution, there is no need to represent the classes of the jobs being served as there was in the non-preemptive priority system. There can be up to 4 transitions from each state. Two corresponding to arrivals in each priority class, and the others corresponding to departures. The rate of departure transitions will depend not only on the class of job departing, but on the state of the system and on the number of processors. The rates of the various transitions out of the state  $(n_1, n_2)$  are given below, depending on the

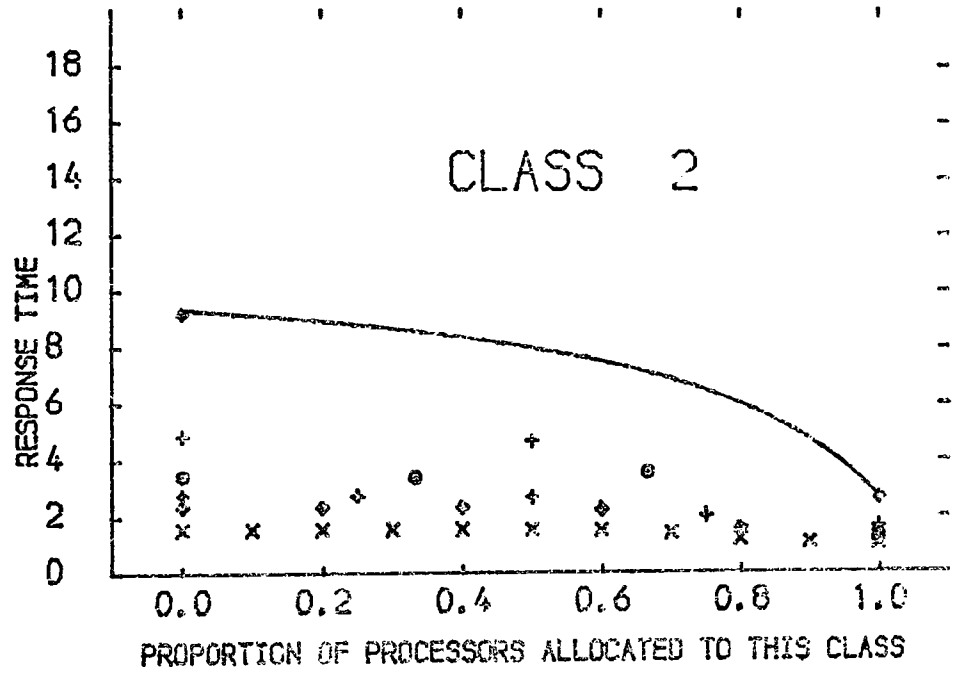
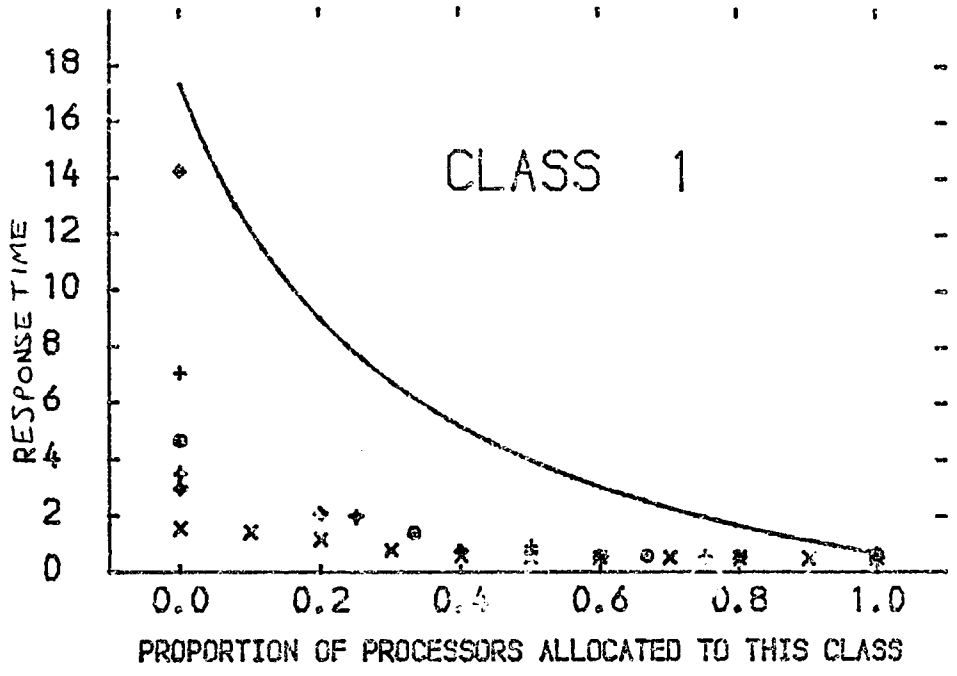
values of  $n$ , the number of processors, and  $k$ , the number of processors dedicated to class 1 jobs.

