September 1, 1978 ESL-FR-834-6

COMPLEX MATERIALS HANDLING AND ASSEMBLY SYSTEMS

Final Report

June 1, 1976 to July 31, 1978

Volume VI

MODELLING AND ANALYSIS OF UNRELIABLE TRANSFER LINES WITH FINITE INTERSTAGE BUFFERS

by

Irvin C. Schick and Stanley B. Gershwin

This report is based on the thesis of Irvin C. Schick, submitted in partial fulfillment of the requirements for the degree of Master of Science at the Massachusetts Institute of Technology in August, 1978. Thesis co-supervisors were Dr. S.B. Gershwin, Lecturer, Department of Electrical Engineering and Computer Science, and Professor C. Georgakis, Department of Chemical Engineering. The research was carried out in the Electronic Systems Laboratory with partial support extended by National Science Foundation Grant NSF/RANN APR76-12036.

(Revised March 30, 1979)

Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

A Markov Chain model of an unreliable transfer line with interstage buffer storages is introduced. The system states are defined as the operational conditions of the stages and the levels of materials in the storages. The steady-state probabilities of these states are sought in order to establish relationships between system parameters and performance measures such as production rate (efficiency), forced-down times, and expected in-process inventory.

Exact solutions for the probabilities of the system states are found by guessing the form of a class of expressions and solving the set of transition equations. Two- and three-stage lines are discussed in detail. Numerical methods that exploit the sparsity and structure of the transition matrix are discussed. These include the power method and a recursive procedure for solving the transition equations by using the nested block tri-diagonal structure of the transition matrix.

Approximate methods to calculate the system production rate are introduced. These consist in lumping machines together, so as to reduce the length of the transfer line to two stages, or in lumping workpieces together in order to reduce the capacity of the storages and thereby render the dimensions of the state space tractable.

The theory is applied to a paper finishing line, as well as to batch and continuous chemical processes. These serve to illustrate the flexibility of the model and to discuss the relaxation of certain assumptions.

ACKNOWLEDGMENTS

Acknowledgments are due to Professor Christos Georgakis for his advice on the chemical engineering examples; to Professor Y.-C. Ho and to M. Akif Eyler of Harvard University for motivating the principles behind Section 6.2; to Professor Stephen G. Graves of the M.I.T. Sloan School of Management for the work presented in Section 9.1; to Dr. Matthew R. Gordon-Clark of the Scott Paper Company for long discussions on the system decribed in Chapter 7; to Dr. Alan J. Laub for his help with Section 3.2.2; to Professors Alvin Drake and Richard Larson for their valuable comments and help; and to Arthur Giordani and Norman Darling for great drafting work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Complex manufacturing and assembly systems are of great importance, and their significance can only grow as automation further develops and enters more areas of production. At the same time, the balance between increased productivity and high cost is rendered more acute by the limitations on world resources, the precariousness of the economy, and the sheer volume of material involved. It is thus necessary to carefully study such systems, not only out of scientific inquisitiveness but also because of their important economical implications.

A suitable starting point in the study of production systems is the transfer line. For the purposes of the present work, a transfer line may be thought of as a series of work stations which serve, process, or operate upon material which flows through these stations in a predetermined order. Transfer lines are the simplest non-trivial manufacturing systems, and it appears that future work on more complex systems will by necessity be based on the concepts and methods, if not the results, derived in their study. Furthermore, transfer lines are already extremely widespread: they have become one of the most highly utilized ways of manufacturing or processing large quantities of standardized items at low cost. Production line principles are used in many areas, from the metal cutting industry, through the flow of jobs through components of a computer system, to batch manufacturing in the pharmaceutical industry. At the same time, the accelerated pace of life and crowded cities have institutionalized queues of people waiting to be served through series of stages, from cafeterias to vehicle inspection stations. The work presented here is devoted to methods of obtaining important measures of performance and design parameters for transfer lines, such as average production rate, in-process inventory, component reliability, and forced-down times.

The transfer line considered here may be termed unflexible: the material flowing through the system is of only one type, and must go though all the stations. A fixed sequence of operations is performed before the material is considered finished and can leave the system. Such a system can be studied as a special case of flexible manufacturing systems.

The stations (also termed machines in the discussion that follows) are unreliable, in that they fail at random times and remain inoperable for random periods during which they are repaired. It is possible to compensate for the losses in production caused by these failures by providing redundancy, i.e. reserve machines that enter the network in case of failures. However, this is often prohibitively expensive, especially in the case of systems involving very costly components.

An alternative appears (Buzacott[1967a]) to have been discovered in the U.S.S.R. in the early fifties. This consists in placing buffer storages between unreliable machines in order to minimize the effects of machine failures. Buffers provide temporary storage space for the products of upstream machines when a downstream machine is under repair, and provide a temporary supply of unprocessed workpieces for downstream machines when an upstream machine is under repair. Although providing storage space and possibly machinery to move parts in and out of storages may be cheaper than redundancy of machines, the cost of floor space and in-process inventory are far from negligible. It is thus necessary to find in some predefined sense the "best" set of storage capacities, in order to minimize cost while keeping productivity high. This leads to what may be refered to as the buffer size optimization problem, which is discussed in section 1.1. Before this important optimization problem can be solved, however, the effect of buffers on productivity, in-process inventory, and other measures of performance must be quantified. This quantification is the purpose of the research reported here.

1.1 Considerations on the Economic Analysis of Interstage Buffer Storages

It is known from experimentation, simulation, and analysis that the average production rate and in-process inventory of a transfer line increase with buffer storage capacity. Before studying in detail the precise methods for finding the relations between these parameters, however, it may be necessary to describe the context for which they are intended. These results are considered in the optimal allocation of interstage buffer storage space.

In some systems, it is desirable to maximize production rate; in others, such as lines that produce components to be assembled with parts produced elsewhere at known rates, it is desirable to keep the production rate as close as possible to a given value, while minimizing cost. In the former case, storages are often of significant value in increasing production rate and compensating for the losses due to the unreliability of machines. However, large storages mean high in-process inventory, a situation that is usually not desirable. In the latter case, ways will be described to find the least costly configuration of interstage buffers to give the desired production rate. In both cases, however, there is need for analysis techniques in order to understand the exact relation between the various design parameters and performance measures.

Since production rate is known to increase with storage size, maximizing production rate could be achieved by providing the system with very large buffers. However, there are important costs and constraints associated with providing buffers, including the costs of storage space and equipment, and in-process inventory. Thus, the buffer size optimization problem must take into account a number of constraints, including the following:

(i) There may be a limit on the <u>total</u> storage space to be provided to the line, i.e. on the sum of the capacities of all individual interstage buffer storages, due to cost of or limitations on floor space.

(ii) Furthermore, the capacity of <u>each</u> interstage storage may be limited due to limitations on floor space, or else, the weighted sum of storage capacities may be limited. This is the case, for example, if the system is an assembly line in which parts are mounted onto the workpieces so that their sizes increase in the downstream direction; this would not only necessitate tighter constraints on downstream storages, but it may also place a low upper limit on them because of floor space limitations.

(iii) It may be desirable to limit the expected (i.e. average) total number of jobs or parts in the system at any time, that is the in-process inventory. (It may be noted that Elmaghraby[1966] calls only those parts that are actually being serviced in-process inventory, while he denotes those in the buffer storages as in-waiting inventory. Here, as in most other works, the term is taken to mean the material waiting in buffer storages.) In-process inventory is an important consideration in the design and operation of manufacturing systems, particularly when the parts are costly or when delay is particularly undesirable due to demand for finished products.

(iv) It may be necessary to limit the expected number of parts in certain storages only. This is the case, for example, if very costly elements are mounted onto the workpieces at a certain station, so that the in-process inventory beyond that point must be limited; if parts equipped with the costly components are allowed to wait in storages, the time between the purchase or manufacture of the costly elements and the sale of the finished products may become long, and this is undesirable. More generally, since each operation at subsequent stages gives more added value to each part, it may be necessary to weigh the cost of downstream inventory more than upstream inventory.

It may also be desirable to limit the amount of in-process inventory between certain specific stations. This is the case, for example, if a workpiece is separated into two parts at a certain station, and one of the parts is removed, possibly processed in a separate line or server, and the two parts are then reassembled at some downstream station. In this case, it is not desirable to have large amounts of inventory waiting between the separation and assembly stations, since that would imply that at certain times, in the presence of failures, the ratio in which the two parts arrive at the assembly station would significantly deviate from the desired one-to-one ratio. Complex network topologies, such as lines splitting and merging, separate lines sharing common servers or storage elements, are not treated here. The present work applies only to simple transfer lines. This is believed to be only a necessary first step towards the study of more complex systems.

It must be noted here that, as will be shown in section 5.3, items (i) and (ii) are not equivalent to (iii) and (iv). In other words, although limiting storage size certainly does impose an upper limit on the amount of in-process inventory, the relation between these two quantities is not necessarily linear.

The constraints outlined above are, of course, not exhaustive; specific applications may require additional considerations or constraints.

Calculating the costs involved in designing, building and operating transfer lines with interstage buffer storages involves numerous factors. Kay[1972] who studied the related problem of optimizing the capacity of conveyor belts by analytical as well as simulation techniques, found that conveyor capacity is an important parameter in the design of production systems. Yet, he found that none of the industrial designers that he encountered had considered this as a design parameter. The techniques and results presented here may serve the dual purpose of reiterating the importance of methods and approaches for calculating the relation of buffer capacity and other design parameters to the performance of transfer lines.

The economic aspects of production lines with interstage buffer storages have been studied by numerous researchers, in some cases by simulation, and in others by analytical methods based on queueing theory. Barten [1962] uses computer simulation to obtain mean delay times for material flowing through the system; he then bases his economic analysis on the cost of providing storage and labor and overhead costs as a function of delay time. Love[1967], who studied the related problem of modeling and policy optimization of a two-station (e.g. warehouse-retailer) inventory system, gives a cost model for inventory including the expected cost per time to operate the system, the cost of providing buffer facility, and that of the expected inventory at each storage. Soyster and Toof[1976] investigate the cost versus reliability tradeoff, and obtain conditions for providing a buffer in a series of unreliable machines. Young[1967] analyzes multi-product production lines and proposes cost functionals for buffer capacities, which he then uses in optimization studies by computer simulation. Kraemer and Love[1970] consider costs incurred by in-process inventory as well as actual buffer capacity, and solve the optimal buffer capacity problem for a line consisting of two reliable servers with exponentially distributed service times and an interstage finite buffer storage.

The approaches proposed in these works may be followed in deriving appropriate cost models for an economic analysis of the system. It is beyond the scope of the present work to attempt to solve, or even formally state, the buffer size optimization problem. For this reason, the economics of unreliable transfer lines with interstage buffer storages are not discussed here in depth. It will suffice to list some of the important elements that must be considered in the cost analysis of such production systems. These include:

(i) Cost of increasing the reliability of machines. While the production rate generally increases with the reliability of individual machines, bottleneck stages eventually dominate. At the same time, increased machine reliability involves increased capital cost, possibly due to additional research, high quality components, etc. In cases where machines are already chosen, there may be no control on their individual efficiency.

(ii) Cost of providing materials handling equipment for each storage. Buzacott[1967b] observes that providing storages involves a fixed cost, independently of the capacity of the storage, because of equipment needed to transfer pieces to and from the buffer, maintaining the orientation of the workpieces, etc. This complicates the decision problem on how many stages a production process must be broken into for optimal performance.

(iii) Cost of providing storage capacity. Floor space can be very expensive, so that buffer storages may involve considerable cost which is linear with the capacity of the buffer. It is sometimes possible, however, to use alternate types of storage elements, such as vertical (chapter 7) or helical (Groover[1975]) buffers, in order to reduce the area occupied by the buffer.

(iv) Cost of repair of failed machines. There is clearly a tradeoff between investing in increased machine reliability (item(i)) and in repairing unreliable machines. Although this cost may not be controlled by providing interstage storages, it enters the design of transfer lines.

(v) Cost of maintaining in-process inventory. One of the major goals in production engineering is minimizing in-process inventory. This is important not only when expensive raw material is involved, but also when the value added to the parts by machining is considerable.

(vi) Cost due to delay or processing time. Apart from the cost of operating the system, there may be a cost due to delaying the production or increasing the expected total processing time. This is especially true of transfer lines involving perishable materials, such as in the food, chemical, or pharmaceutical industries. Delay in response to demand is also an important consideration, although this is most important in flexible lines where the product mix may be changed to conform to demand.

(vii) The production rate of the system: the objective of the optimization problem is maximizing profit rate, a function of production rate as well as cost rate. The latter involves labor and overhead costs, and may be computed in terms of mean-time needed to process a workpiece, including machining times, in-storage waiting times, and transportation. The former requires a more difficult analysis, since its relation to other system parameters such as reliability and storage size, is highly complex.

It is evident from this discussion that the problem of optimally designing a production line has many aspects. These include the choice of machines on the basis of reliability and cost; the division of the line into stages once the machines have been chosen; and the optimal allocation of buffer capacity between these stages. Yet, the relations between these steps and between the various design parameters are not well known, and most previous work in this area has centered on fully reliable lines, on simple two-machine systems, or on simulation. The lack of analytical work on unreliable lines with buffer storages has prompted Buxey, Slack and Wild[1973] to write "the only way to achieve realistic buffer optimization is through the use of computer simulations adapted to apply to specific rather than general situations." Numerical and analytical ways to obtain exact as well as approximate values for production rate, as well as some other performance measures, given the characteristics of the machines and storages, constitute the primary contribution of the present work.

1.2 A Brief Review of Past Research

Transfer lines and transfer line-like queueing processes have been the subject of much research, and numerous approaches as well as results have been reported in the literature. The first analytical studies were the works of Vladzievskii[1952,1953] and Erpsher[1952], published in the U.S.S.R. in the early fifties.

Applications of queueing networks and transfer line models can be found in a large number of seemingly unrelated areas. These include the cotton industry (Goff[1970]), computer systems (Giammo[1976], Chandy[1972], Chandy, Herzog and Woo[1975a,1975b],Shedler[1971,1973], Gelenbe and Muntz [1976], Baskett, Chandy, Muntz and Palacios[1975], Buzen[1971], Lam[1977], Konheim and Reiser[1976], Lavenberg, Traiger and Chang[1973], Wallace[1969], Wallace and Rosenberg[1966], etc.), coal mining(Koenigsberg[1958]), batch chemical processes (Stover[1956], Koenigsberg[1959]), aircraft engine overhauling (Jackson[1956]), and the automotive and metal cutting industries (Koenigsberg[1959]).

A large portion of related research is based on the assumption that parts arrive at the first stage of the transfer line in a Poisson fashion. This greatly simplifies computation, and may be applicable to models of systems where parts arrive from the outside at a random rate, such as jobs in computer systems, people at service stations, cars at toll booths, etc. Most if not all of the computer-related work, as well as the results of Burke[1956], Hunt[1956], Avi-Itzhak and Naor[1963], Neuts[1968,1970], and Chu[1970] are based on the Poisson input assumption. As Soyster and Toof[1976] point out, however, this approach is not realistic when it comes to industrial systems such as assembly and production lines. Here, it is more reasonable to assume that parts are always available at the first stage, so that to follow Koenigsberg[1959], the approach may be termed "stochastic" as opposed to "queueing."

The production rate of transfer lines in the absence of buffers and in the presence of buffers of infinite capacity have been studied (Buzacott[1967a, 1968], Hunt[1956], Suzuki[1964], Rao[1975a], Avi-Itzhak and Yadin[1965], Morse[1965]). Some researchers have analyzed transfer lines with fully reliable components, in which the buffers are used to minimize the effects of fluctuations in the non-deterministic service times (Neuts[1968,1970], Purdue[1972], Muth[1973], Knott[1970a],Hillier and Boling[1966], Patterson [1964], Hatcher[1969] (It should be noted that Knott[1970b] disputes Hatcher's results and provides a counter-example)). Two-stage systems with finite interstage buffers have also been studied (Artamonov[1976], Gershwin [1973a,1973b], Gershwin and Schick[1977], Gershwin and Berman[1978], Buzacott[1967a,1967b,1969,1972], Okamura and Yamashina[1977], Rao[1975a, 1975b], Sevast'yanov[1962]). Longer systems have been more problematic because of the machine interference when buffers are full or empty (Okamura and Yamashina[1977]). Such systems have been formulated in many ways (Gershwin and Schick[1977], Sheskin[1974,1976], Hildebrand[1968], Hatcher [1969], Knott[1970a,1970b]) and studied by approximation (Buzacott[1967a, 1967b], Sevast'yanov[1962], Masso and Smith[1974], Masso[1973]), as well as simulation (Anderson[1968], Anderson and Moodie[1969], Hanifin, Liberty and Taraman[1975], Barten[1962], Kay[1972], Freeman[1964]), but no analytic technique has been found to obtain the expected production rate of a multistage transfer line with unreliable components and finite interstage buffer storages.

1.3 Outline of Research and Contributions

The present work aims at devising analytical, numerical, and approximate methods for solving the problem of obtaining the production rate and other important performance measures of transfer lines with more than two unreliable stages and finite interstage buffers, while at the same time furthering the understanding of two-machine transfer lines.

The problem is formally stated in chapter 2: a description of the transfer line is followed by a state-space formulation in section 2.1, and a discussion of the modeling assumptions in section 2.2. The Markov chain model is introduced and discussed in section 2.3.

An analytical approach is developed in chapter 3: the states of the system are classified as internal and boundary, and these are studied in sections 3.1 and 3.2 respectively. A sum-of-products solution for the steady-state probabilities of internal states of the system is introduced in section 3.1.2, and the analysis is extended to the boundary states for two-machine lines, and three-machine and longer lines, in sections 3.2.1 and 3.2.2 respectively.

Numerical methods for solving the transfer line problem are developed in chapter 4: the iterative multiplication scheme known as the power method is introduced and discussed in section 4.1. An algorithm which solves the large system of transition equations by taking advantage of the sparsity and block-tri-diagonal structure of the transition matrix is developed in section 4.2: the structure of the transition matrix is studied in section 4.2.1 and the algorithm is formulated in section 4.2.2. Some important computer storage problems associated with this algorithm are discussed in section 4.2.3.

The state probabilities obtained by the analytical and numerical methods discussed in chapters 3 and 4 are used to calculate important system performance measures in chapter 5: these include efficiency and production rate, forced-down times, and in-process inventory. The production rate of the system is discussed in section 5.1: alternate ways to compute production rate are given in section 5.1.1, and the effects of start-up transients on this quantity are investigated by dynamic simulation in section 5.1.2. The dependence of production rate, forced-down times and expected in-process inventory on the failure and repair rates of individual machines and the capacities of individual storages is studied in sections 5.1.3, 5.2, and 5.3 respectively.

Approximate methods for computing the system's production rate with less computation than is required by the exact methods developed in earlier chapters are introduced in chapter 6: dynamic simulation and its limited uses in the present work are briefly reviewed in section 6.1. An aggregate method for computing the approximate average production rate of a long transfer line is introduced in section 6.2: this method is based on the quasi-geometric input and output characteristics of two-machine lines, as demonstrated in section 6.2.1. Since a single machine has exactly geometric input and output characteristics, the approximate equivalence of a single machine to a two-machine, one-storage segment of a transfer line is proposed in section 6.2.2. It is shown, however, that the approximation is best when the line is not well balanced, a rare occurrence in actual industrial systems. A mathematical operation on the system parameters referred to as the δ-transformation is introduced in section 6.3.1. It is shown in section 6.3.2 that this transformation leaves production rate nearly unchanged. The major consequence is that the state space can be considerably reduced through this approach, thus decreasing the amount of computation and memory necessary to solve the problem.

Chapters 7,8, and 9 are devoted to applications of the theory. The aim of these chapters is primarily to demonstrate the wide-range applicability of the model, while at the same time pointing out its shortcomings and weaknesses and discussing ways of extending the model to more closely conform to actual situations.

Chapter 7 outlines a paper finishing line: this system is shown to lend itself to a three-machine, two-storage transfer line model, although several important differences exist between the system and the model. These are discussed in section 7.1, while attempts at modeling the system are reviewed in section 7.2.

Chapters 8 and 9 investigate the application of the transfer line model to chemical systems. It appears that Stover's [1956] pioneering work in the application of queueing theory to chemical plants has not been followed up or developed subsequently. Yet, as is shown here, this approach can be particularly useful in estimating the production rates of chemical systems in the presence of unreliable equipment: pumps or valves that fail, heating, cooling, or control mechanisms that break down, etc.

A queueing theory approach to the study of batch chemical processes, in which pumps, reactors, and other unreliable components are represented by machines and holding tanks by storages, is introduced in section 8.1. Major differences between actual systems and the model are discussed in sections 8.1.1 and 8.1.2. The model is extended to account for cases where servicing times are not deterministic. This includes reactors where batches of chemicals take periods of time which deviate from a known mean holding time to reach a desired conversion. This may happen because of variations in the temperature or concentration of the feed, or because the kinetics of the reaction are not understood well enough to predict reaction times exactly. The new model is applied to a simple system consisting of a batch reactor and a still, separated by unreliable pumps and parallel holding tanks, in section 8.2.1. A numerical example is worked out, and more complex systems are discussed, in sections 8.2.2 and 8.2.3 respectively.

The δ-transformation introduced in section 6.3 is taken to its limit as $\delta \to 0$ and the model is shown to become equivalent to a continuous system in chapter 9. Results obtained by differential equations for a continuous line are outlined in section 9.1, and the limiting case of the δ-transformation is studied in section 9.2. The two approaches are shown to yield identical results. A numerical example of a continuous chemical process, in which a plug-flow reactor and a distillation column are separated by unreliable pumps and a holding tank is worked out.

Conclusions and suggestions for future research appear in chapter 10.

2. PROBLEM STATEMENT AND MODEL FORMULATION

Formulating a mathematical model in order to study the relations between certain parameters and measures of performance in transfer lines requires a formal and unambiguous statement of the problem.

Section 2.1 gives a general description of a multistage transfer line with unreliable components and interstage buffer storages. The line is discussed in section 2.1.1 and a state space formulation is introduced in section 2.1.2.

The various assumptions made in the process of translating the system into a mathematical model are outlined and discussed in section 2.2. These assumptions are necessary in order to render the mathematical model tractable, while not losing sight of the physical properties of the actual system. Many of these assumptions are standard (Feller[1966], Koenigsberg [1959]). The assumptions are stated, justification is given, and possibilities of relaxation are investigated.

The Markov chain approach to modeling the transfer line is discussed in section 2.3. This approach is frequently used in the study of queueing networks arising from computer systems (Wallace[1972,1973], Wallace and Rosenberg[1966]) or manufacturing systems (Buzacott[1967a,1967b,1969,1971, 1972]). A brief review of the properties of Markov chains is given in section 2.3.1.(An excellent and exhaustive text on Markov systems is Howard [1971]). The Markov model of the transfer line is discussed in section 2.3.2.

2.1 Modeling the Transfer Line

2.1.1 Description of a Multistage Transfer Line with Buffer Storages

The system under study is illustrated in figure 2.1. It consists of a linear network of machines separated by buffer storages of finite capacities. Workpieces enter the first machine from outside the system. Each piece is processed (drilling or welding in a metal cutting line, reacting or distillating in a chemical plant, data processing in a computer network, etc.) by machine 1, after which it is moved into storage 1. For the purposes of this study, the nature of the machine operation may be ignored, and a machine is taken to be an unreliable mechanism which moves one workpiece per cycle in the downstream direction. The buffer is a storage element in which a workpiece is available to a downstream machine with a negligible delay. The part moves in the downstream direction, from machine i to storage i to machine i+1 and so on, until it is processed by the last machine and thereby leaves the system.

Machines fail at random times. While some of these failures are easy to diagnose and quick to repair, such as some tool failures, temporary power shortages, etc., others involve more serious and time-consuming breakdowns, such as jamming of workpieces or material shortages. Thus, the down-times of the machines, like the up-times, are random variables. When a failure occurs, the level in the adjacent upstream storage tends to rise due to the arrival of parts produced by the upstream portion of the line; at the same time, the level in the downstream adjacent storage tends to fall, as the parts contained in that storage are drained by the downstream portion of the line. If the failure lasts long enough, the upstream storage fills up, at which time the machine immediately preceeding it gets blocked and stops. Similarly, given that the failure takes long enough to repair, the downstream storage eventually empties, and causes the machine following it to starve and stop. This effect propagates up and down the line if the repair is not made promptly.

If it is assumed that machines cannot operate faster than their usual rates in order to catch up the time lost because of such failures, it is clear that

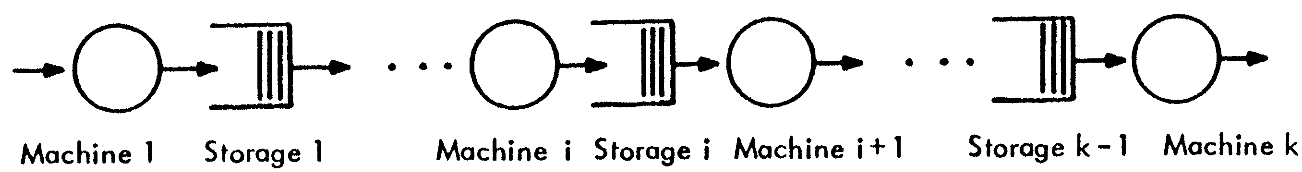


Figure 2.1. A k-machine transfer line.

breakdowns have the effect of reducing the average production rate of the transfer line. Although machine failures are to a certain extent inevitable, it is desirable to avoid situations in which operational machines are affected by failures elsewhere and are forced to stop. Such situations can to a certain extent be avoided by the use of buffer storages, which act so as to partially decouple adjacent machines. As the capacities of these storages are increased, the effects of individual failures on the production rate of the system are decreased.

It is desirable to study the interactions between the elements of the system and the relations between various system parameters, so as to be able to quantify the advantages of using buffer storages and their effect on system production rate.

2.1.2 State Space Formulation

A probabilistic approach is taken in the study of unreliable transfer lines. Starting with probabilities of failure and repair for each individual machine in the line, the probabilities of producing a piece, of being forced down, or of having a given number of parts in a given storage within any time cycle are sought. These are used in evaluating the system's performance. The transfer line problem was studied through such a probabilistic approach for the first time (see Buzacott[1967a]) by Vladzievskii[1952].

In order to carry out the analysis in this direction, it is necessary to formulate a state space for the probabilistic model. A system state is defined as a set of numbers that indicate the operational status of the machines and the number of pieces in each storage, as described below.

For each machine in a k-machine line, the variable α_i is defined as follows:

$$\alpha_i \triangleq \begin{cases} 0 \text{ if machine } i \text{ is under repair} \\ 1 \text{ if machine } i \text{ is operational} \end{cases} \qquad i=1,..,k \qquad (2.1)$$

It is important to note that operational is defined to mean "capable of processing a piece," as opposed to "actually processing a piece." This accounts for cases where the machines are in good working order, but are not processing parts because they are starved or blocked. Several authors (Haydon[1972], Okamura and Yamashina[1977], Kraemer and Love[1970]) define four states, by adding to the above separate states for blocked and starved machines. It will be shown, however, that since probabilities of transition between states are taken here to depend not only on the states of machines but also on the levels of storages, the two approaches are equivalent, though the one given by equation (2.1) is more compact.

The variable n_j is defined as the number of pieces in (the level of) storage j. Each storage is defined to have a finite maximum capacity N_j, so that

$$0 \leq n_j \leq N_j \quad ; \; j=1,..,k-1 \tag{2.2}$$

The state of the system at time t is defined to be the set of numbers

$$s(t) = (n_1(t),..,n_{k-1}(t),\alpha_1(t),..,\alpha_k(t)) \tag{2.3}$$

It may be noted that time, though denoted by the letter t, is discrete. As will be described in section 2.2.2, time is measured in machining cycles.

The <u>efficiency</u> of a transfer line is defined to be the probability of producing a finished piece within any given cycle. It may be thought of as the expected ratio of time in which the system actually produces finished parts to total time. Efficiency, E, satisfies

$$0 \leq E \leq 1 \tag{2.4}$$

State transition probabilities are treated in section 2.2.3. Methods of obtaining steady state probabilities are developed in chapters 3 and 4, and their relation to efficiency are discussed in chapters 5 and 6.

2.2 Assumptions of the Model

2.2.1 Input and Output of the Transfer Line

It is assumed that an endless supply of workpieces is available upstream of the first machine in the line and an unlimited storage area downstream of the last machine is capable of absorbing the parts produced by the line. Thus, the first machine is never starved and the last machine is never blocked.

Although a large portion of computer-related work assumes that jobs arrive at the system at random rates, often in Poisson fashion, it is more realistic in industrial systems to assume that parts are available when needed (Soyster and Toof[1976]). Nevertheless, it is possible to think of cases in which delays in reordering raw materials etc. may cause a shortage of workpieces at the head of the line. Similarly, it is conceivable that congestion downstream in the job shop may cause blocking at the end of the line. These events would clearly not have Poisson time distributions: in that case, parts arrive singly, with random interarrival times. In most industrial cases, it may be expected that parts are delivered in batches. In such cases, it is possible to think of the first and last machines in the model as representing loading and unloading stations. Then, temporary shortages of workpieces or temporary congestion downstream may be modeled as failures in these machines. In other words, unreliable first and last machines may model delivery to and from the production line, especially if parts are moved in bulks (Bagchi and Templeton[1972]).

A single machine, i.e. a one-machine line, stays up for a random length of time, and once a failure occurs, it remains down for a random length of time. Both of these periods are geometrically distributed (as will be shown in section 2.2.3). Thus, the arrival of bulks (or batches) of geometrically distributed sizes, with geometrically distributed interarrival times, may be modeled by a fictitious first machine. This may involve some additional considerations, however. Subsequent deliveries must be independent, and the first storage may have to have infinite capacity.

In general, the assumption of infinite workpiece supply will be justified for most industrial applications. Entire production lines seldom have to stop because of lack of raw material; major shutdowns due to strikes or accidents are of an entirely different nature and are not considered stochastic failures in the sense described in section 2.2.3. There may, nevertheless, be cases where loading and unloading batches takes some time. This is the case, for example, with the paper finishing line (chapter 7) where paper is supplied to the line in the form of extremely large, but necessarily finite rolls. As discussed in section 7.1.2, the effect of starving the line during loading may be ignored if the period of time in which the system is starved is negligible compared to other times involved in the system.

2.2.2 Service Times of the Machines

It is assumed that all machines operate with equal and deterministic service times. The temporal parameter t is chosen so that one time unit is equal to the duration of one machine cycle. Thus, the line has a production rate determined only by its efficiency. The efficiencies of individual machines in isolation, on the other hand, are functions of their mean times between failures and mean times to repair, or alternately their repair and failure probabilities. (This is discussed in detail in section 5.1).

Although deterministic service times may be encountered in certain actual systems (Koenigsberg[1959] mentions an automobile assembly line), this assumption does not hold in many industrial applications. Not only is machining time often a random variable, but downstream machines frequently operate on the average at a faster rate than upstream ones, in order to avoid as much as possible the blocking of upstream machines.

The assumption of constant machining times is justifiable if service times do not deviate appreciably from the mean, compared to the mean service time. This is because variances in service times do not significantly affect the system behavior and average production rate at the condition that the system is not driven to boundaries, i.e. storages are not emptied or filled up. As will be shown in later chapters, the largest steady-state probabilities belong to states with 1 or N_i-1 pieces in storages. Thus, the system runs most often near boundaries. As a result of that, small deviations from the mean may average out, although large deviations may starve certain machines and block others, thereby reducing the line production rate.

Solutions have been obtained for queueing networks with servers having exponential time distributions (See section 8.2). The assumption of exponential distribution reduces the complexity of the problem, but numerous researchers point out that this is often not a reasonable assumption (e.g. Rao[1975a]). Gaussian distributions have been proposed by some (Vladzievskii[1952], Koenigsberg[1959]) and certain Erlang (See Brockmeyer, Halstrøm and Jensen[1960]) distributions may be considered in that they have applicability to industrial cases and satisfy the Markov property of no memory (Section 2.3).

Transportation takes negligible time compared to machining times.

2.2.3 Failure and Repair of Machines

Machines are assumed to have geometrically distributed times between failures and times to repair. This implies that at every time cycle, there is a constant probability of failure given that the machine is processing a piece, equal to the reciprocal of the mean time between failures (MTBF). It is further assumed that machines only fail while processing a piece. Similarly, there is a constant probability of repair given that the machine has failed, equal to the reciprocal of the mean time to repair (MTTR).

The assumption of geometric failure rate is common (Vladzievskii[1952], Koenigsberg[1959], Esary, Marshall and Proschan[1969], Barlow and Proschan [1975], Goff[1970], Buzacott[1967a,1967b,1969], Feller[1966], Sarma and Alam [1975]). It makes it possible to model the system as a Markov chain, since it satisfies the memoryless property of Markov systems (Section 2.3). However, there are certain difficulties with this assumption. While it applies to those cases where the overwhelming majority of failures are due to accidental, truly stochastic events, such as tool breakage or workpiece jams, it does not account for scheduled down-times or tool wear. Such stoppages are predictable given knowledge of the history of the system. Yet, when there is a very large number of possible causes of failure, so that even if some are scheduled, the time distribution including stochastic failures is close to a geometric distribution, this assumption can be made.

Geometric repair time distributions imply that the repair is completed during any cycle with a constant probability, regardless of how long repairmen have been working on the machine. This assumption may not be far from the truth if there are many possible causes of failure, each of which take different lengths of time to repair.

Some examples of actual up- and down-time distributions from an industrial manufacturer appear in figures 2.2 and 2.3. Although these are for relatively small numbers of runs, totalling no more than several hundred time cycles, the distribution is in fact seen to be remarkably close to geometric. (These charts represent typical data obtained from an industrial manufacturer. The actual data is the proprietory information of the industrial manufacturer.)

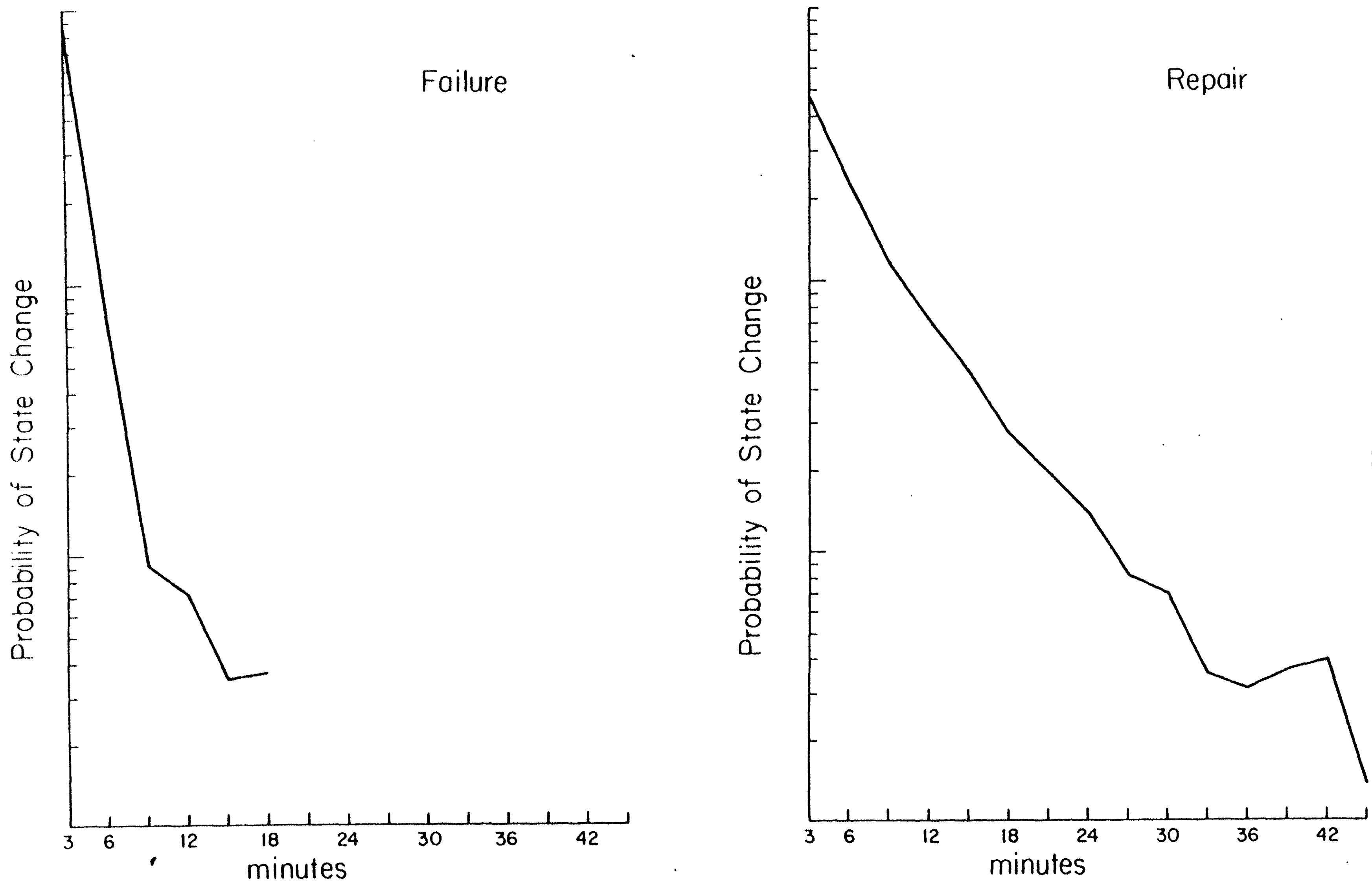


Figure 2.2. Some examples of almost exponential failure and repair time distributions (finite-time samples of data).

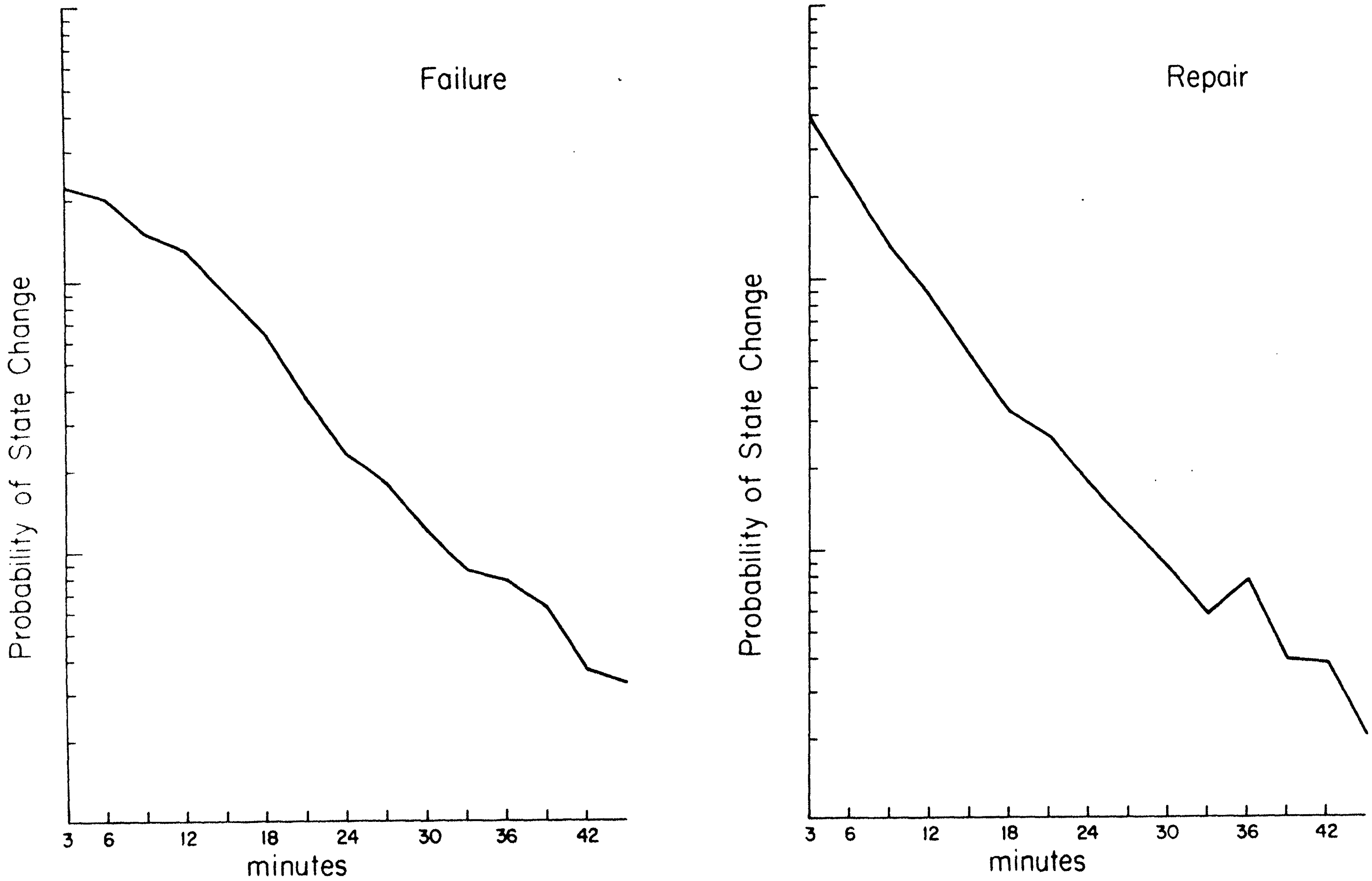


Figure 2.3. Some examples of almost exponential failure and repair time distributions (finite-time samples of data).

The model does not take into account the problem of machine interference (Benson and Cox[1951], Cox and Smith[1974]), in which the limited number of repairmen affects the repair probabilities when more than one machine are down simultaneously. Not only are the repair probabilities reduced in such cases, but they further depend on which machine broke down first, since the repairmen will be at work at that machine with greatest probability. Ways of taking this problem into account are discussed to some detail in section 7.1.6.

While repair takes place independently of storage levels or the number of failed machines, a failure can only occur when the machine is actually processing a part. This implies that the upstream storage is not empty and the downstream storage is not full. In the former case, the machine has no workpiece to operate on, and in the latter, it is not allowed to operate since there is no place to discharge a processed workpiece. In other words, a forced-down machine cannot fail. In research reported by Koenigsberg[1959], Finch assumed that forced-down machines have the same probability of failure as running machines, an assumption that is not realistic. (Buzacott[1967a,1967b], Okamura and Yamashina[1977]).

The assumption that machines only fail while actually processing a piece is consistent with the assumption that the great majority of failures is due to stochastic events such as tool breakage, as opposed to scheduled shutdowns or major system failures that may happen at any time.

2.2.4 Conservation of Workpieces

The model does not account for any mechanism for destroying or rejecting workpieces, or for adding semi-finished workpieces into the line. Thus, the average rate at which pieces are processed by each stage in the line is the same for all stages. It is shown in section 5.1.1 that the solution to the two-machine line satisfies the conservation of pieces. The proof is not complete for longer lines.

The fact that pieces are not created by the system is true except when machines cut workpieces into identical parts, all of which are then processed by downstream machines; this is the case in the paper finishing line (See section 7.1.1). That pieces are not destroyed, however, assumes that a workpiece is not scrapped when a machine fails while processing it (as in the work of Okamura and Yamashina[1977]), that there are no interstage inspection stations where defective parts are removed, etc. In systems satisfying these requirements, all stages process the same average number of pieces per cycle, and it is thus only necessary to compute the production rate of one stage, e.g. the last one (Koenigsberg[1959]). There is an important exception to this rule, and that involves infinite buffer storages for which the upstream portion of the line is more efficient than the downstream portion. This is examined in greater detail in section 5.1.3.

Cases in which workpieces are cut into parts or parts are assembled or packaged together are briefly treated in section 7.1.1. It is possible to approximate such lines by considering the smallest part as a unit and analyzing larger parts, either before they are cut or after they are assembled, as multiples of the smallest unit. This approach is not exact, and errors are introduced by effective changes in the flexibility of the system. (See section 6.3).

2.2.5 Dynamic Behavior of the System

It is assumed as a convention that machines make their state transitions first, conditional on the level of the adjacent storages. Once these changes take place, the storage levels undergo state transitions, within the same time cycle. This is only a convention and the actual system does not have to operate this way. Thus, the transition $\alpha_i(t) \rightarrow \alpha_i(t+1)$ is conditional on $\alpha_i(t)$, $n_{i-1}(t)$ and $n_i(t)$. However, the transition $n_i(t) \rightarrow n_i(t+1)$ is conditional on $n_{i-1}(t), n_i(t), n_{i+1}(t)$, as well as $\alpha_i(t+1)$ and $\alpha_{i+1}(t+1)$, where these latter indices are the final machine states while the former are the initial storage states. Note that the machine and storage transitions depend only on the adjacent machine and storage states, and do not depend on the states of machines and storages further removed.

This assumption makes the computation easier, because it implies that the final storage state is uniquely determined once the initial storage states and the final machine states are known. The advantages of this approach in the mathematical derivation are made clearer in section 3.1.1.

This assumption is consistent with those stated previously: a machine can not fail if the adjacent upstream storage is empty, so that there are no parts to process, or if the adjacent downstream storage is full, so that there is no place to put the processed piece. Furthermore, a piece is not destroyed when a machine fails, but merely remains in the upstream storage until the machine is repaired. Finally, since all machines work synchronously, there is no feed forward information flow, so that the knowledge that a place will be vacant in the downstrean storage or that a piece will emerge from the upstream machine in the time cycle to follow does not influence the decision on whether or not to attempt to process a piece.

It is important to note that this is mostly for mathematical convenience and need not represent the operation of the actual system. One consequence of this assumption is important, however, and must be consistent with the actual system. Because there is no feed forward information flow, a machine can not decide to process a piece if the upstream storage is empty, even though the upstream machine may be ready to discharge a part. Similarly, the machine cannot start processing a piece if the downstream storage is full, even though the downstream machine may have just been repaired and is ready to take in a piece. Thus, there is a delay of at least one cycle between subsequent operations by adjacent machines on any given workpiece, and between a change in the system state and decisions on the part of the machines towards the next state transition. This is unlike the models analyzed by Hatcher[1969] and Masso[1973], in which a part may emerge from a machine and go into the next, bypassing the storage element, within the same time cycle.

2.2.6 The Steady State Assumption

It is assumed that the probabilistic model of the system is in steady state, i.e. that all effects of start-up transients have vanished and that the system may be represented by a stationary probabilistic distribution.

A stochastic system is never at rest. Thus, as explained in section 5.1.2, the steady state assumption does not imply that the system is not fluctuating. What it does imply is that a sufficiently long period of time has passed since start-up, so that knowledge of the initial condition of the system does not give any information on the present state of the system. Thus, the average performance of the system approaches the steady state values calculated by assuming that the probabilistic model of the system is stationary.

There may be cases, however, in which transients take very long to die down, compared to the total running time of the system. In such cases, the steady state values may not represent the average performance of the system. The effects of start-up transients are briefly discussed in section 5.1.2.

2.3 Formulation of the Markov Chain Model

2.3.1 The Markovian Assumption and Some Basic Properties

A stochastic process may be defined as a sequence of events with random outcomes. A process is said to be Markovian if the conditional joint probability distribution of any set of outcomes of the process, given some state, is independent of all outcomes prior to that state. Thus, defining the state of the system at time t as s(t),

$$p[s(t+1)\,|\,s(t),s(t-1),..,s(t-\tau)] = p[s(t+1)\,|\,s(t)] \qquad (2.5)$$

This implies that at any given time, the transition probability depends only on the state occupied at that time; it is independent of the past history of transitions. Another way of saying this is that the transition from one state to another is independent of how the system originally got to the first state. This is what is meant by the memorylessness of Markov processes.

The expression appearing on the right-hand-side of equation (2.5) is the probability of transition from the state occupied at a given time to the state occupied one time step later. This probability is assumed to be independent of time. Thus, the state transition probabilities are defined as

$$t_{ij} \triangleq p[s(t+1)=j\,|\,s(t)=i] \quad ; \text{ all } i,j \qquad (2.6)$$

Given that there are M states, the transition probabilities defined by equation (2.6) obey the following relations:

$$t_{ij} \geq 0 \quad ; \quad \text{all } i,j \qquad (2.7)$$

$$\sum_{j=1}^{M} t_{ij} = 1 \quad ; \quad \text{all } i \qquad (2.8)$$

It is possible to represent the state transition probabilities in matrix form.

The <u>transition matrix</u> is defined as

$$T \triangleq \begin{bmatrix} t_{11} & t_{21} & \cdots & t_{M1} \\ t_{12} & t_{22} & & \vdots \\ \vdots & & & \\ t_{1M} & & \cdots & t_{MM} \end{bmatrix} \tag{2.9}$$

At time t, the probabilities that the system is in state i=1,..,M may be represented as a <u>state probability vector</u>, defined as

$$\underline{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \vdots \\ p_M(t) \end{bmatrix} \triangleq \begin{bmatrix} p[s(t)=1] \\ p[s(t)=2] \\ \vdots \\ p[s(t)=M] \end{bmatrix} \tag{2.10}$$

where

$$\sum_{i=1}^{M} p_i(t) = 1 \tag{2.11}$$

Then, the state probability vector at time t+1 is given by

$$\underline{p}(t+1) = T\,\underline{p}(t) \tag{2.12}$$

and recursive application of equation (2.12) gives

$$\begin{aligned} \underline{p}(t) &= T^t\,\underline{p}(0) \\ &\triangleq \Phi(t)\,\underline{p}(0) \end{aligned} \tag{2.13}$$

Here, $\underline{p}(0)$ is a given initial (<u>a priori</u>) probability vector, and T^t denotes the t^{th} power of the transition matrix T. The chain is termed <u>ergodic</u> if the limit

$$\lim_{t\to\infty} \Phi(t) \overset{\Delta}{=} \Phi \qquad (2.14)$$

exists and if the <u>steady-state probability vector</u> defined as

$$\underline{p} \overset{\Delta}{=} \Phi\, \underline{p}(0) \qquad (2.15)$$

is independent of the value of the initial state probability vector $\underline{p}(0)$. As $t\to\infty$, equation 2.12 becomes

$$\underline{p} = T\, \underline{p} \qquad (2.16)$$

since the vectors $\underline{p}(t)$ and $\underline{p}(t+1)$ converge to $\underline{p}$.