	Destination	Rate
All $(n_1, n_2)$	$\rightarrow (n_1+1, n_2)$	$n\lambda_1$
	$(n_1, n_2+1)$	$n\lambda_2$
if $n_1 \geq k$ and $n_2 \geq n-k$	$\rightarrow (n_1-1, n_2)$	$k \cdot c \cdot \mu_1$
	$(n_1, n_2-1)$	$(n-k) \cdot c \cdot \mu_2$
if $n_1 < k$	$\rightarrow (n_1-1, n_2)$	$n_1 \cdot c \cdot \mu_1$
	$(n_1, n_2-1)$	$\min\{n_2, n-n_1\} \cdot c \cdot \mu_2$
if $n_2 < n-k$	$\rightarrow (n_1-1, n_2)$	$\min\{n_1, n-n_2\} \cdot c \cdot \mu_1$
	$(n_1, n_2-1)$	$n_2 \cdot c \cdot \mu_2$

Without loss of generality, we shall assume that  $c=1$ . This system was solved numerically in order to find the expected response times for the two classes of customer. A fairly saturated system with  $\rho=0.875$  was used. The system was investigated under the following conditions;  $n$ , the number of processors, taking the values 1,2,3,4,5, and 10;  $k$ , the number dedicated to class 1 jobs, taking all possible values,  $k=0,1,2,\dots,n$ .

When there is only a single processor  $n$  equals 1, and we have a two class preemptive priority system in which class 1 has priority if  $k$  is 1 and class 2 has priority when  $k$  is 0. As we increase the number of processors, staying for the moment with the preemptive priority system, the situation becomes more complex. Intuitively, one might conclude that the response time would not change, since the traffic intensity remains constant. This is what Fayolle et al.'s result predicts. However,

examination of the response times which are tabulated in Table 5.12, shows that the mean response time drops as the number of processors increases. This is true even for the lower priority class. It appears from Table 5.12, which is graphically displayed in figure 5.3, that the response time tends to the service time. For comparison, the response times predicted by Fayolle et al. (5.1) and (5.2) are displayed on the same graph. This intuitively surprising result can be explained as follows. A random arrival will expect to find a proportion  $\rho$  of the servers busy, where  $\rho$  is the traffic intensity. Thus the expected number of idle servers is  $n \cdot (1 - \rho)$ . For large enough  $n$ , the random arrival has a very high expectation that there will be at least one free server and that he will not have to wait. This argument can be applied to the more mixed system, when only some of the processors are dedicated to a particular class. The same phenomenon is observed, and the same heuristic explanation is offered, although the behaviour of the system is modified by the possibility of "borrowing" an idle processor which has been dedicated to the other class. Of course, in the strictly preemptive case, the probability that a higher priority job will have to queue can be calculated using Erlang's C formula. This is a decreasing function of  $n$ , and certainly has a lower bound since the differences between its value for successive values of  $n$  tend to zero, and of course, probabilities are non-negative. However, attempts to prove that this lower bound is zero have failed.



RESPONSE TIME

NO OF PROCESSORS 1=+ 2=+ 3=+ 4=+ 5=+ 10=+

FIGURE 5.3

Table 5.12.

Class 1	Arrival rate=	0.500000	Service rate=	2.000000
Class 2	Arrival rate=	0.625000	Service rate=	1.000000
Number of processors		Response time		
Total	Class 1	Class 2	Class 1	Class 2
1	1	0	0.6666337	9.2005407
1	0	1	14.2364923	2.6279103
2	2	0	0.5333082	4.8490108
2	1	1	0.9291195	4.6728568
2	0	2	7.0494053	1.6204611
3	3	0	0.5097764	3.4337964
3	2	1	0.5770905	3.4023259
3	1	2	1.4033058	3.5576889
3	0	3	4.6700526	1.3294645
4	4	0	0.5033677	2.7412747
4	3	1	0.5220412	2.7323192
4	2	2	0.6425932	2.6753015
4	1	3	1.9765216	2.0284232
4	0	4	3.4893669	1.2009225
5	5	0	0.5012523	2.3339086
5	4	1	0.5075151	2.3308560
5	3	2	0.5409117	2.3145667
5	2	3	0.7374313	2.2207717
5	1	4	2.0686257	1.5434506
5	0	5	2.9463677	1.1360835
10	10	0	0.4998907	1.5520505

10	9	1	0.4999628	1.5520109
10	8	2	0.5002776	1.5518409
10	7	3	0.5015418	1.5511630
10	6	4	0.5062088	1.5486715
10	5	5	0.5222930	1.5401531
10	4	6	0.5768641	1.5115197
10	3	7	0.7729156	1.4069984
10	2	8	1.1522268	1.1907322
10	1	9	1.4233566	1.0778378
10	0	10	1.5629823	1.0310214

## 6 Conclusions

We have investigated 3 methods for estimating the steady state probability distribution of an infinite state Markov process. It has been shown that numerical methods can be successfully used to calculate the steady state distributions of infinite state space processes as well as finite state processes. We recommend the method based on Tweedie's results whenever enough core storage is available to make use of it. It is not only as efficient as the others, but also gives bounds on the possible values of the distribution. The iterative methods are both capable of solving much larger systems than Tweedie's method, but they provide very little information about their accuracy. Of the two, Brandwajn's method seems to be no less accurate than Stewart's, and in the majority of cases it is faster. It has the advantage of being simpler, both in terms of its theoretical basis and vis-a-vis its programming complexity.

A method was developed to map arbitrary state spaces into the natural numbers and to allow for the effect of any extraneous states introduced by the mapping. Although developed for Markov processes, there is no reason why the transformation should not be used in other cases where it is required to number an infinite mesh uniquely.

Numerical investigation of non-preemptive priority systems revealed the slightly surprising result that a geometric distribution is an



excellent approximation to the marginal distribution of the number of customers in each priority class. The multiple micro-processor system study also produced an unexpected result. Although the traffic intensity remained constant, the mean response time decreased as the number of processors increased.

Further research is needed into the behaviour of the bounds with different systems. Simple birth-death processes have the upper bound giving the exact answer. In the case of priority queueing systems, the lower bound was a much better estimate than the upper. Some work is needed to find the characteristics of systems which govern these properties. Stewart suggests that his method may be better for nearly completely decomposable systems. Both Tweedie's and Brandwajn's methods need more investigation into their response to such systems.

## References

- [1] Allen, B., Anderssen, R.S., and Seneta, E.  
Numerical Investigation of Infinite Stochastic Matrices  
Technical Report No. 48, Computer Centre, Australian  
National University. (1975)
- [2] Allen, B., Anderssen, R.S., and Seneta, E.  
Computation of Stationary Measures for Infinite Markov Chains  
In :Algorithmic Methods in Probability, edited by M.F.  
Neuts, North Holland Publishing Company, Amsterdam (1977)
- [3] Bauer, F.L., Deutsch, E., and Stoer, J.  
Abschätzungen für die Eigenwerte positiver linearer Operatoren  
Linear Algebra and Applications 2, Pp 275-301 (1969)
- [4] Bhat, U.N. and Raju, S.N.  
A State Space Truncation Procedure for Queueing Systems  
Department of Computer Science and Operations Research,  
Technical Report CP75005, Southern Methodist University,  
Texas (1975)