Equations (2.11) and (2.16) are shown to uniquely determine the value of $\underline{p}$ for the system under study in section 4.2.1. These two equations form the basis of both analytical methods derived in chapter 3 and the sparse block tri-diagonal system of equations solving algorithm introduced in section 4.2. The power method discussed in section 4.1 is based on equations (2.12)-(2.15).

2.3.2 System Parameters

Assumption 2.2.3 implies that whenever a machine is processing a part, it has a probability of failure p_i. Since the up-times of the machines are geometrically distributed, the failure probability for a given machine is equal to the reciprocal of its mean time between failures. When the machine is operational, i.e. in good working order, but forced down either because the upstream storage is empty or because the downstream storage is full, it can not fail; thus, the failure probability of a starved or blocked machine is zero. Finally, when processing a piece, a machine can either fail or successfully complete the machining cycle; thus, since its failure probability is p_i, the probability that it successfully completes the part is $1-p_i$.

Repair of a failed machine starts at the beginning of the time cycle following the failure. By assumption 2.2.3, the probability that a failed machine is repaired at the end of any cycle is r_i. This value is independent of storage levels or the status of other machines. Since down-times of machines are geometrically distributed, the repair probability of a given machine is equal to the reciprocal of its mean time to repair. The probability that a failed machine remains down at the end of a time cycle is $1-r_i$. These probabilities are summarized in table 2.1.

As discussed in section 2.2.5, storage level transitions are uniquely determined by the knowledge of the initial storage levels and the final machine states. Consequently, these transitions have probabilities either equal to 1 (certain) or to zero (impossible). The transitions with probability 1 are listed in table 2.2. Some of these are discussed below.

The level of storage i at time t+1 depends on its level at time t, as well as on whether or not a part is added to or withdrawn from it by the adjacent machines. The upstream machine adds a piece to the storage if it is operational and if it is allowed to process parts, i.e. if it is neither starved nor blocked. Similarly, the downstream machine withdraws a piece from the storage if it is operational, as well as neither starved nor blocked. Consequently, $n_i(t+1)$, the level of the storage at time t+1, is determined by the upstream and downstream machine states at time t+1 $(\alpha_i(t+1)$

$$\text{prob}[\alpha_i(t+1) \mid n_{i-1}(t),\ \alpha_i(t),\ n_i(t)]$$

$n_{i-1}(t)$	$n_i(t)$	$\alpha_i(t)$	$\alpha_i(t+1)$	probability
-	-	0	0	$1-r_i$
-	-	0	1	r_i
0	-	1	0	0
0	-	1	1	1
-	N_i	1	0	0
-	N_i	1	1	1
>0	$<N_i$	1	0	p_i
>0	$<N_i$	1	1	$1-p_i$

Table 2.1. Machine State Transition Probabilities

Table 2.2. Storage Level Transition Probabilities.
$\text{prob}[n_i(t+1) \mid n_{i-1}(t), \alpha_i(t+1), n_i(t), \alpha_{i+1}(t+1), n_{i+1}(t)]$

$n_{i-1}(t)$	$n_i(t)$	$n_{i+1}(t)$	$\alpha_i(t+1)$	$\alpha_{i+1}(t+1)$	$n_i(t+1)$	probability
0	0	$<N_{i+1}$	0	0	0	1
			0	1	0	1
			1	0	0	1
			1	1	0	1
0	0	N_{i+1}	0	0	0	1
			0	1	0	1
			1	0	0	1
			1	1	0	1
0	$>0,<N_i$	$<N_{i+1}$	0	0	$n_i(t)$	1
			0	1	$n_i(t)-1$	1
			1	0	$n_i(t)$	1
			1	1	$n_i(t)-1$	1
0	$>0,<N_i$	N_{i+1}	0	0	$n_i(t)$	1
			0	1	$n_i(t)$	1
			1	0	$n_i(t)$	1
			1	1	$n_i(t)$	1
0	N_i	$<N_{i+1}$	0	0	N_i	1
			0	1	N_i-1	1
			1	0	N_i	1
			1	1	N_i-1	1
0	N_i	N_{i+1}	0	0	N_i	1
			0	1	N_i	1
			1	0	N_i	1
			1	1	N_i	1

(Table 2.2 continued)

$n_{i-1}(t)$	$n_i(t)$	$n_{i+1}(t)$	$\alpha_i(t+1)$	$\alpha_{i+1}(t+1)$	$n_i(t+1)$	probability
>0	0	$<N_{i+1}$	0	0	0	1
			0	1	0	1
			1	0	1	1
			1	1	1	1
>0	0	N_{i+1}	0	0	0	1
			0	1	0	1
			1	0	1	1
			1	1	1	1
>0	$>0, <N_i$	$<N_{i+1}$	0	0	$n_i(t)$	1
			0	1	$n_i(t)-1$	1
			1	0	$n_i(t)+1$	1
			1	1	$n_i(t)$	1
>0	$>0, <N_i$	N_{i+1}	0	0	$n_i(t)$	1
			0	1	$n_i(t)$	1
			1	0	$n_i(t)+1$	1
			1	1	$n_i(t)+1$	1
>0	N_i	$<N_{i+1}$	0	0	N_i	1
			0	1	N_i-1	1
			1	0	N_i	1
			1	1	N_i-1	1
>0	N_i	N_{i+1}	0	0	N_i	1
			0	1	N_i	1
			1	0	N_i	1
			1	1	N_i	1

All other transitions have probability = 0.

and $\alpha_{i+1}(t+1)$) and the levels of the upstream and downstream storages, as well as its own level, at time t $(n_{i-1}(t), n_i(t), n_{i+1}(t))$. This follows from assumption 2.2.5.

As an example, consider the first two sets of four cases in table 2.2. Storage i is initially empty, and so is storage i-1. Since parts may not be removed from an empty storage, the level at time t+1 does not depend on the downstream portion of the line; the outcome is $n_i(t+1)=0$ whether the downstream machine is up or down, as well as whether $n_{i+1}(t)=N_{i+1}$ or not.

In the third set in table 2.2, the storage is initially neither empty nor full; again, the upstream storage is empty, so that parts may not be added to the storage whether the upstream machine is up or down. However, since the downstream storage is not full, parts may be removed if the downstream machine is up. Consequently, the level of storage i at time t+1 is equal to $n_i(t)$ if the downstream machine is down, and to $n_i(t)-1$ if it is up.

Since not all machine state and storage level transitions have non zero probabilities, it is not possible to go from every system state to every other one in one time step. Furthermore, it is impossible to reach certain states, while others may only be reached from states that are impossible to reach in the first place. A simple example of a two-machine line with storage capacity equal to 4 will serve to illustrate this; its state transition diagram appears in figure 2.4.

It is first noted that for a two machine line, the state of the system as given by equation (2.3) is

$$s \triangleq (n, \alpha_1, \alpha_2) \tag{2.17}$$

It may be seen in figure 2.4 that states (0,1,0) and (0,1,1) can be reached from no other states; at the same time, (0,0,0) can only be reached from itself and from (0,1,0). The arguement may be extended to all states drawn with a dotted line. These cannot be reached once the system leaves them: such states are termed <u>transient states</u>, and their steady state probabilities are equal to zero. For this reason, they are often referred to as impossible states in the discussions that follow. In general, it is not difficult to verify whether

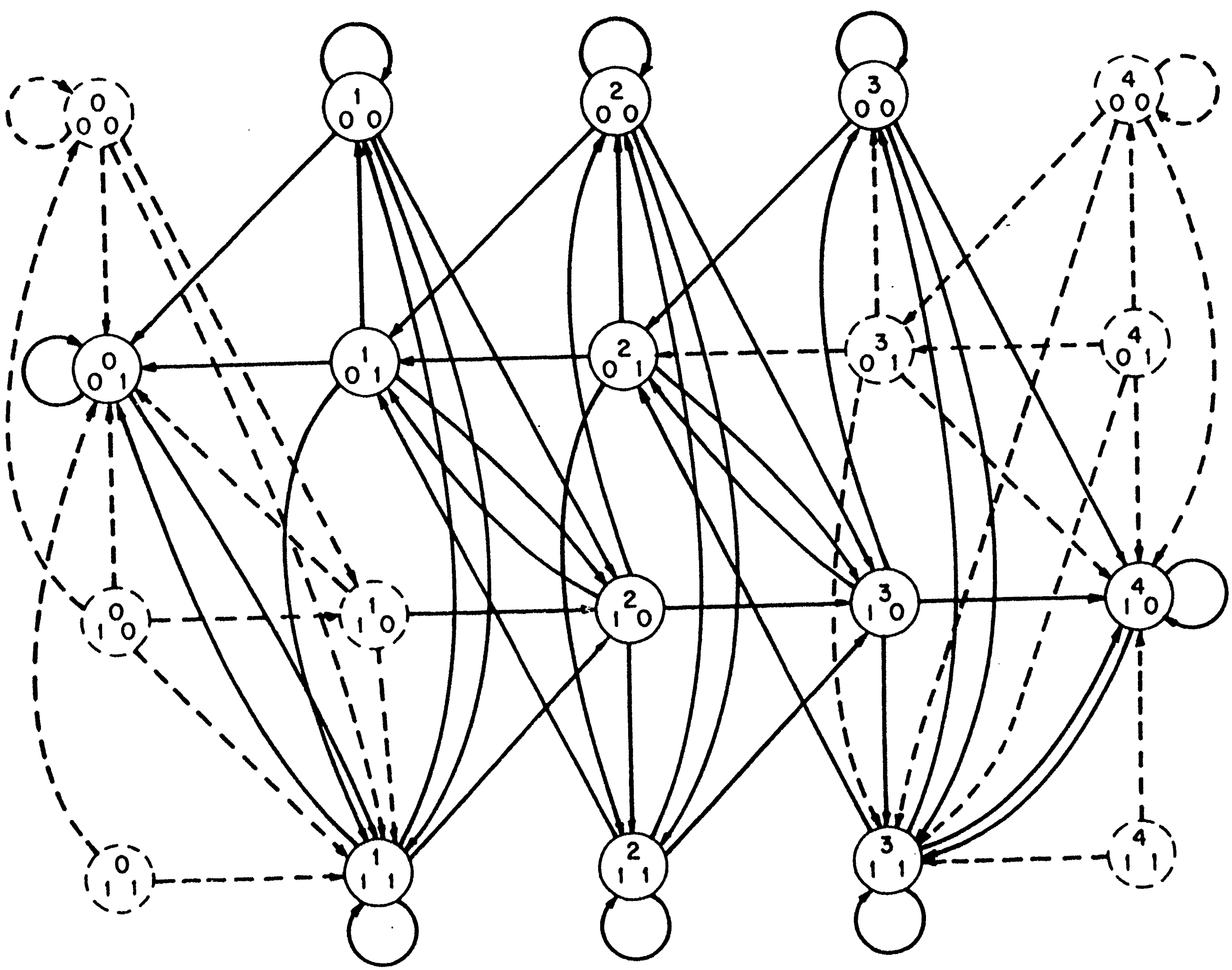


Figure 2.4. Markov State Transition Diagram for a two-machine transfer line with N=4. (Note that the states are given as $\alpha_1{}^{n}\,\alpha_2$ for clarity).

or not a state is transient. A procedure that serves this purpose for a general k-machine line appears in the FORMAC program in Appendix A.2.

It is necessary to make a distinction between two types of states before going any further. The set of boundary states contains all states in which at least one of the storages obeys one of the following two relations:

$$n_i \leq 1 \tag{2.18}$$

$$n_i \geq N_i-1 \tag{2.19}$$

It will be shown that these states must be treated separately from all other states because of differences in transition equations.

The set of internal states contains all other states, i.e. all states for which the relation

$$2 \leq n_i \leq N_i-2 \quad ; \quad i=1,..,k-1 \tag{2.20}$$

is true for every storage. The significance of this classification becomes more apparent in chapter 3.

The steady state probabilities of the transfer lines are defined, in accordance with the definition of system states in equation (2.3), as

$$p[s(t)] \triangleq p[n_1(t),..,n_{k-1}(t),\alpha_1(t),..,\alpha_k(t)] \tag{2.21}$$

The production rate of the system will be shown to be the sum of a certain set of these probabilities. Analogously, in-process inventory, forced down times and other important quantities will be derived as sums of sets of state probabilities.

Analytical and numerical methods for obtaining these probabilities are derived in chapters 3 and 4.

The number of states in a k-machine line with storage capacities $N_1,..,N_{k-1}$ is given by

$$m = 2^k (N_1+1)..(N_{k-1}+1) \tag{2.22}$$

3. DERIVATION OF ANALYTICAL METHODS

By guessing a sum-of-products solution for internal states and using the Markov model presented in chapter 2, it is possible to obtain analytical expressions for the steady-state probabilities defined in section 2.3. Analytical expressions are given for two-machine lines in Artamonov[1976], Buzacott[1967a,1967b,1969], Gershwin[1973a], and Gershwin and Schick[1977]. The approach is general in the present chapter, although only solutions for the two- and three-machine transfer lines are explained in detail.

Section 3.1 discusses the guessed solution and the transition equations for internal states. These equations are expressed in terms of failure and repair probabilities, as well as state probabilities, in section 3.1.1. A set of equations is obtained by guessing the form of the expression for internal steady-state probabilities and substituting it into transition equations, in section 3.1.2 The analysis of internal states and transition equations is perfectly general and applies to a k-machine transfer line. The specific cases of two- and three-machine lines are investigated in detail.

Boundary state transition equations are introduced in section 3.2. These are used to complete the analytical solution for the two- and three-machine cases. The two-machine line is worked out in section 3.2.1. An attempt is made to generalize the derivation to longer lines in section 3.2.2, where the three-machine case is described in detail.

3.1 Closed-Form Expressions for Internal States

3.1.1 Internal State Transition Equations

The state of the system is defined in equation (2.3) as the set of numbers

$$s(t) \triangleq (n_1(t),..,n_{k-1}(t),\alpha_1(t),..,\alpha_k(t)) \qquad (3.1)$$

For every state s(t+1), i.e. for every combination of storage levels and machine states, it is possible to write a transition equation of the form

$$p[s(t+1),t+1] = \sum_{\substack{\text{all}\\ s(t)}} p[s(t+1)|s(t)]\cdot p[s(t),t] \qquad (3.2)$$

where the first factor in the summation denotes the probability of transition from the initial state s(t) to the final state s(t+1). Equation (3.2) is completely general, and does not assume steady-state. The summation is performed over all possible initial states s(t). Modeling assumptions outlined in section 2.2 make it possible to express the transition probability as the product of machine transition probabilities and storage transition probabilities. The first factor in the summation in equation (3.2) may be written as

$$p[s(t+1)|s(t)] = P_\alpha \cdot P_n \qquad (3.3)$$

where

$$P_\alpha = \prod_{i=1}^{k} p[\alpha_i(t+1)|n_{i-1}(t),\alpha_i(t),n_i(t)] \qquad (3.4)$$

and

$$P_n = \prod_{i=1}^{k-1} p[n_i(t+1)|n_{i-1}(t),\alpha_i(t+1),n_i(t),\alpha_{i+1}(t+1),n_{i+1}(t)] \quad (3.5)$$

These conditional probabilities follow from the discussion in section 2.3 as well as tables 2.1 and 2.2. The fictitious storages $n_0(\cdot)$ and $n_k(\cdot)$ are defined so that $n_0(\cdot)$ is never empty and $n_k(\cdot)$ is never full; this is consistent with assumption 2.2.1, which states that the first machine is never starved and the last machine is never blocked.

The terms in the product in equation (3.4) are the transition probabilities of individual machine states. These appear in table 2.1. The terms in the product in equation (3.5) are either zero or one. This is because final storage states are uniquely determined by initial storage states and final machine states (See sections 2.2.5 and 2.3). Furthermore, the only possible storage transitions are those in which the levels change by at most one unit (See section 4.2.1), that is,

$$n_i(t+1) = \begin{cases} n_i(t)-1, \\ n_i(t), \text{ or} \\ n_i(t)+1 \end{cases} \quad (3.6)$$

This eliminates a large number of transitions.

<u>Internal state transition equations</u> are defined as those transition equations involving only internal states, i.e. equations in which the final state as well as all the initial states (from which there is a non-zero transition probability) in equation (3.2) are internal.

When all storages are internal, i.e. when they all have levels such that

$$2 \leq n_i \leq N_i-2 \quad ; \quad i=1,..,k-1 \quad (3.7)$$

all the operational machines can transfer parts from their upstream to their downstream storages. In other words, they are neither starved nor blocked, and thus remove a piece from the upstream storage and add one to the downstream storage. Then, the final state of storage i is given in terms of its initial level and the final states of adjacent machines by the equation

$$n_i(t+1) = n_i(t) + \alpha_i(t+1) - \alpha_{i+1}(t+1) \tag{3.8}$$

For example, equation (3.8) indicates that if the upstream machine is down and the downstream machine is up, the final level is equal to the initial level minus one.

For internal state transitions, the machine transition probabilities in table 2.1 may all be combined in a single expression as

$$\begin{aligned} p[\alpha_i(t+1) \mid n_{i-1}(t), \alpha_i(t), n_i(t)] &= \\ &\left[(1-r_i)^{1-\alpha_i(t+1)} \; r_i^{\alpha_i(t+1)} \right]^{1-\alpha_i(t)} \cdot \\ &\left[(1-p_i)^{\alpha_i(t+1)} \; p_i^{1-\alpha_i(t+1)} \right]^{\alpha_i(t)} \end{aligned} \tag{3.9}$$

Since $\alpha_i(\cdot)$ only takes the values 0 or 1, any combination of $\alpha_i(t)$ and $\alpha_i(t+1)$ results in the reduction of the right hand side of equation (3.9) to a single term. For example, if $\alpha_i(t)=0$ and $\alpha_i(t+1)=1$, the transition is one in which machine i is repaired. It may be verified that for this set of $\alpha_i(\cdot)$, the right hand side in equation (3.9) reduces to r_i.

Equation (3.4) may be rewritten as

$$\begin{aligned} P_\alpha = \prod_{i=1}^{k} &\left[(1-r_i)^{1-\alpha_i(t+1)} \; r_i^{\alpha_i(t+1)} \right]^{1-\alpha_i(t)} \cdot \\ &\left[(1-p_i)^{\alpha_i(t+1)} \; p_i^{1-\alpha_i(t+1)} \right]^{\alpha_i(t)} \end{aligned} \tag{3.10}$$

Set S is now defined to be the set of all states s(t) such that given $n_i(t+1)$, $\alpha_i(t+1)$, and $\alpha_{i+1}(t+1)$, $n_i(t)$ satisfies equation (3.8). It then follows that equation (3.2) becomes

$$p[s(t+1), t+1] = \sum_{s(t) \varepsilon S} P_\alpha \; p[s(t), t]$$

$$= \sum_{\alpha_1(t)=0}^{1} \cdots \sum_{\alpha_k(t)=0}^{1} \prod_{i=1}^{k} \left[(1-r_i)^{1-\alpha_i(t+1)} \; r_i^{\alpha_i(t+1)}\right]^{1-\alpha_i(t)}$$

$$\cdot \left[(1-p_i)^{\alpha_i(t+1)} p_i^{1-\alpha_i(t+1)}\right]^{\alpha_i(t)}$$

$$\cdot \, p[n_1(t),..,n_{k-1}(t),\alpha_1(t),..,\alpha_k(t),t] \qquad (3.11)$$

where $n_1(t),..,n_{k-1}(t)$ satisfy equation (3.8) (i.e. are completely determined by $\alpha_i(\cdot)$).

It is now necessary to guess the form of the expression for $p[\cdot]$ in order to analytically solve the problem. This is done in section 3.1.2.

3.1.2 The Sum-of-Products Solution for Internal State Probabilities

It is well known that numerous queueing theory problems result in product-form solutions. These were studied by Jackson[1963]; Gordon and Newell[1967a] obtained product-form solutions for closed queueing systems with negative exponentially distributed service times; Baskett, Chandy, Muntz and Palacios[1975] formulated a theorem applicable to certain types of networks of queues with different classes of customers, stating that the equilibrium state probabilities are given by a product of a set of terms each of which is dependent only on one state variable. Such product form solutions have also been used by numerous researchers, including Denning and Buzen[1977], Lam[1977], and Solberg[1977]. The work of these authors is concerned with flow through networks of queues, and does not deal with aspects of reliability.

For reasons which will become clear later in this chapter, it is assumed here that the steady-state (i.e. time-independent) probability distribution for <u>internal states</u> has a <u>sum-of-products form</u>:

$$p[s] = p[n_1,\ldots,n_{k-1},\alpha_1,\ldots,\alpha_k] \tag{3.12}$$

$$= \sum_{j=1}^{\ell} C_j X_{1j}^{n_1} \ldots X_{k-1,j}^{n_{k-1}} Y_{1j}^{\alpha_1} \ldots Y_{kj}^{\alpha_k} \tag{3.13}$$

where C_j, X_{ij}, and Y_{ij} are parameters to be determined.

The set of constants must satisfy an additional constraint, that the sum of all states, internal and boundary, equals one:

$$\sum_{\text{all } s} p[s] = 1 \tag{3.14}$$

An analogy is made here with differential equations boundary-value problems: in a differential equation of order n, there may be n distinct solutions. Although each of these solutions by itself satisfies the equation, only a certain linear combination of these solutions satisfies the boundary equations (See for example Boyce and DiPrima[1969]).

Suppressing for clarity the index j, <u>one</u> of the terms in this summation is substituted into equation (3.11):

$$
\begin{aligned}
&C\, X_1^{n_1(t+1)} \cdots X_{k-1}^{n_{k-1}(t+1)}\, Y_1^{\alpha_1(t+1)} \cdots Y_k^{\alpha_k(t+1)} = \\
&\sum_{\alpha_1(t)=0}^{1} \cdots \sum_{\alpha_k(t)=0}^{1} \prod_{i=1}^{k} \left[(1-r_i)^{1-\alpha_i(t+1)}\, r_i^{\alpha_i(t+1)}\right]^{1-\alpha_i(t)} \\
&\cdot \left[(1-p_i)^{\alpha_i(t+1)}\, p_i^{1-\alpha_i(t+1)}\right]^{\alpha_i(t)} \cdot C\, X_1^{n_1(t)} \cdots X_{k-1}^{n_{k-1}(t)} \\
&\cdot Y_1^{\alpha_1(t)} \cdots Y_k^{\alpha_k(t)} \qquad (3.15)
\end{aligned}
$$

Using equation (3.8) and cancelling like terms on both sides, this gives:

$$
\begin{aligned}
&X_1^{\alpha_1(t+1)-\alpha_2(t+1)} \cdots X_{k-1}^{\alpha_{k-1}(t+1)-\alpha_k(t+1)}\, Y_1^{\alpha_1(t+1)} \cdots Y_k^{\alpha_k(t+1)} = \\
&\sum_{\alpha_1(t)=0}^{1} \cdots \sum_{\alpha_k(t)=0}^{1} \prod_{i=1}^{k} \left[(1-r_i)^{1-\alpha_i(t+1)}\, r_i^{\alpha_i(t+1)}\right]^{1-\alpha_i(t)} \\
&\cdot \left[(1-p_i)^{\alpha_i(t+1)}\, p_i^{1-\alpha_i(t+1)}\, Y_i\right]^{\alpha_i(t)} \qquad (3.16)
\end{aligned}
$$

or, readjusting the exponent of the first parantheses in the right hand side of equation (3.16),

$$
\begin{aligned}
&\prod_{i=1}^{k} \frac{X_i^{\alpha_i(t+1)-\alpha_{i+1}(t+1)}\, Y_i^{\alpha_i(t+1)}}{(1-r_i)^{1-\alpha_i(t+1)}\, r_i^{\alpha_i(t+1)}} = \\
&\sum_{\alpha_1(t)=0}^{1} \cdots \sum_{\alpha_k(t)=0}^{1} \prod_{i=1}^{k} \left[\frac{(1-p_i)^{\alpha_i(t+1)}\, p_i^{1-\alpha_i(t+1)}\, Y_i}{(1-r_i)^{1-\alpha_i(t+1)}\, r_i^{\alpha_i(t+1)}}\right]^{\alpha_i(t)}
\end{aligned}
\qquad (3.17)
$$

where for convenience, $X_k \triangleq 1$. Note that $\alpha_i(t)$ only occurs as an exponent in the right hand side of equation (3.17); furthermore, $\alpha_i(\cdot)$ only takes the values 0 and 1. The right hand side of (3.17) can be rewritten as

$$\prod_{i=1}^{k}\left[1 + \frac{[(1-p_i)^{\alpha_i(t+1)}\, p_i^{1-\alpha_i(t+1)}\, \dot{Y}_i}{[(1-r_i)^{1-\alpha_i(t+1)}\, r_i^{\alpha_i(t+1)}]}\right] \tag{3.18}$$

This reformulation is not obvious and requires a proof. Proceeding by induction, it is easy to see that the right hand side of equation (3.17) equals (3.18) for k=1. Assuming that the equality holds for k, it is shown that the equality holds for k+1 as follows (for simplicity, the term in the product in the right hand side of (3.17) is referred to as $A_i^{\alpha_i}$):

$$\sum_{\alpha_1=0}^{1}\cdots\sum_{\alpha_k=0}^{1}\ \sum_{\alpha_{k+1}=0}^{1}\ \prod_{i=1}^{k+1} A_i^{\alpha_i} =$$

$$\sum_{\alpha_1=0}^{1}\cdots\sum_{\alpha_k=0}^{1}\ \prod_{i=1}^{k} A_i^{\alpha_i}\cdot 1 + \sum_{\alpha_1=0}^{1}\cdots\sum_{\alpha_k=0}^{1}\ \prod_{i=1}^{k} A_i^{\alpha_i}\cdot A_{k+1}$$

$$= \prod_{i=1}^{k}(1 + A_i)\ \cdot\ (1 + A_{k+1}) \tag{3.19}$$

Equation (3.19) completes the proof. When (3.18) is substituted into equation (3.17), the argument t (though not t+1) vanishes. Multiplying both sides by the denominator in (3.18), it follows that:

$$\prod_{i=1}^{k} X_i^{\alpha_i(t+1)-\alpha_{i+1}(t+1)}\ Y_i^{\alpha_i(t+1)} =$$

$$\prod_{i=1}^{k}[(1-r_i)^{1-\alpha_i(t+1)}\ r_i^{\alpha_i(t+1)} + (1-p_i)^{\alpha_i(t+1)}\ p_i^{1-\alpha_i(t+1)}\ Y_i] \tag{3.20}$$

Equation (3.20) has been derived with no conditions on $\alpha_i(t+1)$; thus, it must hold for all values of $\alpha_i(t+1)$. In particular, if $\alpha_i(t+1) = 0$, for $i=1,..,k$, then (3.20) reduces to:

$$1 = \prod_{i=1}^{k} [(1-r_i) + p_i Y_i] \tag{3.21}$$

if $\alpha_j(t+1) = 1$, and $\alpha_i(t+1) = 0$ for $i=1,..,k$, $i \neq j$, then (3.20) becomes

$$X_{j-1}^{-1} X_j Y_j = \prod_{\substack{i=1 \\ i\neq j}}^{k} [(1-r_i) + p_i Y_i] \cdot [r_j + (1-p_j) Y_j] \tag{3.22}$$

where for convenience, $X_0 \triangleq 1$. Using equation (3.21) on the right hand side of (3.22), the equation can be reduced to

$$\frac{X_j Y_j}{X_{j-1}} = \left[\frac{r_j + (1-p_j) Y_j}{(1-r_j) + p_j Y_j} \right] \quad ; \quad j=1,..,k \tag{3.23}$$

Any other sets of values for $\alpha_i(t+1)$ in equation (3.20) give equations that may readily be derived from (3.21) and (3.23). Since $X_0 \triangleq X_k \triangleq 1$, there are $k+1$ equations in $2k-1$ unknowns. For $k>2$, this implies that there are more unknowns than equations. Furthermore, the weighting and normalizing constants C_j remain to be computed.

Two cases are now analyzed: when $k=2$, there are three equations in three unknowns and the system of equations given by (3.21) and (3.23) can be solved analytically. When $k \geq 3$, a numerical approach is needed to obtain the terms in equation (3.13). Furthermore, there are more unknowns than equations, so that additional information must be found.

In the two-machine case, $k=2$. Equations (3.21) and (3.23) may be solved to give X_{ij} and Y_{ij}. Since these equations are non-linear, they allow multiple solutions. It may be verified that there are two sets of solutions in this case. These are:

$$\left.\begin{aligned} X_{11} &= 1 \\ Y_{i1} &= \frac{r_i}{p_i} \quad ; \quad i=1,2 \end{aligned}\right\} \tag{3.24}$$

$$\left.\begin{aligned} X_{12} &= Y_{22} \,/\, Y_{12} \\ Y_{12} &= \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 + p_2 - p_1 p_2 - p_1 r_2} \\ Y_{22} &= \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - p_2 r_1} \end{aligned}\right\} \tag{3.25}$$

The constants C_j in equation (3.13) are found in section 3.2 by using boundary equations, as well as (3.14).

In the three-machine case, there are only four equations in five unknowns. The solution is therefore not uniquely determined. Furthermore, since the simultaneous equations (3.21) and (3.23) are non-linear, there is the possibility of multiple solutions.

For any set of $\{X_{1j},..,X_{k-1,j},Y_{1j},..,Y_{kj}\}$, there is a set of constants C_j such that equation (3.13) holds. The set of constants is found by analyzing the boundary equations,as discussed in section 3.2.

3.2 The Boundary State Transition Equations

3.2.1 The Two-Machine, One-Storage Line

Internal states and transition equations are analyzed in section 3.1. To complete the problem, it is necessary to study boundary states and transition equations. In section 2.3.2, boundary states are defined as states in which at least one storage level satisfies one of the following two relations:

$$n_i \leq 1 \tag{3.26}$$

$$n_i \geq N_i - 1 \tag{3.27}$$

Boundary state transition equations are defined to be state transition equations in which at least one state (whether initial or final) is a boundary state.

The number of boundary state transition equations increases rapidly with the number of machines in the line, and with storage capacities. In the simplest case of a two-machine line, however, these are not a function of storage size, and are easy to list.

Neglecting transient (zero steady-state probability) states, the lower boundary (n=0 or 1) state transition equations are the following:

$$\begin{aligned} p[0,0,1] = {} & (1-r_1)\, p[0,0,1] + (1-r_1)r_2\, p[1,0,0] \\ & + (1-r_1)(1-p_2)\, p[1,0,1] + p_1(1-p_2)\, p[1,1,1] \end{aligned} \tag{3.28}$$

$$\begin{aligned} p[1,0,0] = {} & (1-r_1)(1-r_2)\, p[1,0,0] + (1-r_1)p_2\, p[1,0,1] \\ & + p_1 p_2\, p[1,1,1] \end{aligned} \tag{3.29}$$

$$p[1,0,1] = (1-r_1)r_2\ p[2,0,0] + (1-r_1)(1-p_2)\ p[2,0,1] + p_1 r_2\ p[2,1,0] + p_1(1-p_2)\ p[2,1,1] \quad (3.30)$$

$$p[1,1,1] = r_1\ p[0,0,1] + r_1 r_2\ p[1,0,0] + r_1(1-p_2)\ p[1,0,1] + (1-p_1)(1-p_2)\ p[1,1,1] \quad (3.31)$$

$$p[2,1,0] = r_1(1-r_2)\ p[1,0,0] + r_1 p_2\ p[1,0,1] + (1-p_1)\ p_2\ p[1,1,1] \quad (3.32)$$

Using the state transition diagram for the two-machine case (figure 2.4), it may be verified that these are the only possible transitions involving lower boundary states. These equations are now analyzed.

For the general k-machine line, boundary state probabilities are expressed as a sum of terms, analogous to the sum-of-products for internal state probabilities in equation (3.13):

$$p[s] = \sum_{j=1}^{\ell} C_j \quad \xi[s, X_{1j}, \ldots, X_{k-1,j}, Y_{1j}, \ldots, Y_{kj}] \quad (3.33)$$

It is noted that equation (3.33) applies to all states, and takes the form of internal state probability expressions when

$$\xi[(n_1, \ldots, n_{k-1}, \alpha_1, \ldots, \alpha_k), X_{1j}, \ldots, X_{k-1,j}, Y_{1j}, \ldots, Y_{kj}] = X_{1j}^{n_1} \ldots X_{k-1,j}^{n_{k-1}}\ Y_{i1}^{\alpha_1} \ldots Y_{kj}^{\alpha_k} \quad (3.34)$$

The analogy with boundary-value differential equations problems is carried over to the analysis of boundary state transition equations. Thus, as in section 3.1.2, only one of the terms in the summation in equation (3.33) is considered. The notation

$$U_j \triangleq \{ X_{1j}, \ldots, X_{k-1,j}, Y_{1j}, \ldots, Y_{kj} \} \quad (3.35)$$

is introduced.

The two-machine boundary state transition equations are studied: Noting that all the right hand side terms in equation (3.30) are internal, it is rewritten as

$$\begin{aligned}\xi[(1,0,1),U_j] &= (1-r_1)r_2 X_{1j}^2 + (1-r_1)(1-p_2) X_{1j}^2 \\ &\quad + p_1 r_2 X_{1j}^2 + p_1(1-p_2) X_{1j}^2 Y_{1j} Y_{2j} \\ &= X_{1j}^2 [(1-r_1) + p_1 Y_{1j}] [r_2 + (1-p_2) Y_{2j}] \qquad (3.36)\end{aligned}$$

Equation (3.23) is used to simplify (3.36). For j=2, noting that $X_{2j} \triangleq 1$, the rightmost term in equation (3.36) is rewritten as

$$[(1-r_2) + p_2 Y_{2j}] \, Y_{2j} \; / \; X_{1j} \qquad (3.37)$$

Substituting (3.37) into equation (3.36), and using (3.21), it follows that

$$\xi[(1,0,1),U_j] = X_{1j} Y_{2j} \qquad (3.38)$$

It can be verified that in general, any state which can only be reached from internal states has the internal (product) form.

Equation (3.38) is substituted into (3.33), giving

$$p[1,0,1] = \sum_{j=1}^{2} C_j X_{1j} Y_{2j} \qquad (3.39)$$

State (2,1,0) is internal. Thus, it has a probability of the form given by equation (3.13):

$$p[2,1,0] = \sum_{j=1}^{2} C_j X_{1j}^2 Y_{1j} \qquad (3.40)$$

The parameters X_{ij} and Y_{ij} are given by equations (3.24) and (3.25). Equations (3.39) and (3.40) are substituted into equations (3.28), (3.29), (3.31), and (3.32), and these four equations are summed up. The coefficients of C_2 cancel eachother out, and the equation reduces to:

$$C_1 \left(\frac{r_1}{p_1} - \frac{r_2}{p_2} \right) = 0 \tag{3.41}$$

Whenever the two machines do not have equal efficiencies, the term in the parantheses is not zero and $C_1=0$. If the two machines have equal efficiencies, it is easy to see that equations (3.24) and (3.25) are identical, i.e. equations (3.21) and (3.23) have one second-order root given by either of (3.24) or (3.25). In this case, it is not necessary to have two terms in the summation in equation (3.33), since the terms are identical. It is possible to set one of the C_j to be zero. The constant C_1 is arbitrarily set equal to zero when the two machines have equal efficiencies. Since $C_1=0$ when they do not, it follows that the steady-state probabilities for a two-machine line have only one term in the summation in equation (3.33).

From equation (3.32), it follows that:

$$\begin{aligned} \xi[(1,1,1),U_j] &= X_{1j}^2 Y_{1j} - r_1 p_2 X_{1j} Y_{2j} \\ &\quad - r_1(1-r_2)\, \xi[(1,0,0),U_j] \end{aligned} \tag{3.42}$$

Equation (3.42) is substituted into equation (3.29), giving, after some simplification and use of equations (3.21) and (3.23),

$$\xi[(1,0,0),U_j] = X_{1j} \tag{3.43}$$

The probability of state (1,0,0) is also seen to have the internal form. Equation (3.43) is now substituted into (3.42), and yields, after using equations (3.21) and (3.23),

$$\xi[(1,1,1),U_j] = \frac{X_{1j}}{p_2}[r_2 + (1-p_2)Y_{2j}] \tag{3.44}$$

Equation (3.44) shows that state (1,1,1) does not have a steady-state probability with an expression of the internal form. Two equations are left, (3.28) and (3.31). These are consistent, and substituting equations (3.38), (3.43) and (3.44) into either of these two equations give the expression

$$\xi[(0,0,1),U_j] = X_{1j}\frac{r_1 + r_2 - r_1r_2 - p_2r_1}{p_2r_1} \tag{3.45}$$

The same reasoning is applied to the upper boundary (n=N-1 or N) state transition equations. These are the following:

$$\begin{aligned} p[N-2,0,1] &= (1-r_1)r_2\, p[N-1,0,0] + p_1r_2\, p[N-1,1,0] \\ &\quad + p_1(1-p_2)\, p[N-1,1,1] \end{aligned} \tag{3.46}$$

$$\begin{aligned} p[N-1,0,0] &= (1-r_1)(1-r_2)\, P[N-1,0,0] + p_1(1-r_2)\, p[N-1,1,0] \\ &\quad + p_1p_2\, p[N-1,1,1] \end{aligned} \tag{3.47}$$

$$\begin{aligned} p[N-1,1,0] &= r_1(1-r_2)\, p[N-2,0,0] + r_1p_2\, p[N-2,0,1] \\ &\quad + (1-p_1)(1-r_2)\, p[N-2,1,0] + (1-p_1)p_2\, p[N-2,1,1] \end{aligned} \tag{3.48}$$

$$\begin{aligned} p[N-1,1,1] &= r_1r_2\, p[N-1,0,0] + (1-p_1)r_2\, p[N-1,1,0] \\ &\quad + (1-p_1)(1-p_2)\, p[N-1,1,1] + r_2\, p[N,1,0] \end{aligned} \tag{3.49}$$

$$\begin{aligned} p[N,1,0] &= r_1(1-r_2)\, p[N-1,0,0] + (1-p_1)(1-r_2)\, p[N-1,1,0] \\ &\quad + (1-p_1)p_2\, p[N-1,1,1] + (1-r_2)\, p[N,1,0] \end{aligned} \tag{3.50}$$

Here, it is noted that all states with storage level n=N-2 are internal. These equations are solved as before. Again, it is found that the equations can be manipulated to obtain equation (3.41), so that C_1 may again be set equal to zero. Furthermore, the additional equations are found to be consistent, giving a unique set of probability expressions.

The complete set of steady-state probabilities for the two-machine transfer line is summarized in table 3.1.

A certain amount of symmetry is visible in these results: for example, the expressions for p[0,0,1] and p[N,1,0] have similar forms; so do p[1,1,1] and p[N-1,1,1], and other pairs. Such considerations give some insight into the derivation of analogous expressions for the three-machine case in section 3.2.2.

A computer program designed to evaluate the steady-state probabilities and some performance measures (See chapter 5) of the two-machine line appears in Appendix A.1.

Table 3.1. Steady-state probabilities of two-machine line.

$$p[0,0,0] = 0$$

$$p[0,0,1] = CX \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2 r_1}$$

$$p[0,1,0] = 0$$

$$p[0,1,1] = 0$$

$$p[1,0,0] = CX$$

$$p[1,0,1] = CXY_2$$

$$p[1,1,0] = 0$$

$$p[1,1,1] = \frac{CX}{p_2} \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 + p_2 - p_1 p_2 - p_2 r_1}$$

$$p[n,\alpha_1,\alpha_2] = CX^n Y_1^{\alpha_1} Y_2^{\alpha_2} \quad ; \quad 2 \leq n \leq N-2$$

$$p[N-1,0,0] = CX^{N-1}$$

$$p[N-1,0,1] = 0$$

$$p[N-1,1,0] = CX^{N-1} Y_1$$

$$p[N-1,1,1] = \frac{CX^{N-1}}{p_1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - p_1 r_2}$$

$$p[N,0,0] = 0$$

$$p[N,0,1] = 0$$

$$p[N,1,0] = CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2}$$

$$p(N,1,1] = 0$$

$X \overset{\Delta}{=} X_1$, Y_1, and Y_2 are given by equation (3.25); C satisfies (3.14)

3.2.2 Longer Transfer Lines

The three-machine, two-storage case is complex enough to make the manual generation of the boundary state transition equations almost intractable. A program was therefore written in the IBM FORMAC Symbolic Mathematics Interpreter language (See Tobey[1969], Trufyn[n.d.]), a superset and extension of PL/I. This program generates the boundary state transition equations for a general k-machine line with given buffer capacities algebraically, i.e. in mathematical symbols rather than numerically. The program listing, as well as a sample output for the lower boundary of a three-machine line with buffer capacities $N_1=N_2=10$, appear in Appendix A.2.

The boundary state transition equations constitute a very large system of linear equations. With considerable work, as well as insight given by the quasi-symmetry of the two-machine results, this system can be solved to give closed-form expressions for the $\xi[\cdot]$ defined in equation (3.33). The procedure is considerably more complex than the solution of the two-machine case presented in section 3.2.1, and involves a great deal of algebraic manipulations.

In the three-machine case, boundary states are subdivided into two classes. Corner states are those in which both storages have boundary levels; edge states are those in which only one of the two storages has a boundary level. It is found that there is a simple relationship between certain edge states: for states with the same machine status configuration (i.e. the same α_i ; i=1,2,3), incrementing the internal storage level n_i corresponds to multiplying $\xi[\cdot]$ by X_{ij}. Thus, for example,

$$\xi[(1,n_2+1,0,0,1),U_j] = X_{2j}\ \xi[(1,n_2,0,0,1),U_j] \tag{3.51}$$

where both n_2 and n_2+1 are internal. Consequently, the number of expressions that need to be derived does not increase with storage size.

The complete derivation is lengthy and is not reproduced here. A sample of the expressions for the lower boundary corner and edge states is given in table 3.2. A more complete account of the derivation appears in Gershwin and Schick[1978].

The crucial fact about these $\xi[\cdot]$ expressions is that, though they all satisfy subsets of the boundary state transition equations, they do not all satisfy all these equations. Only the appropriate linear combination of these solutions satisfies all the transition equations. The procedure followed to obtain this linear combination is outlined below.

For the Markov chain described in section 2.3,

$$\underline{p} = T\,\underline{p} \tag{3.52}$$

or

$$(T - I)\,\underline{p} = \underline{0} \tag{3.53}$$

where I denotes the identity matrix. Following equation (3.33), the probability vector $\underline{p}$ may be rewritten as

$$\underline{p} = \sum_{j=1}^{\ell} c_j \; \underline{\xi}[U_j] \tag{3.54}$$

where

$$\underline{\xi}[U_j] \triangleq \begin{bmatrix} \xi[s_1,U_j] \\ \xi[s_2,U_j] \\ \vdots \\ \xi[s_m,U_j] \end{bmatrix} \tag{3.55}$$

The number of states, m, is given by equation (2.22) as

$$m = 2^k\,(N_1+1)\,\ldots\,(N_{k-1}+1) \tag{3.56}$$

for a k-machine transfer line with storage capacities $N_1,\ldots,N_{k-1}$.

Thus, equation (3.53) becomes:

$$(T - I) \sum_{j=1}^{\ell} c_j \, \underline{\xi}[U_j] = \underline{0} \tag{3.57}$$

Table 3.2. Some boundary state probability expressions for a three-machine transfer line.

Edge states (n_2 internal):

$$\xi[1,n_2,0,0,0] = X_1 X_2^{n_2}$$

$$\xi[1,n_2,0,0,1] = X_1 X_2^{n_2} Y_3$$

$$\xi[1,n_2,0,1,0] = X_1 X_2^{n_2} Y_2$$

$$\xi[1,n_2,0,1,1] = X_1 X_2^{n_2} Y_2 Y_3$$

$$\xi[1,n_2,1,0,0] = 0$$

$$\xi[1,n_2,1,0,1] = 0$$

$$\xi[1,n_2,1,1,0] = X_1 X_2^{n_2} Y_1 (1 - r_2 + p_2 Y_2) / p_2$$

$$\xi[1,n_2,1,1,1] = X_1 X_2^{n_2} Y_1 Y_3 (1 - r_2 + p_2 Y_2) / p_2$$

Edge states (n_1 internal):

$$\xi[n_1,1,0,0,0] = X_1^{n_1} X_2$$

$$\xi[n_1,1,0,0,1] = X_1^{n_1} X_2 Y_3$$

$$\xi[n_1,1,0,1,0] = 0$$

(Table 3.2 continued)

$$\xi[n_1,1,0,1,1] = X_1^{n_1} X_2 Y_2 (1 - r_3 + p_3 Y_3) / p_3$$

$$\xi[n_1,1,1,0,0] = X_1^{n_1} X_2 Y_1$$

$$\xi[n_1,1,1,0,1] = X_1^{n_1} X_2 Y_1 Y_3$$

$$\xi[n_1,1,1,1,0] = 0$$

$$\xi[n_1,1,1,1,1] = X_1^{n_1} X_2 Y_1 Y_2 (1 - r_3 + p_3 Y_3) / p_3$$

Corner states:

$$\xi[0,0,0,1,1] = \frac{1 - r_1}{r_1^2 p_3} (r_1 + r_3 - r_1 r_3 - p_3 r_1) X_1 X_2 Y_1 Y_2$$

$$\xi[0,1,0,1,0] = \frac{1 - r_1}{r_1} X_1 X_2 Y_1 Y_2$$

$$\xi[0,1,0,1,1] = \frac{r_1 + r_3 - r_1 r_3}{p_3 r_1} X_1 X_2 Y_1 Y_2$$

$$\xi[1,0,0,0,1] = \frac{X_1 X_2}{p_3(r_1 + r_2 - r_1 r_2)} \left[\frac{p_1 p_2 r_3}{r_1} Y_1 Y_2 + (1 - p_3) - (1 - p_3 - r_3)(1 - r_1)(1 - r_2) \right]$$

$$\xi[1,0,1,1,1] = \frac{r_1 + r_3 - r_1 r_3 - p_3 r_1}{r_1 p_3} X_1 X_2 Y_1 Y_2$$

(Table 3.2 continued)

$$\xi[1,1,0,0,0] = X_1X_2$$

$$\xi[1,1,0,0,1] = X_1X_2Y_3$$

$$\xi[1,1,0,1,0] = 0$$

$$\xi[1,1,0,1,1] = X_1X_2Y_2\ (1 - r_3 + p_3Y_3)\ /\ p_3$$

$$\xi[1,1,1,0,0] = 0$$

$$\xi[1,1,1,0,1] = 0$$

$$\xi[1,1,1,1,0] = X_1X_2Y_1Y_2$$

$$\xi[1,1,1,1,1] = \frac{X_1X_2Y_1}{p_2p_3}\left[(1 - r_2)(1 - r_3) + (1 - r_2 + p_2Y_2)\ p_3Y_3\right]$$

(A complete list appears in Gershwin and Schick[1978].)

or

$$\sum_{j=1}^{\ell} C_j \quad (T - I)\ \underline{\xi}[U_j] \quad = \quad \underline{0} \qquad (3.58)$$

Defining the vector $\underline{C}$ as

$$\underline{C} \ \triangleq \ \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_\ell \end{bmatrix} \qquad (3.59)$$

and the matrix Ξ as

$$\Xi \ \triangleq \ \left[\underline{\xi}[U_1] \quad \underline{\xi}[U_2] \quad \cdots\cdot \quad \underline{\xi}[U_\ell] \right] \qquad (3.60)$$

equation (3.58) is rewritten as

$$(T - I)\ \Xi\ \underline{C} \quad = \quad \underline{0} \qquad (3.61)$$

For a given set $\{U_1, U_2, .., U_\ell\}$, where all U_j are <u>distinct</u> and satisfy the <u>internal equations</u> (3.21) and (3.23), the system of equations in (3.61) has a unique solution $\underline{C}$ if and only if the matrix $(T-I)\Xi$ has rank equal to ℓ-1. Thus, only ℓ, the number of terms in the summation in equation (3.33), remains to be determined.

Because the expressions $\xi[\cdot]$ satisfy most transition equations, most components of the product vector

$$(T - I)\ \underline{\xi}[U_j] \qquad (3.62)$$

are identically equal to zero for any U_j that satisfies equations (3.21) and (3.23). For example, in the three-machine case with storage capacities N_1=N_2=10, 898 components out of the 968-vector are identically zero. Thus, most of the rows in the matrix equation (3.61) are automatically satisfied, regardless of $\underline{C}$. If ℓ is taken to be the number of rows not automatically satisfied, then the system of equations (3.61) has a unique solution $\underline{C}$, once a set of ℓ distinct U_j is chosen. The system in

(3.61) can be reduced by computing only those ℓ rows of $(T-I)\Xi$ that are not satisfied identically. The new reduced order system can be written as

$$\Gamma \underline{C} = \underline{0} \tag{3.63}$$

where Γ consists of the non-zero rows of $(T-I)\Xi$. This is an $\ell \times \ell$ rather than an $m \times m$ system, and thus, the computational work needed to solve for $\underline{C}$ is drastically reduced. In addition, ℓ as a function of storage capacities increases much more slowly than m. The complexity of the problem thus remains tractable even for very large storages.