- [5] Borland, R.P.  
The AQ algorithm for the solution of a sparse system of linear equations  
Report DNACS 17/79, National Physical Laboratory,  
Teddington (1979)
- [6] Brandwajn, A.  
An Iterative Solution of Two Dimensional Birth and Death Processes  
ENST - D - 77010, Ecole National Superieure des  
Telecommunications, Paris (1975)
- [7] Brandwajn, A.  
An Approach to the Numerical Solution of Some Queueing Problems  
Pp 83-112, Computer Performance, International Symposium  
on Computer performance, Modelling, Measurement, and  
Evaluation, IFIP Working Group 7.3 (1977)
- [8] Cohen, J.W.  
The Single Service Queue  
North Holland Publishing Company, Amsterdam (1959)

- [9] Courtois, P.J.  
Decomposability: queueing and computer system applications  
A.C.M. Monograph , Academic Press, London (1976)
- [10] Curtis, A.R. and Reid, J.K.  
Fortran subroutines for the solution of sparse sets of linear  
equations  
A.E.R.E. Report R6844, H.M.S.O., London (1971)
- [11] Duff, I.S.  
On the number of non-zeroes added when Gaussian elimination is  
performed on random sparse matrices  
Mathematics of Computation 18, Pp 219-233 (1974)
- [12] Duff, I.S.  
MA28 - a set of Fortran subroutines for sparse unsymmetric  
linear equations  
A.E.R.E. Report R8730, H.M.S.O., London. (1977)
- [13] Duff, I.S. and Reid, J.K.  
Some design features of a sparse matrix code  
A.C.M. Transactions on Mathematical Software 5, Pp 18-35  
(1979)

- [14] Erisman, A.M. and Tinney, W.F.  
On Computing Certain Elements of the Inverse of a Sparse  
Matrix  
C.A.C.M. 18, Pp 177-179 (1975)
- [15] Fayolle, G., Iasnogorodski, R. and Mitrani, I.  
On the Sharing of a Processor among Many Job Classes  
I.R.I.A. Rapport de Recherche No. 275, Rocquencourt,  
France (1978)
- [16] Frazer, R.A., Duncan, W.J., and Collar, A.R.  
Elementary Matrices  
C.U.P., Cambridge (1938)
- [17] Gaver, D.P. and Humfeld, G.  
Multi-type Multiprogramming Models  
Acta Informatica 7, Pp 111-121 (1975)

- [18] Golub, G.H. and Seneta, E.  
Computation of the Stationary Distribution of an Infinite  
Markov Matrix  
Bulletin of the Australian Mathematical Society 8,  
Pp 333-341 (1973)
- [19] Golub, G.H. and Seneta, E.  
Computation of the Stationary Distribution of an Infinite  
Stochastic Matrix of special form  
Bulletin of the Australian Mathematical Society 10,  
Pp 255-261 (1974)
- [20] Grassman, W.K.  
Transient Solutions for Queueing Networks  
Research Report 75-5,  
Department of Computational Science, University of  
Saskatchewan (1975)
- [21] Grassman, W.K.  
Some Computational Aspects of the  $M/E_k/1$  Queue in Steady State  
Research Report 76-3,  
Department of Computational Science, University of  
Saskatchewan (1978)

- [22] Grassman, W.K.  
Transient and Steady State results for Two parallel Queues.  
Research Report 78-3,  
Department of Computational Science, University of  
Saskatchewan (1978)
- [23] Hall, C.A. and Porsching, T.A.  
Computing the Maximal Eigenvalue and Eigenvector of a  
Non-negative Matrix  
SIAM Journal on Numerical Analysis 5, Pp 269-274 and  
470-474 (1968)
- [24] Hiemann, D. and Neuts, M.F.  
The Single Server Queue in Discrete Time:  
Numerical Analysis IV  
Naval Research Logistics Quarterly 20, Pp 753-766 (1971)
- [25] Irani, K.B. and Wallace, V.L.  
On Network Linguistics and The Conversational Design of  
Queueing Networks  
J.A.C.M. 18, Pp 616-620 (1971)

- [26] Jensen, A. and Kendall, D.G.  
Denumerable Markov Processes with Bounded Generators: A  
Routine for calculating  $p_{ij}(\infty)$   
Journal of Applied Probability 8, Pp 423-427 (1971)
- [27] Jennings, A and Stewart, W.J.  
Simultaneous Iteration for the Partial Eigensolution of Real  
Matrices  
Journal of the Institute of Mathematics and its  
Applications 15, Pp 351-361 (1975)
- [28] Kemeny, J.G.  
Representation Theory for Denumerable Markov Chains  
Transactions of the American Mathematical Society 125,  
Pp 47-62 (1966)
- [29] Kerridge, D.  
A Numerical Method for the Solution of Queueing Problems  
New Journal of Statistics and Operational Research 2,  
Pp 3-13 (1966)