Unfortunately, serious numerical problems arise in the solution of equation (3.63). Although Γ has rank $\ell-1$, it appears to the computer to have much lower rank because of these numerical problems.

This difficulty is overcome by using singular value decomposition techniques (Golub[1969], Golub and Kahan[1965]). The least squares solution of equation (3.63) is then the weighting (though not yet normalized) constants in the summation in equation (3.33). This procedure is described in detail in Gershwin and Schick[1978].

It only remains to normalize the values obtained by equation (3.33) so that equation (3.14) is satisfied. This is achieved by summing up all the state expressions and dividing each expression by the sum. This may cause substantial round-off errors in systems with very large storages (and hence large state-spaces).

The analytic solution for the three-machine transfer line is now complete. The main difficulty in extending the above results to longer transfer lines lies in the derivation of the $\xi[\cdot]$ expressions. General forms for these expressions for k-machine lines have not yet been obtained; thus, even larger sets of boundary equations may have to be solved in order to obtain the steady-state probabilities of longer lines.

4. NUMERICAL METHODS FOR EXACT SOLUTIONS

While analytical solutions often have the advantage of being compact and easy to implement, they are hard to derive; furthermore, they depend strongly on modeling assumptions in such a way that they offer little flexibility for relaxing or modifying such assumptions. It is thus often useful to use numerical approaches to solve the problems; while these may not necessarily be as compact as analytical solutions, they do offer greater flexibility.

An iterative multiplication scheme known as the power method is introduced in section 4.1. Computational problems caused by the convergence of this algorithm are stressed, and possible improvements are suggested.

An algorithm which exploits the sparsity and block tri-diagonal structure of the transition matrix is developed in section 4.2. Because a large proportion of storage level transitions have zero probability, the transition matrix T is extremely sparse. Furthermore, if the system states are listed in the appropriate order, the matrix has a useful and interesting nested block tri-diagonal structure. These properties are used in solving the large set of transition equations. The motivation for developing such an algorithm, as well as the structural properties of the transition matrix, are discussed in section 4.2.1. The algorithm is derived in section 4.2.2. The flexibility and usefulness of this approach and the computer memory and programming problems involved are discussed in section 4.2.3.

4.1 The Power Method

The property of ergodicity is defined in section 2.3.1 as follows: Given the transition matrix T, and setting $\Phi(t) \triangleq T^t$, a process is ergodic if and only if

$$\lim_{t\to\infty} \Phi(t) = \Phi \tag{4.1}$$

exists, and the value of

$$\underline{p} \triangleq \Phi \; \underline{p}(0) \tag{4.2}$$

is independent of $\underline{p}(0)$, provided that $\sum_i p_i(0) = 1$.

A closed class is defined as a set of states C such that no state outside C can be reached from any state inside C. Two states communicate if each can be reached from the other. A closed communicating class is a closed class in which all pairs of states communicate. A final class is one that includes no transient states.

A process is periodic if a state can be reached from itself in d, 2d, 3d, ..., nd, ... trials. If d=1 only, the process is termed aperiodic. The existence of a self-loop (a transition such that $t_{ii} \neq 0$ for some i) on at least one state in a final class is sufficient for its aperiodicity.

The Markov chain model of a transfer line described in chapter 2 contains only one final closed communicating class; furthermore, several states in that class contain self-loops. These conditions are sufficient for ergodicity. Thus, for an arbitrary initial probability vector $\underline{p}(0)$, the steady-state probability vector $\underline{p}$ may be computed from equation (4.2). Since Φ is not known, equation (4.2) is rewritten as follows:

$$\lim_{t\to\infty} T^t \underline{p}(0) = \underline{p} \tag{4.3}$$

Equation (4.3) suggests a "brute force" method for obtaining the steady-state probability vector $\underline{p}$. This method consists in an iterative multiplication scheme

$$\underline{p}(t+1) = T\,\underline{p}(t) \tag{4.4}$$

with a given $\underline{p}(0)$.

Convergence criteria such as

$$\left.\begin{array}{ll} & \|\ \underline{p}(t+1)-\underline{p}(t)\ \| < \varepsilon \\ \text{or} & \\ & \max\limits_{i}\ |\ p_i(t+1)-p_i(t)\ | < \varepsilon \end{array}\right\} \tag{4.5}$$

may be used to decide when a vector has been obtained that is sufficiently close to the steady-state probability vector.

In devising a computer implementation of the iterative multiplication algorithm, it is necessary to take advantage of the sparsity (See section 4.2.1) of the transition matrix. This is not only desirable, it is imperative in view of the large dimensions of the matrix (See table 6.1).

A sparse matrix need not be stored in full. Rather, its nonzero elements and their coordinates are stored, making it possible to represent extremely large sparse matrices with relatively small arrays (Tewarson[1973]). For example, given that a certain element t_{ij} of the sparse matrix T is nonzero, it is sufficient to store t_{ij}, i, and j. The full matrix may be reconstituted from this information. Thus, it is only necessary to store $\underline{p}(t)$ and $\underline{p}(t+1)$, in addition to the arrays giving the nonzero elements of the transition matrix, while implementing the power method algorithm.

The major limitation of this algorithm is the computation necessary for the convergence of $\underline{p}(t) \rightarrow \underline{p}$. Although the properties of the ergodic Markov chain outlined above guarantee that the vector converges to the steady-state distribution, the number of iterations required to satisfy convergence criteria such as those in equation (4.5) may get very large as the number of machines in the line or the capacity of the storages increase, thereby increasing the dimension of the vector. Some results of runs for a three-machine line with the computer program given in Appendix A.3 are presented in figure 4.1. (It must be noted that these results do not correspond exactly to the iterative procedure given in equation (4.4): some improvements were made, as described below. Still, these results can give an idea of the way in which computation increases as storage size increases.) The machine parameters are the same as those given in table 6.2. The first storage has capacity $N_1=5$, and the second storage capacity is varied.

The rate at which this algorithm converges depends most strongly on two factors. These are the accuracy of the initial guess, and the second largest eigenvalue of the transition matrix (the largest eigenvalue is always 1). The latter factor is dependent on the system parameters, such as failure and repair probabilities and storage capacities. Furthermore, the computation of the eigenvalues of a matrix as large as T is far from trivial. Thus, there is no control over the eigenvalues, and even evaluating them in order to estimate how fast the algorithm converges is a difficult problem.

The initial guess, however, can be improved significantly, by making certain observations.

(i) The transient (zero steady-state probability) states are easy to predict (Section 2.3.2). Thus, it is possible to set at least some of the states equal to their final values.

(ii) The steady-state probabilities can be subdivided into three classes according to their orders of magnitude. In ascending order, these are the internal states, the edge boundary states, and the corner boundary states. For example, in the case of a three-machine line with system parameters given in table 6.2 (For the probability distribution, see the sample output in Appendix A.4), these orders of magnitude are 10^{-3}, 10^{-2}, and 10^{-1}, respectively. Thus, it is possible to predict the relative magnitudes of the final values (See Gershwin and Schick[1978]).

(iii) By the δ-transformation techniques outlined in section 6.3, it is possible to solve a smaller problem first (i.e. a problem with smaller storage sizes). The results of the smaller problem may then be used, by also taking the order of magnitude considerations into account, to set up

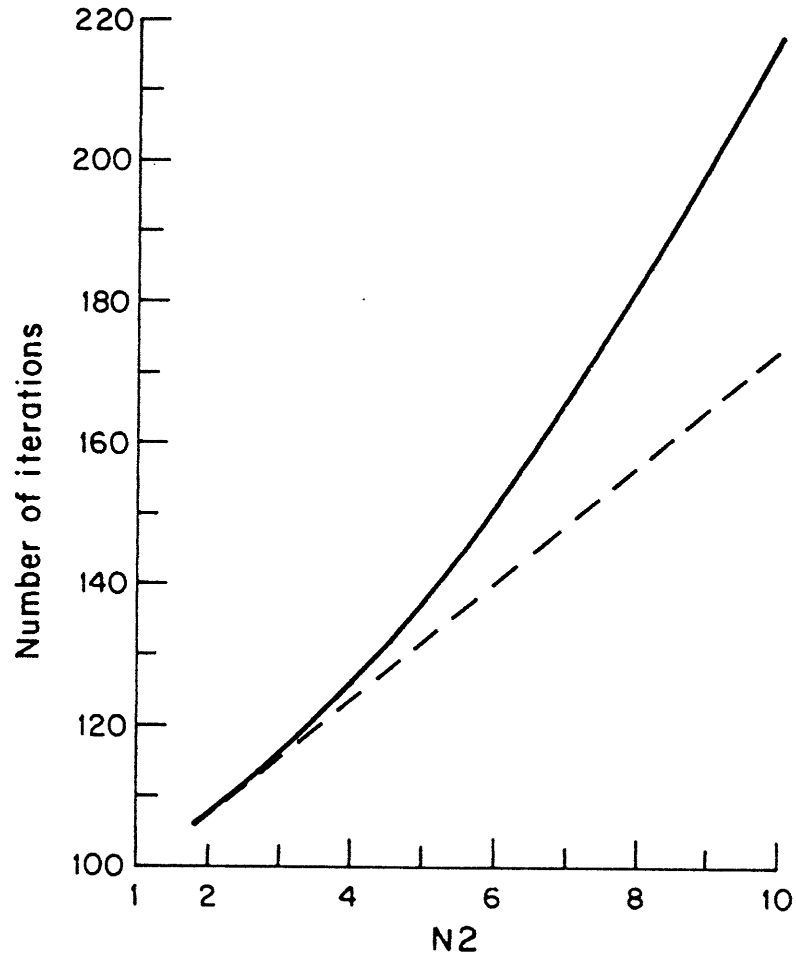


Figure 4.1. The number of iterations in which the computer program in Appendix A.3 converges for $N_1=5$, $\varepsilon=10^{-5}$, p_i and r_i given in sample output, Appendix A.4.

an initial guess for the larger problem.

It is sometimes desirable to perform a sensitivity analysis on specific storages, by incrementing their capacities up while keeping all other system parameters constant. In this case, a combination of items (ii) and (iii) may be used. The upper boundary probabilities for the already solved problem with capacity N_i are shifted so as to become the upper boundary probabilities of the new problem with capacity N_i'. Internal states are set equal to an internal set of probabilities in the already solved problem. This procedure is illustrated in figure 4.2.

(iv) During the iterative multiplication procedure, it is possible to save some computation by using interpolation at regular intervals. For example, the program for a three-machine line given in Appendix A.3 uses the vectors $\underline{p}(t-1)$ and $\underline{p}(t)$ to interpolate $\underline{p}(t+1)$ once every ten iterations. This essentially gives a "free" iteration, since interpolating involves less computation than multiplying the vector by the transition matrix.

In order to avoid the propagation of computational errors, it is useful to normalize the $\underline{p}(t)$ vector at regular intervals. The program in Appendix A.3 does this once every ten iterations.

It may also be noted that it is possible to further save storage by only storing numerically distinct transition probabilities. It is shown in section 4.2.1 that the transition matrix T has a block tri-diagonal, block Toeplitz form. Thus, most probabilities reoccur many times along the diagonals of the matrix. Storing only distinct probabilities thus saves a significant amount of computer memory.

In general, many numerical devices may be used to improve the rate of convergence of the iterative multiplication algorithm. Still, for moderately large systems, the number of iterations remains large. It may thus be necessary to turn to more efficient methods in some cases.

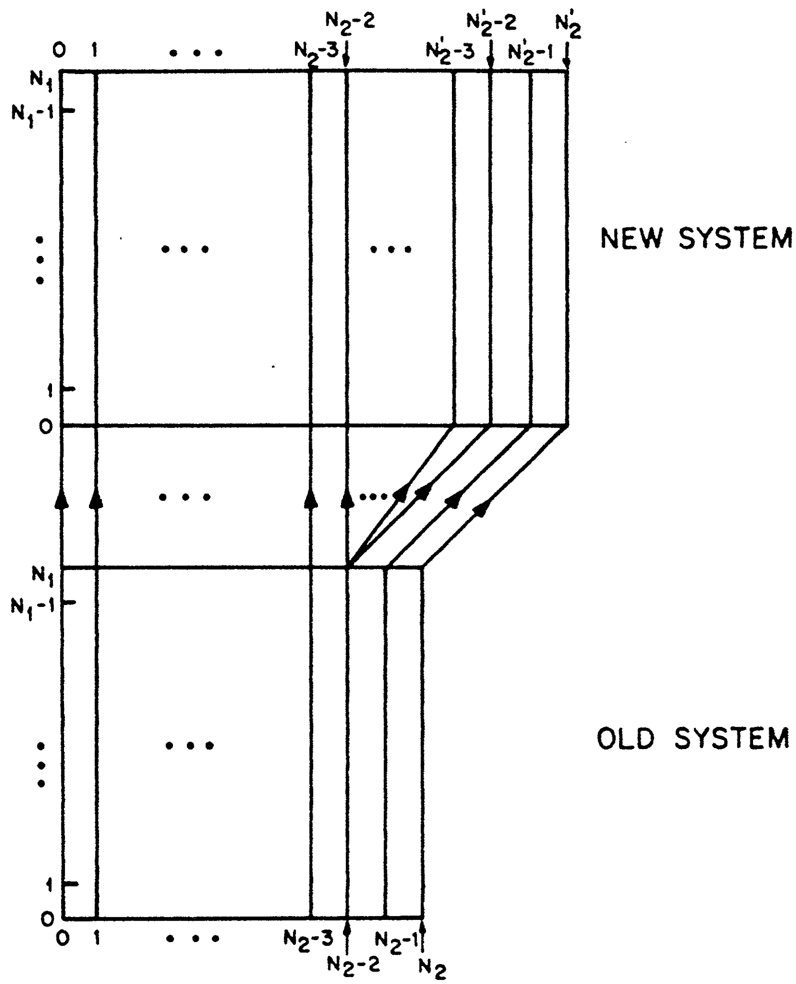


Figure 4.2. Building up initial guess for power method based on the results for a smaller storage capacity case. (Phase plane for the levels in the two storages)

4.2 Solution of the System of Transition Equations by Use of Sparsity and Structure

4.2.1 The Transition Matrix and its Structural Properties

It is shown in section 2.3.1 that the transition matrix T and the steady-state probability distribution $\underline{p}$ are related by

$$\underline{p} = T\,\underline{p} \tag{4.6}$$

or

$$(T - I)\,\underline{p} = \underline{0} \tag{4.7}$$

Because $\underline{p}$ satisfies equation 3.14, it cannot be identically zero. Thus, $\underline{p} \neq \underline{0}$ and (T - I) is a singular matrix. The following two theorems are now stated:

Theorem 4.1: If in the matrix T all rows and all columns corresponding to states outside the closed class C are deleted, there remains a stochastic matrix that defines a Markov chain on C. This subchain may be studied independently of all other states (Feller[1966]).

Theorem 4.2: In a finite recurrent aperiodic class, the steady-state probability distribution $\underline{p}$ is uniquely determined by the set of equations

$$\sum_i p_i = 1 \tag{4.8}$$

$$p_j = \sum_i p_i\, t_{ij} \qquad ;\ j=1,..,m \tag{4.9}$$

where as in 2.3.1, $T \triangleq [t_{ji}]$ (Karlin[1968]).

As pointed out in section 4.1, there is only one final (recurrent) aperiodic closed communicating class in the Markov chain under study. It may thus be concluded from the above two theorems that the deficiency of (T - I) in equation (4.7) is one. In other words, its rank is one less than full.

The vector $\underline{\nu}$ is defined as:

$$\underline{\nu} \overset{\Delta}{=} [1\ 1\ \ldots\ 1]^T \qquad (4.10)$$

Then, equation (3.14) is rewritten as

$$\underline{\nu}^T \underline{p} = 1 \qquad (4.11)$$

The vector $\underline{\nu}^T$ is substituted for a row in (T - I). Calling this new matrix T^*, and defining $\underline{b} \overset{\Delta}{=} [0\ldots0\ 1\ 0\ldots0]^T$ where the 1 entry corresponds to the location of $\underline{\nu}^T$ in T^*, it follows that:

$$T^* \underline{p} = \underline{b} \qquad (4.12)$$

Equation (4.12) is solved to give

$$\underline{p} = T^{*^{-1}} \underline{b} \qquad (4.13)$$

In principle, the problem thus reduces to solving a system of linear equations. Okamura and Yamashina[1977] solve the two-machine transfer line problem precisely in this way, by solving simultaneous linear equations in terms of the state probabilities. Because of memory limitations, they can only solve systems for which the storage capacity is less than 36. However, the dimensions of the system are generally very large (See table 6.1). Thus, solving (4.12) would involve an extremely large amount of computation and computer memory, as the number of machines in the line or the capacities of the storages increase. It is therefore necessary to fully exploit the sparsity and structure of T*. Tha sparsity follows from the fact that many storage transitions have zero probability (See table 2.2). The structure follows from the following two observations relating to the transition matrix T:

(i) During a single transition, storage levels can each change by a maximum of 1, i.e.,

$$|n_i(t+1) - n_i(t)| \leq 1 \quad ; \quad i=1,..,k-1 \tag{4.14}$$

This is due to the facts that parts only travel in one direction, each stage consists of only one machine, and the time cycle is defined such that a machine processes one part per cycle. Thus, as seen in the storage transition table 2.2, a storage can go up by 1, down by 1, or stay at a constant level.

(ii) During a single transition, adjacent storages cannot change in the same direction, i.e. they cannot both gain or both lose a piece within a single cycle. This is a consequence of the two facts mentioned in item (i), as well as the conservation of pieces (Assumption 2.2.4) in the system. As an example, a single machine i is analyzed. The level of the upstream storage i-1 can decrease only if machine i-1 is down or starved, and machine i is up and not blocked. At the same time, the downstream storage i can decrease only if machine i is down or starved, and machine i+1 is up and not blocked. Now machine i cannot be starved, since if it were, storage i-1 could not go down. Thus, for storages i-1 and i to both decrease, it is necessary that machine i be both up and down, a contradiction. A similar argument can be made for the case in which both storages are hypothesized to go up.

If the system states, as defined by equation (2.3), are listed semi-lexicographically (i.e. the order of the indices from the fastest changing one to the slowest is $\alpha_k, \alpha_{k-1}, .., \alpha_1, n_1, n_2, .., n_{k-1}$), then observation (i) implies that the matrix T is block tri-diagonal. If there is more than one storage, the main-diagonal blocks are themselves block tri-diagonal. This nested block tri-diagonal structure persists for as many <u>levels</u> as there are storages. Furthermore, in the case where there are more than one storage , observations (i) and (ii) together imply that the off-diagonal blocks are block bi-diagonal, and this structure persists for one fewer levels than there are storages.

The lowest level blocks, which are smallest and most basic, are $2^k x 2^k$. These each represent a specific storage level transition; for example, a particular block in the matrix of a three-machine line may represent a transition in which the first storage stays constant while the second goes up; another, a transition in which the first storage goes down while the second goes up, etc. Each of the 2^k columns and rows of a basic block represent each of the 2^k machine states, from (0 0 .. 0) = all machines are down, to (1 1 .. 1) = all machines are up. Finally, because T premultiplies $\underline{p}$ in equation (4.6), columns correspond to initial states, and rows to final states.

Some examples should help clarify the structures of the blocks and of the transition matrix.

(i) Two-machine line: the storage is initially internal, and remains constant through the transition (Main-diagonal block).

The basic blocks for a two-machine line are $2^2 x 2^2$=4x4 square matrices. From table 2.2, it follows that for an internal initial storage level, the number of pieces does not change in either of the following cases: either the machines after the transition are both down (0 0), so that no parts go into or out of the system, or the machines are both up (1 1), so that a part enters and another leaves the system. Thus, only two out of the four rows in this block are non-zero. The elements in these rows are computed by using table 2.1. For example, given that the storage is initially internal, (0 1) $\rightarrow$ (0 0) with probability equal to $(1-r_1)p_2$.

The block is thus completely determined, and appears in table 4.1.

(ii) Same as above, except that the storage is initially full (Upper boundary, main-diagonal block).

Once again, the final machine states that ensure the desired storage level transition are determined by using table 2.2. Given that the storage is initially full, its level remains constant only if the second machine is down, i.e. for final machine states (0 0) and (1 0). In the former case, nothing enters or leaves the system; in the latter, the first machine is blocked and the second is down, so that pieces do not enter or leave the system. If the second machine were operational, then the storage level would necessarily go down, since the first machine is not allowed to process a piece when the downstream storage is full. Again, only two out of the four rows in the block are non-zero. While evaluating the elements in these rows, it is noted that the probability of failure of the first machine is zero, since it is blocked.

$(1-r_1)(1-r_2)$	$(1-r_1)p_2$	$p_1(1-r_2)$	p_1p_2
0	0	0	0
0	0	0	0
r_1r_2	$r_1(1-p_2)$	$(1-p_1)r_2$	$(1-p_1)(1-p_2)$

Table 4.1. Two-machine line, lowest level, internal, main-diagonal block.

The block thus defined appears in table 4.2.

(iii) Three-machine line: both storage levels are initially internal; during the transition, the first stays constant, while the second loses a piece (Main-diagonal block in the second level upper off-diagonal block - see figure 4.4).

The basic blocks of a three-machine line are $2^3 x 2^3$ = 8x8 square matrices. It follows from table 2.2 that when initial storages are internal, the second storage loses a piece only if the second machine is down while the third machine is up. On the other hand, the first storage level remains unchanged if the first two machines are either both up or both down. Since the second machine is known to be down, the first machine must be down as well. Thus, only one final machine state satisfies the given conditions: (0 0 1), and only one out of the eight rows is non-zero. All machine transition probabilities are conditional on internal initial storages.

The block is thus determined and appears in table 4.3.

As stated earlier, the blocks are arranged in the transition matrix in a very useful nested block tri-diagonal structure. Two examples for two- and three-machine lines are given in figures 4.3 and 4.4 respectively. For clarity, the blocks in these figures are represented only by the final machine states corresponding to non-zero rows. It is noted that corner boundaries, where more than one storage is empty or full, differ from edge or internal transition blocks. These latter blocks are arranged in a convenient block Toeplitz form (Grenander and Szegö[1958]), which greatly facilitates computer implementations of the algorithm described in section 4.2.2. The block Toeplitz form is visible in figure 4.3, because the storage capacity is larger than 2 (Thus, internal transitions for initial storage levels equal to 1,2 and 3 involve identical basic blocks arranged on the diagonals of the matrix). In figure 4.4, the storages have capacities $N_1=N_2=2$, in order for the diagram to fit on one page and be readable. Here, the matrix is essentially subdivided into three second level block-columns (for n_2=0,1, and 2), each of which is made up of three lowest level block-columns (for n_1=0,1, and 2). The lowest level block-columns in the centers of the larger block-columns would be extended in Toeplitz form if $N_1>2$.

$(1-r_1)(1-r_2)$	$(1-r_1)p_2$	0	0
0	0	0	0
$r_1(1-r_2)$	r_1p_2	$(1-r_2)$	p_2
0	0	0	0

Table 4.2. Two-machine line, lowest level, upper boundary, main-diagonal block.

0	0	0	0	0	0	0	0
$(1-r_1)(1-r_2)r_3$	$(1-r_1)(1-r_2)(1-p_3)$	$(1-r_1)p_2r_3$	$(1-r_1)p_2(1-p_3)$	$p_1(1-r_2)r_3$	$p_1(1-r_2)(1-p_3)$	$p_1p_2r_3$	$p_1p_2(1-p_3)$
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 4.3. Three-machine line, lowest level, internal; Main-diagonal block in second level upper off-diagonal block.

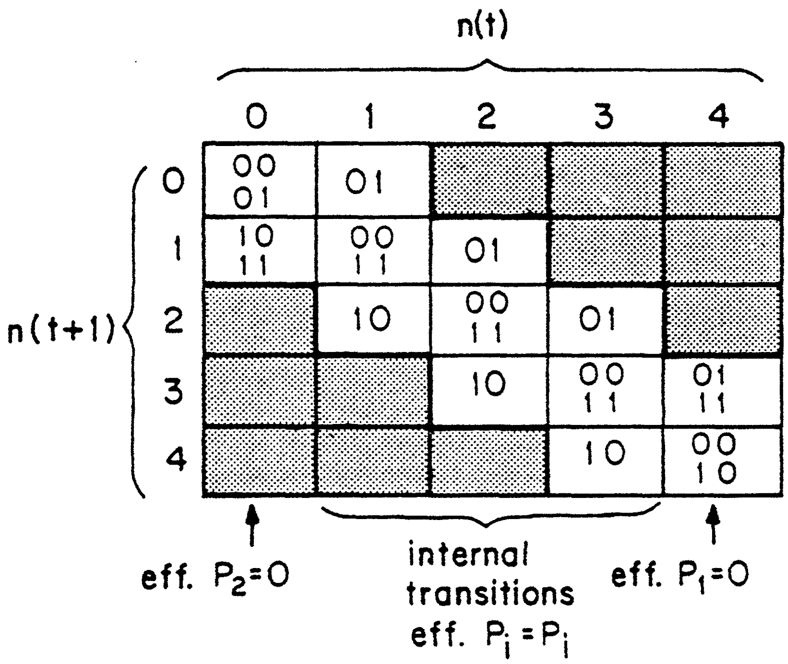


Figure 4.3. Structure of the transition matrix T for a two-machine case with N=4. (The numbers indicate the non-zero rows in the block. Eg. 00 → first row in block is nonzero. Shaded areas indicate zero blocks. The notation eff.p_i denotes the effective failure probability in that block-column.)

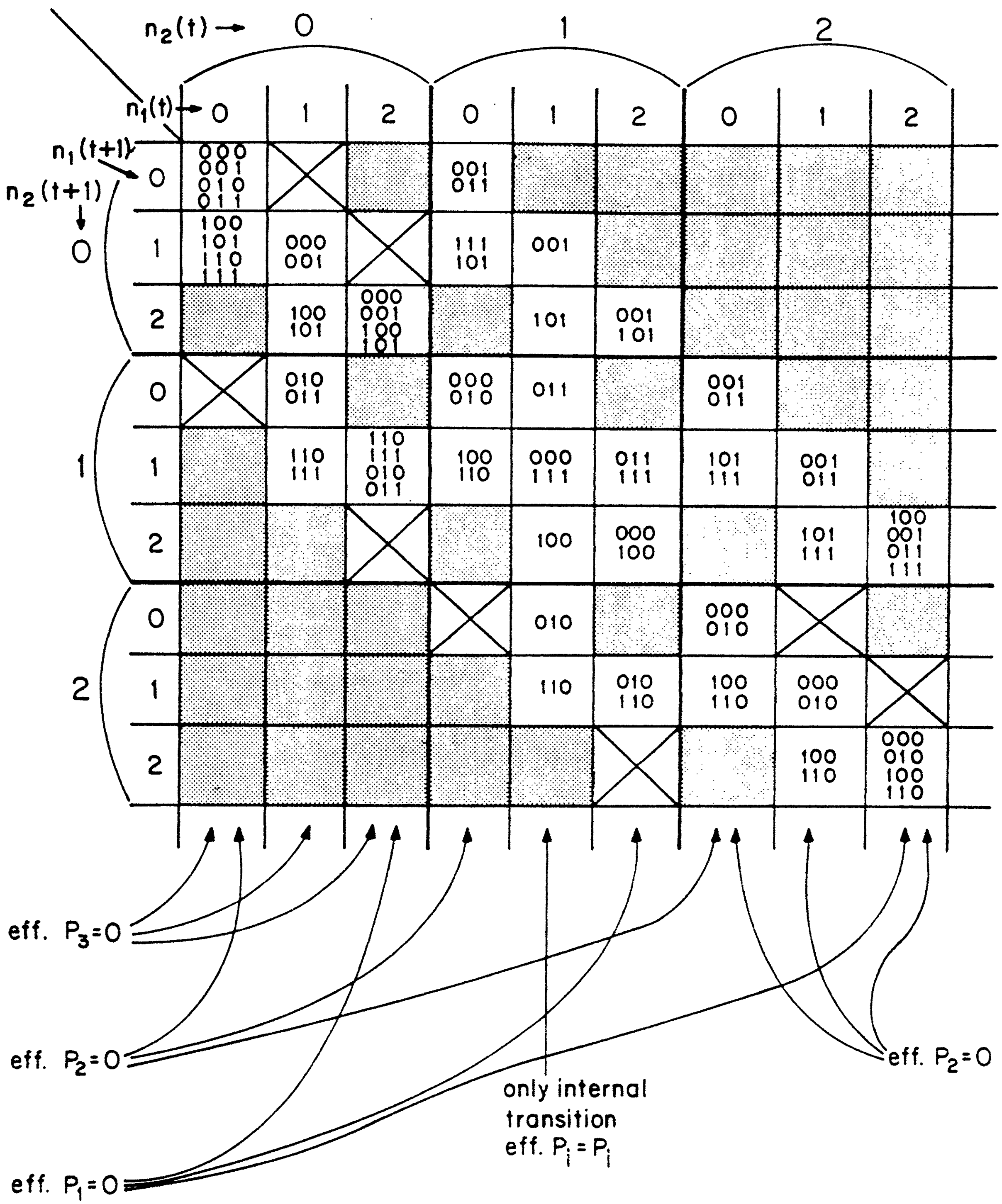


Figure 4.4. Structure of the transition matrix T for a three-machine line with $N_1=N_2=2$. (Crosses indicate those blocks which are all-zero blocks because of boundary transitions. For other notation, see fig. 4.3.)

Similarly, the middle second level block-column (That corresponding to $n_2=1$) would be extended in Toeplitz form if $N_2>2$.

It must be noted that although system states are said to be internal if all storage levels are greater than 1 and less than N_i-1, internal transitions are those for which neither storage is empty ($n_i=0$) or full ($n_i=N_i$). The reason for this difference is evident from the transition tables 2.1 and 2.2. These require that for a machine to be able to process a piece, there should be at least one piece in the upstream storage (i.e. $n_i>0$) and at least one vacant slot in the downstream storage (i.e. $n_i<N_i$).

4.2.2 Solution of the System of Transition Equations

It is shown in section 4.2.1 that the set of equations

$$(T - I)\,\underline{p} = \underline{0} \tag{4.15}$$

and

$$\underline{\nu}^T \underline{p} = 1 \tag{4.16}$$

has a unique solution $\underline{p}$, and that (4.15) and (4.16) may be combined and rewritten in the form

$$T^* \underline{p} = \underline{b} \tag{4.17}$$

or

$$\underline{p} = T^{*-1} \underline{b} \tag{4.18}$$

Equation (4.17) is to be solved by making use of the sparsity and structure of the transition matrix T. It is noted that while (T - I) is of the same block tri-diagonal form as T, the substitution of $\underline{\nu}^T$ for one row of (T - I) disturbs this structure. A new matrix T' is defined. This is a slightly modified version of (T - I) designed to make it non-singular. The nature of the modification is not of capital importance. In the program appearing in Appendix A.4, it consists in substituting in only that part of $\underline{\nu}^T$ which falls within the main-diagonal block. The rest of $\underline{\nu}^T$ is later taken into consideration, by applying the matrix inversion lemma (See below) on the inverse of T' calculated by the algorithm described in this section. Thus, the nested block tri-diagonal structure and sparsity of T is not lost. The following system of equations must now be solved:

$$T' \underline{p} = \underline{b} \tag{4.19}$$

The matrix inversion lemma (Householder[1965]) is now stated:

Lemma 4.1: Given the non-singular matrices H and G and their inverses H^{-1} and G^{-1}, and the compatible matrices F and G, the following identity holds:.

$$(H - EG^{-1}F)^{-1} = H^{-1} - H^{-1}E(-G + FH^{-1}E)^{-1}FH^{-1} \tag{4.20}$$

This lemma is used below to show that changing a single row in a matrix H (whose inverse H^{-1} is known) by defining $EG^{-1}F$ appropriately amounts to inverting a scalar. In this way, modifying $T'^{-1}\underline{b}$ to obtain $T^{*-1}\underline{b}$ is shown to be very simple.

The row in T' to be modified is chosen to be the row corresponding to the location of $\underline{\nu}^T$ in T*. Thus, its position corresponds to the location of the 1 entry in the otherwise all-zero $\underline{b}$ vector. The vector $\underline{\partial}^T$ is defined to be a correction vector such that

$$T' + \underline{b}\,\underline{\partial}^T = T^* \tag{4.21}$$

Then, by lemma 4.1,

$$\begin{aligned} T^{*-1} &= (T' - \underline{b}(-1)\underline{\partial}^T)^{-1} \\ &= T'^{-1} - T'^{-1}\underline{b}(1 + \underline{\partial}^T T'^{-1}\underline{b})^{-1}\underline{\partial}^T T'^{-1} \end{aligned} \tag{4.22}$$

where $(1 + \underline{\partial}^T T'^{-1}\underline{b})$ is a scalar and its inversion trivial. However, it is known from equation (4.18) that all of T^{*-1} is not needed. This inverse only appears in (4.18) as post-multiplied by the vector $\underline{b}$. Thus,

$$\begin{aligned} \underline{p} &= T^{*-1}\underline{b} \\ &= (T' + \underline{b}\,\underline{\partial}^T)^{-1}\,\underline{b} \\ &= T'^{-1}\underline{b} - T'^{-1}\underline{b}(1 + \underline{\partial}^T T'^{-1}\underline{b})^{-1}\underline{\partial}^T T'^{-1}\underline{b} \\ &= \left[1 - \frac{\underline{\partial}^T T'^{-1}\underline{b}}{1 + \underline{\partial}^T T'^{-1}\underline{b}} \right] T'^{-1}\underline{b} \\ &= \left[\frac{1}{1 + \underline{\partial}^T T'^{-1}\underline{b}} \right] T'^{-1}\underline{b} \end{aligned} \tag{4.23}$$

It is noted that the term in the parantheses in (4.23) is a scalar. Furthermore, T'^{-1} only appears in (4.23) as post-multiplied by $\underline{b}$. Since $\underline{b}$ is a vector of zeros with one 1 entry, post-multiplying a matrix by $\underline{b}$ amounts to reading off one column of that matrix. It will be shown that because of this, it is useful to substitute $\underline{\nu}^T$ for one of the first 2^k rows in the (T - I) matrix.

Equation (4.23) has an interesting implication: whatever the slight modification discussed earlier actually is, it follows from (4.23) that the result will merely be a scalar multiple of the true probability vector $\underline{p}$. Thus, finding the solution vector $\underline{p}$ amounts to normalizing the result obtained by solving the modified system in equation (4.19). The denominator $(1 + \underline{\partial}^T T'^{-1} \underline{b})$ performs this normalization. As a consequence of this, it is possible to modify T* in absolutely any way, as long as the resulting matrix is still block tri-diagonal, and has become non-singular. The desired result $\underline{p}$ can then be obtained by normalizing $T'^{-1}\underline{b}$.

For simplicity of notation, the T' matrix is now partitioned into blocks as seen in table 4.4. Matrices of this form have been studied by various authors. Disney[1972] solves the two-server queue with overflow problem (See also Çınlar and Disney[1967]) by using the block tri-diagonal structure of the transition matrix. Evans[1967] proposes a quadratic matrix equation of the form

$$B + AK + CK^2 = 0 \tag{4.24}$$

based on the assumption that the solution vector may be partitioned into vectors $\underline{y}_i$ such that

$$\underline{y}_{i+1} = K\, \underline{y}_i \tag{4.25}$$

Wallace[1969] develops this algorithm, derives conditions for such a solution to exist, and proposes an iterative algorithm to obtain the K matrix. These studies apply to block tri-diagonal matrices of infinite dimension and block Toeplitz form, except possibly at the lower boundary.

$$
\begin{bmatrix}
A_0 & C_1 & 0 & 0 & \cdots & & 0 \\
B_0 & A_1 & C_2 & 0 & \cdots & & 0 \\
0 & B_1 & A_2 & C_3 & \cdots & & 0 \\
\vdots & & \ddots & \ddots & \ddots & & \vdots \\
0 & \cdots & 0 & B_{N-2} & A_{N-1} & C_N \\
0 & \cdots & 0 & 0 & B_{N-1} & A_N
\end{bmatrix}
$$

Table 4.4. General form of the T' matrix.

Such matrices are termed by Wallace quasi-birth-death processes.

Solutions of systems of equations with tri-diagonal or block tri-diagonal matrices have been studied by Hindmarsh[1977], Varah[1972], and Temperton[1975], in relation to systems of difference equations and other applications. Navon[1977] presents several algorithms including an LU-decomposition scheme for block tri-diagonal matrices, and investigates the numerical stability of such algorithms.

As stated above, the entire inverse matrix T^{*-1} is not needed, since it only appears in equation (4.23) as post-multiplied by $\underline{b}$. Nevertheless, an algorithm for computing T^{*-1} for a two-machine line is described in full for completeness. The algorithm is then generalized to a k-machine line. It is shown that in order to obtain the inverse of a matrix such as the one appearing in table 4.4, it is necessary to know the inverses of the main-diagonal blocks. However, it was shown earlier that in the case of a k-machine line, there are k-1 levels of nested block tri-diagonal matrices. In other words, the main-diagonal blocks are themselves block tri-diagonal. Thus, it is necessary to obtain the full inverse of a block tri-diagonal matrix at all levels except the lowest and the highest levels. In the former case, the lowest level (basic) blocks are not tri-diagonal; in the latter, only one column of the inverse is needed in equation (4.23). This is done by means of the algorithm described below.

The following matrices are defined for a two-machine system: T' is the slightly modified non-singular version of $(T - I)$. Thus, it has the form of the matrix given in table 4.4. It consists of $(N+1)^2$ blocks, each of which is of dimension $2^2 x 2^2 = 4x4$. Here, N is the storage capacity as in the earlier discussion. The rectangular matrix Y is defined as a matrix of dimensions 4(N+1)x4. It is partitioned into N+1 blocks of dimension 4x4, as seen below:

$$Y \triangleq \begin{bmatrix} Y_0 \\ \vdots \\ Y_N \end{bmatrix} \tag{4.26}$$

The rectangular matrix E has the same dimensions as Y. It is partioned into N+1 blocks of dimension 4x4, as seen below:

$$E \triangleq \begin{bmatrix} E_0 \\ \vdots \\ E_N \end{bmatrix} \tag{4.27}$$

Thus, the dimensions of the matrices are compatible, and it is possible to write the equation

$$T' \; Y = E \tag{4.28}$$

From table 4.4 and equation (4.28), it follows that:

$$- \; A_0Y_0 + C_1Y_1 = E_0 \tag{4.29}$$

$$B_0Y_0 + A_1Y_1 + C_2Y_2 = E_1$$

$$B_1Y_1 + A_2Y_2 + C_3Y_3 = E_2$$

$$\dots$$

$$B_{N-2}Y_{N-2} + A_{N-1}Y_{N-1} + C_NY_N = E_{N-1} \tag{4.30}$$

$$B_{N-1}Y_{N-1} + A_NY_N = E_N \tag{4.31}$$

Assuming that A_N is invertible, equation (4.31) may be solved to give

$$Y_N = A_N^{-1} \, [\, E_N - B_{N-1}Y_{N-1} \,] \tag{4.32}$$

Substituting (4.32) into (4.30), and assuming once again that the desired inverses exist, it follows that:

$$Y_{N-1} = [A_{N-1} - C_NA_N^{-1}B_{N-1}]^{-1}[E_{N-1} - C_NA_N^{-1}E_N - B_{N-2}Y_{N-2}] \tag{4.33}$$

Equations (4.32) and (4.33) suggest a recursion. Defining the matrices X_i and D_i as

$$\left.\begin{aligned} X_0 &= A_N \\ X_i &= A_{N-i} - C_{N-i+1} X_{i-1}^{-1} B_{N-i} \end{aligned}\right\} \quad i=1,..,N \tag{4.34}$$

and

$$\left.\begin{aligned} D_0 &= A_N^{-1} E_N = X_0^{-1} E_N \\ D_i &= X_i^{-1} [E_{N-i} - C_{N-i+1} D_{i-1}] \end{aligned}\right\} \quad i=1,..,N \tag{4.35}$$

it may be verified that

$$Y_{N-i} = D_i - X_i^{-1} B_{N-i-1} Y_{N-i-1} \quad ; \quad i=0,..,N \tag{4.36}$$

The set of equations (4.29) through (4.31) are thus solved backwards, until equation (4.29), which gives

$$Y_0 = A_0^{-1} [E_0 - C_1 Y_1] \tag{4.37}$$

From (4.36), it follows that (4.37) can be rewritten as

$$Y_0 = A_0^{-1} [E_0 - C_1 D_{N-1} + C_1 X_{N-1}^{-1} B_0 Y_0] \tag{4.38}$$

Solving (4.38) for Y_0, and factoring,

$$\begin{aligned} Y_0 &= [A_0 - C_1 X_{N-1}^{-1} B_0]^{-1} [E_0 - C_1 D_{N-1}] && (4.39) \\ &= X_N^{-1} [E_0 - C_1 D_{N-1}] && (4.40) \\ &= D_N && (4.41) \end{aligned}$$

where (4.40) and (4.41) follow from equations (4.34) and (4.35) respectively. The recursion is thus complete. Equations (4.36) and (4.41) are rewritten as

$$\left.\begin{aligned} Y_0 &= D_N \\ Y_i &= D_{N-i} - X_{N-i}^{-1} B_{i-1} Y_{i-1} \end{aligned}\right\} \quad i=1,..,N \tag{4.42}$$

where D_i and X_j are defined by equations (4.34) and (4.35).

Now, the matrix E is successively set equal to a set of matrices which together constitute the identity matrix. This is achieved by defining the blocks of E as follows:

$$\left.\begin{aligned} E_i &= I \\ E_j &= 0 \quad ; \quad j=0,1,..,N;\ j\neq i \end{aligned}\right\} \quad i=0,1,..,N \tag{4.43}$$

Then, solving equation (4.28) by the recursion formulas defined above gives a solution matrix Y which is the $i+1^{st}$ block-column of T'^{-1}. Clearly, then, it is only necessary to obtain the inverses of $X_0,..,X_N$ in order to find T'^{-1}. However, these blocks may be large in systems with more than two machines. It is therefore not desirable to use a direct inversion procedure, since even the best computer implementations of inversion algorithms involve considerable amounts of computation. It is necessary to make use of the sparsity of C_i and obtain X_i^{-1} more efficiently.

It is noted that C_i has few non-zero rows: by a rough estimation, only about 25% of the rows in C_i are non-zero. It is easy to verify that a product in which C_i appears as a pre-multiplier has its only non-zero rows in the same positions as those of C_i. Thus, the product

$$C_{N-i+1} X_{i-1}^{-1} B_{N-i} \tag{4.44}$$

in equation (4.34) has approximately three quarters of its rows equal to zero. It follows that only about one quarter of the rows of A_{N-i} in equation (4.34) are altered when the term in (4.44) is subtracted from it. If A_{N-i}^{-1} is known, it is then possible to use the matrix inversion lemma (lemma 4.1) to alter these rows and obtain X_i^{-1} efficiently. It only remains to show that A_i^{-1} are relatively easy to obtain.

In the general k-machine case, the main-diagonal blocks are themselves block tri-diagonal. As noted above, this nested structure persists for k-1 levels. This suggests a recursive procedure whereby the inverses of the main-diagonal blocks are computed by the algorithm described above.

At the lowest level, as in the two-machine case, the blocks are basic, in the sense described in section 4.2.1, and their dimensions are $2^k x 2^k$. The main diagonal blocks represent those transitions in which none of the storage levels change. They are thus similar to the blocks discussed in examples (i) and (ii) in section 4.2.1. The important difference between these examples and the main-diagonal blocks of T' is that T' is a slightly modified version not of T but of (T - I) . Thus, the identity matrix is subtracted from each of the main diagonal basic blocks. The inverses of these blocks have relatively simple closed-form solutions. For example, the inverse of the block described in example (i) of section 4.2.1 (minus the identity matrix) has the following form. Given that the block is represented by $A=[a_{ij}]$ and its inverse by $A^{-1}=[a'_{ij}]$,

$$\left.\begin{aligned}
a'_{11} &= a_{44} / \Delta \\
a'_{1j} &= (a_{1j}a_{44} - a_{4j}a_{14}) / \Delta \qquad ; j=2,3 \\
a'_{14} &= -a_{14} / \Delta \\
a'_{41} &= -a_{41} / \Delta \\
a'_{4j} &= (a_{4j}a_{11} - a_{1j}a_{41}) / \Delta \qquad ; j=2,3 \\
a'_{44} &= a_{11} / \Delta \\
a'_{jj} &= -1 \quad ; j=2,3
\end{aligned}\right\} \tag{4.45}$$

where

$$\Delta = a_{11}a_{44} - a_{14}a_{41} \tag{4.46}$$

All other entries in A^{-1} are zero. This closed inverse carries over to larger blocks which have the same form as the block in example (i) of section 4.2.1. For the three-machine analogous block, for example, all 3 and 4 in the above equations become 7 and 8, respectively; nothing else is changed.

Here, it is important to note that although the algorithm described in this section adopts the notation $A_0, A_1, .., A_N$ for generality, not all of these matrices are in fact distinct. Basic blocks are defined as the transition probabilities for given initial and final storage levels. The important point to note, however, is that these transitions are not conditional on the precise values of these levels. Rather, it is whether the initial level was empty, full, or otherwise which conditions the transition probability. These three possibilities are termed <u>regions</u>. In a k-machine line, there are k-1 storages, each of which can be in any of three regions. Thus, there are at most 3^{k-1} different matrices among the $(N_1+1)..(N_{k-1}+1)$ basic main diagonal blocks. This implies the block Toeplitz form of the transition matrix. The fact that the number of different A_i is small is especially important if the storage capacities are large. The number of distinct matrices among the A_i is independent of storage capacity and remains small. This is further discussed in section 4.2.3.

It is important to reiterate that the entire T'^{-1} matrix is not needed. In fact, if $\underline{\nu}^T$ is substituted for one of the first 2^k rows of $(T - I)$, the problem becomes significantly simpler. In that case, it is only necessary to compute a column from the first block-column of T'^{-1}. This means that the solution Y corresponding to $E_0=I$, $E_i=..=E_N=0$ is sought. Then, the problem becomes

$$\left.\begin{aligned} Y_0 &= X_N^{-1} \\ Y_i &= -X_{N-i}^{-1} B_{i-1} Y_{i-1} \end{aligned}\right\} \qquad i=1,..,N \tag{4.47}$$

where X_i are defined, as before, by equation (4.34). The column of Y corresponding to the $\underline{\nu}^T$ row in T' is then equal to $T'^{-1}\underline{b}$. The solution vector $\underline{p}$ is found from equation (4.23).

The problem is now reformulated for the general k-machine transfer line. The T' matrix for such a line is block tri-diagonal; its main diagonal blocks are themselves block tri-diagonal, and this persists downwards for k-1 levels. The lowest level (ℓ=1) is defined to be the basic level. Then, the solution to equation (4.19) is to be found recursively; solving the ℓ^{th} level problem requires the results of level

ℓ-1. The following notation is now established: A_i^1, B_i^1 and C_i^1 are the main-, lower off-, and upper off-diagonal basic blocks. At the ℓ^{th} level, the main diagonal block in the i^{th} block-column is A_i^ℓ; similarly, the ℓ^{th} level upper off-diagonal block in the i^{th} block-column is C_i^ℓ, and the ℓ^{th} level lower off-diagonal block in the i^{th} block-column is B_i^ℓ. The X_i^ℓ, E_i^ℓ, and D_i^ℓ matrices are defined analogously. This notation will be clarified by the example on table 4.5.

At ℓ=1, the inverses of the main-diagonal blocks, $(A_i^\ell)^{-1}$, are found by explicit inversion, or by means of closed-form solutions such as that in equations (4.45) and (4.46).

At the ℓ^{th} level, where ℓ<k-1, the following recursions are defined in order to calculate $(A_i^{\ell+1})^{-1}$:

$$\left.\begin{aligned} X_0^\ell &= A_{N_\ell}^\ell \\ X_i^\ell &= A_{N_\ell-i}^\ell - C_{N_\ell-i+1}^\ell (X_{i-1}^\ell)^{-1} B_{N_\ell-i}^\ell \end{aligned}\right\} \quad i=1,..,N_\ell \tag{4.48}$$

$$\left.\begin{aligned} D_0^\ell &= (X_0^\ell)^{-1} E_{N_\ell}^\ell \\ D_i^\ell &= (X_i^\ell)^{-1} [E_{N_\ell-i}^\ell - C_{N_\ell-i+1}^\ell D_{i-1}^\ell] \end{aligned}\right\} \quad i=1,..,N_\ell \tag{4.49}$$

and

$$\left.\begin{aligned} Y_0^\ell &= D_{N_\ell}^\ell \\ Y_i^\ell &= D_{N_\ell-i}^\ell - (X_{N_\ell-i}^\ell)^{-1} B_{i-1}^\ell Y_{i-1}^\ell \end{aligned}\right\} \quad i=1,..,N_\ell \tag{4.50}$$

where $(X_i^\ell)^{-1}$ is found by applying lemma 4.1 on $(A_i^\ell)^{-1}$; for each but the lowest level, $(A_i^\ell)^{-1}$ are the inverses obtained at the level immediately below. This is done by setting $E_i^{\ell-1}$ successively equal to the identity matrix I, as stated in equation (4.43).