- [30] Kleinrock, L.  
Time-Shared Systems : A Theoretical Treatment  
J.A.C.M. 14, Pp 242-261 (1967)
- [31] Klimko, E.M. and Neuts, M.F.  
The Single Server Queue in Discrete Time:  
Numerical Analysis II  
Naval Research Logistics Quarterly 20, Pp 305-319 (1971)
- [32] Klimko, E.M. and Neuts, M.F.  
The Single Server Queue in Discrete Time:  
Numerical Analysis III  
Naval Research Logistics Quarterly 20, Pp 557-567 (1971)
- [33] Lynn, M.S. and Timlake, W.P.  
Bounds for Perron eigenvectors and subdominant eigenvalues of  
Positive Matrices  
Linear Algebra 2, Pp 143-152 (1969)
- [34] Mitrani, I.  
Networks of Unreliable Computers  
Technical Report No. 62, Computing Laboratory, University  
of Newcastle upon Tyne (1974)

- [35] Mitrani, I. and Hine, J.H.  
Complete Parameterized Families of Job Scheduling Strategies  
Acta Informatica 8, Pp 61-73 (1977)
- [36] Neuts, M.F.  
The Single-server Queue in Discrete Time: Numerical Analysis I  
Naval Research Logistics Quarterly 20, Pp 289-304 (1971)
- [37] Paige, C.C., Styan, G.P.M., and Wachter, P.G.  
Computation of the Stationary Distribution of a Markov Chain  
Journal of Statistical Computation and Simulation 4,  
Pp 173-186 (1975)
- [38] Ponstein, J  
Theory and Numerical Solution of a Discrete Queueing Problem  
Statistica Neerlandica 28, Pp 139-152 (1974)
- [39] Reid, J.K.  
A note on the stability of Gaussian elimination  
Journal of the Institute of Mathematics and its  
Applications 8, Pp 176-179 (1971)

- [40] Saaty, T.L.  
Elements of Queueing Theory  
McGraw Hill, New York (1961)
- [41] Seneta, E.  
Finite Approximations to Infinite Non-negative Matrices  
Proceedings of the Cambridge Philosophical Society 63,  
Pp 983-992 (1967)
- [42] Seneta, E.  
Finite Approximations to Infinite Non-negative  
Matrices.,Part II : Refinements and Applications  
Proceedings of the Cambridge Philosophical Society 64,  
Pp 465-470 (1968)
- [43] Seneta, E.  
Non-negative Matrices  
Allen and Unwin, London (1973)

- [44] Stewart, W.J.  
Markov Analysis of Operating Systems Techniques  
Ph.D. Thesis, Queen's University of Belfast. (1974)
- [45] Stewart, W.J.  
MARCA: Markov Chain Analyser  
I.R.I.S.A. Publication Interne No.45, Universite de  
Rennes (1976)
- [46] Stewart, W.J.  
A Comparison of Numerical Techniques in Markov Modelling  
C.A.C.M. 21, Pp 144-152 (1978)
- [47] Stewart, W.J.  
A Direct Numerical Method for Queueing Networks  
Unpublished note (1978)
- [48] Tweedie, R.L.  
Truncation Procedures for Non-negative Matrices  
Journal of Applied Probability 8, Pp 311-320 (1971)

- [49] Tweedie, R.L.  
The Calculation of Limit Probabilities for Denumerable Markov  
Processes from the Infinitesimal Properties  
Journal of Applied Probability 10, Pp 84-99 (1973)
- [50] Wallace, V.L. and Rosenberg, R.S.  
Markovian Models and Numerical Analysis of Computer System  
Behaviour  
Proceedings AFIPS SJCC No. 28, Pp 141-148 (1966)
- [51] Yamamoto, T.  
A Computational Method for the Dominant Root of a Non-negative  
Irreducible Matrix  
Numerische Mathematike 8, Pp 324-333 (1966)