An example may serve to clarify this procedure: it is desired to find $\underline{p}$ for a three-machine transfer line with storage capacities N_1 and N_2. Thus, equation (4.19) must be solved to obtain $T'^{-1}\underline{b}$.

$$A_i^{\ell} = \begin{bmatrix} A_0^{\ell-1} & C_1^{\ell-1} & 0 & \cdots & & \\ B_0^{\ell-1} & A_1^{\ell-1} & C_2^{\ell-1} & & & \\ \vdots & \ddots & \ddots & \ddots & & \\ & & B_{N_\ell-2}^{\ell-1} & A_{N_\ell-1}^{\ell-1} & C_{N_\ell}^{\ell-1} \\ & & 0 & B_{N_\ell-1}^{\ell-1} & A_{N_\ell}^{\ell-1} \end{bmatrix}$$

Table 4.5. The ℓ^{th} level main-diagonal block.

Finding the inverse of a block tri-diagonal matrix requires knowledge of the inverses of its main-diagonal blocks. Furthermore, every level corresponds to a storage, and each storage has three regions: empty, full, and otherwise. Thus, for a three-machine system, k=3 and there are only $3^2=9$ distinct basic blocks, and 3 distinct second level blocks. The procedure followed in solving this problem is schematically illustrated in figure 4.5. The numbers on the arrows indicate the order of the steps of the recursion. For reasons that directly follow from equations (4.48) to (4.50), the recursion proceeds from top to bottom and from right to left in the diagram.

Since only the first block-column of T'^{-1} is needed at the highest level (at the condition that $\underline{\nu}^T$ is substituted for any one of the first 2^k rows), for $\ell=k-1$, the recursion described by equations (4.48)-(4.50) reduces to:

$$\left.\begin{aligned} X_0^{k-1} &= A_{N_{k-1}}^{k-1} \\ X_i^{k-1} &= A_{N_{k-1}-i}^{k-1} - C_{N_{k-1}-i+1}^{k-1} (X_{i-1}^{k-1})^{-1} B_{N_{k-1}-i}^{k-1} \end{aligned}\right\} \quad i=1,..,N_{k-1} \qquad (4.51)$$

and

$$\left.\begin{aligned} Y_0^{k-1} &= (X_{N_{k-1}}^{k-1})^{-1} \\ Y_i^{k-1} &= -(X_{N_{k-1}-i}^{k-1})^{-1} B_{i-1}^{k-1} Y_{i-1}^{k-1} \end{aligned}\right\} \quad i=1,..,N_{k-1} \qquad (4.52)$$

Equations (4.48) through (4.52), together with (4.23), completely determine the solution vector $\underline{p}$, i.e. the vector of state probabilities for a k-machine transfer line.

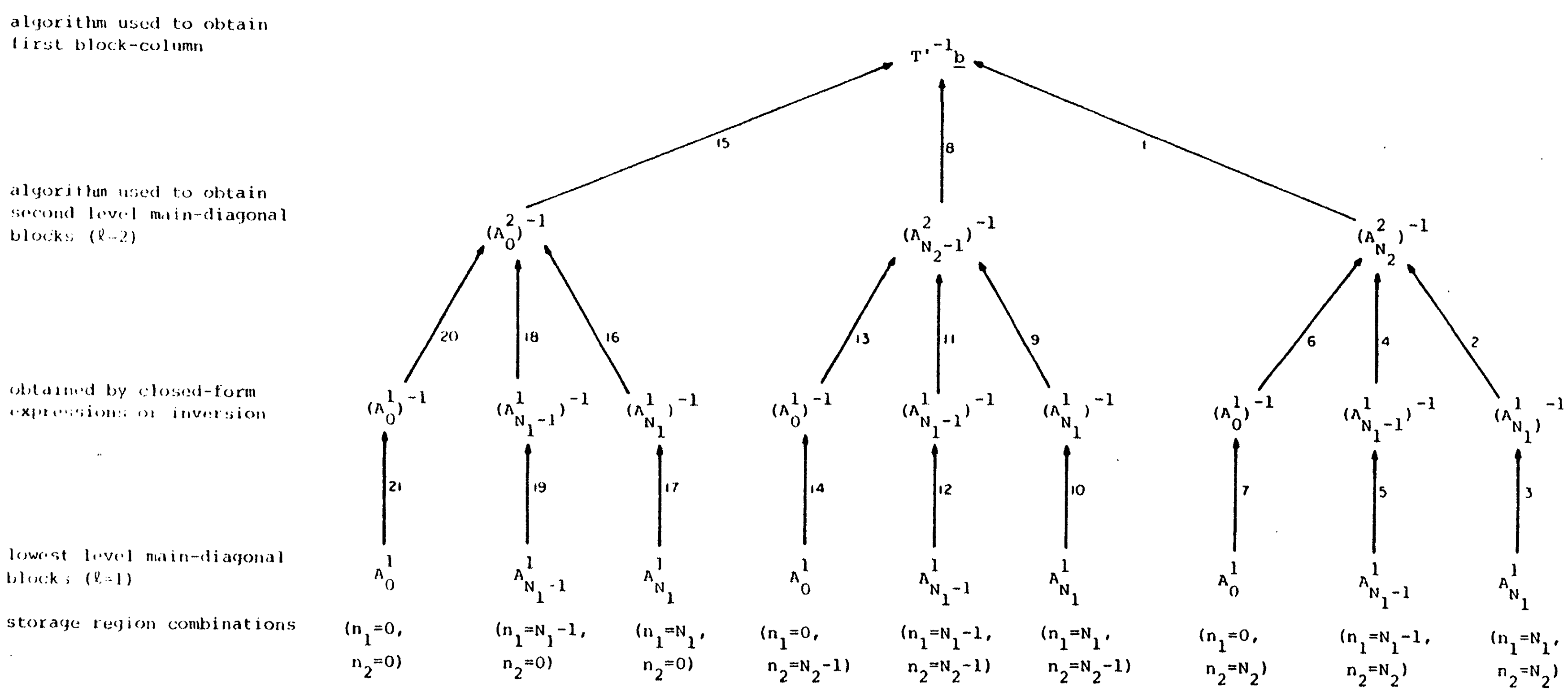


Figure 4.5. Recursive procedure to solve the three-machine line problem by using the sparsity and nested block tri-diagonal structure of the transition matrix T.

4.2.3 Discussion of the Algorithm and Computer Implementations

The basic blocks represent the state transition probabilities given the initial and final storage levels. If the storage levels are ordered semi-lexicographically, then the blocks are arranged in a block tri-diagonal form. Because of the way in which the states are ordered, each level corresponds to allowing the level of a particular storage to take all values from zero to maximum capacity N_i (See figure 4.4).

At each level, all blocks on any given diagonal are equal except at the upper and lower boundaries (figure 4.6). This is because transition probabilities are conditional on the adjacent storages being empty, full, or otherwise. As long as a storage is not empty or full, it influences the transition probability in the same way, regardless of the value of its level. Each diagonal matrix is thus in nearly block Toeplitz form. Since there are only three storage regions, there are three distinct main-diagonal blocks in any higher level block. Thus, at each level ℓ, it is not necessary to obtain <u>all</u> $(A_i^\ell)^{-1}$ in order to compute $(X_i^\ell)^{-1}$. Only three different $(A_i^\ell)^{-1}$ must be obtained: for $i=N_\ell, N_\ell-1$, and 0. These are computed in this order because of the way in which equation (4.48) is set up. Furthermore, since there are three distinct matrices at each level, the total number of distinct main-diagonal blocks for all branches of the recursion at any level ℓ is given by $3^{k-\ell}$. This may be verified in figure 4.5. At the lowest level (ℓ=1), there are 3^2=9 different storage region combinations, and thus there are 9 distinct main-diagonal blocks. At the second level, there are only 3 distinct main-diagonal blocks. As storage size (and thus the total number of blocks) increases, the savings in computation made by the observation that only three inverse matrices have to be obtained at any level of the recursion becomes more and more important.

Secondly, it may be noted that on the boundaries, some of the blocks which (according to the formulation of the nested block tri-diagonal matrix) should be non-zero diagonal blocks are in fact zero blocks. This

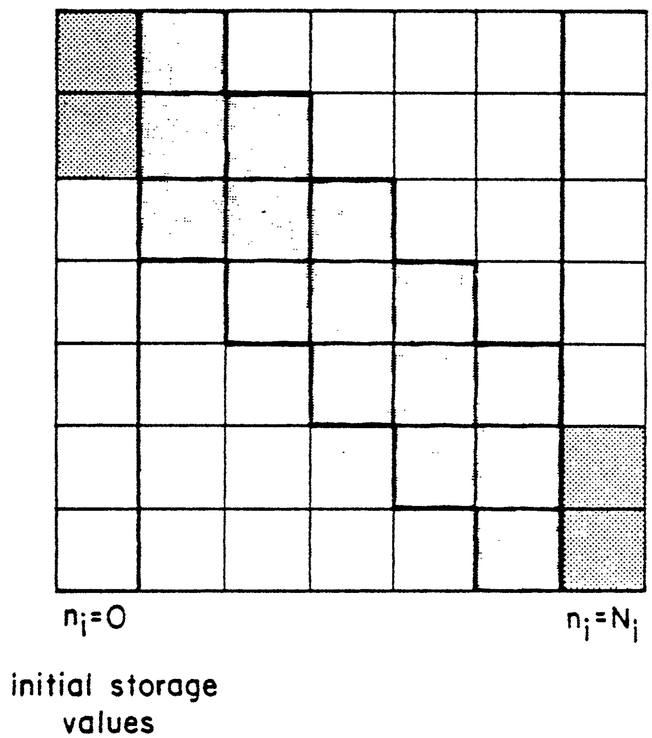


Figure 4.6. Location of boundary block-columns
in any main-diagonal block.

may be verified in figure 4.4. This is caused by the fact that the level of an empty storage cannot go down, and that of a full storage cannot go up. The positions of these blocks are easily predictable, so that it is possible to save computation by avoiding multiplications involving large, all-zero blocks. Furthermore, at the highest level, the upper and lower main-diagonal blocks are bi-diagonal, instead of tri-diagonal, for these reasons. This too allows great computational savings, since for those two cases (out of a total of three distinct highest level main-diagonal blocks), equations (4.48)-(4.50) reduce to

$$\left.\begin{aligned} Y_0^\ell &= (A_0^\ell)^{-1}\, E_0^\ell \\ Y_i^\ell &= (A_i^\ell)^{-1}\, [E_i^\ell - B_{i-1}^\ell Y_{i-1}^\ell] \end{aligned}\right\} \qquad i=1,..,N_\ell \tag{4.53}$$

A third and very important point is that if some storage capacity is increased from N_ℓ to N'_ℓ at some level ℓ, then it is only necessary to recompute $(X_{N_\ell}^\ell)^{-1},\ldots,(X_{N'_\ell}^\ell)^{-1}$ in equation (4.48) (as well as higher level matrices). The savings in computation allowed by this are crucial especially for large N_ℓ. The reason for this is that increasing the capacity of a storage amounts to appending new block-columns and block-rows to the matrix at level ℓ. If the very last storage is being incremented up, then ℓ=k-1 and the only change in the original matrix is in the final block-column, since transitions with initial storage $n_{k-1}=N_{k-1}$ are no longer boundary transitions when the capacity is increased to $N'_{k-1} > N_{k-1}$. As a result, only X-matrices with indices beginning at the old storage capacity and at levels beyond the storage whose capacity is changed need be recomputed. Since the bulk of computations in the algorithm is the computation devoted to generating $(X_i^\ell)^{-1}$ (equation 4.48), this is an important consideration in computer implementations.

The major problem in computer implementations of this algorithm is not computation time but memory requirements. Because $(X_i^\ell)^{-1}$ can only be generated "upwards", i.e. from i=0 to $i=N_\ell$ (equations (4.48)) and are used "downwards", i.e. from $i=N_\ell$ to i=0 (equations (4.49) and (4.50)), they must all be stored in memory and can not be computed as needed. This causes very serious memory problems. At high levels, the X matrices are very large. In IBM double precision, the computer program given in Appendix A.4 required 1M bytes for a small ($N_1=N_2=10$) storage capacity case.

This difficulty can be overcome by reverting to slow memory: this may be done by creating disk files and storing the unused $(X_i^{\ell})^{-1}$ matrices in these files. Since these matrices are not needed at all times, the time loss incurred by this procedure may not be very significant. A better way is to use the IBM Virtual Machine System (See references under IBM). This process allows unlimited virtual memory and enables the program to be loaded and executed even with large storage capacities.

Computational complexity and error stability studies remain to be performed for this algorithm.

5. COMPUTATION OF SYSTEM PERFORMANCE MEASURES

Calculating the steady-state probabilities by using the methods outlined in chapters 3 and 4 is not an end in itself. It is a means for obtaining certain system performance measures that are of use in designing transfer lines or job shops. This chapter discusses some of these important performance measures and how they are obtained from the state probabilities.

The steady-state (long-time average) efficiency and production rate of the system are defined and ways of computing them are discussed in section 5.1. Section 5.1.1 discusses different expressions for efficiency, as well as various ways to calculate it. Some conclusions pertaining to the conservation of pieces are derived from these equivalent methods for obtaining efficiency. The transients of the system are investigated in section 5.1.2 and their effects on the production rate of the system are discussed. Some computational results on the dependence of efficiency on storage size are given and discussed in section 5.1.3.

Section 5.2 investigates the effect of buffer size on the performance of individual machines. The asymptotic behavior of forced-down times as functions of storage size is established.

Section 5.3 studies the dependence of in-process inventory on storage capacity. It is shown that inventory does not necessarily increase linearly with storage size, and that in some cases it approaches an asymptote as storage capacity is increased.

5.1 Efficiency and Production Rate of the System

5.1.1 Computation of Efficiency

A transfer line can produce a part during a time cycle if and only if the last machine is operational and the last storage is non-empty. The efficiency of the transfer line is defined as the probability that the line produces a part during any cycle. This probability is equivalent to the expected value of the ratio of the number of cycles during which the line produces a part to the total number of cycles. The production rate of the transfer line is the expected number of completed parts produced by the system per unit time. Thus, since all machines have equal, deterministic service times (Assumption 2.2.2), it follows that

$$\text{Production Rate} \stackrel{\Delta}{=} \frac{\text{Efficiency}}{\text{Length of Machining Cycle}} \tag{5.1}$$

Throughout this work (with the exception of chapter 8) the length of a machining cycle is taken to be one time unit, so that production rate equals efficiency. These two terms are thus used interchangeably.

The efficiency in isolation of the i^{th} machine in a k-machine transfer line is defined as

$$e_i \stackrel{\Delta}{=} \frac{\text{average up time}}{\text{average up time + average down time}} \tag{5.2}$$

$$= \frac{(1/p_i)}{(1/p_i) + (1/r_i)} \tag{5.3}$$

$$= \frac{r_i}{r_i + p_i} \tag{5.4}$$

Physically, this is the efficiency of that machine removed from the line and supplied with an unlimited reservoir of workpieces and an unlimited sink for processed parts. It is easy to see that for a one-machine line, where assumption 2.2.1 holds, this quantity is equivalent to the definition for transfer line efficiency given by equation 5.1.

When the machine operates within an unreliable transfer line, however, the assumption of unlimited supply of workpieces does not hold, since the adjacent upstream storage is sometimes empty due to failures upstream in the line. Similarly, the assumption of unlimited storage space for machined pieces does not hold, since the adjacent downstream storage is sometimes full due to failures downstream in the line. Thus, the actual production rate of the machine <u>in</u> the unreliable line, defined as its <u>utilization</u>, is lower than its efficiency in isolation. The utilization of the i^{th} machine in a k-machine transfer line is defined as

$$E_i \stackrel{\Delta}{=} p[\text{a piece emerges from machine i during any cycle at steady-state}] \tag{5.5}$$

$$= p[\alpha_i(t+1)=1, n_{i-1}(t)>0, n_i(t)<N_i] \tag{5.6}$$

The difference between the arguments of $\alpha_i(\cdot)$ and $n_j(\cdot)$ is required by assumption 2.2.5. It is clear that the efficiency of the transfer line, as defined in section 5.1, is equal to the utilization of the last machine, E_k, as given by equation (5.5). Here, $n_k(t)$ is taken to be non-full, as stated in section 3.1.1. Intuitively, the expected utilization of all machines should be equal, since pieces are neither created nor destroyed by the line (Assumption 2.2.4). This proposition is developed later.

Another numerical quantity is now defined for the i^{th} machine in the transfer line:

$$S_i \stackrel{\Delta}{=} p[\alpha_i(t)=1, n_{i-1}(t)>0, n_i(t)<N_i] \tag{5.7}$$

It is noted that the difference between E_i and S_i is that in the latter, all events take place at the <u>same</u> time t, whereas in the former, the machine state and the storage levels are examined at different times.

Proposition 5.1: $E_i = S_i$

Since from section 2.3.2,

$$p[\alpha_i(t+1)=1|\alpha_i(t)=0,n_{i-1}(t)>0,n_i(t)<N_i] \quad = \quad r_i \tag{5.8}$$

$$p[\alpha_i(t+1)=1|\alpha_i(t)=1,n_{i-1}(t)>0,n_i(t)<N_i] \quad = \quad 1-p_i \tag{5.9}$$

it follows that by Bayes' theorem,

$$\begin{aligned} E_i \quad = \quad & r_i p[\alpha_i(t)=0,n_{i-1}(t)>0,n_i(t)<N_i] \quad + \\ & (1-p_i)\ p[\alpha_i(t)=1,n_{i-1}(t)>0,n_i(t)<N_i] \end{aligned} \tag{5.10}$$

All events in equation (5.10) take place at the same time t. The two terms on the right hand side of (5.10) may be rewritten as

$$p[\alpha_i(t)=0,n_{i-1}(t)>0,n_i(t)<N_i] \quad = \quad \sum_{s\varepsilon\Omega_0} p[s] \tag{5.11}$$

$$p[\alpha_i(t)=1,n_{i-1}(t)>0,n_i(t)<N_i] \quad = \quad \sum_{s\varepsilon\Omega_1} p[s] \tag{5.12}$$

where

$$\Omega_j \quad \stackrel{\Delta}{=} \quad \{s|\alpha_i(t)=j,n_{i-1}(t)>0,n_i(t)<N_i\} \qquad ;j=0,1 \tag{5.13}$$

Thus, equation (5.10) becomes

$$E_i \quad = \quad r_i \sum_{s\varepsilon\Omega_0} p[s] \quad + \quad (1-p_i) \sum_{s\varepsilon\Omega_1} p[s] \tag{5.14}$$

Using the same notation, equation (5.7) may be rewritten as

$$S_i \quad = \quad \sum_{s\varepsilon\Omega_1} p[s] \tag{5.15}$$

It is thus necessary to show that

$$E_i - S_i = r_i \sum_{s\epsilon\Omega_0} p[s] - p_i \sum_{s\epsilon\Omega_1} p[s] \tag{5.16}$$

$$= 0 \tag{5.17}$$

Consider the set Ω_0. Since machine repair does not depend on storage levels, the probability that the system leaves Ω_0 in any time cycle is given by

$$r_i\, p[\Omega_0] = r_i \sum_{s\epsilon\Omega_0} p[s] \tag{5.18}$$

To get into set Ω_0, which consists of all states in which machine i is down and the upstream storage is non-empty and the downstream storage is non-full at the same time cycle, it is necessary that at the previous time cycle, these storages were non-empty and non-full, respectively. This is because by assumption 2.2.3, a machine can only fail while processing a piece, and it can only process a piece if there is at least one piece in the upstream storage and at least one vacant slot in the downstream storage. Thus, set Ω_0 can only be reached from set Ω_1, and this takes place if machine i fails. The probability that the system enters set Ω_0 in any time cycle is therefore given by

$$p_i\, p[\Omega_1] = p_i \sum_{s\epsilon\Omega_1} p[s] \tag{5.19}$$

Because of the steady-state assumption (Section 2.2.6), the probability of the system entering any set of states during a given period must equal the probability of it leaving the same set during a period of the same length. Thus, equations (5.18) and (5.19) give the balance equation

$$r_i \sum_{s\epsilon\Omega_0} p[s] = p_i \sum_{s\epsilon\Omega_1} p[s] \tag{5.20}$$

so that

$$E_i - S_i = 0 \tag{5.21}$$

or

$$E_i = S_i \tag{5.22}$$

This proves proposition 5.1. The consequence of this proof is that the computation of efficiency is considerably simplified.

Proposition 5.1 can also be demonstrated intuitively*. Consider for the sake of illustration the last machine in a two-machine line. Neglecting transient (zero steady-state probability) states, it may be verified in the state transition diagram (figure 2.4) that all transitions to states in which the storage is non-empty and the second machine is up, with exactly one exception, result in the production of a piece. Thus, calculating the sum of the probabilities of these states is equivalent to finding the ratio of the number of transitions that result in the production of a piece to the total number of transitions. The only exception is the transition $(0,0,1)\rightarrow(1,1,1)$. This does not result in a part because the storage must first become non-empty before the second machine can operate. However, it is also observed that all possible transitions to $(0,0,1)$ from other states do result in the production of a piece. This is the case with the three transitions $(1,0,0)\rightarrow(0,0,1)$, $(1,0,1)\rightarrow(0,0,1)$, and $(1,1,1)\rightarrow(0,0,1)$. This means that for every transition to $(1,1,1)$ that does not result in a piece, there is exactly one transition to $(0,0,1)$ that does. Thus, if the probabilities of states in which the storage is non-empty and the second machine is up are summed (thereby obtaining S_2), the $(0,0,1)\rightarrow(1,1,1)$ transition that is included in the sum but did not result in a part is esactly counter-balanced by transitions to $(0,0,1)$ that were not included in the sum but that did result in a part. Since the sum of the probabilities of

---

* This demonstration is due to Mr. M. Akif Eyler of Harvard University.

states which have been reached by a transition that resulted in the production of a part is E_i, it follows that

$$E_i = S_i = \sum_{s \varepsilon \Omega_1} p[s] \tag{5.23}$$

as before.

Proposition 5.2: $S_i = S_j$; all i,j (Proof for k=2).

The proof that all machines have equal utilizations is considerably more complex. In the two-machine case, it involves closed-form expressions derived in chapter 3, and explicit relations between X, Y_1, and Y_2. Since such relations have not yet been obtained for lines with more than two machines, the proposition has not yet been proved for transfer lines longer than two machines. The consequence of this proposition is that the assumption that pieces are neither created nor destroyed by the line (Section 2.2.4) implies that all stages in the line have equal steady-state production rates.

For the two-machine case, the proof proceeds as follows:
Defining the sets

$$B_1 \triangleq \{s|\alpha_1=1, n<N\} \tag{5.24}$$

$$B_2 \triangleq \{s|\alpha_2=1, 0<n\} \tag{5.25}$$

It is easy to verify from equation (5.7) that

$$S_j = p[B_j] \quad ; \quad j=1,2 \tag{5.26}$$

Introducing the more compact notation

$$B_1 \triangleq \{(n<N, 1, \alpha_2)\} \tag{5.27}$$

$$B_2 \triangleq \{(0<n, \alpha_1, 1)\} \tag{5.28}$$

and defining the intersection of these two sets as

$$C \triangleq B_1 \cap B_2 \tag{5.29}$$

$$= \{(0<n<N,1,1)\} \tag{5.30}$$

equation (5.26) may be rewritten as

$$S_j = p[B_j - C] + p[C] \quad ; \quad j=1,2 \tag{5.31}$$

Thus, proving that the two machines have equal utilizations is equivalent to showing that

$$p[B_1 - C] = p[B_2 - C] \tag{5.32}$$

This is demonstrated as follows: the set on the left hand side of equation (5.32) is

$$B_1 - C = \{(0,1,\alpha_2)\} \cup \{(n<N,1,0)\} \tag{5.33}$$

It is noted that both elements in the first set in this union have zero steady-state probabilities. Thus,

$$p[B_1 - C] = \sum_{n=0}^{N-1} p[n,1,0] \tag{5.34}$$

Similarly,

$$B_2 - C = \{(N,\alpha_1,1)\} \cup \{(0<n,0,1)\} \tag{5.35}$$

Again, the elements in the first set of the union have zero steady-state probabilities. It follows that

$$p[B_2 - C] = \sum_{n=1}^{N} p[n,0,1] \tag{5.36}$$

The results of section 3.2.1 are now used. From table 3.1,

$$p[n,1,0] = \begin{cases} CX^n\, Y_1 & ; \quad n=2,..,N-1 \\ 0 & ; \quad n=0,1 \end{cases} \tag{5.37}$$

$$p[n,0,1] = \begin{cases} CX^n\, Y_2 & ; \quad n=1,..,N-2 \\ 0 & ; \quad n=N-1,N \end{cases} \tag{5.38}$$

and from equation (3.25),

$$X = Y_2 \;/\; Y_1 \tag{5.39}$$

Thus,

$$\sum_{n=0}^{N-1} p[n,1,0] = \sum_{n=2}^{N-1} CX^n\, Y_1 \tag{5.40}$$

$$= \sum_{n=1}^{N-2} CX^{n+1}\, Y_1 \tag{5.41}$$

$$= \sum_{n=1}^{N-2} CX^n\, Y_2 \tag{5.42}$$

$$= \sum_{n=1}^{N} p[n,0,1] \tag{5.43}$$

It follows that

$$p[B_1 - C] = p[B_2 - C] \tag{5.44}$$

which proves the proposition for the two-machine case.

In the general k-machine case, equations (5.24) and (5.25) are extended to include both upstream and downstream storages:

$$B_i \triangleq \{s \,|\, \alpha_i=1, 0<n_{i-1}, n_1<N_i\} \tag{5.45}$$

so that once again,

$$S_i = p[B_i] \tag{5.46}$$

Then, demonstrating that all machines have equal steady-state utilizations is equivalent to proving that

$$p[B_i - C] = p[B_j - C] \quad ; \quad \text{all } i,j \tag{5.47}$$

where, as before,

$$C \triangleq B_i \cap B_j \tag{5.48}$$

As stated earlier, the proof for the two-machine case involves the explicit relationship between X_i and Y_j given by equation (5.39). Since this relation is not known to hold for a k-machine line, the proposition has not been proved for the general case.

5.1.2 System Transients and Efficiency

Both the analytical and the numerical methods discussed in the past chapters give steady-state solutions. In some problems, steady-state signifies the theoretical description of a system when time is allowed to approach infinity and the system itself becomes static as all transients die down. For example, a released pendulum oscillates for a certain period of time, but it eventually comes to rest. In stochastic systems, steady-state does not imply that the system itself is at rest, but that the probabilistic model of the system has become stationary. For ergodic systems, one consequence of this is that the system behavior at steady-state is independent of initial conditions (See section 2.3.1). Assumption 2.2.6 therefore implies that the system has been running long enough so that it is governed by steady-state probability distributions, and the effects of start-up have vanished. Knowing the system's initial state thus gives no information on its present state. Essentially, therefore, the practical equivalent of the abstract concept of steady-state is the long-time average. In practical situations, this may approach the steady-state values in a time that is relatively short compared to the mathematical calculation of the time required to approximate steady-state conditions. When the system has run long enough, the average efficiency obtained is equal to the steady-state value computed on the basis of the theory developed in chapters 3 and 4.

How long is "long enough", however, is a question that is difficult to answer. The speed with which the system approaches steady-state is a function of the second largest eigenvalue of the transition matrix (See section 4.1). The eigenvalues are related to the system parameters (the probabilities of failure and repair and the storage sizes) and are not easy to compute. It is possible to estimate how many cycles it would take for the transients to vanish, given an initial condition, by using the power method (Section 4.1). The number of iterations the algorithm needs to converge is a measure of the expected speed with which the system reaches a stationary probability distribution.

There are two main consequences of the effect of start-up transients on the efficiency of the system:
(i) The steady-state efficiency may be calculated, by analytical or numerical methods, to as many decimal places as the computer is capable of handling. However, if the start-up transients take very long to vanish, so that the system does not sufficiently approach steady-state during finite-time operations, this accurate efficiency computed on the basis of the steady-state probabilities will not reflect the actual behavior of the system. As a result of this, the model will not adequately describe and predict the production of the actual system.
(ii) On the other hand, the transients may not dominate enough to render the model useless, although they may have some effects on the system. Then, the fact that actual efficiency is close but not exactly equal to the steady-state efficiency suggests that approximate methods may be used to calculate with less work an approximate efficiency that is sufficiently precise for actual (e.g. industrial) applications. This theme is developed in chapter 6.

Since no system has yet actually operated for an infinite length of time, it is important to understand the finite-time, non-steady-state behavior of the system. The dynamic simulation program described in section 6.1 was used and runs were made for different lengths of time and system parameters. Some results are presented in figures 5.1-5.5.

In one set of runs, the average number of pieces produced per cycle was sampled at regular intervals, for several interval lengths. These are not cumulative averages, but were computed over each non-overlapping interval. It should be noted, however, that intervals were taken consecutively and without long time periods between them. Thus, the sample efficiencies obtained in this manner are not independent. As expected, deviations from the mean become smaller as the length of the time intervals are increased. At the limit, an infinite-length time interval would give the steady-state efficiency. This is confirmed by the fact that the cumulative average efficiency approaches the steady-state value after about 500 time cycles and does not appreciably deviate from that value thereafter even though the sample averages continue to fluctuate. The important point to note is therefore not that the output of the system fluctuates, but that the cumulative (long-time) average converges on the steady-state value. Three examples for a two-machine line with the parameters appearing in table 5.1 are given in figures 5.1-5.3. These are for interval lengths of 1, 10, and 100 cycles, respectively. Since the three graphs are drawn to the same vertical scale, the fact that the magnitude of the deviations from the mean decrease with interval length is clearly illustrated in these plots.

A different set of runs consist of simulating systems with different parameters, keeping the time interval constant. Figures 5.4 and 5.5 illustrate some results for the system parameters given in table 5.2. For very small failure or repair probabilities, the system spends long periods of time in few states, while it does not reach some of the lower probability states during simulations of short durations. Thus, it does not fluctuate often enough in short time periods, and the cumulative average does not approach the steady-state value during these short periods. On the other hand, large failure or repair probabilities imply that transitions take place often, and all states are visited more or less frequently. The cumulative average approaches the steady-state efficiency much faster in this case. This experiment confirms that the applicability of the steady-state assumption to actual systems depends strongly on the system parameters.

Sections 5.1.3, 5.2, and 5.3 discuss the relations between system parameters and three basic performance measures: production rate, forced-down times, and in-process inventory. All computations make the steady-state assumption. How close such results are to actual values depends on the system parameters and the length of time the actual system is continuously operated.

$p_1 = 0.1$	$p_2 = 0.05$	$N = 4$
$r_1 = 0.2$	$r_2 = 0.2$	

Table 5.1. System parameters for dynamic simulation of system transients.

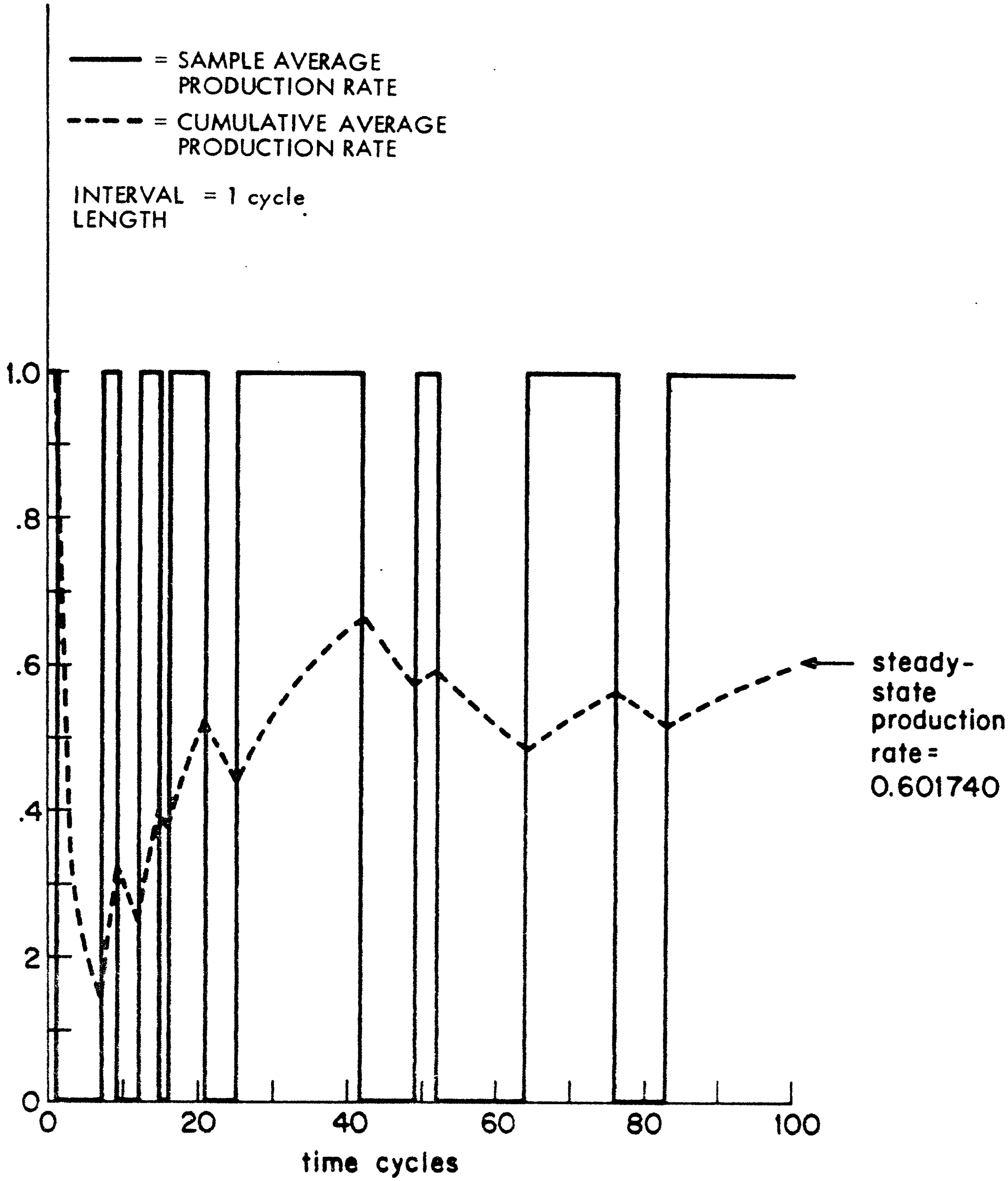


Figure 5.1. Sample and cumulative average production rates for a two-machine line and intervals of length 1.

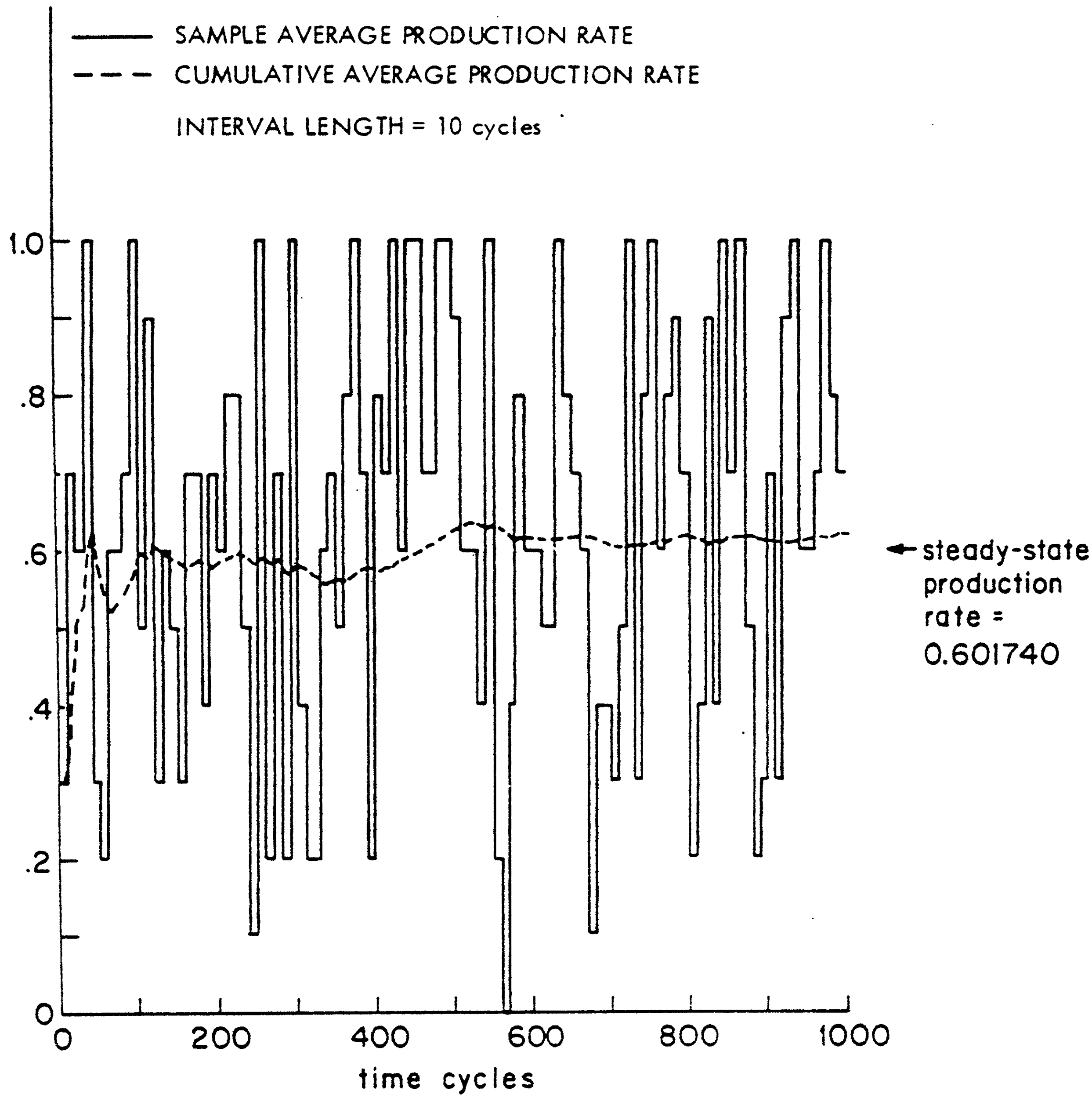


Figure 5.2. Sample and cumulative average production rates for a two-machine line and intervals of length 10.

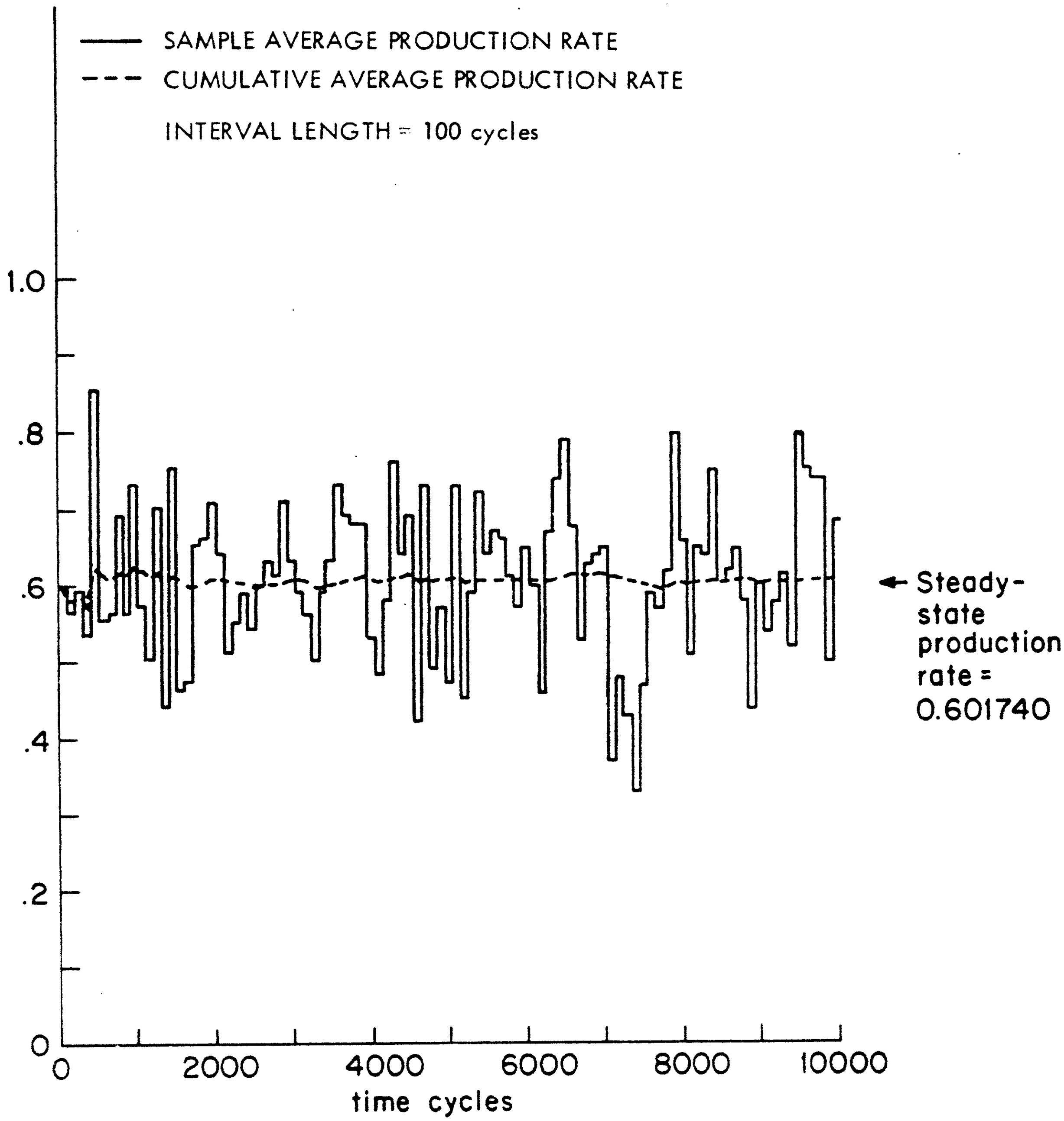


Figure 5.3. Sample and cumulative average production rates for a two-machine line and intervals of length 100

Case 1:	$p_1 = 0.001$	$p_2 = 0.002$	$N = 4$
	$r_1 = 0.001$	$r_2 = 0.003$	
Case 2:	$p_1 = 0.9$	$p_2 = 0.9$	$N = 4$
	$r_1 = 0.8$	$r_2 = 0.8$	

Table 5.2. System parameters for dynamic simulation of system transients.

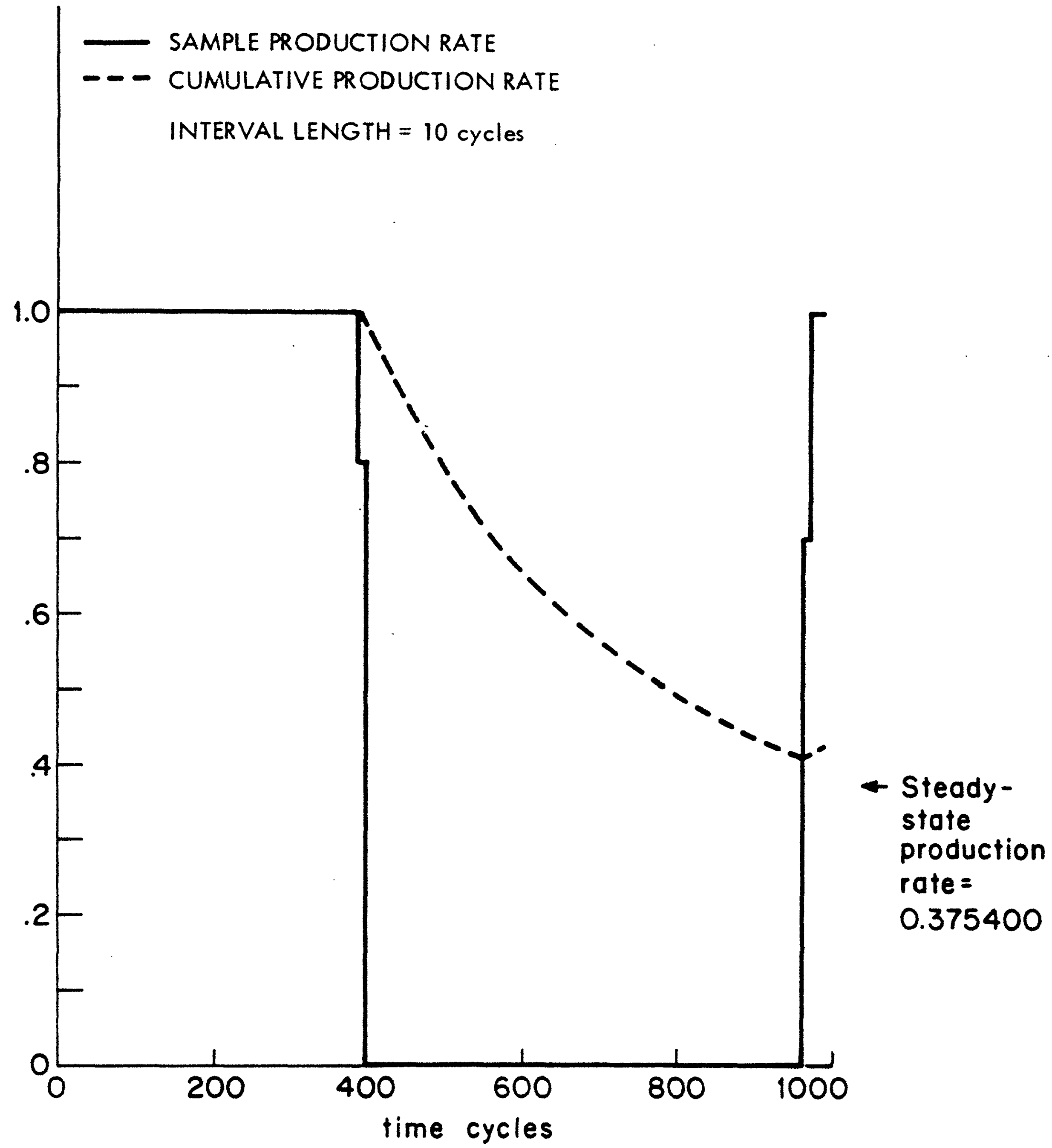


Figure 5.4. Sample and cumulative average production rates for a two-machine line with small probabilities of failure and repair.

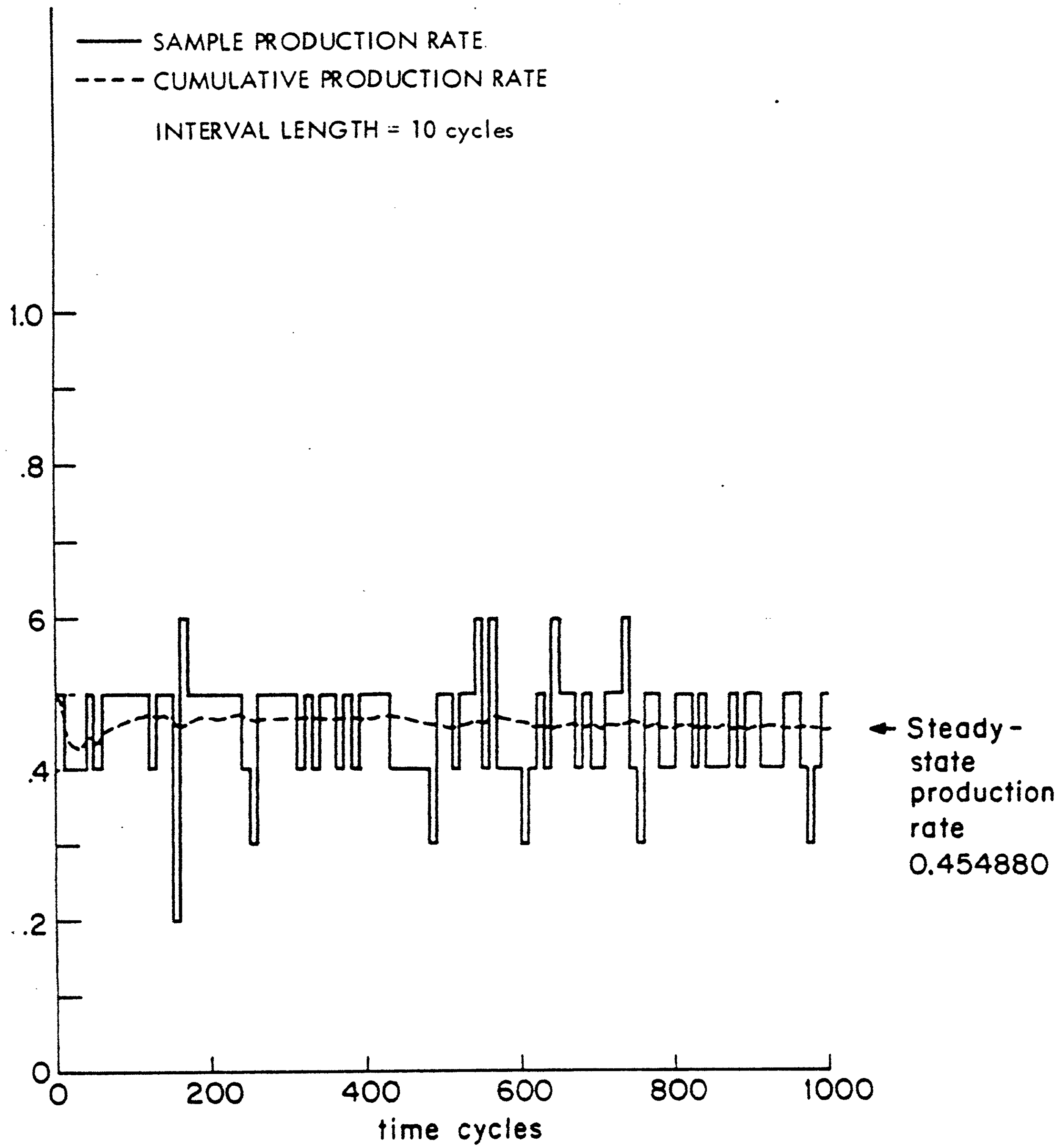


Figure 5.5. Sample and cumulative average production rates for a two-machine line with large probabilities of failure and repair.

5.1.3 Production Rate and Storage Size

Studies of transfer lines may be subdivided into three classes, on the basis of the assumption they make with regard to the capacity of interstage buffers. These are (Barten[1962]):

(i) No storage: In the case of servers in tandem with no storage space between them, the machines are most tightly coupled, in that when one of them breaks down, the entire line must stop. Such lines are often encountered in industry, as in the case of continuous production lines.

(ii) Infinite storage: In this case, the machines are as decoupled as possible, as is shown below. Although infinite buffer capacities give the highest possible production rate for a given set of machines, this assumption does not have wide applicability to actual situations. Costs of storage capacity and in-process inventory make even very large buffers relatively rare in industry.

(iii) Finite storage: In this case, a limited storage capacity is provided between machines or stages consisting of several machines. This is the most common case in industry, as well as in many other areas.

The no-storage case was treated by numerous researchers, including Buzacott[1967a,1968], Hunt[1956], Suzuki[1964], Rao[1975a], Avi-Itzhak and Yadin[1965], and Barlow and Prochan[1975]. Masso and Smith[1974] state that adjacent stages with no storage between them may, in some cases, be combined and treated as a single machine. This simplifies the analysis of long transfer lines considerably. The most complete analysis of the efficiency of systems without buffer storages appears in Buzacott[1968], where various network topologies are considered.

In the present case, the system consists of k machines in series, with equal and deterministic cycle times, taken to be 1 time unit. The derivation of the efficiency of a line with no storages presented below follows Buzacott[1968]. For a given machine i, where i=1,..,k, the mean up- and down-times are given by $1/p_i$ and $1/r_i$, respectively. Assuming that during some long time period T, machine i produces M pieces, it follows that during that period, the expected number of breakdowns is Mp_i. Thus, the total expected down time of machine i is Mp_i/r_i time units. However, the whole line is forced down whenever one machine fails. It follows that

$$T = M + \sum_{i=1}^{k} Mp_i / r_i \tag{5.49}$$

The efficiency of the line is equal to that of any machine, since the line is up only when all machines are up. As defined in equation 5.1, efficiency is the ratio of expected up time to total time. Thus,

$$E(0) = \frac{M}{T} \tag{5.50}$$

$$= \frac{1}{1 + \sum_{i=1}^{k} p_i / r_i} \tag{5.51}$$

where E(0) is defined to be the efficiency of a line with no buffer storages. This value gives a lower bound to the transfer line production rate that can be obtained with the given set of machines.

An upper bound is given by the limit of efficiency as the storage capacities go to infinity. Although this is, as stated earlier, an unrealistic assumption, it does sometimes give remarkably accurate results (See Solberg[1977]). Infinite buffer models have been the subject of considerable research, including Buzacott[1967a,1967b], Hunt[1956], Morse[1965], and Schweitzer[1976].

A common mistake (Buzacott[1967b], Koenigsberg[1959], Masso and Smith[1974], Barten[1962] and others) is to assume that infinite buffers truly decouple the machines, so that each machine may be considered independent from all others. It follows then that the efficiency of the line, $E(\infty)$, is equal to that of the least efficient machine. Basing himself on Burke[1956], Koenigsberg[1959] notes that this is the case when machines are reliable and have exponentially distributed service times, and the input to the line is Poisson. In this case, Burke shows that the output of a machine is also Poisson, and since an infinite storage implies that there is no blocking, each stage is indeed independent. Assumption 2.2.1, however, requires that a machine in isolation be never starved or blocked. Okamura and Yamashina[1977] point out that in some cases, the average number of pieces in a storage approaches a limit as the capacity of the storage increases (See also section 5.3). In such cases, the storage may be empty and starve the downstream machine with positive probability. It is thus incorrect to assume that machines are truly decoupled by infinite storages.

However, this does not invalidate Buzacott's thesis that the production rate of a line with infinite buffers is equal to that of the least efficient machine. Furthermore, Okamura and Yamashina's counter argument leaves much to be desired: although they are able to solve two-machine lines with storage capacities less than 36 only (because of memory limitations), they deduce from a graph a value for $E(\infty)$ which they say is lower than the efficiency of the least efficient machine. The assertion that the production rate of a line with infinite buffers is equal to the efficiency of the least efficient machine is proved by two different approaches. Mathematical induction is used below; it is also demonstrated in section 5.2 that in the two-machine case, the forced-down times of the least efficient machine approaches zero as storage capacity increases.

To prove the proposition, it is first noted that for a transfer line with infinite capacity buffer storages, any segment of the line (be it a single machine or a series of machines and storages) behaves independently from the downstream part of the line. This is because blocking can not occur with infinite buffers. Thus, the segment operates at its efficiency in isolation, i.e. at its highest possible production rate, without being hampered by what is appended downstream of it.

A storage is defined to be <u>stable</u> if and only if its level increases without bound as $t\to\infty$ with zero probability (Lavenberg[1975], Hildebrand[1967]).

The proof that the production rate of a line with infinite capacity buffers is equal to the efficiency of its least efficient machine proceeds as follows: Concentrating first on the first storage (storage 1), two cases may be considered. Either the first machine is <u>less</u> efficient than the second one, in which case the storage is stable (See section 5.3), or else it is unstable.

(i) Storage is stable: Since the first machine is never blocked, it operates with a production rate equal to its efficiency in isolation. The second machine can do no better than the first, so that the flow of pieces through the second machine must be equal to or less than the efficiency of the first one. If the average rate of flow through the second machine were less than that through the first one, the level in the first storage would increase without bound as $t\to\infty$. This contradicts the hypothesis. Thus, for a stable storage, conservation of pieces (See sections 2.2.4 and 5.1.1) holds, and the production rate of the downstream machine is equal to that of the upstream one, which is the least efficient of the two.

(ii) Storage is unstable: By definition, the number of pieces in the storage increases without bound as $t\to\infty$. By modeling the level of the storage as a birth-death process with states $\stackrel{\Delta}{=} 0,1,2,\ldots$ and assigning $p_{i,i+1}$, the probability of transition from state i to state i+1 to be greater than $p_{i,i-1}$ for all i, it may be shown that the probability of being in state 0 (storage is empty) decreases to zero as $t\to\infty$. (Note that this is also true for states 1,2,...). Thus, the probability that the second machine is starved goes to zero as $t\to\infty$. Since the second machine is never blocked, the rate of flow through it approaches its efficiency in isolation as $t\to\infty$. In case (i), the first machine is less efficient than the second one by hypothesis. Here, the second machine is at most as efficient as the first one. Thus, the assertion has been proved for storage 1.

Storage i is considered next. It is between machines i and i+1 (See figure 2.1). It is assumed, following the usual induction argument, that the assertion is true for storage i-1. Thus, the rate of flow through machine i is equal to the efficiency of the least efficient among machines

1 to i. Once again, there are two possibilities.

(i) If storage i is stable, conservation of pieces holds by the same argument as above. Then, the rate of flow through machine i+1 is equal to that of machine i. For the storage to be stable, it is noted that machine i+1 must be more efficient than the upstream portion of the line. Thus, the rate of flow through machine i+1 is equal to the efficiency of the least efficient machine among machines 1 to i+1.

(ii) If storage i is unstable, the same argument as above implies that the probability that it is empty approaches zero as $t \to \infty$. Then, the rate of flow through machine i+1 approaches its efficiency in isolation as $t \to \infty$. The storage is unstable only if the efficiency of machine i+1 is less than or equal to that of the upstream portion of the line.

Thus, it has been shown that assuming that the assertion holds for storage i-1 implies that it also holds for storage i. The proof is now complete. Defining the efficiency of the line $E(\infty)$ to be the rate of flow out of the last machine (which may or may not be equal to the rate of flow into the line, depending on the stability of the storages), it follows that (Buzacott[1967a,1967b]):

$$E(\infty) = \min_{i=1,..,k} \left\{ \frac{r_i}{r_i + p_i} \right\} \tag{5.52}$$

$$= \frac{1}{1 + \max\limits_{i=1,..,k} \left\{ \frac{p_i}{r_i} \right\}} \tag{5.53}$$

The lower bound on production rate given by equation (5.51) and the upper bound given by (5.53) are now analyzed. It is noted that for a perfectly reliable machine, $p_i/r_i \to 0$ and for a completely unreliable one, $p_i/r_i \to \infty$. Thus, if all machines are very reliable, $E(0) \to E(\infty) \to 1$. On the other hand, if a single machine is much less reliable than all the others, $E(0) \to E(\infty)$. Since the difference between the upper and lower bounds indicates how much production can be increased by the addition of buffer storages, it follows that storages are most useful when each machine is not extremely reliable and no single machine is much less efficient than all others (Buzacott[1967a]). An interesting consequence is that the production rate of a balanced line, in which all machines have equal efficiencies in isolation, is likely to improve more by the addition of buffer storages than that of an unbalanced line.

Although the difference between $E(0)$ and $E(\infty)$ indicates how much can be gained by adding storages with infinite capacities to the line, it is useful to have a measure of how effective a given storage configuration $(N_1,..,N_{k-1})$ is in reducing the loss of production due to breakdowns. The effectiveness of a set of storage capacities for a given line with known $E(0)$ and $E(\infty)$ is defined as

$$\eta(N_1,..,N_{k-1}) = \frac{E(N_1,..,N_{k-1}) - E(0)}{E(\infty) - E(0)} \tag{5.54}$$

(Equation (5.61) follows Freeman[1964] and Buzacott[1969], rather than Buzacott[1967b], who takes the denominator to be $1-E(0)$). Since

$$E(0) \leq E(N_1,..,N_{k-1}) \leq E(\infty) \tag{5.55}$$

it follows that

$$0 \leq \eta(\cdot) \leq 1 \tag{5.56}$$

It is clear that $\eta(N_1,..,N_{k-1})$ may have identical values for different sets of storage capacities $(N_1,..,N_{k-1})$. However, providing storage space at different locations may have different costs. Thus, the optimization problem of section 1.1 involves also minimizing cost for a given $\eta(\cdot)$.

Although efficiency is known to vary between $E(0)$ and $E(\infty)$, it is important to know the rate at which this increase occurs with respect to buffer capacity. This is because there are cases in which very large buffers can improve production rate significantly; however, in some of these, most of the improvement is achieved with small buffers, while larger buffers do not further increase production rate appreciably.

Okamura and Yamashina[1977] classify curves of efficiency against storage capacity in three groups: those in which the curve is almost linear, those in which it displays a marked concavity, and those that initially rise quickly, but soon approach a limit. These types of curves are illustrated by the graphs of efficiency against storage capacity in figure 5.6 (Gershwin[1973b]). These results are for two-machine lines with the system parameters given in table 5.3.

In case 1, both machines are very efficient. Hence, both $E(0)$ and $E(\infty)$ are close to 1 (and to each other), so that there is little to be gained by the addition of buffer storage between the machines. Since the increase is very gradual, this case is similar to those which Okamura and Yamashina call almost linear. In case 2, the machines are less efficient and neither machine is significantly less efficient than the other. Thus, the addition of buffer storage can be expected to improve the line production rate considerably.

Cases 3 and 4 are interesting to compare. The least efficient machines in these two cases have equal efficiencies, so that the limiting production rates as the storages increase are equal. However, the other machines in these cases have different efficiencies: the second machine in case 3 is more efficient than that in case 4. Consequently, the first machine is more of a bottleneck in case 3 than the first machine in case 4. Thus, the difference between $E(0)$ and $E(\infty)$ is larger in case 4.

On the other hand, for small storages, the two curves have approximately the same slopes. Thus, while in case 3, $\eta(10)=0.28$ and $\eta(20)=0.50$, the effectiveness values for the same storage capacities in case 4 are $\eta(10)=0.16$ and $\eta(20)=0.27$. It is therefore more effective to use relatively small buffers in case 3, although very large buffers gain much more in case 4. Case 3 approaches the limiting value faster, and thus falls into the third class described by Okamura and Yamashina. Case 4, however, displays a marked concavity for a much broader range of storage capacity, and belongs to the second group of curves in their classification.

Case 1:	$p_1 = 0.001$	$p_2 = 0.001$
	$r_1 = 0.04$	$r_2 = 0.04$
Case 2:	$p_1 = 0.01$	$p_2 = 0.01$
	$r_1 = 0.08$	$r_2 = 0.08$
Case 3:	$p_1 = 0.01$	$p_2 = 0.01$
	$r_1 = 0.04$	$r_2 = 0.08$
Case 4:	$p_1 = 0.01$	$p_2 = 0.01$
	$r_1 = 0.04$	$r_2 = 0.04$

Table 5.3. System parameters for two-machine lines.

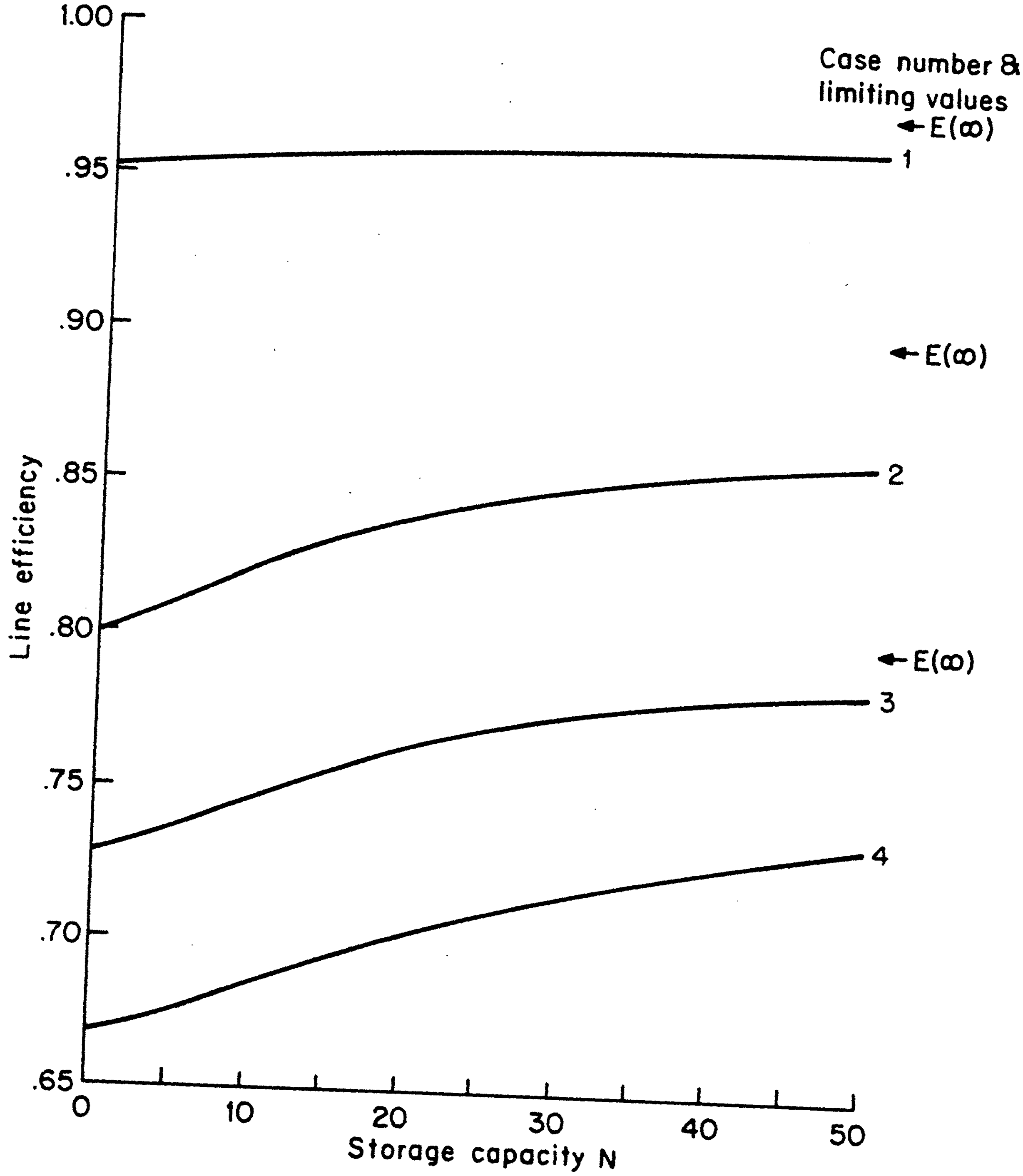


Figure 5.6. Steady-state line efficiency for two-machine transfer lines.

It follows that two numbers are of interest when deciding whether or not to provide buffer storage between stages: while the difference between $E(0)$ and $E(\infty)$ gives the total increase in production rate that can be achieved by buffer storage, the effectiveness $\eta(\cdot)$ indicates what fraction of this total increase is gained by a given storage capacity.

The difference between $E(0)$ and $E(\infty)$ depends on the efficiency in isolation of each machine in the line, i.e. on the <u>ratio</u> of p_i and r_i. The rate at which the efficiency versus storage size curve increases (which determines $\eta(\cdot)$) depends on the <u>magnitudes</u> of these probabilities. For example, if $p_i=r_i=0.1$, $i=1,2$, then equations (5.51) and (5.53) yield that $E(0)=0.33$ and $E(\infty)=0.50$. If $p_i=r_i=0.001$, $i=1,2$, $E(0)$ and $E(\infty)$ have the same values. Thus, the line efficiency can be increased from 0.33 to at amost 0.50 by the addition of buffers in both cases. Yet, in the former case, a storage of capacity $N=4$ yields a production rate equal to $E(4)=0.35938$, corresponding to an effectiveness of $\eta(4)=0.15630$; in the latter, the production rate for the same storage capacity is $E(4)=0.33361$, corresponding to an effectiveness of only $\eta(4)=0.00168$. This is not difficult to explain intuitively: little decoupling can be exercised on the machines in a line by adding a buffer storage if the machines fail extremely rarely, and when they fail, take very long to be repaired. In such cases, relatively small capacity storages empty or fill up in a length of time which is small compared to the total up or down times; their influence on production rate is therefore negligible. If machines fail often and are repaired easily, a small capacity storage may improve production rate significantly. On the other hand, very large storages may improve efficiency significantly in the former case, since they take longer to empty or fill up. Thus, there is a certain relationship between the magnitudes of transition probabilities and storage capacity. This relation is the basis of the δ-transformation outlined in section 6.3.

Results have been obtained for a three-machine line by the methods of section 4.2. Some of these appear in figures 5.7-5.9, where the line efficiency is plotted against the capacity of one of the storages, while the other is varied as a parameter. System parameters are given in table 5.4.

In case 1 (figure 5.7), the last machine is most efficient, so that workpieces produced by the second machine are most often instantly processed by the third machine. Thus, the second storage is often nearly empty, and little is gained by providing it with a large capacity. On the other hand, the efficiency in isolation of the first machine is close to that of the downstream segment of the line (i.e. the portion of the line downstream of it, consisting of machine 2, storage 2, and machine 3). Thus, it is not profitable to provide storage space between machines 2 and 3, though it is useful to provide a buffer between machines 1 and 2.

In case 2 (figure 5.8), the first machine is most efficient. Thus, the first storage is often nearly full, and the downstream segment of the line operates most of the time as if in isolation. On the other hand, the efficiency of the third machine is close to that of the upstream segment of the line (machines 1 and 2, storage 1). Thus, little is gained by providing the first storage with a large capacity, although it is useful to have a large storage between machines 2 and 3.

In case 3 (figure 5.9), all machines have equal efficiencies in isolation, and the effects of added storage capacity are most clearly visible in this case. Furthermore, it is observed that the production rate is symmetrical with respect to the orientation of the system. Since all machines are identical, for example $E(2,5) = E(5,2)$, etc.

These examples indicate once again that storages act best as buffers to temporary fluctuations in the system. If the efficiencies of machines are very different, storages do not improve production rate; if the line is well balanced, the temporary breakdowns are to a certain extent compensated for by buffer storages.

Case 1:	$p_1 = 0.1$	$p_2 = 0.1$	$p_3 = 0.01$
	$r_1 = 0.1$	$r_2 = 0.1$	$r_3 = 0.2$
Case 2:	$p_1 = 0.01$	$p_2 = 0.1$	$p_3 = 0.1$
	$r_1 = 0.2$	$r_2 = 0.2$	$r_3 = 0.2$
Case 3:	$p_1 = 0.1$	$p_2 = 0.1$	$p_3 = 0.1$
	$r_1 = 0.2$	$r_2 = 0.2$	$r_3 = 0.2$

Table 5.4. System parameters for three-machine lines.

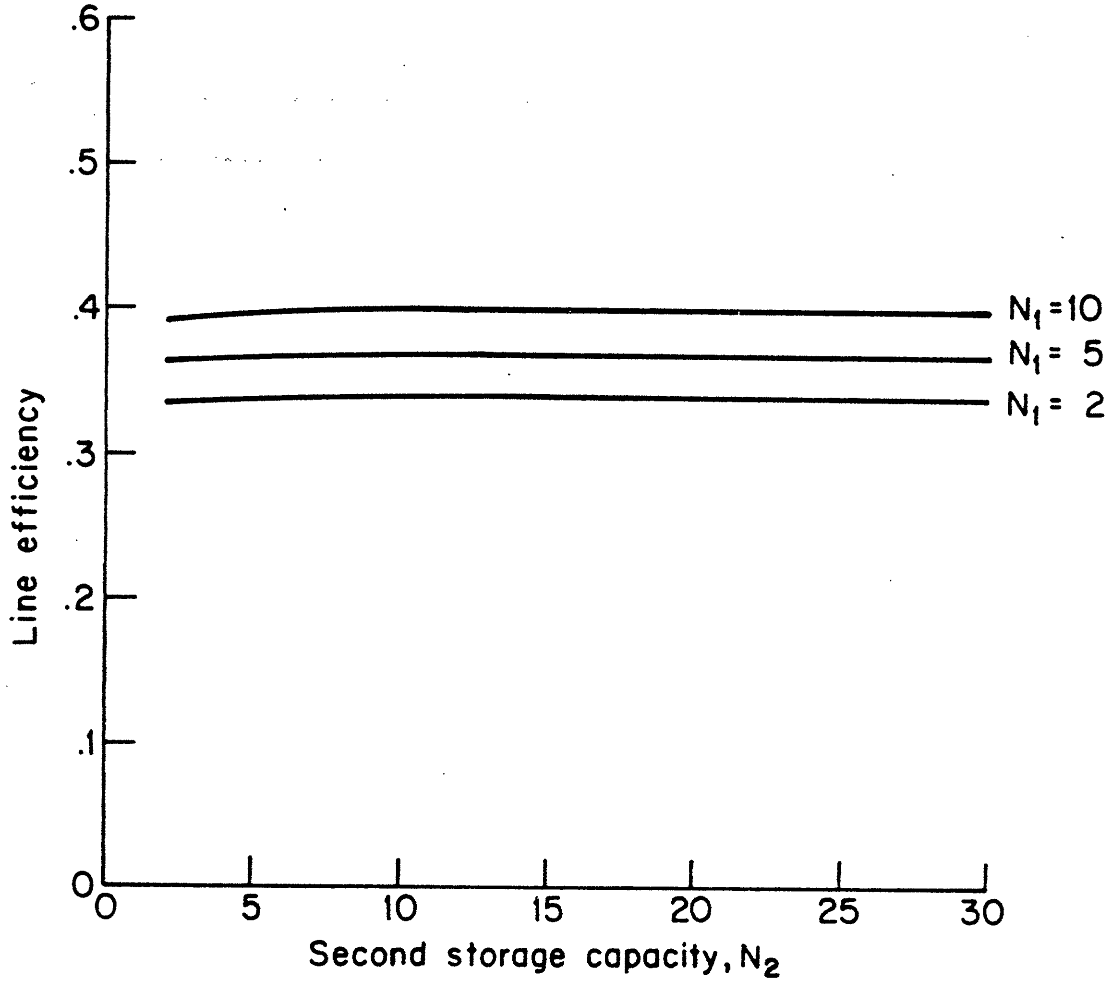


Figure 5.7. Steady-state line efficiency for a three-machine transfer line with a very efficient third machine.

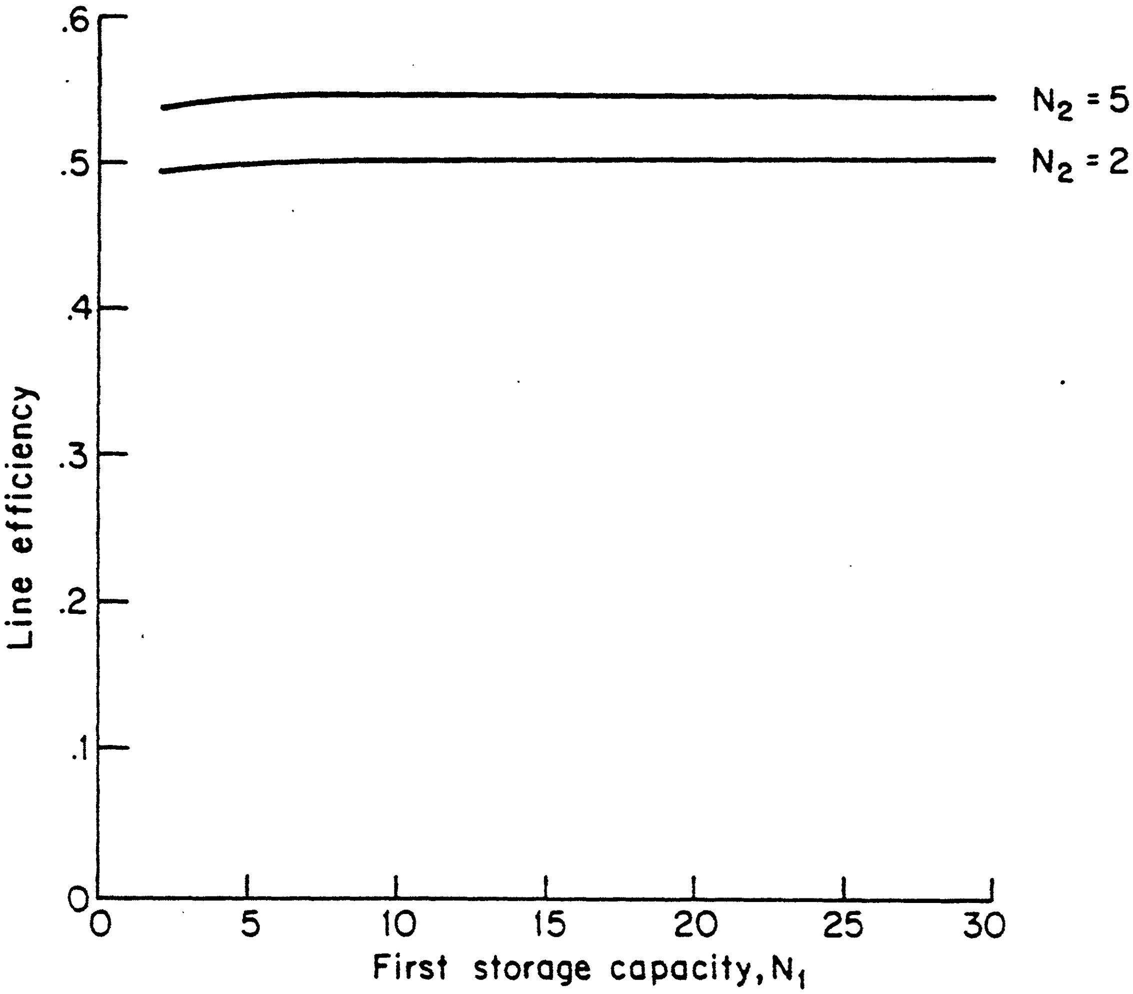


Figure 5.8. Steady-state line efficiency for a three-machine transfer line with a very efficient first machine.

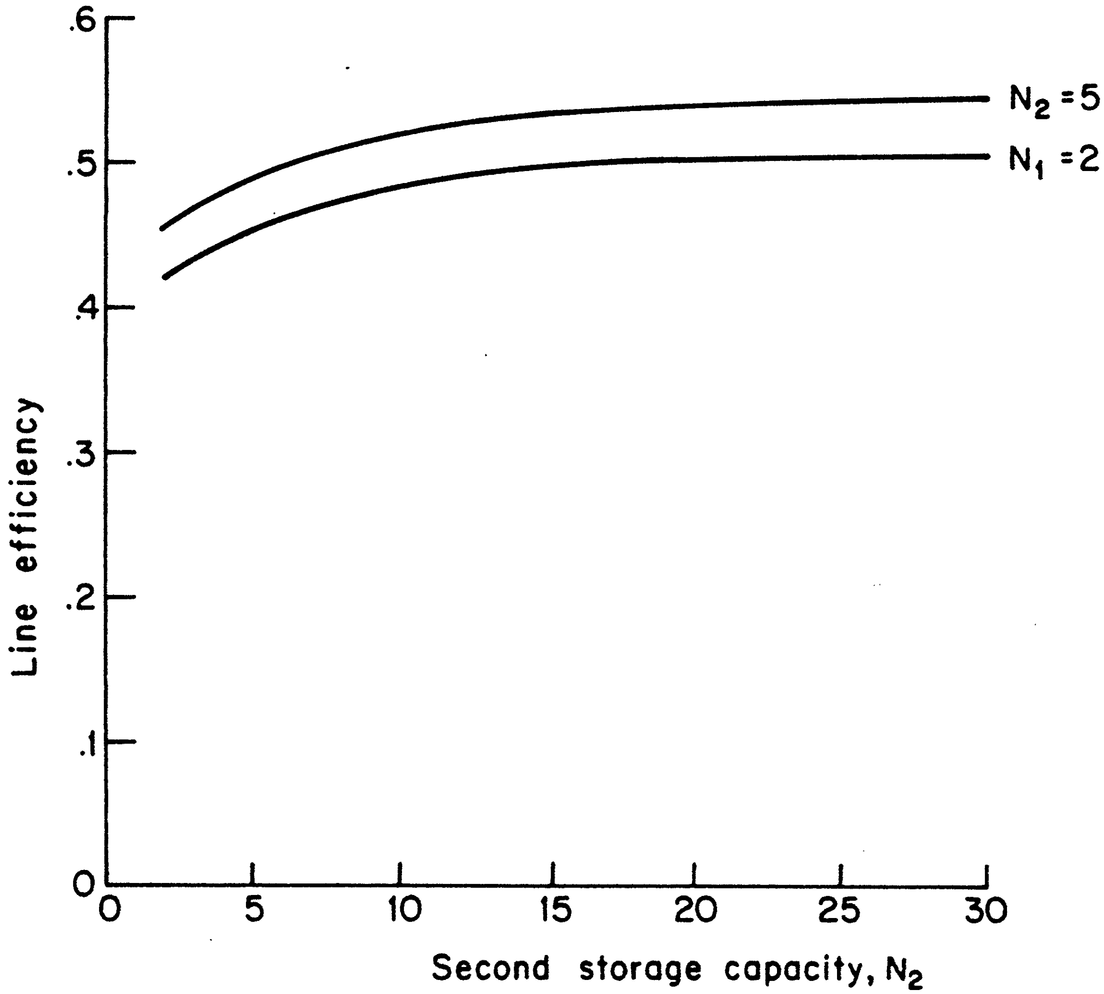


Figure 5.9. Steady-state line efficiency for a three-machine transfer line with identical machines.

5.2 Forced-Down Times, Storage Size, and Efficiency

It is suggested in section 5.1.1 that the rates of flow of workpieces through each machine in a k-machine line are equal at steady-state; this is proved for k=2. In lines with finite storages, the utilization of the machines is always lower than their efficiencies in isolation, since they are occasionally blocked or starved. As storage capacities are increased, the utilizations asymptotically approach the efficiency in isolation of the least efficient machine. This is proved by induction in section 5.1.1. An alternate proof is given in this section, by showing that the forced-down times of the least efficient machine go to zero as storage capacity is increased.

Often, reliability involves increased cost. Thus, it may be undesirable to design and build highly efficient components if they are required to operate within lines involving significantly less efficient components. Since a good measure of how efficient a given component is with respect to the rest of the line is its forced-down times, or alternately the steady-state probabilities that it is idle or blocked, it is necessary to study the relationship between this probability, the efficiency, and storage size.

The two-machine case is discussed here. Similar results may be obtained for longer lines as well.

Although increasing the efficiency of an individual machine has the overall effect of increasing the production rate of the transfer line, this effect is far from simple to calculate. The utilization of the improved machine (and hence, the production rate of the transfer line) does not increase linearly with the efficiency of an individual machine. As is shown below, the effects of system bottlenecks are significant. Since a transfer line may contain less efficient stages, and the line production rate cannot possibly exceed the efficiency of its worst stage, it would appear that the utilization of an individual machine should approach an asymptote as its efficiency is increased.

In addition, it is shown in section 5.1.3 that buffer storages contribute most to the system production rate when the machines are not extremely efficient and no machine is significantly less efficient than the others (i.e. the line is balanced). Such observations are important when designing machines and transfer lines, in that they may provide guidance in decisions involving reliability, storage capacity, and cost tradeoffs. A specific example is analyzed here, and some conclusions are drawn on the effect of buffers and machine efficiencies on each other and on the production rate of the transfer line.

The parameters considered in the numerical example appear in table 5.5. The first machine is not altered, while the efficiency of the second machine is increased (here, this is done by decreasing the failure probability while keeping the repair probability constant). The performance measures sought are machine utilization and forced-down times (alternately, the probabilities of being starved or blocked).

In case 1, the efficiency in isolation of the second machine is very low (e_2=0.15). The efficiency of machine 2 is increased, past that of the first machine (when e_1=e_2=0.50 in case 3) up to e_2=0.85 in case 5. Thus, the system bottleneck is machine 2 in cases 1 and 2, and machine 1 in cases 4 and 5. This is well illustrated by the graphs of line efficiency and probability of blocking and starving appearing in figures 5.10-5.13.

The line efficiency is plotted against storage capacity for each of the five cases in figure 5.10. In cases 3-5, the value of $E(\infty)$ is the same, since the least efficient machine is the first one (and it is not altered). In cases 1-2, on the other hand, the least efficient machine is the second one. Thus, $E(\infty)$ changes as e_2 is varied. This effect is clearly seen in figure 5.11, where the line efficiency is plotted against the efficiency in isolation of the second machine, e_2, for various values of storage capacity. The production rate increases with e_2 until $e_2 \approx e_1$, after which the first machine acts as a bottleneck and the production rate approaches an asymptote. Thus, beyond a certain point, increasing the efficiency of the second machine becomes less and less effective. This result agrees with those for the flow through

All cases:	$p_1 = 0.1$	$r_1 = 0.1$
Case 1:	$p_2 = 0.567$	$r_2 = 0.1$
Case 2:	$p_2 = 0.2$	$r_2 = 0.1$
Case 3:	$p_2 = 0.1$	$r_2 = 0.1$
Case 4:	$p_2 = 0.05$	$r_2 = 0.1$
Case 5:	$p_2 = 0.018$	$r_2 = 0.1$

Table 5.5. System parameters for two-machine lines.

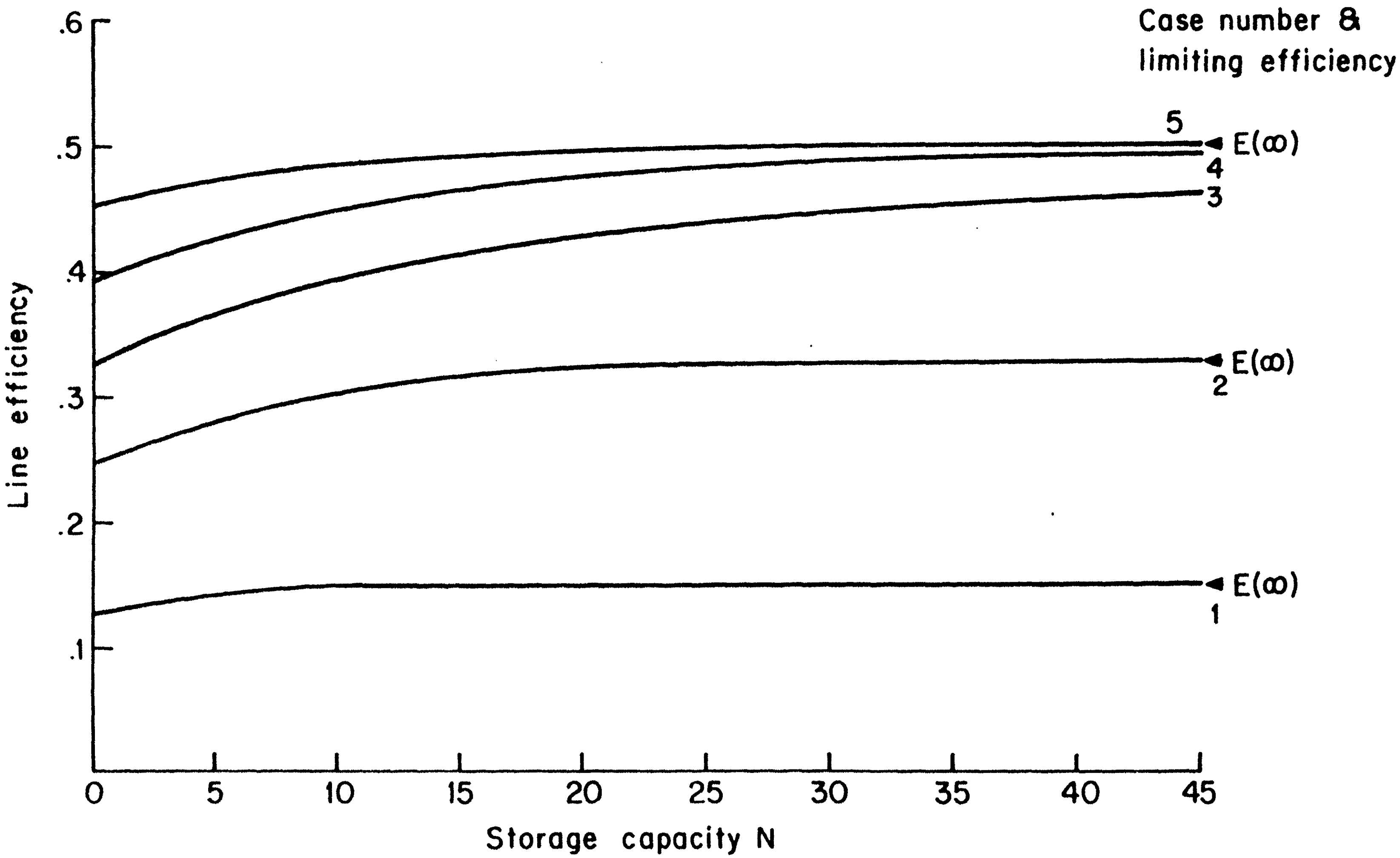
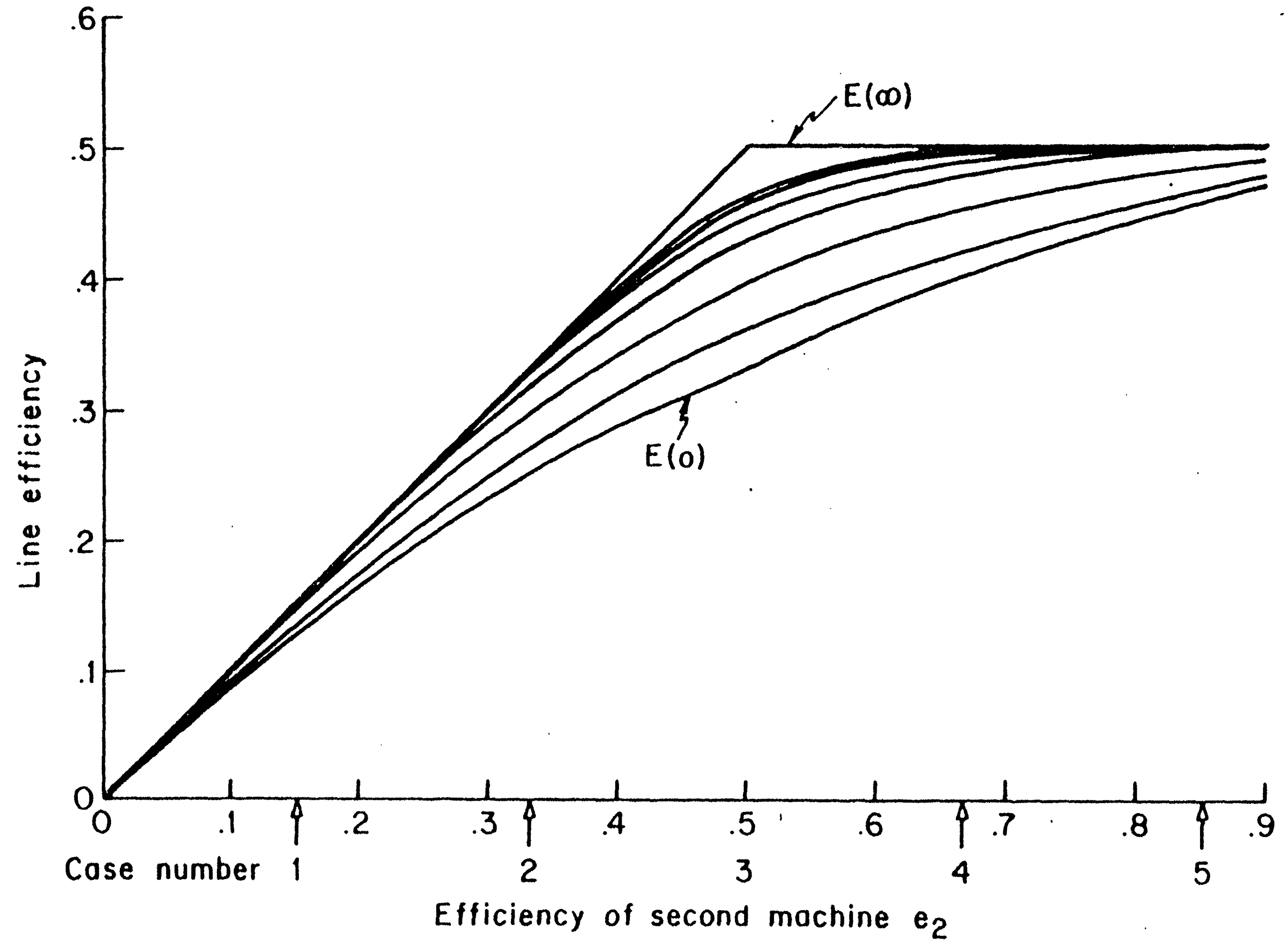


Figure 5.10. Steady-state line efficiency for two-machine lines with the same first machine and different second machines.



5.11. Steady-state line efficiency plotted against the efficiency in isolation of the second machine, for two-machine lines with identical first machines. (The curves are for N = 0, 4, 10, 20, 30, 40, 50, ∞)

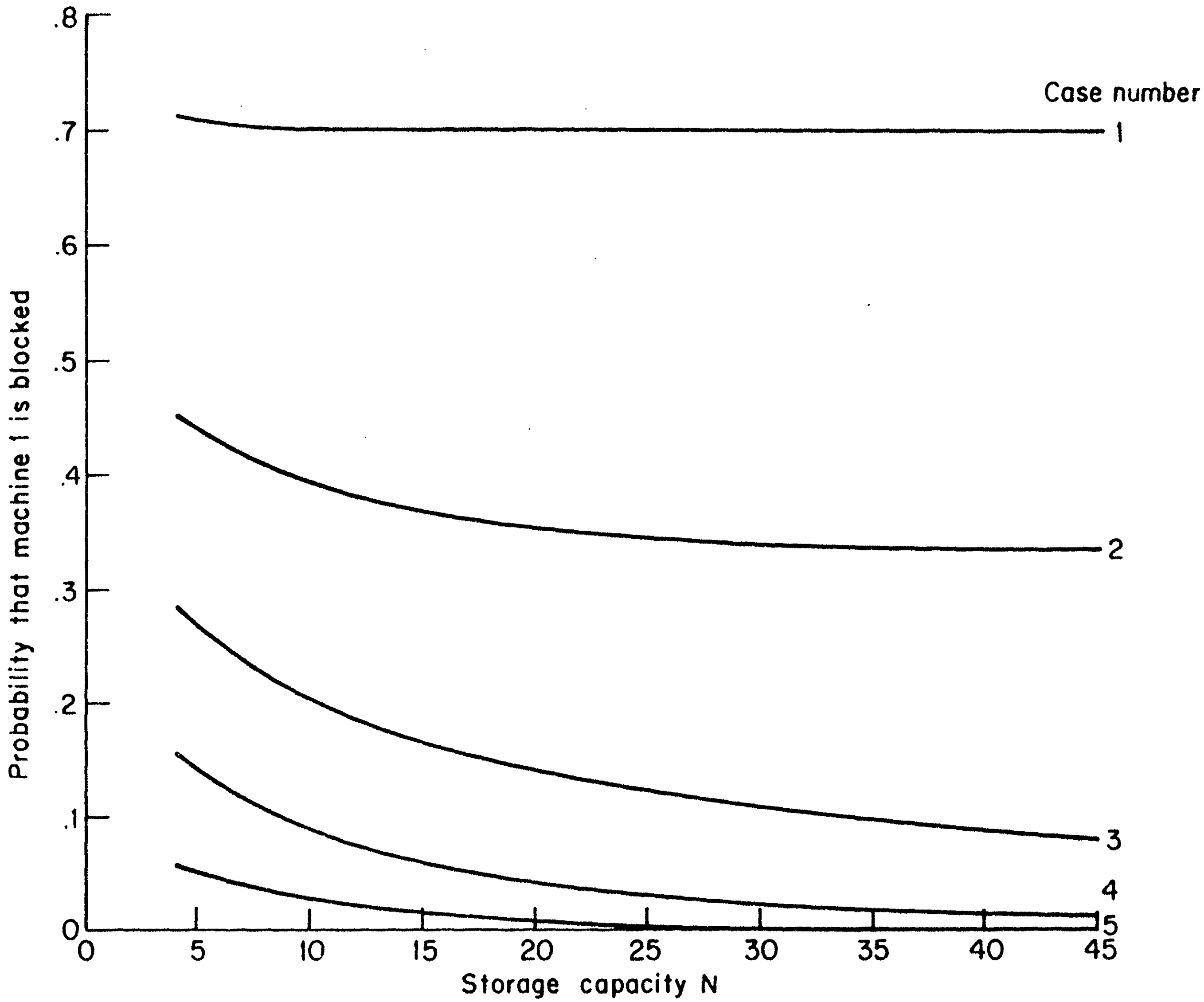


Figure 5.12. The probability that the first machine in a two-machine line is blocked, plotted against storage capacity.

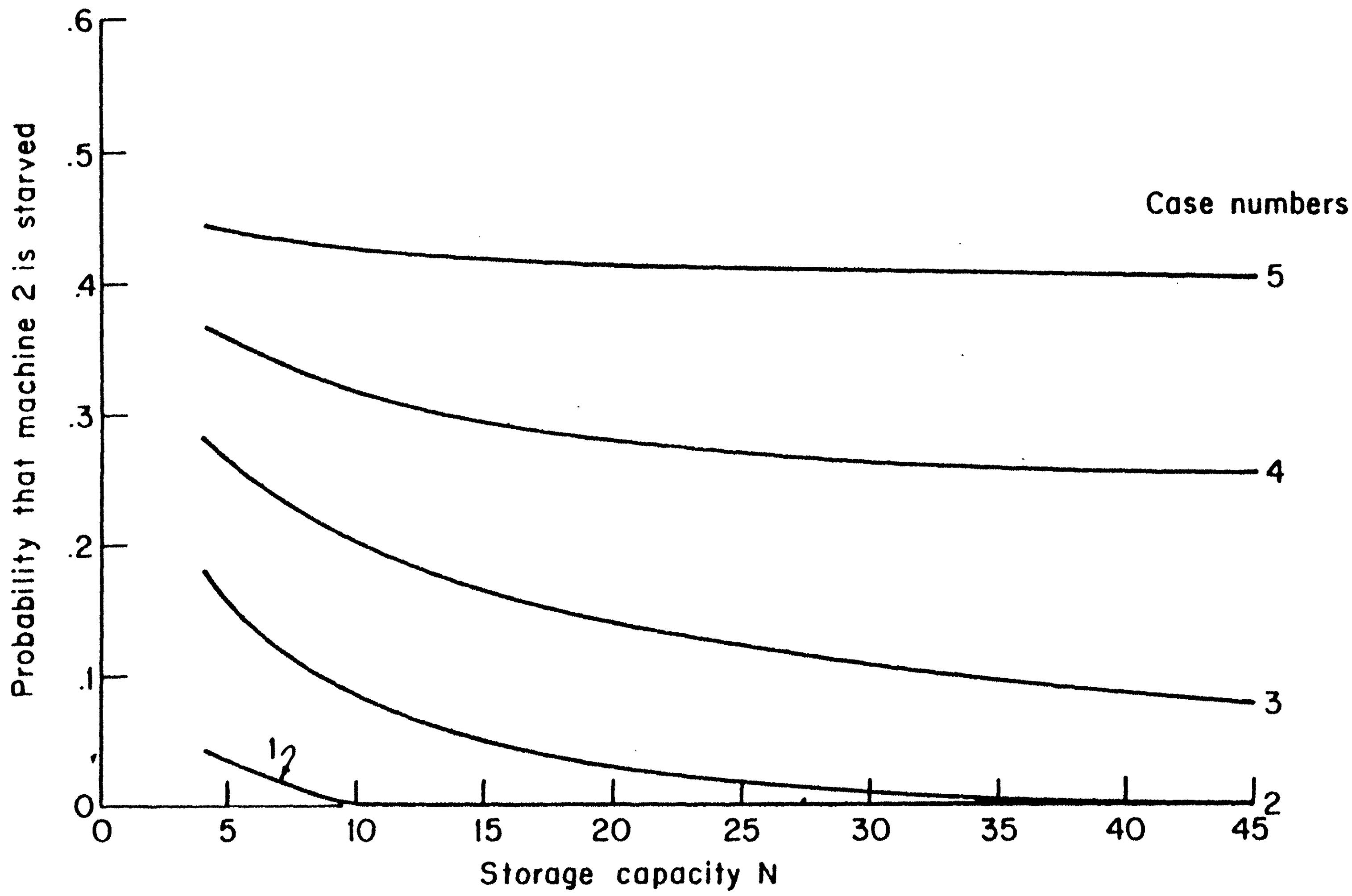


Figure 5.13. The probability that the second machine in a two-machine line is starved, plotted against storage capacity.

networks of queues conducted by Kimemia and Gershwin[1978]. It is found that in general, when a given attribute is limiting, the flow through the network increases linearly with that attribute; as the attribute increases, it is no longer limiting and some other attribute is. Thus, the flow rate reaches an asymptote.

It is noteworthy that for a certain range of e_2, it appears that providing small amounts of storage can improve the production rate as much as increasing e_2; for example, e_2=0.67 and no storage gives approximately the same efficiency as e_2=0.6 and N=4, or e_2=0.5 and N=10. This is significant, because improving the efficiency of a machine may involve a great deal of research and capital investment or labor costs, and may thus be more expensive than providing a small amount of buffer capacity. It is especially important that this effect is strongest when the machines have approximately the same efficiency, i.e. when the line is balanced. Since this is most often the case in industry (although deliberately unbalancing a line may at times be profitable - see Rao[1975b], Hillier and Boling[1966]), the fact that increasing buffer capacity is most effective when the line is balanced is of great importance.

Figures 5.12 and 5.13 are also revealing, in that they show the dependence of forced-down times on the efficiency of the second machine and the storage capacity. The probability that the first machine is blocked (p[N,1,0]) is plotted against storage capacity in figure 5.12. It is seen that this probability approaches a positive asymptote when the second machine is least efficient, and hence the bottleneck. It approaches zero when the first machine is least efficient, so that as the storage capacity is allowed to increase without bound, the first machine is fully utilized because it is the system bottleneck. This result agrees with the findings of Secco-Suardo[1978] and Kimemia and Gershwin[1978]: as the speed (and thus the production rate in isolation) of a machine increases, the average size of the queue decreases.

Conversely, the probability that the second machine is starved (p[0,0,1]) is plotted against storage capacity in figure 5.13, approaches a positive asymptote when the first machine is limiting. When the second machine is the system bottleneck, this probability approaches zero as storage increases.

This may be demonstrated analytically by using the two machine state probability expressions derived in chapter 3. In the case where the first machine is more efficient than the second, $(p_1/r_1)<(p_2/r_2)$, so that $Y_1<Y_2$ and $X>1$ (See equation (3.25)). Now from table 3.1,

$$p[0,0,1] = CX \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2 r_1} \tag{5.57}$$

where C is chosen so that the probabilities sum up to one. Thus,

$$\lim_{N\to\infty} \frac{1}{C} = \sum_{n=2}^{\infty} X^n (1+Y_1)(1+Y_2) + \ldots \tag{5.58}$$

where this first term is sufficient to guarantee that $C\to 0$ as $N\to\infty$ (since $X>1$ and all other terms are positive). Thus, for $e_1>e_2$,

$$\lim_{N\to\infty} p[0,0,1] = 0 \tag{5.59}$$

On the other hand, in the case where the first machine is less efficient than the second, $(p_1/r_1)>(p_2/r_2)$, so that $Y_1>Y_2$ and thus, $X<1$ (equation (3.25). From table 3.1,

$$p[N,1,0] = CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2} \tag{5.60}$$

Since $X^{N-1}\to 0$ as $N\to\infty$ for $X<1$, it is sufficient to show that C does not tend towards infinity as N is increased. Neglecting terms containing X^{N-1} (because $X<1$), the limit for $1/C$ is written as

$$\lim_{N\to\infty} \frac{1}{C} = \sum_{n=2}^{\infty} X^n (1+Y_1)(1+Y_2) + X(1+Y_2) + \ldots + X\left[\frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2 r_1} + \frac{1}{p_2}\,\frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 + p_2 - p_1 p_2 - p_2 r_1}\right] \tag{5.61}$$

Since for $X<1$,

$$\sum_{n=0}^{\infty} X^n = \frac{X}{1-X} \tag{5.62}$$

equation (5.61) has a non-zero right hand side. Thus, C is bounded for $e_1<e_2$ and

$$\lim_{N\to\infty} p[N,1,0] = 0 \tag{5.63}$$

These proofs show once again, as stated in section 5.1.3, that at least for the two-machine case, the infinite-buffer production rate is such that the least efficient machine is never forced down. This implies that it is equal to the efficiency in isolation of the least efficient machine.

5.3 In-Process Inventory and Storage Size

The cost of providing storage may increase linearly with its capacity, in terms of floor space etc. However, the cost incurred by maintaining in-process inventory is not linear with buffer capacity. Calculating the expected number of workpieces in a storage or in the entire production line therefore involves the use of state probabilities. The expected inventory in storage i is given by

$$I_i = \sum_{n_1=0}^{N_1} \cdots \sum_{n_{k-1}=0}^{N_{k-1}} \sum_{\alpha_1=0}^{1} \cdots \sum_{\alpha_k=0}^{1} p[n_1,\ldots,n_{k-1},\alpha_1,\ldots,\alpha_k] \cdot n_i \quad (5.70)$$

Solving the buffer size optimization problem described in section 1.1 involves the cost of maintaining in-process inventory. This cost must be calculated on the basis of the expected inventory, as given in equation (5.70).

Two-machine and longer lines are reviewed here.

Okamura and Yamashina[1977] observe that for large enough buffer capacities, an increase in the capacity does not necessarily imply an increase in the expected number of pieces in the storage. This is illustrated by the results presented in figures 5.14 and 5.15. These are for a two-machine line with state parameters as given in table 5.5.

In figure 5.14, the expected number of pieces in the storage is plotted against storage capacity. In cases 1 and 2, the first machine is more efficient than the second, and the expected in-process inventory is seen to increase with storage capacity. In case 3, the two machines have equal efficiencies, and the expected inventory increases linearly with storage capacity. In cases 4 and 5, the second machine is more efficient than the first, and the expected inventory approaches an asymptote. This is even more evident in figure 5.15, where the expected in-process inventory as a fraction of the storage capacity is plotted against storage size. These curves are seen to approach limiting values.

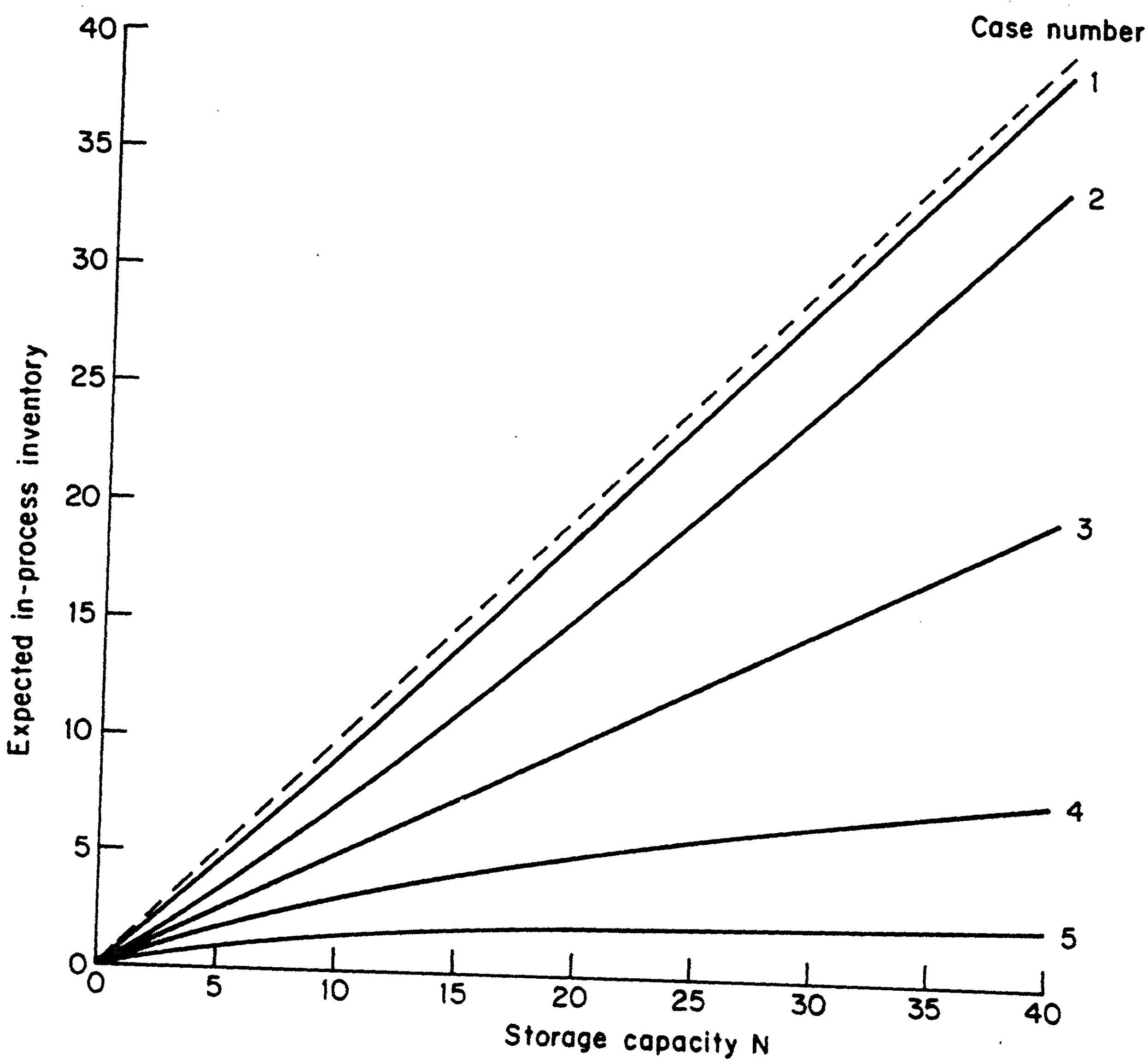


Figure 5.14. Expected in-process inventory plotted against storage capacity, for two-machine lines with identical first machines.

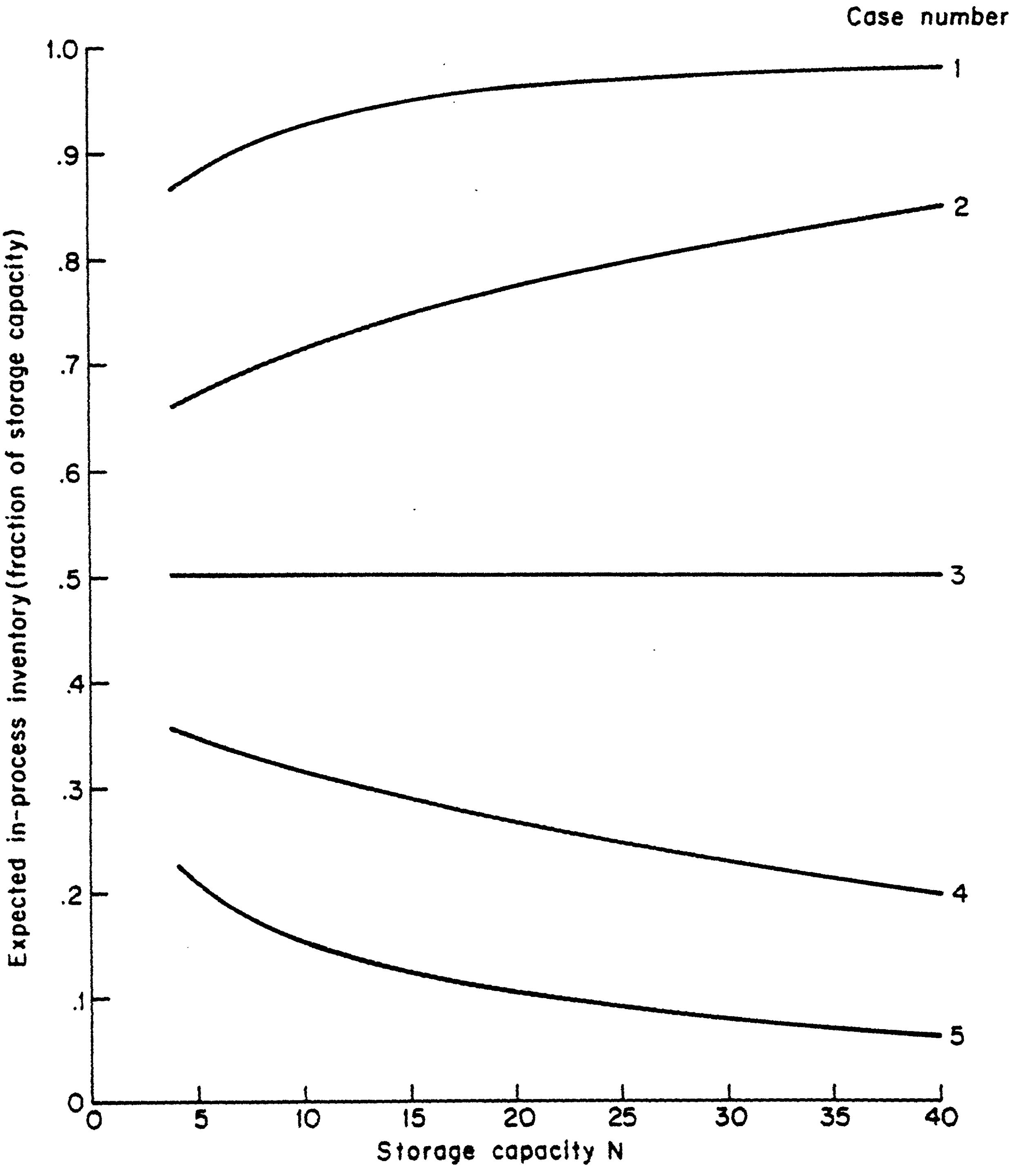


Figure 5.15. Expected in-process inventory as a fraction of storage capacity plotted against storage capacity, for two-machine lines with identical first machines.

That the expected inventory approaches an asymptote when the second machine is more efficient than the first ($e_2 > e_1$) may be proved analytically by using equation (5.70) and the results of chapter 3. As in section 5.2, it is first noted that for $e_1 < e_2$, $X<1$ and C approaches a limiting value as $N \to \infty$ (See equations (5.61) and (5.62)). Furthermore, for $X<1$,

$$\lim_{N \to \infty} X^{N-1} (N-1) = 0 \tag{5.71}$$

$$\lim_{N \to \infty} X^{N-1} N = 0 \tag{5.72}$$

For the two-machine case, equation (5.70) may be explicitely written as

$$\begin{aligned} I = & \sum_{n=2}^{N-2} CX^n (1+Y_1)(1+Y_2)\, n + CX + CXY_2 + \\ & \frac{CX}{p_2} \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 + p_2 - p_1 p_2 - p_2 r_1} + CX^{N-1}(N-1) + \\ & CX^{N-1} Y_1 (N-1) + \frac{CX^{N-1}}{p_1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - p_1 r_2} (N-1) \\ & + CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2} N \end{aligned} \tag{5.73}$$

From equations (5.71) and (5.72), it is seen that the last four terms approach zero as $N \to \infty$. Furthermore, since C approaches a constant as $N \to \infty$, the second, third, and fourth terms also approach constants. Thus, to prove that I approaches an asymptote as $N \to \infty$, it is sufficient to show that the first term approaches a limiting value. This is done as follows:

$$\begin{aligned} \sum_{n=2}^{N-2} n\, X^n &= X \sum_{n=2}^{N-2} n\, X^{n-1} \\ &= X \frac{d}{dX} \sum_{n=2}^{N-2} X^n \end{aligned}$$

$$= X \frac{d}{dx}\left[\frac{X^2 - X^{N-1}}{1 - X}\right]$$

$$= X \left[\frac{2X - (N-1)X^{N-2}}{1 - X} + \frac{X^2 - X^{N-1}}{(1 - X)^2}\right] \qquad (5.74)$$

As $N\to\infty$, equation (5.74) reduces to

$$\lim_{N\to\infty} \sum_{n=2}^{N-2} n\, X^n = \frac{2X^2}{1 - X} + \frac{X^3}{(1 - X)^2} \qquad (5.75)$$

Thus, I approaches an asymptote as $N\to\infty$.

An important consequence follows from this: In cases with $e_1<e_2$, added storage capacity is utilized less and less as the storage capacity increases. This asymptotic behavior is similar to that exhibited by production rate as a function of storage capacity (Section 5.1.3). As $N\to\infty$, increasing the storage capacity becomes less useful and contributes less to improving the system efficiency. How quickly the line efficiency approaches the limiting value is related to the speed with which I_i approaches the limit. However, it seems incorrect to say, as Okamura and Yamashina[1977] do, that in general, curves of efficiency and expected inventory against buffer size have the same shape. This should be obvious from cases 1 through 3 in figure 5.14.

While in the two-machine case, the expected in-process inventory depends on the relationship between the efficiencies in isolation of the upstream and downstream machines, this is not so in longer lines. In general, the expected inventory in storage i depends on the efficiencies in isolation of the upstream segment of the line (machines 1 through i and the storages between them) and the downstream segment of the line (machines i+1 through k, and the storages between them).

As a consequence, the expected inventory of storage i increases if the capacity of an upstream storage is increased, since that has the effect of increasing the efficiency of the upstream segment of the line. Similarly, the expected inventory in storage i decreases if the capacity of a downstream storage is increased, since that has the effect of increasing the efficiency of the downstream segment of the line.

This is illustrated by the results plotted in figures 5.16 and 5.17 for a three-machine line with parameters given in table 5.6. In figure 5.16, the capacity of storage 2 is increased, and the expected inventory as a fraction of storage capacity in storage 1 is seen to decrease. Since the production rate of the downstream portion of the line (machines 2 and 3, storage 2) approaches an asymptote as $N_2 \to \infty$, the expected inventory in storage 1 also approaches an asymptote. In figure 5.17, the capacity of storage 1 is increased, and the expected inventory as a fraction of storage capacity in storage 2 is seen to increase. Again, since the production rate of the upstream portion of the line approaches an asymptote as $N_1 \to \infty$, the expected inventory in storage 2 also approaches an asymptote.

This has an important consequence: how effective a buffer is generally depends on its utilization by the system. Thus, if a storage is very often empty or full, it serves little purpose in the line (Buzacott[1967a]). It follows that altering the capacities of upstream or downstream storages affects the contribution of a given storage to the production rate of the transfer line. It is necessary to consider this interaction between storages while computing the optimal buffer capacity allocation for a system.

$p_1 = 0.1$	$r_1 = 0.2$
$p_2 = 0.1$	$r_2 = 0.2$
$p_3 = 0.1$	$r_3 = 0.2$

Table 5.6. System parameters for a three-machine line.

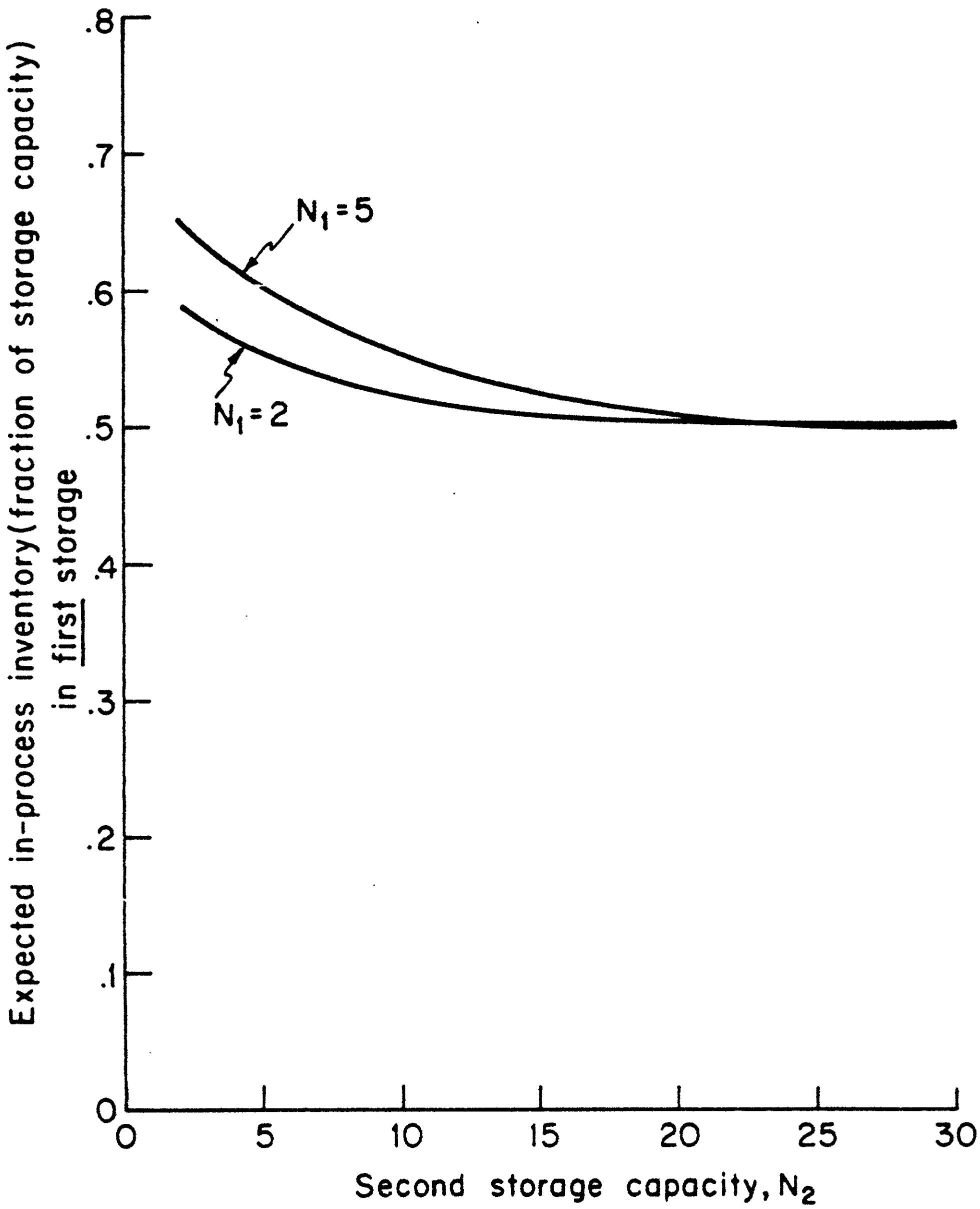


Figure 5.16. Expected in-process inventory as a fraction of storage capacity in the first storage plotted against the capacity of the second storage, for a three-machine line.

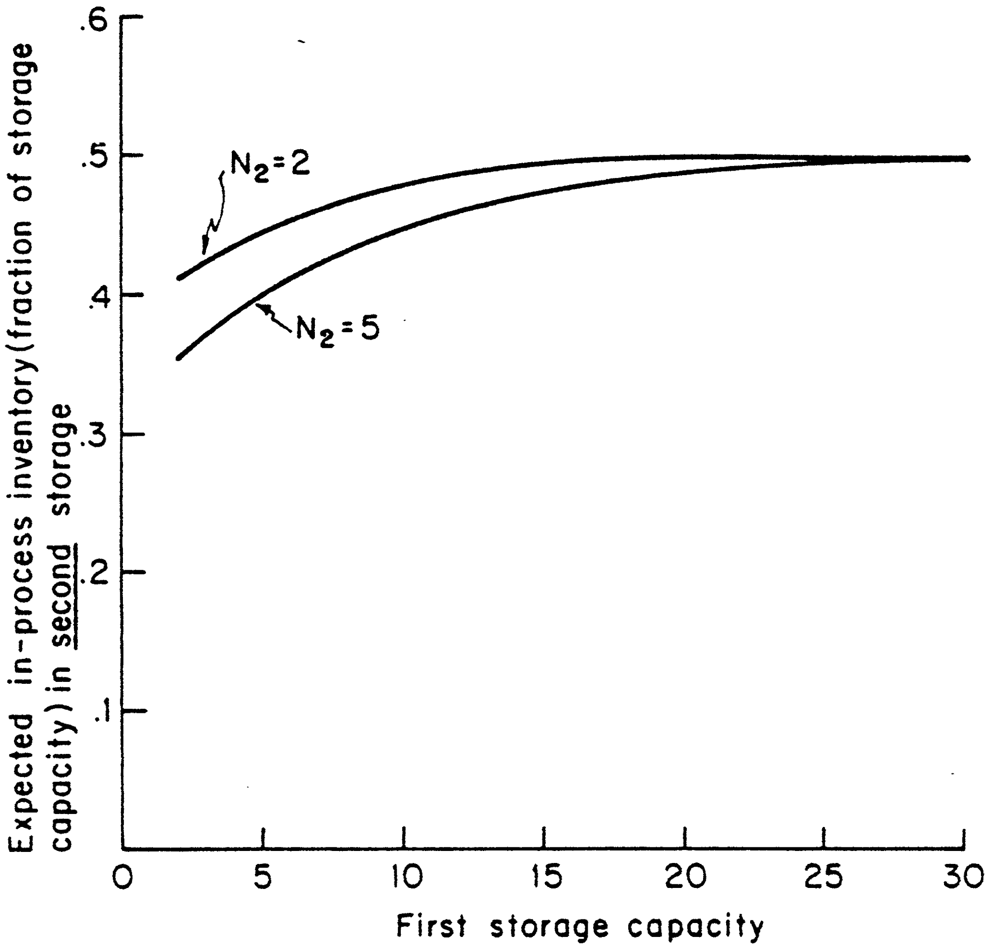


Figure 5.17. Expected in-process inventory as a fraction of storage capacity in the second storage plotted against the capacity of the first storage, for a three-machine line.

6. APPROXIMATE METHODS FOR SOLVING MULTISTAGE LINE PROBLEMS

An important difficulty in calculating the production rates of transfer lines with more than two machines and relatively large storages is that the state space very rapidly reaches intractable dimensions. From equation (2.22), the number of states for a k-machine line with storage capacities $N_1,..,N_{k-1}$ is given by

$$m = 2^k (N_1 + 1) \dots (N_{k-1} + 1) \tag{6.1}$$

Some examples of only moderately large problems are given in table 6.1. Considering the fact that certain processes, for example in the automotive industry, may involve many tens of machines, it becomes extremely difficult or even impossible to solve the problem exactly, whether by the analytical methods derived in chapter 3, or by numerical approaches outlined in chapter 4 (See Buzacott[1969]).

In such cases, approximate methods such as computer simulation are often used. Simulations can often be expensive and inefficient, although the particular details of specific systems can better be considered in simulation than in analytical approaches. A simulation program that corresponds exactly to the model dexcribed in chapter 2 is reviewed in section 6.1. This program (See Appendix A.5) was used at various stages of the research, both for gaining insight into the behavior of the system, and for checking the validity of analytical and numerical results obtained.

An aggregate method that lumps two-machine, one-storage segments of longer transfer lines into almost equivalent single machines is discussed in section 6.2. Although the agreement with exact results is best when the line is unbalanced (rarely the case in practice), the accuracy for balanced lines may be satisfactory for many applications.

Based on the relationship between the magnitude of failure and repair

k	N_1	N_2	N_3	m
2	10	-	-	44
2	100	-	-	404
3	10	10	-	968
3	100	100	-	81,608
4	10	10	10	21,296
4	100	100	100	16,484,816

Table 6.1. The number of system states in a k-machine transfer line with buffer storage capacities $N_1,..,N_{k-1}$.

probabilities and storage size described in section 5.1.3, the δ-transformation is introduced in section 6.3. This approximate approach effectively lumps workpieces together, and reduces the capacity of the storages. The dimensions of the state space are thereby reduced, while the transformation leaves the line efficiency virtually unchanged. An intuitive explanation of the transformation is given, followed by some numerical examples. The transformation is taken to its limit as $\delta \to 0$ in section 9.2.

6.1 Dynamic Simulation of the System

6.1.1 State Frequency Ratios

To test the hypothesis that the solution to a three-machine line for internal states has a product form (rather than a sum-of-products form - see chapter 3), the state frequencies $\omega[\cdot]$ obtained by simulation were used to calculate the following ratios:

$$\left[\frac{\omega[n_1+n,n_2,\alpha_1,\alpha_2,\alpha_3]}{\omega[n_1,n_2,\alpha_1,\alpha_2,\alpha_3]}\right]^{\frac{1}{n}} = \hat{X}_1 \qquad (6.2)$$

$$\left[\frac{\omega[n_1,n_2+n,\alpha_1,\alpha_2,\alpha_3]}{\omega[n_1,n_2,\alpha_1,\alpha_2,\alpha_3]}\right]^{\frac{1}{n}} = \hat{X}_2 \qquad (6.3)$$

$$\frac{\omega[n_1,n_2,1,\alpha_2,\alpha_3]}{\omega[n_1,n_2,0,\alpha_2,\alpha_3]} = \hat{Y}_1 \qquad (6.4)$$

$$\frac{\omega[n_1,n_2,\alpha_1,1,\alpha_3]}{\omega[n_1,n_2,\alpha_1,0,\alpha_3]} = \hat{Y}_2 \qquad (6.5)$$

$$\frac{\omega[n_1,n_2,\alpha_1,\alpha_2,1]}{\omega[n_1,n_2,\alpha_1,\alpha_2,0]} = \hat{Y}_3 \qquad (6.6)$$

(where n_i and n_i+n are all internal). If there were only one term in equation (3.13), these estimates would be very close to the true values for X_i and Y_j, for long enough simulation runs. $\hat{X}_i$ and $\hat{Y}_j$ were calculated for all pairs of internal states, and their averages and variances were computed. These values for the parameters in table 6.2 appear in table 6.3. The variances were seen to decrease but did not vanish. This suggests that $\ell\neq 1$ in equation (3.13), i.e. that the internal state probabilities have a sum of products, rather than a product form. Similar calculations were subsequently performed on exact numerical results obtained by the power method (Section 4.1). The values obtained confirmed these findings.

$p_1 = 0.10$	$r_1 = 0.20$	
		$N_1 = 10$
$p_2 = 0.05$	$r_2 = 0.20$	
		$N_2 = 10$
$p_3 = 0.05$	$r_3 = 0.15$	

Table 6.2. System parameters for dynamic simulation.

Estimates of Parameters	Averages	Sample Variances
$\hat{X}_1$	0.937	0.0084
$\hat{X}_2$	0.974	0.0082
$\hat{Y}_1$	2.46	0.241
$\hat{Y}_2$	3.24	0.426
$\hat{Y}_3$	2.44	0.434

Table 6.3. Estimates of parameters computed by taking ratios of state frequencies from simulation results.

6.1.2 System Transients

In order to check the validity of approximate methods, as well as to see how well steady-state results represent the behavior of the actual system for finite-time runs, sample averages of production rate over time intervals of various lengths were calculated. These are the ratios of the number of pieces produced during a time interval to the interval length. The cumulative average, i.e. the ratio of the total number of pieces produced to the total time elapsed was also calculated. these results are discussed in section 5.1.2. It is shown that while the cumulative average approaches the steady-state value for long times, the rate at which it approaches this value depends strongly on system parameters.

It must be noted that the sample average production rates calculated in this way are not uncorrelated: although intervals do not overlap, they follow eachother immediately. More nearly independent sample averages could be obtained by skipping alternate time intervals.

Steady-state values may also be misleading when considering the loading and unloading of the transfer line. Assumption 2.2.1 states that parts are always available to the first machine and storage space is always available to the last machine in the line. Thus, the line is assumed never to be blocked or starved. In practice, however, it is often the case that workpieces are delivered to and finished parts are removed from the transfer line area in batches. Thus, in actual systems, there usually are input and output queues, i.e. external buffers, upstream and downstream of the line. It is therefore necessary to design these buffers and schedule deliveries to and from the line in such a way that the probability of starving or blocking the line is very nearly zero.

This may be done by obtaining the distribution of up times and down times for the transfer line as a whole. The output distribution determines the probabilities of producing exactly 1,2,..,n,.. pieces consecutively, and of not producing pieces for exactly 1,2,..,n,.. time cycles consecutively. The input distribution determines the probabilities of taking in exactly 1,2,..,n,.. pieces consecutively, and of not taking pieces in for exactly 1,2,..,n,.. time cycles consecutively. It is important to note that these distributions are not uniquely determined by the steady-state line production rate. For example, a line which produces an average of 1000 pieces consecutively and is down for an average of 500 consecutive time cycles has a steady-state efficiency of 0.667. A line which produces an average of 10 pieces consecutively and is down for an average of 5 consecutive time cycles has the same steady-state efficiency. Yet, in the former case, a very large external buffer is needed to ensure that the line is almost never blocked; in the latter, a much smaller buffer is sufficient. Furthermore, although the average rates of flow through the first and last machines are equal (for finite storages - see section 5.1.1), the input and output distributions are generally not the same.

A method for obtaining these distributions analytically is described in section 6.2. These distributions are used in an approximate approach for finding the production rates of long transfer lines.

6.2 An Aggregate Method for Obtaining the Production Rate of Multistage Transfer Lines

6.2.1 Quasi-Geometric Input/Output Distributions of a Two-Machine Line

Sevast'yanov[1962] describes an approximate procedure for solving problems involving multistage transfer lines where storage level is modeled as a continuous variable. He bases his method on the observation that as articles move in the downstream direction, there is an equal but reverse flow of "anti-articles", or holes, in the upstream direction. It may be noted that Gordon and Newell[1967b] independently introduce the concept of duality on the basis of the same observation in their work on closed cyclic queueing systems. Basing himself of Sevast'yanov's work, Buzacott[1967b] describes a method for approximating a three-machine line by a two-machine line: this can be done by dividing the line into two stages, either at the first storage, or at the second one. Buzacott states that this method can be applied if the two-stage line up-time distribution is not far from geometric and the stage repair distributions are identical.

In order to verify the applicability of the first of these conditions, the simulation program was used as described in section 6.1.2. The program was designed to record the numbers of times that in a run of given length, a two-machine line produced parts for exactly 1,2,..,n,.. consecutive cycles, as well as the numbers of times that it failed to produce parts for exactly 1,2,..,n,.. consecutive cycles. These quantities are normalized to give the frequencies of producing (or failing to produce) parts for 1,2,..,n,.. consecutive cycles, given that it produced (or failed to produce) for at least one cycle. Results are plotted in figures 6.1-6.3 for a given two-machine line (See table 6.4) and three different storage capacities. The logarithms of the frequencies of producing exactly n parts given that at least one part has been produced are plotted against n, the number of parts produced. Since the down-times are not

p_1	=	0.10	r_1	=	0.20
p_2	=	0.05	r_2	=	0.20
Case 1:			N	=	4
Case 2:			N	=	8
Case 3:			N	=	16

Table 6.4. System parameters for output distributions of a two-machine line.

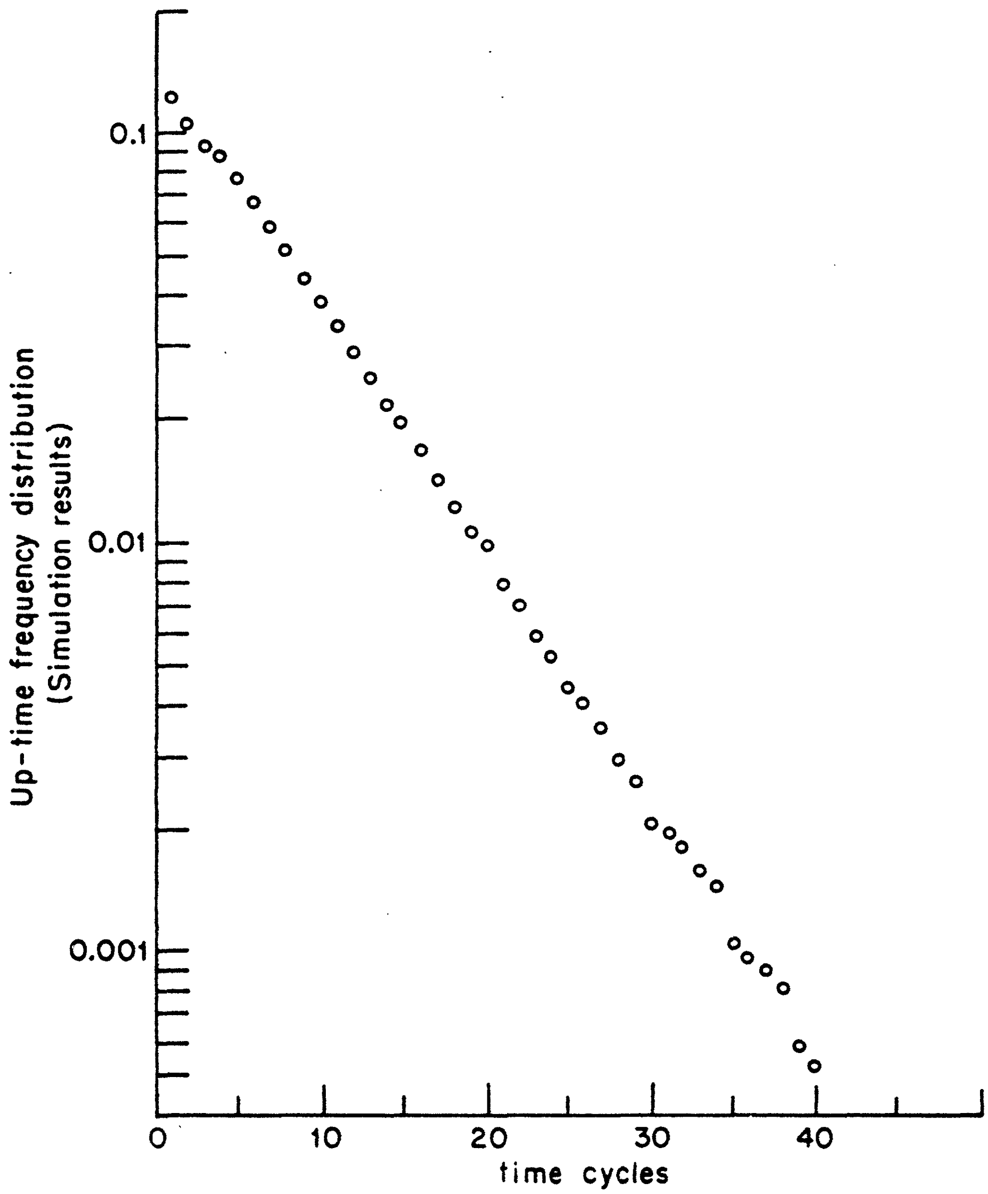


Figure 6.1. Up-time frequency distribution for a two-machine line with N = 4.

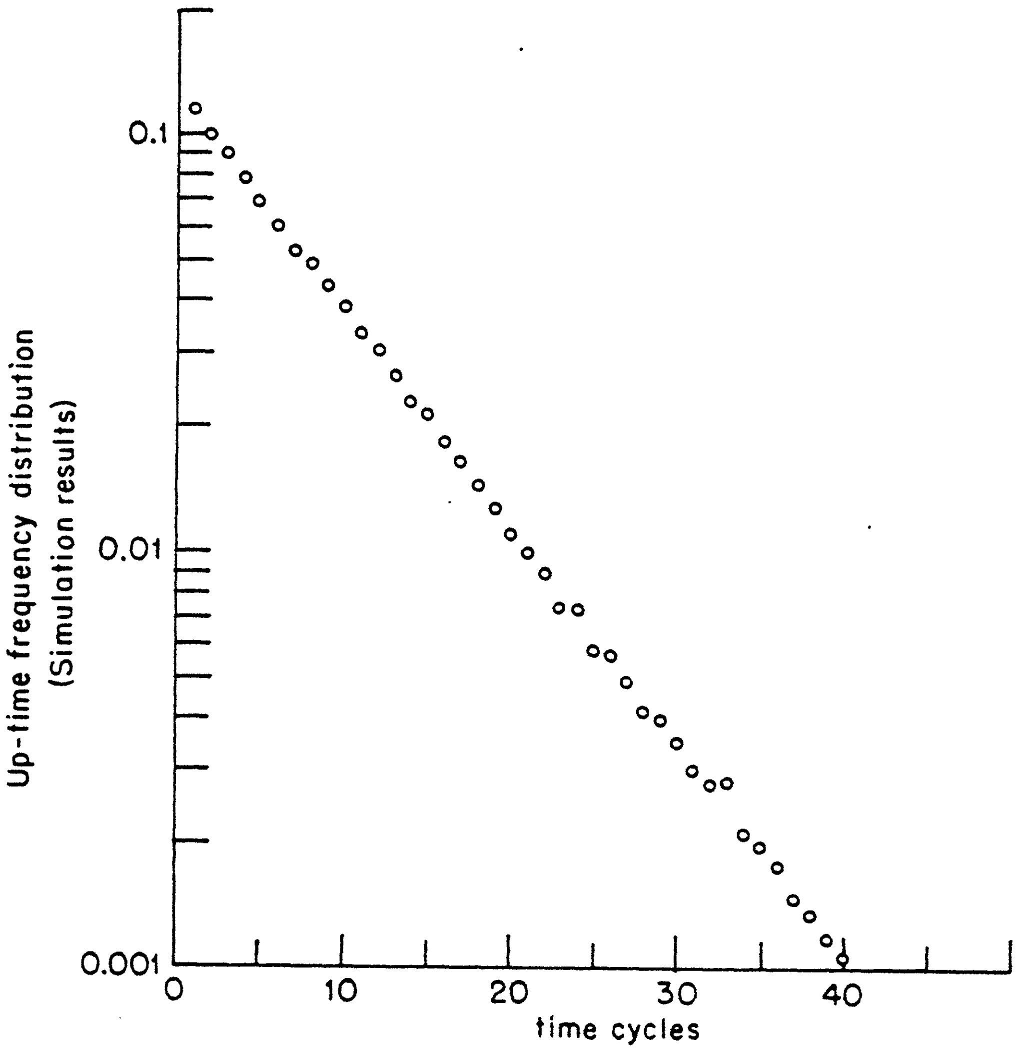


Figure 6.2. Up-time frequency distribution for a two-machine line with N = 8.

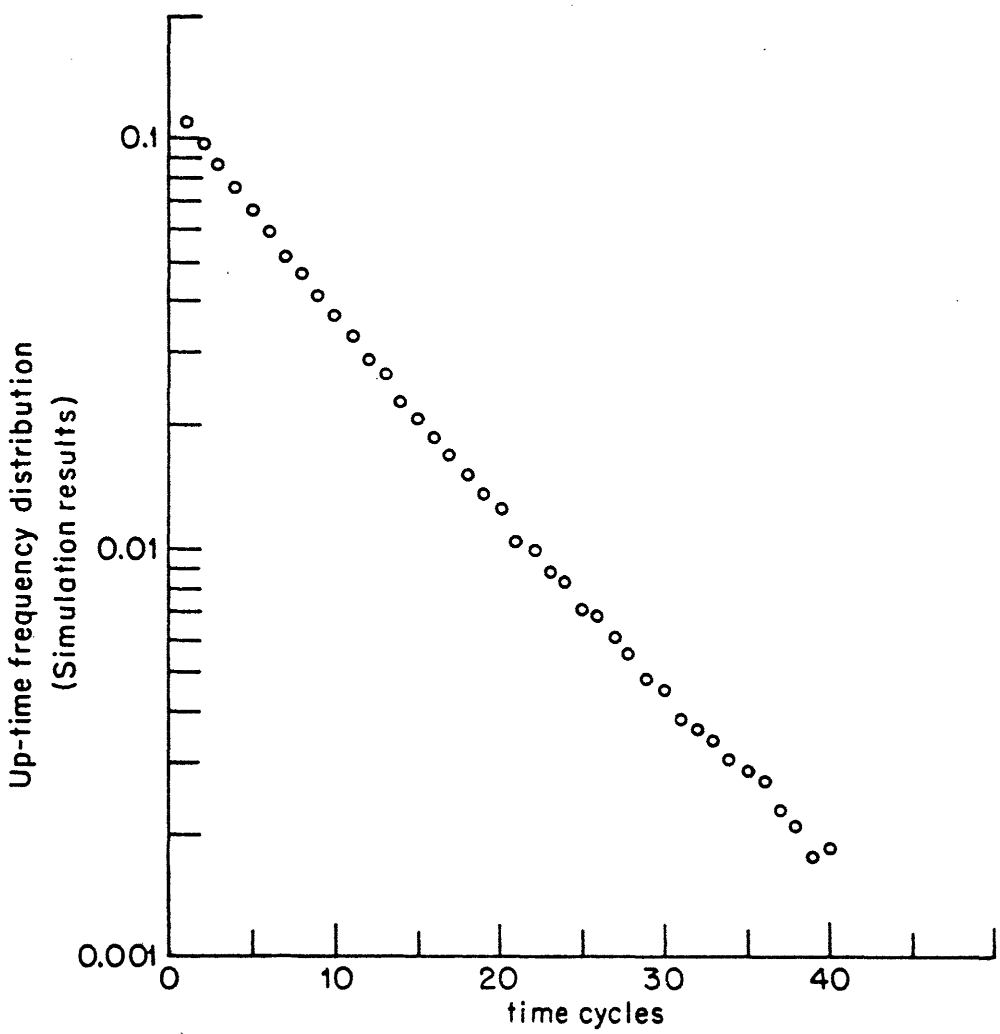


Figure 6.3. Up-time frequency distribution for a two-machine line with n = 16.

dependent on storage capacities in a two-machine line (down times depend on the time it takes to repair the last machine or to render the storage non-empty (i.e. to repair the first machine)), the down-time frequency distribution is the same for all three cases and appears in figure 6.4.

The logarithms of the distributions are very close to straight lines. The slopes of these lines depend on the storage capacity in a way which will be discussed below. This implies that the frequency distributions are very close to geometric. Since finite time simulations by their nature can not give exact steady-state results, it is necessary to derive the probability distributions analogous to these frequency distributions analytically. Output processes of single stages and transfer lines have been studied by various authors (Burke[1956,1972], Çınlar and Disney[1967], Chang[1963], Wyner[1974], Aleksandrov[1968]). These studies include stages with exponential service times, Poisson arrivals, and overflow processes. However, the output of a two-stage line with deterministic service times and a finite interstage buffer (as well as an unlimited supply of workpieces upstream) has not been investigated.

The following events are defined:

$\mathcal{D}_n \triangleq$ Event that the system fails to produce a piece for exactly n time cycles. (6.7)

$\mathcal{U}_n \triangleq$ Event that the system produces pieces for exactly n time cycles. (6.8)

$$\mathcal{D} \triangleq \bigcup_{n=1}^{\infty} \mathcal{D}_n \tag{6.9}$$

= Event that the system has failed to produce a piece for at least one cycle.

$$\mathcal{U} \triangleq \bigcup_{n=1}^{\infty} \mathcal{U}_n \tag{6.10}$$

= Event that the system has produced at least one piece.

Corresponding to the frequencies described in section 6.1.2, the following conditional probabilities are now defined:

$$p_u[n] \triangleq p[\mathcal{U}_n | \mathcal{U}] \tag{6.11}$$

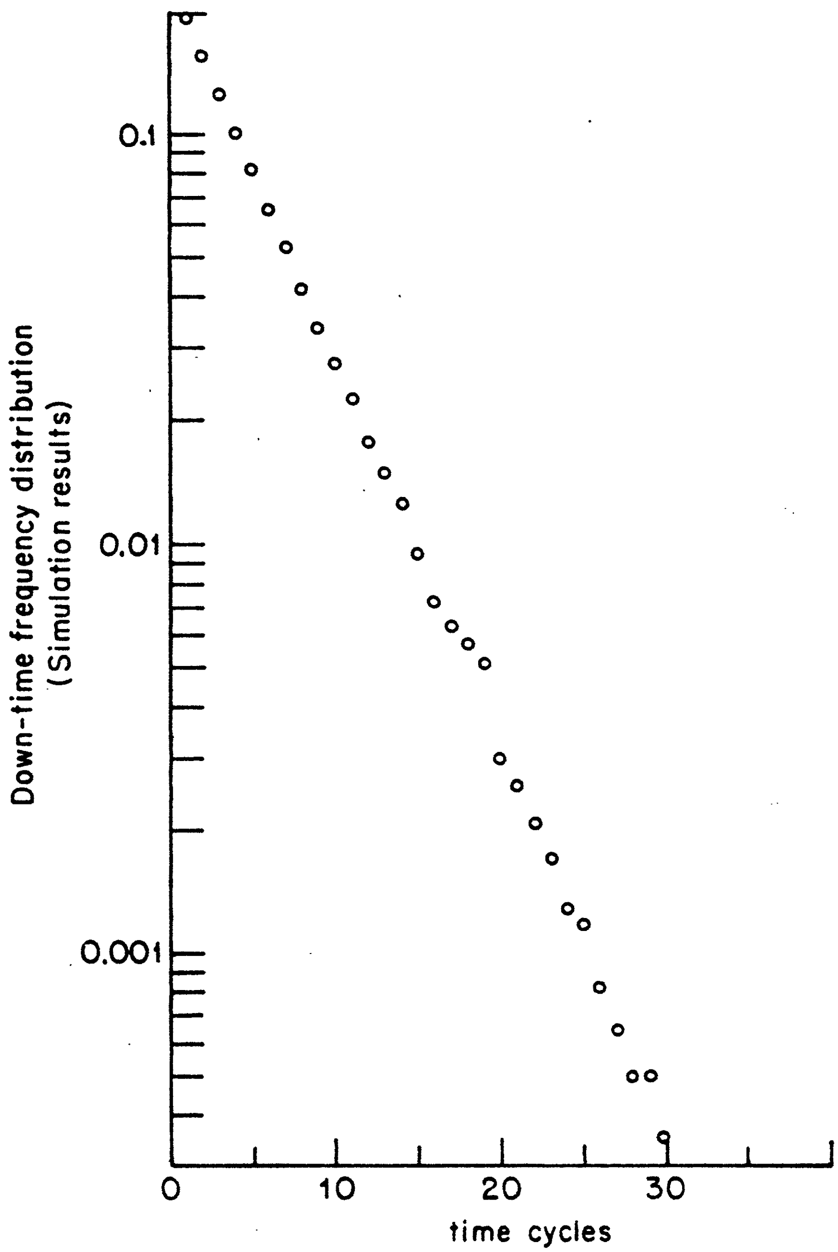


Figure 6.4. Down-time frequency distribution for a two-machine line.

$$p_d[n] \triangleq p[\mathscr{D}_n | \mathscr{D}] \tag{6.12}$$

By equations (6.9) and (6.10),

$$\mathscr{U}_n \subset \mathscr{U} \tag{6.13}$$

$$\mathscr{D}_n \subset \mathscr{D} \tag{6.14}$$

As a result,

$$p[\mathscr{U}_n, \mathscr{U}] = p[\mathscr{U}_n] \tag{6.15}$$

$$p[\mathscr{D}_n, \mathscr{D}] = p[\mathscr{D}_n] \tag{6.16}$$

Using equations (6.15) and (6.16) and Bayes' theorem, it follows from (6.11) and (6.12) that

$$p_u[n] = p[\mathscr{U}_n] \;/\; p[\mathscr{U}] \tag{6.17}$$

$$p_d[n] = p[\mathscr{D}_n] \;/\; p[\mathscr{D}] \tag{6.18}$$

To compute the unconditional probabilities in equations (6.17) and (6.18), the analytical expressions for the steady-state probabilities of a two-machine line (See section 3.2.1) are used.

In order to produce exactly n pieces, the system must start out being down, i.e. not producing parts. The system starts producing, remains up for exactly n cycles, and then stops again. This happens either because the last machine fails or because the storage empties. Similarly, in order to fail to produce pieces for exactly n cycles, the system must start out having produced at least one piece. It then stops producing, remains down for exactly n cycles, and starts producing again. This happens either because the last machine is repaired or because the storage becomes non-empty.

The output process of a two-machine line is analyzed below. The state transitions that result in the production of a finished piece are studied, and it is shown that by modifying slightly theMarkov chain, it is possible to subdivide all recurre t states into two classes. This information is then used in computing the probabilities of equations (6.17) and (6.18).

A simple two-machine line with storage capacity N=4 is used for illustration. The state transition diagram of the system as described in section 2.3 is given in figure 6.5 (only recurrent states are included). The heavy lines represent those transitions during which a part is produced by the system. It is seen that except for two states, all states are reached by transitions of only one kind, i.e. either those that result in the production of a part, or those that do not (See also section 5.1.1). All states in which the last machine is operational are reached through transitions that result in the production of a part, with the exception of the transitions $(0,0,1)\to(0,0,1)$ and $(0,0,1)\to(1,1,1)$. All other transitions to (0,0,1) and (1,1,1) result in the production of a part.

It is possible to modify the Markov chain by splitting states, so that the states are subdivided into two sets. These sets are defined as follows:

$$\Omega_1 = \{s \mid s \text{ is reached by a transition that produces a part}\} \quad (6.19)$$

$$\Omega_0 = \{s \mid s \text{ is reached by a transition that does not produce a piece}\} \quad (6.20)$$

In the discussion that follows, the system is referred to as being in an <u>up state</u> at time t if $s(t)\varepsilon\Omega_1$, where s(t) is the state of the system. Conversely, the system is in a <u>down state</u> at time t if $s(t)\varepsilon\Omega_0$.

As shown in figure 6.6, state (0,0,1) is split into states (0,0,1)' and (0,0,1)". State (0,0,1)' is reached from (1,0,0), (1,0,1), and (1,1,1), through transitions that always result in a part. Thus,

$$(0,0,1)' \ \varepsilon \ \Omega_1 \quad (6.21)$$

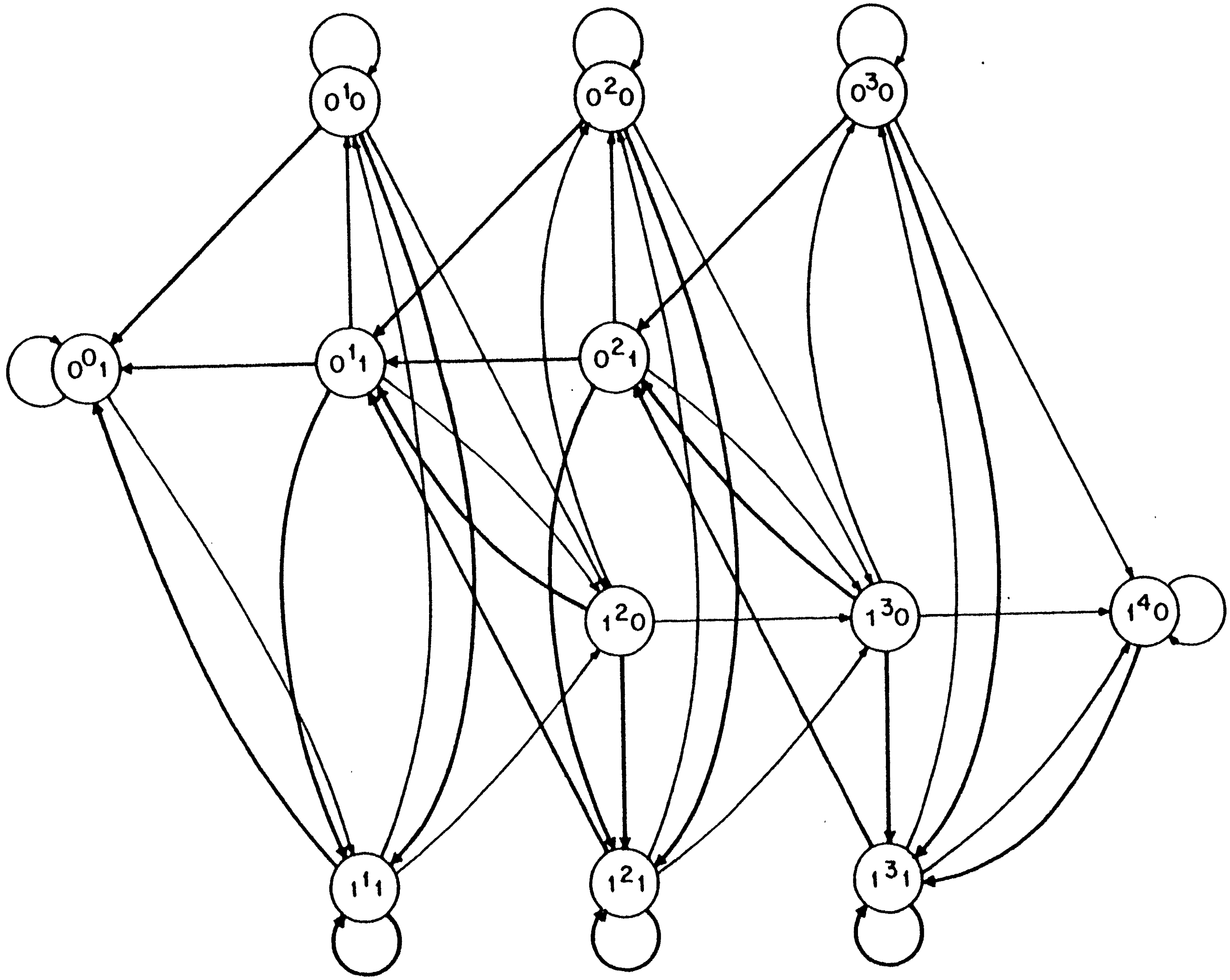


Figure 6.5. Markov State Transition Diagram for a two-machine transfer line with N=4. (Note that the states are given as $\alpha_1 {}^{n} \alpha_2$ for clarity).

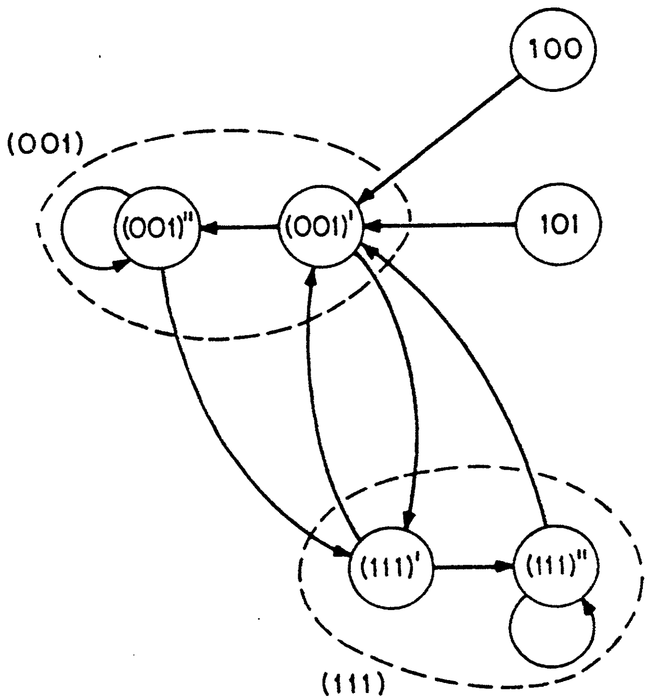


Figure 6.6. Splitting states (0,0,1) and (1,1,1).

On the other hand, state (0,0,1)" is only reached from (0,0,1)' or from itself, through transitions which do not result in parts, so that

$$(0,0,1)'' \;\varepsilon\; \Omega_0 \tag{6.22}$$

Physically, (0,0,1)' is the first occupancy of state (0,0,1). Since it has no self-loops, it leads either to (0,0,1)" (all subsequent occupancies of (0,0,1)) or to outside of state (0,0,1).

A similar argument is made for state (1,1,1), which is split into two states, (1,1,1)' and (1,1,1)". Here again, (1,1,1)' is the first occupancy of (1,1,1) coming from (0,0,1). Since it is necessary for the storage to become nonempty before a part can be produced, no pieces are produced when (1,1,1)' is reached. Thus,

$$(1,1,1)' \;\varepsilon\; \Omega_0 \tag{6.23}$$

$$(1,1,1)'' \;\varepsilon\; \Omega_1 \tag{6.24}$$

The states of the Markov chain are thus subdivided into two sets, as shown in equations (6.19) and (6.20). The steady-state probability vector of the modified system is denoted by $\tilde{\underline{p}}$. The transition matrix of the modified system is denoted by $\tilde{T}$.

The steady-state transition equations involving the split states are the following:

$$p[(0,0,1)'] = (1-r_1)r_2\, p[1,0,0] + (1-r_1)(1-p_2)\, p[1,0,1] + p_1(1-p_2)\, p[(1,1,1)'] + p_1(1-p_2)\, p[(1,1,1)''] \tag{6.25}$$

$$p[(0,0,1)''] = (1-r_1)\, p[(0,0,1)'] + (1-r_1)\, p[(0,0,1)''] \tag{6.26}$$

$$p[(1,1,1)'] = r_1\, p[(0,0,1)'] + r_1\, p[(0,0,1)''] \tag{6.27}$$

$$p[(1,1,1)''] = r_1(1-p_2)\, p[1,0,1] + r_1 r_2\, p[1,0,0] + (1-p_1)(1-p_2)\, p[(1,1,1)'] + (1-p_1)(1-p_2)\, p[(1,1,1)''] \tag{6.28}$$

The transition probabilities multiplying the state probabilities on the right hand side of equations (6.25)-(6.28) comprise the rows corresponding to the split states in $\tilde{T}$.

These equations are simplified by noting that since states (0,0,1) and (1,1,1) are split to give (0,0,1)', (0,0,1)", (1,1,1)', (1,1,1)", it follows that

$$p[0,0,1] = p[(0,0,1)'] + p[(0,0,1)''] \tag{6.29}$$

$$p[1,1,1] = p[(1,1,1)'] + p[(1,1,1)''] \tag{6.30}$$

Furthermore, state (0,0,1) can only be reached through (0,0,1)' and can only be left through (1,1,1)'; since these states have no self-loops and (1,1,1)' can be reached from no other state, it follows that

$$p[(0,0,1)'] = p[(1,1,1)'] \tag{6.31}$$

This equation directly follows from (6.26) and (6.27). Once equation (6.30) and the results of section 3.2.1 are used to solve (6.28) for p[(0,0,1)'], the probabilities of all the other split states are easily computed. The expressions for these state probabilities are given in table 6.5.

Thus, the vector $\underline{\tilde{p}}$ is known, and may be used to obtain the output up- and down-time distributions, as described below. For the sake of illustration, only the up-time probability distribution is derived. The down-time distribution is obtained analogously.

The vector $\underline{q}(n)$ is defined as the probability distribution given that the system was running for a long time prior to t=0, was in a down state at t=0, and has been in up states for t=1,..,n. At n=0, $\underline{q}(0)$ is therefore the steady-state conditional probability vector given that the system is in a down state:

$$q_i(0) \triangleq \begin{cases} 0 & \text{if } i\varepsilon\Omega_1 \\ \tilde{p}_i/\sigma & \text{if } i\varepsilon\Omega_0 \end{cases} \tag{6.32}$$

Table 6.5. Steady-state probabilities of split states

$$p[(0,0,1)'] = CX \; \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2}$$

$$p[(0,0,1)''] = CX \; \frac{(1 - r_1)}{r_1} \; \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2}$$

$$p[(1,1,1)'] = CX \; \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2}$$

$$p[(1,1,1)''] = CX \; \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 + p_2 - p_1 p_2 - p_2 r_1} \; \frac{(1 - p_1)(1 - p_2) + p_2 r_1}{p_2}$$

where i denotes the index of the state in the modified system and

$$\sigma \triangleq \sum_{i\varepsilon\Omega_0} \tilde{p}_i \tag{6.33}$$

The matrix Q is defined to be a stochastic matrix of the same dimensions as $\tilde{T}$. The elements of the matrix are given by

$$q_{ij} \triangleq \begin{cases} p[s(t+1)=j \mid s(t)=i] \;/\; \Theta_i & \text{if } j\varepsilon\Omega_1 \\ 0 & \text{if } j\varepsilon\Omega_0 \end{cases} \tag{6.34}$$

where

$$\Theta_i \triangleq \sum_{k\varepsilon\Omega_1} p[s(t+1)=k \mid s(t)=i] \tag{6.35}$$

Then, the vector $\underline{q}(\cdot)$ and the matrix Q are related by

$$\underline{q}(n+1) = Q\,\underline{q}(n) \tag{6.36}$$

The probability $p[\mathscr{U}]$ is the probability of producing at least one piece (i.e. that the system was in a down state at t=0, and in an up state at t=1, regardless of the states s(t) for t>1). Thus, it is given by

$$p[\mathscr{U}] = p[s(1)\varepsilon\Omega_1 \mid s(0)\varepsilon\Omega_0] \tag{6.37}$$

$$= \sum_{i\varepsilon\Omega_1} p[s(1)=i \mid s(0)\varepsilon\Omega_0] \tag{6.38}$$

Defining the vector $\underline{u}$ such that

$$u_i \triangleq \begin{cases} 1 & \text{if} \quad i\varepsilon\Omega_1 \\ 0 & \text{if} \quad i\varepsilon\Omega_0 \end{cases} \tag{6.39}$$

equation (6.38) is rewritten as

$$p[\mathscr{U}] = \underline{u}^T \tilde{T} \underline{q}(0) \tag{6.40}$$

In order to produce exactly one piece, the system must next enter a down state:

$$p[\mathscr{U}_1] = p[s(2)\varepsilon\Omega_0, s(1)\varepsilon\Omega_1, s(0)\varepsilon\Omega_0] \tag{6.41}$$

Using Bayes' theorem, equation (6.41) is rewritten as

$$p[\mathscr{U}_1] = p[s(2)\varepsilon\Omega_0 | s(1)\varepsilon\Omega_1, s(0)\varepsilon\Omega_0] \cdot p[s(1)\varepsilon\Omega_1 | s(0)\varepsilon\Omega_0] \cdot p[s(0)\varepsilon\Omega_0] \tag{6.42}$$

The last factor in (6.42) is given by σ in equation (6.33), and the second by $p[\mathscr{U}]$ in equations (6.37) and (6.40). The first factor is the sum of the probabilities of down states at t=2. Defining the vector $\underline{d}$ such that

$$d_i \triangleq \begin{cases} 0 & \text{if } i\varepsilon\Omega_1 \\ 1 & \text{if } i\varepsilon\Omega_0 \end{cases} \tag{6.43}$$

the first factor in (6.42) is given by

$$p[s(2)\varepsilon\Omega_0 | s(1)\varepsilon\Omega_1, s(0)\varepsilon\Omega_0] = \underline{d}^T \tilde{T} \underline{q}(1) \tag{6.44}$$

Thus, (6.42) becomes

$$p[\mathscr{U}_1] = \underline{d}^T \tilde{T} \underline{q}(1) \cdot \underline{u}^T \tilde{T} \underline{q}(0) \cdot \sigma \tag{6.45}$$

and from (6.17), it follows that

$$p_u[1] = \underline{d}^T \tilde{T} \underline{q}(1) \cdot \underline{u}^T \tilde{T} \underline{q}(0) \cdot \sigma \; / \; \underline{u}^T \tilde{T} \underline{q}(0) \tag{6.46}$$

$$= \underline{d}^T \tilde{T} \underline{q}(1) \cdot \sigma \tag{6.47}$$

Equation (6.42) is generalized to obtain the probability of producing exactly n pieces:

$$p[\mathscr{U}_n] = p[s(n+1)\varepsilon\Omega_0 | s(n)\varepsilon\Omega_1,..,s(1)\varepsilon\Omega_1,s(0)\varepsilon\Omega_0] \cdots$$
$$\cdot \; p[s(1)\varepsilon\Omega_1 | s(0)\varepsilon\Omega_0] \cdot p[s(0)\varepsilon\Omega_0] \qquad (6.48)$$

Combining (6.48) with equations (6.17) and (6.37), it follows that

$$p_u[n] = \underline{d}^T \tilde{T} \underline{q}(n) \cdot \underline{u}^T \tilde{T} \underline{q}(n-1) \ldots \underline{u}^T \tilde{T} \underline{q}(1) . \sigma \qquad (6.49)$$

Note that $\underline{q}(0)$ is given by equation (6.32).

Equation (6.49) is used to obtain the up-time distribution $p_u[n]$ of the output of a two-machine transfer line. An analogous method may be used to obtain the down-time distribution, $p_d[n]$.

Some numerical results for $p_u[n]$ for the system parameters in table 6.4 are given in table 6.6. The simulation values appear only for comparison. The use of these distributions is discussed in section 6.2.2.

n	$p_u[n]$ analytical	$p_u[n]$ simulation
1	0.121909	0.121479
2	0.106641	0.105821
3	0.092271	0.092769

Table 6.6. Probability of producing exactly n pieces given that the system has produced at least one piece, for a two-machine line.

6.2.2 Single Machine Equivalence of Two-Machine Line

It follows from assumption 2.2.3 that a single machine has geometric up- and down-time distributions. For a single machine, the system state is given only by α. Then,

$$p_u[n] = \frac{p[\alpha=0]\ r\ (1-p)^{n-1}\ p}{p[\alpha=0]\ r} \tag{6.50}$$

$$= (1-p)^{n-1}\ p \tag{6.51}$$

$$p_d[n] = \frac{p[\alpha=1]\ p\ (1-r)^{n-1}\ r}{p[\alpha=1]\ p} \tag{6.52}$$

$$= (1-r)^{n-1}\ r \tag{6.53}$$

Taking the logarithm of these distributions gives

$$\ln p_u[n] = (n-1)\ \ln(1-p) + \ln p \tag{6.54}$$

$$\ln p_d[n] = (n-1)\ \ln(1-r) + \ln r \tag{6.55}$$

These functions are linear in n. Thus, graphs of the logarithms of the up- and down-time distributions against up and down times respectively are straight lines with slopes ln(1-p) and ln(1-r) respectively. It is shown in section 6.2.1 that the corresponding graphs for two-machine transfer lines are <u>almost</u> straight lines. The fact that they are not exactly straight may be explained by the following arguments:

A single machine has no "memory". In other words, the past history of the system does not affect its transition probabilities. In a two-machine transfer line, the storage acts as a memory. For example, if the last machine fails, the storage tends to fill up; if the machine is later repaired but after some time the first machine breaks down, it takes a longer period of time for the line to stop producing pieces because the storage is full, due to the previous failure. On the other hand, if the first machine fails twice consecutively, the storage has few parts in it and it takes a shorter time for the line to stop producing pieces. Thus, the storage provides information on the past history of the system, and this affects the up- and down- time distributions.

The transfer line does not produce finished parts if the last machine is down or if the last storage is empty (See section 5.1.1). If the storage were never empty, the two machines would be effectively decoupled, in the sense that the output behavior of the line would only depend on the status of the last machine. In that case, the distributions would be exactly geometric and identical to those of the second machine. Similarly, if the last machine never failed, it would have no effect on the output behavior of the system: since service times are deterministic, it would merely introduce a delay of one cycle, but would not affect the actual distributions of up and down times. In that case, the distributions would be determined by the first machine only, and would be exactly geometric.

Thus, the deviation of the distributions of a two-machine line from exactly geometric are due to the coupling effects of the two machines. These effects are insignificant when the machines have very different efficiencies. From the discussion in section 5.2, it follows that the probability that the storage is empty decreases as the first machine is made more efficient. At the limit (e_1=1.0), the probability that the storage is empty is zero. The up- and down-time distributions are then only determined by the second machine and are exactly geometric.

Similarly, the probability that the second machine is down decreases as the second machine is made more efficient. At the limit (e_2=1.0), this probability is zero. The up- and down-time distributions are then only determined by the first machine and are exactly geometric.

A similar argument can be made in the case where the efficiency in isolation of one of the machines approaches zero. As the efficiency of one machine is decreased, the forced-down probability of the other machine increases (See section 5.2). From assumption 2.2.3, machines cannot fail when forced down. Thus, the time a machine spends under repair becomes insignificant compared to total time as the efficiency of the other machine is decreased. Consequently, the up- and down-time distributions of the line approach those of the least efficient machine as its efficiency approaches zero.

In general, the up- and down-time distributions of a two-machine line is closest to exactly geometric when one machine is strongly limiting, i.e. when the line is not well balanced. In such cases, the two-machine line may be approximately represented by a single machine. The failure and repair probabilities of a single machine may be obtained from the slopes of the graphs of the logarithms of up- and down-time distributions, as seen in equations (6.54) and (6.55). Analogously, the failure and repair probabilities of the single machine that is approximately equivalent to a two-machine line are obtained from the slopes of the straight lines which best fit the logarithm of the up- and down-time distributions of the line. It is important to note that the two-machine line and the approximately equivalent single machine must have equal efficiencies. Thus, it may be necessary to adjust the p and r values obtained from the slopes of the distributions in order to obtain the efficiency of the two-machine line.

The use of this procedure in reducing long transfer lines to approximately equivalent two-machine lines that can be solved by the closed-form expressions given in chapter 3 is discussed in section 6.2.3.

6.2.3 Solution of a k-Machine Line by the Aggregate Method

It is shown in section 6.2.2 that a two-machine line in isolation may be approximately represented by a single machine. This suggests a method for approximately computing the production rate of a k-machine line by successively lumping together two-machine segments of the line. This procedure is illustrated by figure 6.7.

If the limiting machine is the second one in the two-machine segment, appending more machines downstream of it can only serve to make its utilization even lower. In that case, the output distributions are still close to geometric.

Often in practice, however, downstream machines are faster or more efficient than upstream ones (See section 7.1.4). This is done in order to reduce the probability of blocking upstream machines and to avoid having to use large storages (See section 5.3). In that case, it is possible to use input up- and down-time distributions, and start lumping machines from the end of the line towards the beginning

For three-machine lines, the approximation is often accurate within one or two percent. As stated before, it is worst when the line is balanced, since the coupling effects of the machines in the line are strongest then. When one machine strongly acts as a bottleneck, the approximation is much better.

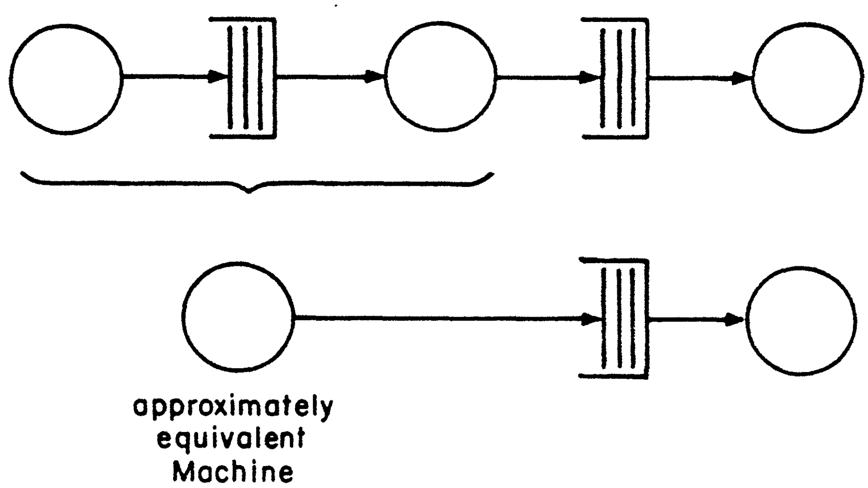


Figure 6.7. Reduction of a three-machine line to an approximately equivalent two-machine line by the aggregate method.

6.3 The δ-Transformation

It is shown in section 5.1.3 that there is a relationship between the magnitudes (i.e. not the ratios between them) of failure and repair probabilities and the buffer capacities required to achieve a given efficiency.

In the example given in section 5.1.3, two two-machine lines are considered. These have $p_i=r_i=0.1$, $i=1,2$, and $p_i'=r_i'=0.001$, $i=1,2$, respectively (The primes are only for clarity and are intended to serve to differentiate the two lines. This also applies to $E(\cdot)$ and $E'(\cdot)$). Both lines are shown to have the same limiting efficiencies, $E(0)=E'(0)=0.333$ and $E(\infty)=E'(\infty)=0.500$. Yet, in the former line, a buffer of capacity 4 gives an efficiency of $E(4)=0.35938$, while in the latter, the same buffer capacity yields only $E'(4)=0.33361$. Further investigation reveals that $E'(400)=0.36834$, a value which is close to $E(4)$. It is seen that going from the former to the latter line, the failure and repair probabilities are multiplied by 10^{-2} while the storage capacity is divided by the same number. The resulting efficiencies are close to eachother. The following proposition is now introduced:

Proposition 6.1. The δ-transformation: letting

$$\bar{p}_i = p_i \delta \qquad (6.56)$$

$$\bar{r}_i = r_i \delta \qquad (6.57)$$

$$\bar{N} = N / \delta \qquad (6.58)$$

For a wide range of δ, the efficiency of the original system and that of the transformed system are nearly equal.

It is observed that the transformed system with the parameters on the left hand side of equations (6.56)-(6.58) is identical with the original system with the parameters on the right hand side of these equations, except that the time cycles are now of length δ. Thus, the fact that the transformation leaves the line efficiency almost unchanged can be explained intuitively by the argument illustrated by figure 6.8.

At the top of the diagram, a workpiece is shaved by the tool in a machine. The cycle length is 1, and the probability that a machine fails or is repaired within a time cycle are p_i and r_i. The storage size is N; thus, when full, it takes N cycles to empty. At the center of the diagram, the workpiece is sliced into $1/\delta$ identical parts. However, the slices are held together $1/\delta$ at a time. Thus, the sliced parts are treated exactly as unsliced, i.e. slicing has no consequence. The probability that the machine fails or is repaired within the time it takes to process any slice are $p_i\delta$ and $r_i\delta$, respectively. Since the slices are held together, if a machine fails while processing any slice within a set, the entire set is moved back into the upstream storage and must be reprocessed (This is a mathematical abstraction; in a real system, a machined piece need not be machined again). In the third case, the slices are allowed to go through the system independently. The probabilities of failure and repair during each cycle are $p_i\delta$ and $r_i\delta$, respectively. Because the slices are independent of eachother, if a machine fails while processing a slice, only that part is moved back into the upstream storage. Thus, there is no loss of time due to reprocessing.

A numerical example is illustrated by figure 6.9. The system is a two-machine line with parameters given by table 6.7. The value of δ is varied from 1.0 to 0.01. The efficiency of the original system (δ=1.0) is E(4)=0.65764; that of the transformed system (δ=0.01) is E'(400)=0.66892. As δ is changed from 1.0 to 0.01, the line efficiency remains within 0.011. The line efficiency is plotted against δ in figure 6.9. It is noted that as $\delta \to 0$, the curve in figure 6.9 approaches a straight line. This line can be used to make the δ-transformation even more accurate.

It must be noted that E(0) and E(∞) are unchanged by the transformation, since they only depend on the ratios between p_i and r_i, not on their magnitudes (See section 5.1.3). The difference between E(0) and E(∞) is shown in section 5.1.3 to determine how much can be gained by providing the system with storage capacity. This difference is also an indicator of

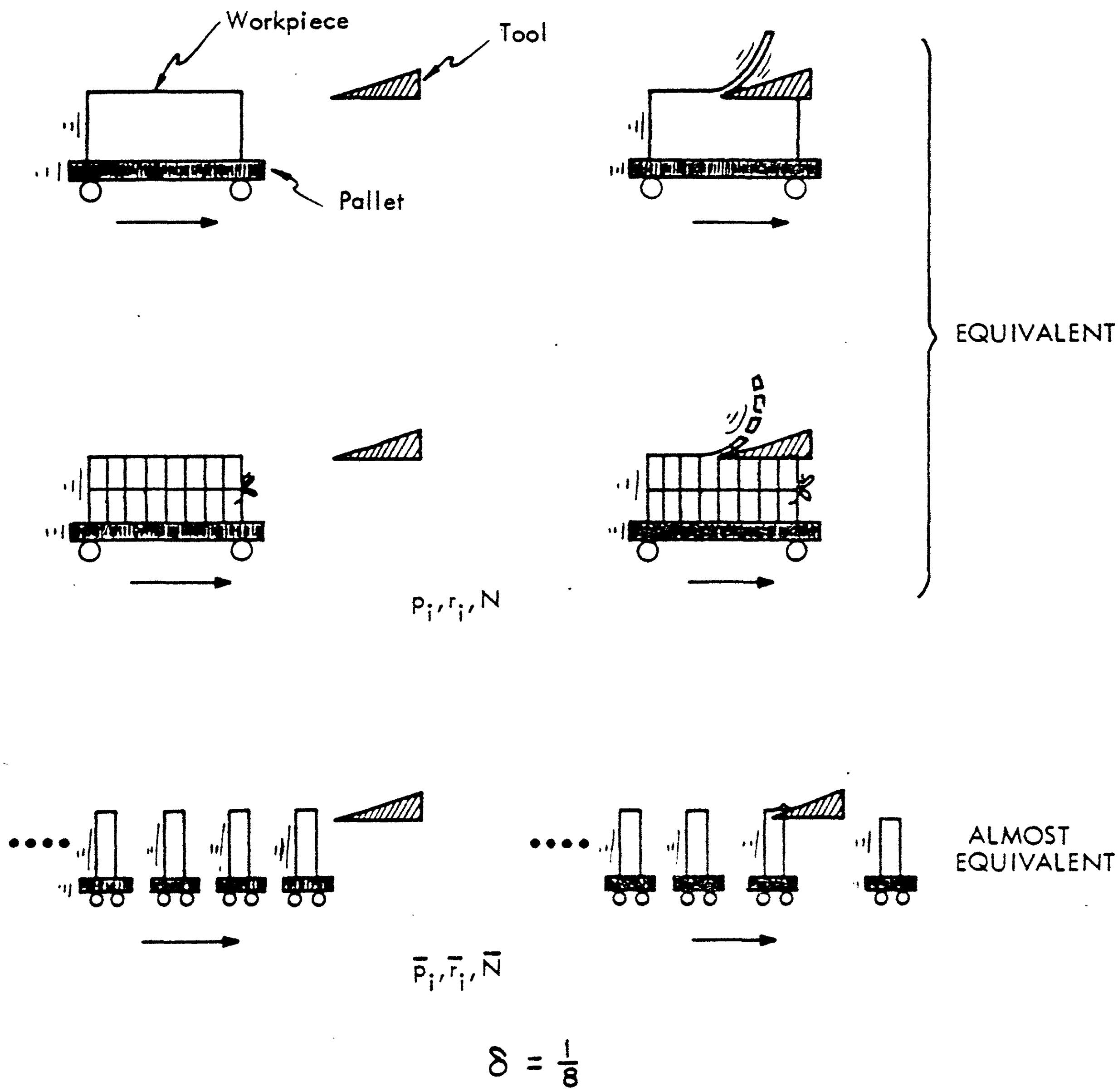


Figure 6.8. Intuitive explanation of the δ-transformation.

$p_1 = 0.05$	$r_1 = 0.20$	
		$N = 4$
$p_2 = 0.05$	$r_2 = 0.15$	

Table 6.7. System parameters for δ-transformation.

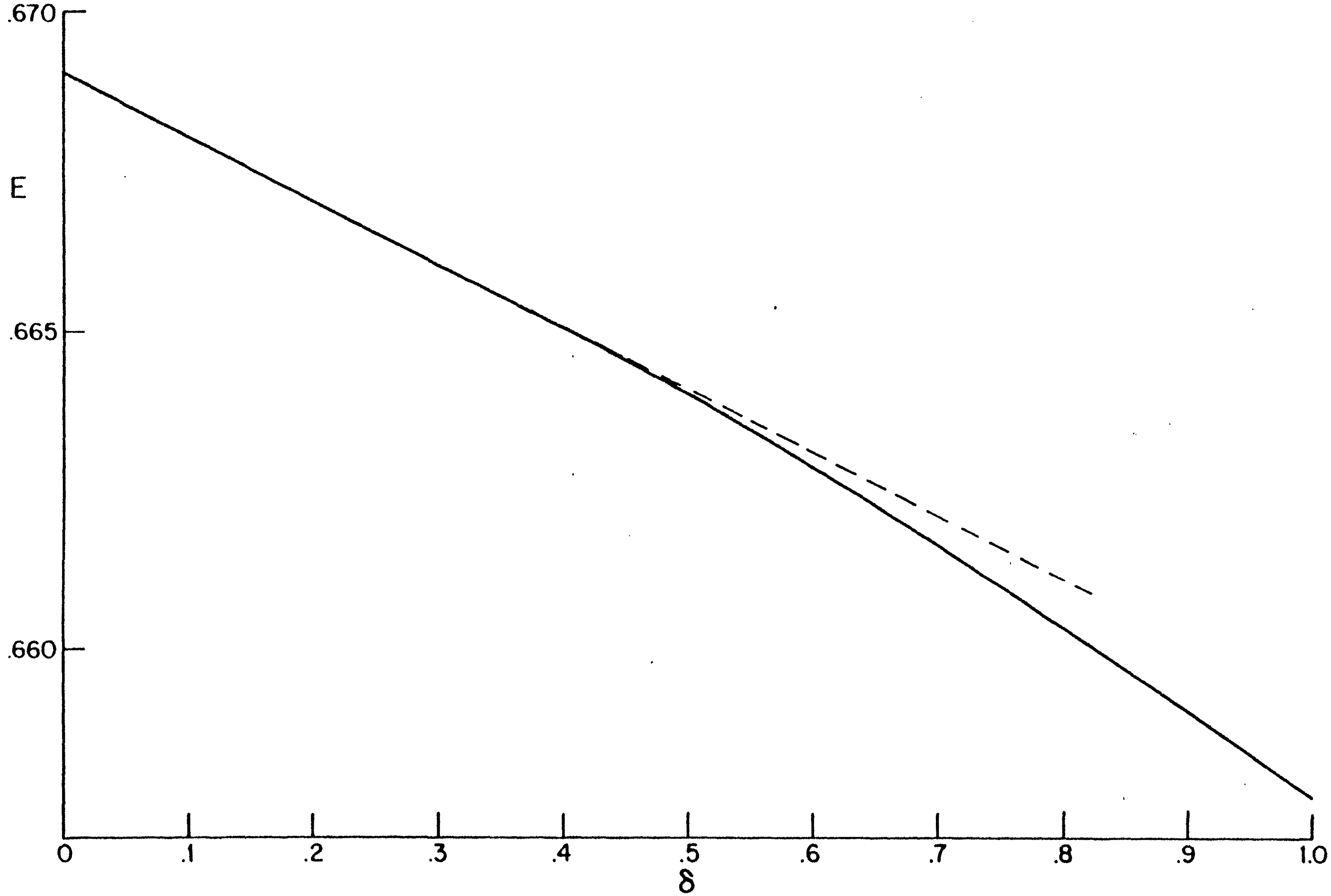


Figure 6.9. Efficiency against δ for a two-machine line. (Note that the vertical scale only goes from about .655 to .670).

how accurate an approximation is obtained by the δ-transformation. Some numerical results appear in table 6.8. In case 1, machine is much less efficient than machine 1. Thus, the difference between the limiting efficiencies is $E(\infty)-E(0)=0.00219$; the line efficiency stays constant through the transformation to within 10^{-5} in this case. In case 2, neither machine is extremely limiting, and $E(\infty)-E(0)=0.12122$. The line efficiency stays constant only to within 10^{-2} in this case. In both cases, however, the approximation is good enough to be useful in many engineering applications.

The major consequence of the δ-transformation technique is that systems with large storages can be reduced to approximately equivalent systems with smaller storages. The transformation is thus equivalent to lumping workpieces together, thereby reducing the capacities of the storages. Smaller storages mean reduced state space dimensions, and this significantly decreases the computational burden. This is illustrated by figure 6.10. The solid curve is the efficiency versus storage capacity graph for a small system, i.e. a system whose efficiency rises sharply with small storage capacities. The dotted curve is the approximate efficiency of the original system, which has much larger storages and whose efficiency rises more smoothly with storage size. Since the efficiency of the smaller system at any storage capacity N is approximately equal to that of the larger system at storage capacity N/δ, where δ is the ratio of the failure and repair rates of the two systems, the efficiency of the large system may be approximated by that of the smaller system with considerable savings in computation.

As pointed out in section 4.1, the δ-transformation method is also useful in estimating the state probabilities of a system with large storages by solving the problem for a system with smaller storages. This is related to the order of magnitude considerations mentioned in section 4.1 (See also Gershwin and Schick[1978]). To illustrate this argument, the two-machine line probability expressions given in chapter 3 are analyzed.

The orders of magnitude of X_i and Y_j in equation (3.25) are related to the ratios between p_i and r_i and the relative efficiencies in isolation

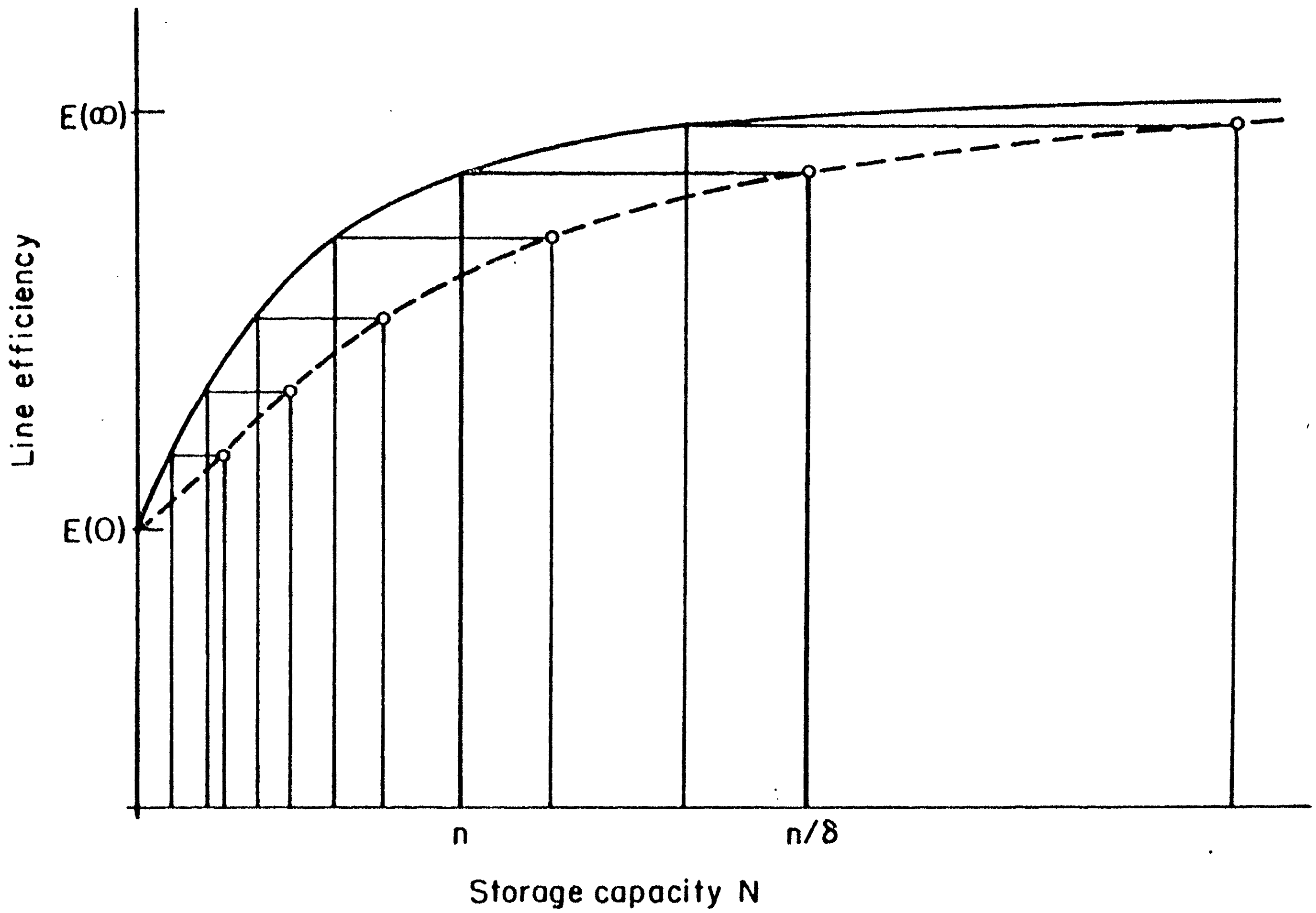


Figure 6.10. Obtaining the approximate efficiency of a large system by solving a smaller system, through the δ-transformation.

p_1	r_1	p_2	r_2	N	δ	E(N)	Case
.100	.700	.700	.100	4	1	0.12499	1
.020	.140	.140	.020	20	.2	0.12498	
.004	.028	.028	.004	100	.04	0.12498	
.001	.007	.007	.001	400	.01	0.12498	
.100	.200	.050	.150	4	1	0.57515	2
.020	.040	.010	.030	20	.2	0.58475	
.004	.008	.002	.006	100	.04	0.58629	
.001	.002	.0005	.0015	400	.01	0.58663	

Table 6.8. System parameters and line efficiencies for various values of δ.

of the two machines. It can be shown that these orders of magnitude are not affected by the δ-transformation (See section 9.2). The numbers of states for a two-machine system is linear with storage capacity (See equation (2.22)). Thus, if the orders of magnitude of $\xi[\cdot]$ in equation (3.33) do not change for most states, the value of the normalizing constant C can be expected to change inversely with the number of states, and thus, with storage size. On the other hand, some of the boundary probabilities have different orders of magnitude from that of internal states. These are p[0,0,1], p[1,1,1], p[N-1,1,1], and p[N,1,0]. It is seen in table 3.1 that the orders of magnitude of the former two probabilities have a ratio of $1/p_2$ with the order of magnitude of internal probabilities; the orders of magnitude of the latter two probabilities have a ratio of $1/p_1$ with that of the internal probabilities. From equations (6.56) and (6.57), it follows that these ratios should change when the system undergoes a δ-transformation. This change is inversely proportional to $\bar{p}_i$, and therefore to δ. On the other hand, C is inversely proportional to $\bar{N}$ and therefore proportional to δ. Thus, the two effects cancel eachother out in these boundary state probabilities. In consequence, it can be expected that the four boundary state probabilities given above are approximately unchanged by the transformation, while all other probabilities change proportionally to δ. The numerical example given in table 6.9 confirms this proposition. The order of magnitude considerations are discussed with respect to the three machine case in Gershwin and Schick[1978].

The major limitation on the δ-transformation applies to the range of δ. Since $\bar{p}_i$ and $\bar{r}_i$ are probabilities, it is necessary that

$$0 \leq \bar{p}_i, \bar{r}_i \leq 1 \tag{6.59}$$

Given p_i and r_i, only a limited range of δ satisfies (6.59). Furthermore, it is necessary that $\bar{N}$, the storage capacity of the transformed system, be an integer. Equation (6.58) implies that not all δ satisfy this condition. Consequently, it may not always be possible to reduce a system

Case 1: $\delta=0.5$	Case 2: $\delta=1.0$
$p_1=0.001$ $r_1=0.002$	$p_1=0.002$ $r_1=0.004$
$p_2=0.0005$ $r_2=0.002$	$p_2=0.001$ $r_2=0.004$
N = 400	N = 200
P[0,0,1] = 0.234439	0.234543
p[1,1,1] = 0.312898	0.313351
p[399,1,1] = 0.108616	p[199,1,1] = 0.108966
p[400,1,0] = 0.081327	p[200,1,0] = 0.081452
p[100,0,0] = 0.535855×10^{-4}	0.980038×10^{-4}
p[100,0,1] = 0.142823×10^{-3}	0.261082×10^{-3}
p[100,1,0] = 0.142954×10^{-3}	0.261562×10^{-3}
p[100,1,1] = 0.381021×10^{-3}	0.696800×10^{-3}

Table 6.9. System parameters and some boundary and internal state probabilities for δ-transformation (Note that the ratio between the left hand side and right hand side sets of probabilities is 1 in the upper (boundary) sets and δ in the lower (internal) sets).

with large storage capacities to a smaller problem which is efficiently solvable. This is especially true if the system with large storage has a large failure or repair probability or if, in case there is more than one storage, the storage capacities differ widely. Work should be directed towards investigating whether or not it is possible to extend the transformation so that it may be applied to different segments of a transfer line with different values of δ. This approach would also be useful in analyzing systems where parts are cut or assembled by stages in the line so that each machine does not process the same average number of parts (See section 7.1.1).

The limit of this transformation as $\delta \to 0$ is a continuous system. It is shown in chapter 9 that the limiting system can be analyzed by differential equations in the two-machine case (the three-machine or general k-machine cases are not yet solved). This renders the analysis of the system considerably simpler.

7. AN APPLICATION OF THE MODEL: A PAPER FINISHING LINE

Although an effort was made to conform as much as possible to actual systems by choosing the modeling assumptions realistically, the transfer line model presented here remains an idealized abstraction, a mathematical tool. The following three chapters are concerned with applications of the model, discussions of its limitations and shortcomings, and investigations of possible changes and extensions to adapt the model to real situations.

One existing system that may lend itself to the transfer line model is a roll products paper finishing line: here, paper from large rolls is winded into cylinders of smaller diameter, which are then cut into user-size rolls. These are then packaged, several rolls at a time. The system can be thought of as a three-machine, two-storage transfer line.

Yet, it is shown in section 7.1 that the system does not satisfy many of the modeling assumptions described in section 2.2. The paper finishing line is used in the present chapter to illustrate possible discrepencies between actual systems and the transfer line model and to discuss ways of relaxing the assumptions that do not hold. Attempts at modeling the system and a discussion of the models are presented in section 7.2.

7.1 The Paper Finishing Line

The paper finishing line considered here consists of three stages separated by two storage elements. These components are the following:

(i) The Continuous Winder

(ii) The First-in-last-out Buffer Storage

(iii) The Log Saw

(iv) The Conveyor Belt

(v) The Wrapper

The system is sketched in figure 7.1. These components differ from the idealized machines and storages in the transfer line model in a number of ways. In some instances, the effects of these differences may be negligible; in some, major model changes may be necessary to account for these discrepencies. Still others may necessitate an altogether different approach.

The main discrepencies between the model and the actual system, as well as possible approximations, are briefly discussed in sections 7.1.1 to 7.1.6.

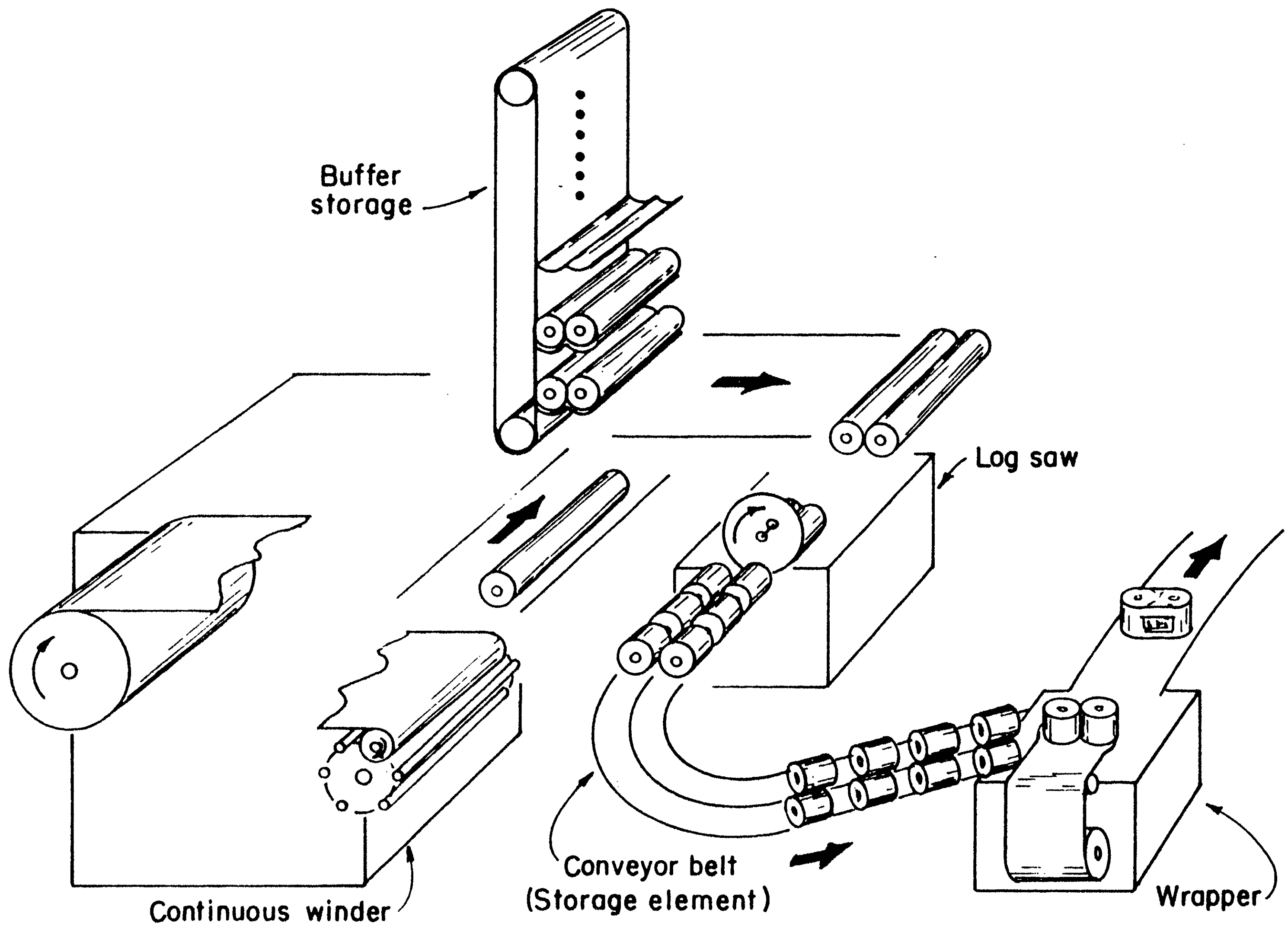


Figure 7.1. Sketch of a roll products paper finishing line.

7.1.1 The Workpieces

The continuous winder takes in a large roll of paper, known as a parent roll, which is approximately 10 ft. in diameter and 10 ft. long. It winds up the paper onto cardboard cores of the same length. Each of these cores takes the length of paper that makes the output of the continuous winder to have the diameter of commercially available rolls. These are termed logs. The log saw takes in these logs two at a time, and saws them each into about twenty short cylinders, the size of user rolls. Because the log saw takes in two logs at a time, the buffer storage between the first two stages stores the logs two at a time. Its capacity is typically about sixty pairs of logs. After coming out of the log saw, the rolls travel on two conveyor belts until they reach the wrapper. The conveyor belts each have a capacity of about forty rolls. The final stage wraps the rolls in packages of two or four.

The simple one-piece-in, one-piece-out machine model introduced in section 2.1.1 is thus not applicable to this considerably more complex system. One possible approximation is to take the smallest unit (the roll or pair of rolls) as the workpiece in the transfer line model. Everything else is then computed in terms of this smallest unit. Thus, the continuous winder, for example, processes twenty rolls, rather than a log; the log saw processes forty rolls, rather than two logs; and the wrapper processes two or four rolls, rather than a package. A variant of the δ-transformation techniques (Section 6.3) may be used to adapt this system to the transfer line model. The duration of a cycle in which the log saw processes two logs is equal to that of forty cycles in which machine 2 processes a unit workpiece. Thus, the probability of failure or repair during a single roll cycle is equal to 1/40 of the respective probabilities during a single actual machining cycle in which the log saw processes two logs. This analysis can be extended to the other stages of the system as well.

It is important to note that this is only an approximation. As pointed out in section 6.3, there is an effective change in the flexibility of the system when the δ-transformation is applied. This is due to the fact that when a failure occurs, there is a loss of time in the lumped-workpieces case that does not take place in the case where pieces travel singly through the system. Thus, the efficiency of the transformed system is not exactly the same as that of the original system.

7.1.2 Input to the Line and System Transients

The input to the first stage (the continuous winder) is a large roll of paper (the parent roll) which takes about one hour to unwind.

The fact that the input to the transfer line is thus in some sense continuous dows not matter, since the machine uses up discrete segments of the paper. However, the assumption that workpieces are always present at the first stage (Section 2.2.1) is not always satisfied: the continuous winder is starved when the parent roll is being loaded.

Since Markov processes are, by definition, memoryless (Section 2.3.1), it is not possible to take scheduled down times and other deterministic events into consideration with the present model. However, if the lengths of time required to load and unwind a parent roll can deviate significantly from the mean value, it may be possible to model this phenomenon as a stochastic event, by lumping it together with other causes of failure and repair times. The time to unwind a parent roll may be assumed to be random if rolls do not contain the same length of paper or if this length is variable because of defects in the paper. Since loading a roll requires human intervention, the time needed to load a new roll may be assumed to be random if workmen are not always available at the transfer line. In this case, termination of the paper on a parent roll and loading a new roll can be modeled as random events and be included in the calculation of the failure and repair probabilities of the continuous winder along with other causes of breakdown.

A consequence of the parent roll loading time is that the storages often empty while the roll is being loaded, so that the line is restarted every time. This introduces important transients which Gordon-Clark[1977] estimates can significantly influence the production rate of the system for as long as about a third of the total time the roll takes to unwind.

This is a more important difficulty than the fact that the continuous winder is occasionally starved. If no real transients were introduced by the loading, it would have been possible to compute the actual production rate of the line from knowledge of the ideal production rate (i.e., assumption 2.3.1 satisfied) and the average loading time of a parent roll. However, the duration of loading time is such that the line is restarted with empty storages every time the roll ends. Thus, the probabilistic model of the line never achieves steady-state. As suggested in section 5.1.2, transients may have a very important effect on the efficiency of the transfer line. When the steady-state assumption (Section 2.2.6) does not hold, the production rate computed by the methods outlined here may not be representative of the actual behavior of the system.

It is possible to deal with system transients through Markov techniques, and in particular, by means of the iterative multiplication method (Section 4.1). Since the initial condition is known (at $t=0$, $s(0)=(0,0,1,1,1)$), the initial probability vector $\underline{p}(0)$ could be defined to have a 1-entry corresponding to that initial state. The transition matrix can then be iteratively multiplied and the probability of producing a part at each time cycle (i.e. at each multiplication) be computed by summing up the probabilities of the appropriate states (Section 5.1.1). Considering the fact that a cycle has a very short duration in this system, however, the iteration would have to be performed a very large number of times and it is therefore doubtful that this approach would be an efficienct way of computing the transient production rate.

7.1.3 Rejects and Loss of Defective Pieces

When the continuous winder is turned on, it takes a certain length of time to reach the operating speed; a number of logs manufactured during the period of acceleration are defective and must be discarded. Similarly, the log saw may be preprogrammed to reject the rolls at certain positions in the logs because the parent roll may be known to be partly defective.

Both of these events have deterministic results: given the rate of failure of the continuous winder, it is possible to estimate the percentage of production lost in the acceleration period. Similarly, since the log saw is preprogrammed to reject a given quantity of rolls, this loss too is predictable. However, these events have an effect on the behavior of the system because they affect the probabilities of storages emptying out and starving downstream machines. Thus, the loss in production is more than the mere fraction of rejects out of the total.

The assumption that parts are not rejected or destroyed at any point in the system (Section 2.2.4) is thus not satisfied. As a result, the expected flow rates through all the machines are not equal. This may or may not be neglected, depending on the percentage of rejects in the total production.

The difficulty may be overcome by extending the model. Non-predictable rejections (e.g. at an inspection station) may be taken into account by defining new transitions with positive probabilities, in which storage levels go down (or fail to rise) even though the appropriate machines are up and processing workpieces. Predictable losses (e.g. the logs lost while the continuous winder is accelerating) may be accounted for by computing the average down-times of machines so as to include the times when parts are processed but rejected, or by defining new system states (e.g. acceleration states).

Some of the methods developed in this study may be extended to obtain the performance results of such a model.

7.1.4 Machining Times

In order to reduce the probability of filling up storages and blocking the machines, particularly as large storages can be expensive, the relative speeds of the machines are designed to increase in the downstream direction. Thus, the relative speed of machine i+1 is greater than that of machine i. This violates assumption 2.2.2, which states that all machines run at equal rates.

It is sometimes possible to approximately compensate for this by adjusting the failure probabilities so that the value of the efficiency of machine i in the model is equal to the value of the production rate of the actual machine in the system. This is done as follows: the production rate in isolation of a machine which operates at the deterministic speed of ζ_i and has failure and repair probabilities p_i and r_i is given by

$$\text{Production rate} = \zeta_i \frac{r_i}{r_i + p_i} \tag{7.1}$$

This is the product of the speed of the machine and its efficiency in isolation (See section 5.1.1). For a certain range of ζ_i, it may be possible to readjust the failure probabilities p_i so that a common time basis is taken: the slower machines appear to operate at the same speed as faster machines, but they are less efficient.

It is important to note that this approach does not give exact results. The production rate of a line with given storages depends not only on the efficiencies of individual machines, but also on the magnitudes of p_i and r_i (See section 5.1.3). Thus, adjusting these probabilities while keeping the production rates in isolation of individual machines constant may introduce significant errors.

The fact that machining times are not only different, but also can be adjustable, gives rise to an optimal control problem which will only be touched upon here: the problem of controlling storage levels.

The continuous winder is sometimes operated below its maximum speed for two reasons (Gordon-Clark[1977]): the limitation imposed upon it by the speed of the next downstream machine (the log saw), and the assumption that higher speed causes a higher failure rate, which is undesirable since the continuous winder is difficult to restart. Tests have indicated, however, that in some cases the continuous winder can be operated at 20% faster than its present speed (though the failure rate is likely to increase). Furthermore, it has been acertained that the continuous winder can also be operated as slowly as 25% of its present speed.

In addition, it is known that the crucial part of the line is the first storage and its adjacent machines, since the wrapper fails considerably less often (See sections 5.1.3, 5.2). Thus, controlling the speed of the continuous winder could significantly improve the production rate of the system.

Such real time control raises a number of interesting questions. These include:

(i) When the log saw (or the wrapper) is down and/or the level of the storage is high, would decreasing the speed of the continuous winder be profitable? Buzacott[1969] states that it can be shown by using linear programming that the optimal policy for operating machines is never to slow them down. However, restarting a forced down continuous winder requires operator action and can result in the loss of several defective logs. Thus, it may be profitable to slow the continuous winder down so as to decrease the probability that it gets blocked. If so, what storage level should be considered high? How must all speeds of the continuous winder be computed to give optimal yield?

(ii) When the parent roll is close to exhausted, would increasing the speed of the continuous winder in order to fill up the storage improve the system production rate? This increase would result in providing the log saw with workpieces to process at least during part of the time in which a new parent roll is loaded. Considering that higher speed is likely to mean a higher failure rate, how much faster should it run? If a failure does occur, when is it more profitable to discard the almost empty parent roll and load a new one?

(iii) While the system operates normally, is it profitable to control the speed of the continuous winder in order to maintain a certain optimal minimum level in the storage? (From Buzacott[1969], decreasing the speed in order to maintain a maximum level is known not to be profitable).

(iv) As stated in section 7.1.3, some logs manufactured while the continuous winder accelerates or decelerates can be defective and are discarded. How is this effect to be taken into account in controlling the speed of the continuous winder?

More could of course be said about this important problem, which may carry over to other systems as well. This question is beyond the scope of the present work, but is clearly of importance and constitutes a possible direction for future research.

7.1.5 The Conveyor Belt

The log saw and the wrapper are connected by a two-channel conveyor belt with a capacity of about forty rolls each.

It is common in the literature to encounter conveyor belts being referred to simply as in-process storage facilities (e.g. Richman and Elmaghraby[1957]). However, these differ from the idealized storage element considered here, in which a workpiece is available to the downstream machine as soon as it enters the storage, because of the delay involved in the transportation of pieces between stages. Pritsker [1966] observes that a power driven conveyor often corresponds to a no-storage line: in such a system, the parts are moved along with the belt at a speed equal to the production rates of the upstream and downstream machines. Thus, if a downstream machine fails, the conveyor must be halted. On the other hand, Pritsker states that a non-powered conveyor is identical with the limited storage case. It is suggested below that this is not necessarily true.

It is stated in section 2.2.5 that there is a delay of one cycle between the time a workpiece is completed at stage i and the time its processing starts at stage i+1 (assuming that the stages are operational and not forced down). A conveyor in which a piece leaving machine i moves fast enough that it reaches machine i+1 in at most one cycle may be considered equivalent to the idealized storage described in section 2.1. If parts move at a slower speed on the conveyor, a different approach may be necessary to account for the time lost in transportation. In either case, a conveyor can be thought of as a storage element only if parts are allowed to accumulate on it, i.e. when the conveyor is not required to stop if a part reaches a machine which is not ready to take it in.

Conveyor belts have been analytically studied by Kwo[1958] and others, and numerous researchers have used simulation techniques in their work (e.g. Kay[1972], Barten[1962]).

In the case of the conveyor belt in the paper finishing line, it may be possible to model the conveyor belt as a series of perfectly reliable machines with limited storage between them. The number of stages is determined by the time (the number of cycles) needed for a part to travel from the log saw to the wrapper. However, this approach has the effect of increasing the number of machines in the model, thereby increasing its complexity.

7.1.6 The Failure and Repair Model

Assumption 2.2.3 implies that a forced-down machine is able to resume production as soon as the upstream storage ceases to be empty or the downstream storage ceases to be full. In accordance with this assumption, the state of a forced down machine is denoted by $\alpha_i=1$, indicating that it is in good repair.

It is often the case in industrial systems that the upstream machine is automatically shut down when a storage fills up and blocks it. In some cases, such as the continuous winder, restarting the machine requires human intervention and may even be costly and cause loss of product due to defects. The model as it now stands does not account for such events, although it can be extended.

For example, it is possible to define a third machine state, forced down. The transition from this state to the down state (i.e. failure when forced down) would have zero probability, while the transition to the up state (i.e. being restarted) would have a probability that may or may not be equal to r_i.

Similarly, the problem of having too few repairmen (or the machine interference problem - see Cox and Smith[1974]) is ignored here. However, it is important in actual systems and in particular, in the case of the paper plant discussed here, where several parallel paper finishing lines share a limited crew. It is possible to extended the model to account for this problem: for example, there may be a lower repair probability when two machines are down simultaneously than when only one machine is down (See Benson and Cox[1951]).

Lastly, the model assumes that a machine is starved if there are no pieces in the upstream storage, while in the actual line, the log saw is not allowed to operate if there is only one pair of logs left in the storage. Since there is always one pair in the storage, it can be ignored, and the storage capacity is given by N_1-1, rather than N_1.

7.2 A Brief Discussion of Some Attempts at Modeling the System

Although the behavior of a system as complex as the roll products paper finishing line can probably best be predicted by an extremely detailed computer simulation, such programs often take a long time to develop and are very costly, both in terms of manpower and computer time. Good mathematical models with analytical or relatively simple numerical solution techniques are therefore extremely useful in studying such systems.

Gordon-Clark[1977] modeled the system as a semi-Markov process; one version of his model was based on an 8-state process, where each state represents a combination of the states of the machines ((0,0,0) through (1,1,1)). The states thus do not take storage levels into account in this model. The state residence times and transition probabilities were estimated from records of the actual operation of the paper finishing line. If the line were still in the designing stage, these would have had to be guessed. Transition probabilities may in some cases be worked out from data from individual machines. However, the state residence times are more difficult to calculate, since they involve knowledge of average storage levels. Since Gordon-Clark's model does not consider storage levels as state variables, the residence times cannot be evaluated theoretically.

The predicted and actual results were not in excellent agreement. Among possible reasons for this discrepency, non-geometric actual failure and repair rates, dominating transients, and the effects of the past history on system behavior were proposed. To these may be added the fact that forced down and failed machines behave differently, a point that was not taken into account by this model, since storage levels were not state variables.

This last difficulty may be accounted for by two approaches: by defining three machine states (operational, failed, and forced down) instead of two; or by extending the definition of a state to include at least the three storage regions (empty, full, or otherwise). That the model would significantly improve if the above was done is suggested by the fact that the best agreement between predicted and actual behavior was for the continuous winder which is forced down least frequently; the worst agreement was for the wrapper, the stage which is idle most often.

Another approach would be to modify the transfer line model developed in chapter 2 to account for the discrepencies outlined in section 7.1. Some modifications proposed to adapt the model to the paper finishing line are relatively easy to implement, such as considering the smallest unit (the roll or pair of rolls) as a workpiece. Assuming that the parent roll loading time has negligible effects on the system, that rejects amount to a negligible fraction of total production, that the conveyor is fast enough to be equivalent to a storage, etc. may give satisfactory results. Some problems may introduce errors into the computation of system performance measures: for example, the fact that repair crews are limited in number implies that the repair probability is reduced when more than one machine fails; furthermore, even the order in which they fail matters. Still other issues, such as dominant transients, may require a completely different approach. A detailed analysis is required to verify the applicability of the transfer line model presented here to this particular system.

This is clearly beyond the scope of this work, which aims primarily at answering generic questions as opposed to studying specific systems. This chapter has tried to show several possible sources of difficulty which may arise in the application of an idealized mathematical model to real systems. The following two chapters investigate two qualitatively different cases, in an attempt to emphasize the flexibility of the model.

8. APPLICATION OF A TRANSFER LINE MODEL TO BATCH CHEMICAL PROCESSES

The discrete nature of the Markov chain model of a transfer line described here allows a wide range of applications, including not only obvious cases such as assembly and transfer lines in the metal cutting or electronic industries, but also chemical processes in which batches of chemicals proceed through stages in the manner of a production line.

A queueing theory approach to batch chemical systems is outlined following Stover's [1956] early work in section 8.1. Some of the differences between a model proposed by Koenigsberg[1959] and the present model are discussed in sections 8.1.1 and 8.1.2.

Applications of the transfer line model to such systems, as well as a discussion of the results thus obtained, appear in section 8.2.

8.1 A Queueing Theory Approach to the Study of Batch Chemical Processes

In an early paper, Stover[1956]* used a queueing theory approach to estimate the production rate of a chemical plant that was planned to be expanded. The system consisted of a stage of parallel reactors, followed by holding tanks, followed by a stage of parallel stills. Stover modeled the stills as exponential service time servers, and computed the number of holding tanks and stills needed to achieve the desired production rate. Essentially, his model was a single-stage, parallel-server queue, and techniques existed for its solution.

Basing himself on Stover's work, Koenigsberg[1959] proposed a schematic representation of a batch chemical process similar to that presented in figure 8.1. Represented thus, the plant may be studied as a transfer line, although certain important particularities of the system are not accounted for by the model as it stands.

Some of the differences between the system and the model are discussed in sections 8.1.1 and 8.1.2.

*Acknowledgment is due to the Archives of the United States Rubber Company, Naugatuck, Connecticut, for supplying Mr. Stover's unpublished manuscript.

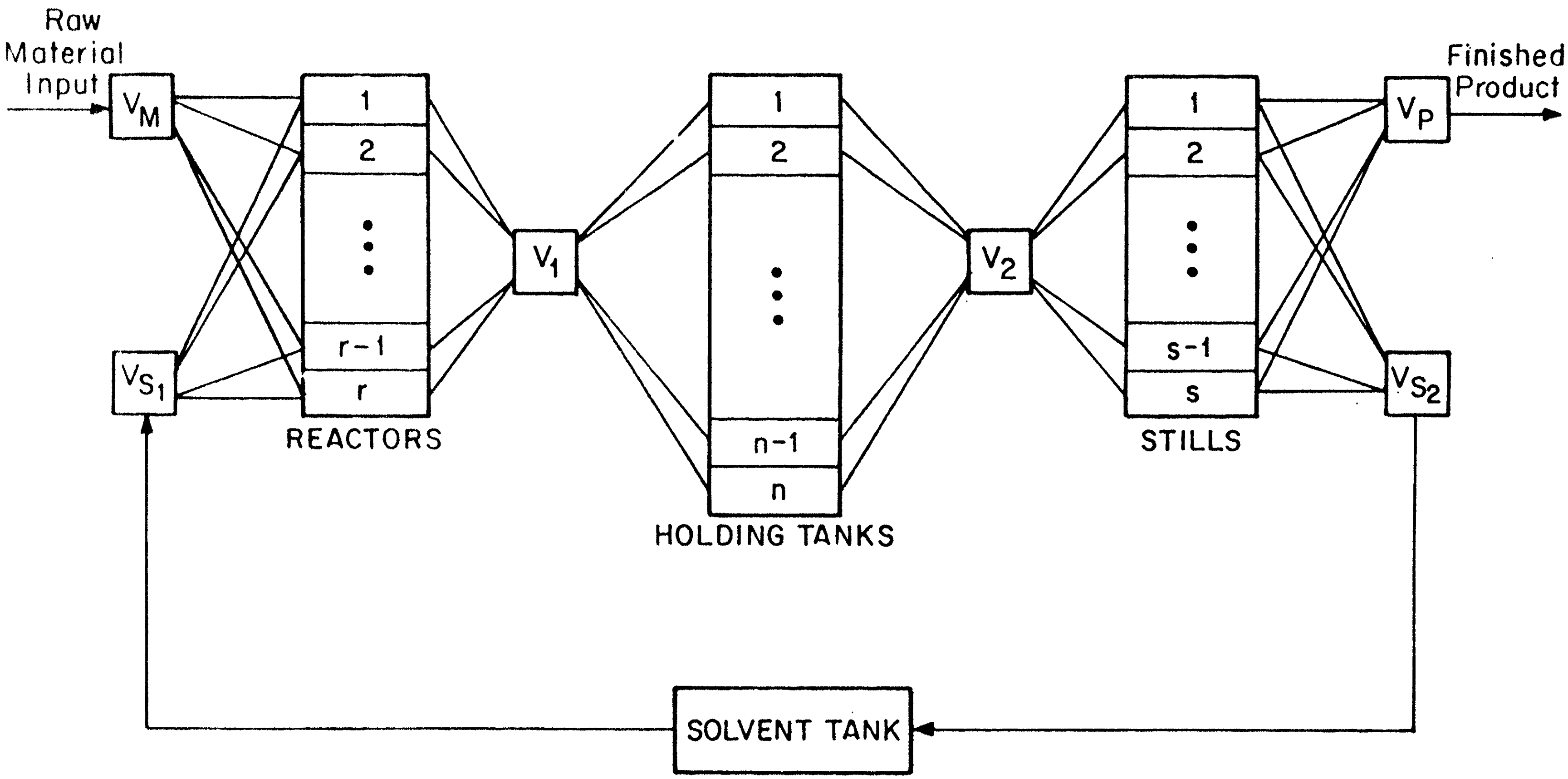


Figure 8.1. Schematic representation of a batch chemical process as a transfer line.

8.1.1 Non-Deterministic Processing Times

Stover[1956] reports that maintaining a fixed schedule and production rate for batches finishing in the stills is not possible because of unpredictable variations in reaction times. These variations may be due to fluctuations in feed temperature or concentration, changes in the activity of catalysts, etc. Thus, batches are modeled as taking random periods of time to be processed in the stills.

By assumption 2.2.2, the model of a transfer line developed in the present work involves stages that have deterministic and equal service times. Reliable lines with random cycle times have been studied by numerous researchers. Most of this work assumes exponentially distributed service times (Hillier and Boling[1967], Konheim and Reiser [1976], Neuts[1968], Muth[1973], Hunt[1956], Hatcher[1969], Knott[1970a, 1970b]). Rao[1975a] studied two stage lines with normal and Erlangian service times and no interstage storage; Neuts[1968] considers a line consisting of two stages, one of which has uniformly distributed service times. Buzacott[1972] studied a two-stage line with identical unreliable machines and exponential service times. Gershwin and Berman[1978] analyze a two-stage line with exponentially distributed service times and unreliable machines. Lines with more than three stages have not been successfully analyzed because of the complexity of the effects of blocking and starving when storages are full or empty (Okamura and Yamashina[1977]).

The transfer line model of chapter 2 is extended, following Gershwin and Berman[1978], to allow exponentially distributed service times, in section 8.2. Rao[1975b] states that exponential service time distributions often do not represent actual systems, where the service times are closer to normal distributions (Vladzievskii[1952], Koenigsberg[1959]). However, the solution of exponential service time models is an important step towards the analysis of models where the service times are given by Erlang distributions (See Brockmeyer, Halstrøm and Jensen[1960]). This is because a stage with Erang distributed service times may be thought of as a machine in which a series of distinct operations are performed on the workpiece, each of which takes an exponentially distributed length of time.

(See Wallace and Rosenberg[1966], Lavenberg, Traiger and Chang[1973], Herzog, Woo and Chandy[1974]). In other words, an Erlang stage is equivalent to a series of exponential stages. The importance of this lies in the fact that Erlang distributions may represent accurately normal distributions, which themselves best model chemical reaction time distributions.

8.1.2 Feedback Loops

The solvent recycling system in figure 8.1 cannot be accounted for by the present model. Given that the amount of solvent in the system is constant, such a system could be modeled by a cyclic queueing network of the kind analyzed by Koenigsberg[1958] or Finch[1959]. However, certain additional assumptions make it possible to use the present model in studying the system in figure 8.1. These are the following:

(i) If the solvent tank is large enough, the inlet and outlet of the plant are effectively decoupled. In this case, the last stage is never blocked because of a failure or blocking in the first stage. Thus, recycling the solvent does not change the structure of the model.

(ii) It may be assumed that in case either the solvent inlet pump V_{s1} or the raw material inlet V_m fail, the first stage fails because both mechanisms must operate for the reactors to be fed. Similarly, if either V_{s2} or V_p fail, the last stage may be assumed to fail.

Thus, it may be possible to model V_{s1} and V_m as a single machine; similarly, V_{s2} and V_p may be considered a single machine. The condition for this to hold is that the repair times of V_{s1} and V_m, as well as those of V_{s2} and V_p, are identical. In this case, the failure and repair probabilities of the single machine equivalents may be computed as follows: Given that V_{s1} fails with probability p_{s1} and V_m fails with probability p_m, rhe single machine equivalent remains up during a cycle if both V_{s1} and V_m remain up. Thus,

$$p = 1 - (1 - p_{s1})(1 - p_m) \tag{8.1}$$

On the other hand, given that V_{s1} and V_m have equal repair rates r, the repair rate of the equivalent single machine is simply equal to r. Thus, the equivalent machine has geometric up and down time distributions and the model of Gershwin and Berman[1978] may be applied. A similar argument can be made for V_{s2} and V_p.

(iii) It may be assumed that the amount of solvent in the system is sufficient so that the first stage is never starved. This assumption completes the decoupling of the first and last stages in the transfer line model of the batch system.

It may be noted that by assuming infinite storage capacities, Secco-Suardo[1978] shows that in a closed network with large numbers of customers (in a production line, pellets, in the batch chemical plant, batches of solvent, etc.), one stage always acts as a bottleneck, so that the system is equivalent to an open network.

By making the above assumptions, the system pictured in figure 8.1 may be treated as a transfer line.

8.2 The Production Rate of a Batch Chemical Process

8.2.1 The Single Reactor, Single Still Case

As outlined in section 8.1, the transfer line model is extended to cover a system of the type described by Koenigsberg[1959]. Stover[1956] studies the system as a single-stage, multiple-server queue; he assumes that the stills have exponentially distributed service times, and that the holding tanks comprise a finite queue. If the reactors are also modeled as having exponentially distributed service times, the single reactor, single still case can be analyzed by means of the results derived by Gershwin and Berman[1978].

The system considered here consists of two stages; these are the reactor and the still. Both stages include all pumps, valves, and other devices through which batches of chemicals are transfered. The stages are unreliable in the sense that they occasionally fail, due to unpredictable failures in pumps, heating or cooling systems, and so on. A finite number of parallel holding tanks are located between the two stages. The object of the study is to compute the effects of the variations in service times on the production rate of the system, and to see how these effects can be mitigated by the use of interstage holding tanks.

As stated in section 8.1.1, the service times are assumed to be exponentially distributed, although this assumption may not hold for batch chemical processes. It is hoped that the exponential results will provide the theory necessary to help analyze and solve Erlangian systems.

The steady-state probabilities of the system are found by Gershwin and Berman[1978] by assuming a solution for internal states of the form of equation (3.13), and substituting the expression into detailed balance equations for internal states. This development is analogous to the derivation in chapter 3. Here, however, the stages operate with variable service times. The mean service time for stage i is given by $1/\mu_i$. A consequence of the fact that the stages are not synchronous is that the boundaries reduce to n=0 and n=N, instead of $n \leq 1$ and $n \geq N-1$ as in the deterministic service time case. Thus, the states with n=1 and n=N-1 have probabilities with internal form expressions.

For the two-stage exponential service time transfer line, the internal equations (analogous to (3.21) and (3.23) in the deterministic case) are the following:

$$\mu_1\left(\frac{1}{X_1} - 1\right) - p_1 Y_1 + r_1 + \frac{r_1}{Y_1} - p_1 = 0 \tag{8.2}$$

$$\mu_2(X_1 - 1) - p_2 Y_2 + r_2 + \frac{r_2}{Y_2} - p_2 = 0 \tag{8.3}$$

$$p_1 Y_1 + p_2 Y_2 - r_1 - r_2 = 0 \tag{8.4}$$

These constitute a set of three non-linear equations in three unknowns, and may be combined into a fourth order polynomial in terms of one of the variables, say Y_1. In this case, it is easy to verify that $Y_1 = r_1/p_1$ is a root, so that the polynomial becomes

$$\left(Y_1 - \frac{r_1}{p_1}\right)\left(Y_1^{\,3} + \beta_1 Y_1^{\,2} + \beta_2 Y_1 + \beta_3\right) = 0 \tag{8.5}$$

where β_j, j=1,2,3 are functions of r_i and p_i, i=1,2.

The cubic polynomial has its roots at

$$Y_{1i} = 2\sqrt{-\frac{a}{3}}\ \cos\theta_i \quad ; \quad i=2,3,4 \tag{8.6}$$

where

$$\left.\begin{aligned} a &= \frac{1}{3}(3\beta_2 - \beta_1^{\,2}) \\ b &= \frac{1}{27}(2\beta_1^{\,3} - 9\beta_1 a + 27\beta_3) \\ \phi &= \operatorname{arc\,cos} \frac{-b/2}{\sqrt{-a^3/27}} \end{aligned}\right\} \tag{8.7}$$

and

$$\theta_i = \frac{1}{3}\phi + 120(i-2) \quad ; \quad i=2,3,4$$

(ϕ and θ_i measured in degrees.)

After Y_{1i} are found from equations (8.6) and (8.7), these values are substituted into (8.2)-(8.4) and Y_{2i} and X_i are found. Thus, the solution is of the form of (3.13):

$$p[n,\alpha_1,\alpha_2] = \sum_{i=1}^{4} C_i\, X_i^{\,n}\, Y_{1i}^{\,\alpha_1}\, Y_{2i}^{\,\alpha_2} \tag{8.8}$$

The coefficients C_i are found by using boundary detailed balance equations (analogous to boundary transition equations in section 3.2). For the root $Y_1=r_1/p_1$ mentioned above, it is found that $Y_2=r_2/p_2$ and $X=1$; the constant C_1 corresponding to this root is found to be zero, as in the deterministic case. This is noteworthy, because the set Y_{11}, Y_{21}, X_1 corresponds in both the exponential and the deterministic case to a solution that assumes the stages in the system to be decoupled. That $C_1=0$ implies that this is not true.

These results are now used in a numerical example.

8.2.2 A Numerical Example

The following system is considered: a batch reactor and a still are separated by N=10 parallel holding tanks. The first stage consists of the reactor as well as valves, pumps, etc. which serve to transmit the batches of chemicals; the second stage similarly consists of the still, as well as pumps etc. Both stages are unreliable, and randomly fail because of breakdowns in the pumps, in temperature control mechanisms, and in other unreliable devices. The production rates in isolation of both stages are equal to 0.5 batches / time unit.

The still has failure and repair rates (in probability / time unit) equal to $p_2=r_2=1.0$. Its service times are exponentially distributed with mean $1/\mu_2=1.0$.

The reactor failure and service rates p_1 and μ_1 are varied in such a way as to hold average production rate constant at 0.5 batches / time unit. The repair rate (in probability / time unit) is equal to $r_1=1.0$.

Results for some values of p_1 and μ_1 appear in table 8.1. It is seen that as the efficiency in isolation of the first stage is increased and its service rate decreased, the line production rate increases. This suggests that for these system parameters, the fluctuations in service times influence line production rate more strongly than the failures in the first machine. This is important, because in practice, chemical systems are often highly reliable, although such variations in service times may sometimes be unavoidable. Random service times can be used to model human intervention in the processing of batches. Although humans may be highly reliable, it is clear that variations in service times cannot be avoided. From these results, it follows that it is more important to control fluctuations in service times than improve the reliability of the first stage in this line.

The experiment is repeated this time by varying p_2 and μ_2 so as to

p_1	r_1	e_1	μ_1	Line efficiency (Production rate)
9	1	.1	5	0.432
2	1	.33	1.5	0.435
1	1	.5	1	0.438
.5	1	.67	.75	0.441
.11	1	.9	.55	0.442

Table 8.1. System parameters and line production rate for a two-machine line with exponential service times.

maintain the production rate of the second stage at a constant value equal to 0.5 batches / time unit. The other system parameters are set at $p_1=r_1=\mu_1=r_2=1.0$.

The same values as those given for p_1 and μ_1 in table 8.1 are assigned to p_2 and μ_2.

The results obtained confirm that efficiency is more important than service rates (or alternately, reducing variations in service times is more beneficial than improving efficiency in isolation, given a constant production rate in isolation) for these system parameters.

Furthermore, it is observed that the line production rate is symmetrical with the orientation of the production line. Thus, when the parameters of the two stages are reversed, the line production rate does not change. This is the case with deterministic service time transfer lines also.

8.2.3 Parallel Reactors or Stills

The system discussed in sections 8.2.1 and 8.2.2 is a simple case of the general class of systems discussed in section 8.1 and schematically illustrated by figure 8.1. The present state of the model is not able to deal with networks with non-linear topologies, such as those involving stages with multiple servers. Ignall and Silver[1977] give an approximate method to calculate the production rate of a two-stage, multiple-server system with deterministic service times. Single stage queues have been treated by several authors including Morse[1965], Galliher[1962], and Disney[1962,1963].

Future studies of multichannel stage transfer lines may be based on defining the system states as not only the operational conditions of the machines, but the number of operational machines in each stage. If each machine has exponentially distributed service times, it may be possible to represent the stage as a single machine with Erlang service time distributions (See Lavenberg, Traiger and Chang[1973]).

A different approach may be modeling the number of operational machines in any stage as a birth-death process (See also Taylor and Jackson[1954]). This assumes that the probability that more than one machine fails simultaneously is small enough to be neglected. The production rate of the stage at any time can then be expressed as a function of the number of operational machines in the stage, as well as the levels of the storages upstream and downstream of the stage. These levels affect the stage if fewer batches are available in the upstream storage than there are operational machines, or if less storage space is available in the downstream storage than there are operational machines.

A sufficient range of applications exists to make the results of linear topology production lines described in this work of interest. However, it is clear that their applicability will greatly increase if these results can be extended to more complex networks as well.

9. APPLICATION OF A TRANSFER LINE MODEL TO CONTINUOUS CHEMICAL PROCESSES

Although the preceding discussion has centered on discrete transfer lines (i.e. lines in which discrete workpieces travel through the system), it is possible to extend the model to continuous transfer lines. In such systems, the storage level is treated as a continuous variable.

Continuous models are often good approximations to discrete queueing networks with large numbers of customers (Newell[1971]); in the case of transfer lines, they can be good approximations to the discrete system if the storage capacities are very large (Sevast'yanov[1962]). Since continuous models can be studied by means of differential equations, the computation needed to obtain the steady-state probability distribution of the system can be greatly reduced by making this approximation. In addition, the continuous transfer line model can be used in the study of unreliable hydraulic systems (Buzacott[1971]) or continuous chemical processes. Here, fluids or chemicals flow through series of unreliable stages separated by holding tanks. By using the steady-state probability distributions, it is possible to find the relations between the failure and repair rates and holding tank sizes, and performance measures such as the flow rate through the system, the amount of material in the tanks, etc.

Two approaches to the problem are discussed here. A differential equations approach for obtaining the probability density functions is reviewed in section 9.1. The two-machine discrete line analytical results of chapter 3 and the δ-transformation of section 6.3 are used to arrive at identical results in section 9.2. A numerical example is worked out and discussed in section 9.3.

9.1 The Continuous Transfer Line Model

By assuming that the buffer storage level may be treated as a continuous variable, Graves[1977] has derived a series of probability density functions that describe the steady-state probability distribution of the system states for a two-machine line. These probability density functions are denoted by $f(x,\alpha_1,\alpha_2)$, where x is the level of the storage $(0 \leq x \leq N)$ and α_1 and α_2 are the machine states as defined in section 2.1.2. Graves' derivation is summarized below.

To obtain the probability density functions $f(\cdot)$, it is necessary to consider transient local balance equations. Denoting the transient probability density function by $f(x,\alpha_1,\alpha_2,t)$, where t is time, for a small time increment Δ and internal storage level $(0<x<N)$,

$$f(x,1,1,t+\Delta) = (1-p_1\Delta-p_2\Delta)\ f(x,1,1,t) + r_1\Delta\ f(x,0,1,t) + r_2\Delta\ f(x,1,0,t) + O(\Delta^2) \qquad (9.1)$$

where $O(\Delta^2)$ denotes terms of order Δ^2 and above. Equation (9.1) is a balance equation on the probability of being in state (x,1,1) at time $t+\Delta$. The parameters p_i and r_i are failure and repair rates, not probabilities. Thus, for small Δ, the products $p_i\Delta$ and $r_i\Delta$ are the probabilities of failure and repair of machine i. Given that the system is in state (x,1,1) at time t, it stays in that state over the increment Δ with probability $(1-p_1\Delta)(1-p_2\Delta)$; if the system is in states (x,0,1) or (x,1,0), the transition probabilities in the small time increment Δ are $r_1\Delta(1-p_2\Delta)$ and $(1-p_1\Delta)r_2\Delta$ respectively. Finally, the probability of transition from (x,0,0) to (x,1,1) is $r_1\Delta r_2\Delta$. The terms of order Δ^2 are lumped together as a first-order approximation, and equation (9.1) directly follows. Letting $\Delta\rightarrow 0$ and making the steady-state assumption (i.e. assuming that $\frac{d}{dt}(\cdot)=0$), it follows that

$$(-p_1-p_2)\ f(x,1,1) + r_1\ f(x,0,1) + r_2\ f(x,1,0) = 0 \qquad (9.2)$$

Similarly,

$$f(x,0,0,t+\Delta) = (1-r_1\Delta-r_2\Delta)\ f(x,0,0,t) + p_1\Delta\ f(x,1,0,t) + p_2\Delta\ f(x,0,1,t) + o(\Delta^2) \tag{9.3}$$

Again, letting $\Delta\to 0$ and making the steady-state assumption, equation (9.3) gives

$$(-r_1-r_2)\ f(x,0,0) + p_1\ f(x,1,0) + p_2\ f(x,0,1) = 0 \tag{9.4}$$

In both equations (9.1) and (9.3), the final machine states are such that the storage levels do not change within the time increment Δ. Given that the storage levels are internal, the level goes down by Δ in the time increment Δ if the second machine is up while the first is down. The balance equation is

$$f(x-\Delta,0,1,t+\Delta) = (1-r_1\Delta-p_2\Delta)\ f(x,0,1,t) + p_1\Delta\ f(x,1,1,t) + r_2\Delta\ f(x,0,0,t) + o(\Delta^2) \tag{9.5}$$

When $\Delta\to 0$ and the steady-state assumption is made, (9.5) gives

$$-\frac{d}{dx}\ f(x,0,1) = (-r_1-p_2)\ f(x,0,1) + p_1\ f(x,1,1) + r_2\ f(x,0,0) \tag{9.6}$$

Similarly, the differential equation giving the probability density of internal states with an operational first machine and a failed second machine is:

$$\frac{d}{dx}\ f(x,1,0) = (-r_2-p_1)\ f(x,1,0) + p_2\ f(x,1,1) + r_1\ f(x,0,0) \tag{9.7}$$

Equations (9.2), (9.4), (9.6), and (9.7) determine the steady-state probability density functions for the internal states. The boundary probability mass functions are found by using balance equations analogous to the boundary transition equations of section 3.2. Assumption 2.2.3 states that machines can only fail while processing parts. Thus, it is not possible for a failed machine to be preceeded by an empty storage or followed by a full storage.* As a consequence, some probability mass functions $p[\cdot]$ at the boundaries $x=0$ and $x=N$ are found to be identically zero:

$$p[0,0,0] = p[0,1,0] = 0 \tag{9.8}$$

$$p[N,0,0] = p[N,0,1] = 0 \tag{9.9}$$

In setting up the boundary balance equations, it is noted that boundary transition rates differ from internal transition rates, because of assumption 2.2.3. For example, the transition $(N,1,0)\to(N,1,0)$ occurs with probability $(1-r_2\Delta)$, rather than $(1-p_1\Delta)(1-r_2\Delta)$, since the the first machine cannot fail when it is blocked. On the other hand, the transition $(N,1,1)\to(N,1,1)$ occurs with probability $(1-p_1\Delta)(1-p_2\Delta)$, since the first machine is not blocked as long as the storage is drained simultaneously by the second machine, even if $x=N$. As a result, it is necessary to redefine the terms blocked and starved for the continuous transfer line. Here, a machine is blocked if its downstream storage is full <u>and</u> the downstream machine is down. Similarly, a machine is starved if its upstream storage is empty <u>and</u> the upstream machine is down. These definitions differ from those for the discrete line given in chapter 2.

From these conditions, it follows that at the upper boundary $(x=N)$,

$$p[N,1,0,t+\Delta] = (1-r_2\Delta)\, p[N,1,0,t] + p_2\Delta\, p[N,1,1,t] + \int_{N-\Delta}^{N} f(x,1,0,t)\, dx + o(\Delta^2) \tag{9.10}$$

---

* This assumption causes the boundary conditions presented here to differ slightly from those in the work of Graves.

The integral in (9.10) accounts for the probability that the first machine remains up and the second remains down through the time increment Δ. Since the machines operate at unit rate (as before - note that Graves' results include the case when they operate at different rates), the limits of the integral go from $N-\Delta$ to N. It is noted that

$$\lim_{\Delta \to 0} \int_{N-\Delta}^{N} \frac{f(x,1,0,t)}{\Delta} dx = f(N,1,0,t) \tag{9.11}$$

Thus, letting $\Delta \to 0$ and assuming steady-state, (9.10) becomes

$$0 = -r_2 \, p[N,1,0] + p_2 \, p[N,1,1] + f(N,1,0) \tag{9.12}$$

Similarly,

$$p[N,1,1,t+\Delta] = (1-p_1\Delta-p_2\Delta) \, p[N,1,1,t] + r_2\Delta \, p[N,1,0,t] + o(\Delta^2) \tag{9.13}$$

which gives

$$0 = (-p_1-p_2) \, p[N,1,1] + r_2 \, p[N,1,0] \tag{9.14}$$

At the lower boundary (x=0), analogous balance equations are

$$p[0,0,1,t+\Delta] = (1-r_1\Delta) \, p[0,0,1,t] + p_1\Delta \, p[0,1,1,t] + \int_{\Delta}^{0} f(x,0,1,t) \, dx + o(\Delta^2) \tag{9.15}$$

$$p[0,1,1,t+\Delta] = (1-p_1\Delta-p_2\Delta) \, p[0,1,1,t] + r_1\Delta \, p[0,0,1,t] + o(\Delta^2) \tag{9.16}$$

which give

$$0 = -r_1\, p[0,0,1] + p_1\, p[0,1,1] + f(0,0,1) \tag{9.17}$$

$$0 = (-p_1-p_2)\, p[0,1,1] + r_1\, p[0,0,1] \tag{9.18}$$

Equations (9.12), (9.14), (9.17), and (9.18) are the steady-state boundary transition equations. These may be simultaneously solved, giving:

$$p[N,1,0] = \left[\frac{p_1 + p_2}{r_2 p_1}\right] f(N,1,0) \tag{9.19}$$

$$p[N,1,1] = \frac{1}{p_1} f(N,1,0) \tag{9.20}$$

$$p[0,0,1] = \left[\frac{p_1 + p_2}{r_1 p_2}\right] f(0,0,1) \tag{9.21}$$

$$p[0,1,1] = \frac{1}{p_2} f(0,0,1) \tag{9.22}$$

Equations (9.19)-(9.20) are found by solving (9.12) and (9.14) simultaneously; equations (9.21)-(9.22) are found by solving (9.17) and (9.18) simultaneously.

The internal balance equations may be solved by first adding (9.2), (9.4), (9.6), and (9.7) together, giving

$$\frac{d}{dx} f(x,1,0) - \frac{d}{dx} f(x,0,1) = 0 \tag{9.23}$$

Equation (9.23) is solved to give

$$f(x,1,0) = f(x,0,1) + K_1 \tag{9.24}$$

The constant K_1 is now evaluated. Given that the system is in state $(N,1,1)$ at time t and that the first machine fails during the time increment Δ (this is possible, since machine 1 is not blocked as long as machine 2 is operational), it follows that

$$f(N,0,1,t+\Delta) = p_1\Delta\, p[N,1,1,t] + o(\Delta^2) \qquad (9.25)$$

which gives, as $\Delta\to 0$ and steady-state is assumed,

$$f(N,0,1) = p_1\, p[N,1,1] \qquad (9.26)$$

This is the third boundary balance equation at the upper boundary. Adding (9.12), (9.14), and (9.26), it follows that

$$f(N,0,1) = f(N,1,0) \qquad (9.27)$$

Similarly, at the lower boundary,

$$f(0,1,0,t+\Delta) = p_2\Delta\, p[0,1,1,t] + o(\Delta^2) \qquad (9.28)$$

which gives the third lower boundary balance equation,

$$f(0,1,0) = p_2\, p[0,1,1] \qquad (9.29)$$

Adding (9.17), (9.18), and (9.29), it follows that

$$f(0,1,0) = f(0,0,1) \qquad (9.30)$$

From either of equations (9.27) and (9.30), it follows that $K_1=0$ in equation (9.24). Thus,

$$f(x,1,0) = f(x,0,1) \qquad (9.31)$$

Using (9.31), equations (9.2) and (9.4) give

$$f(x,1,1) = \left[\frac{r_1 + r_2}{p_1 + p_2}\right] f(x,0,1) \qquad (9.32)$$

$$f(x,0,0) = \left[\frac{r_1 + r_2}{p_1 + p_2}\right] f(x,0,1) \qquad (9.33)$$

Substituting (9.32) and (9.33) into (9.6), it is found that

$$-\frac{d}{dx} f(x,0,1) = \left(-r_1 - p_2 + p_1\left[\frac{r_1+r_2}{p_1+p_2}\right] + r_2\left[\frac{p_1+p_2}{r_1+r_2}\right]\right) f(x,0,1)$$

$$= \left((p_1 r_2 - p_2 r_1)\left[\frac{1}{p_1+p_2} + \frac{1}{r_1+r_2}\right]\right) f(x,0,1)$$

$$\triangleq \lambda\, f(x,0,1) \tag{9.34}$$

Solving (9.34) gives

$$f(x,0,1) = K_2 \ e^{-\lambda x} \tag{9.35}$$

and from (9.31),

$$f(x,1,0) = K_2 \ e^{-\lambda x} \tag{9.36}$$

As Graves notes, if the machines have equal efficiencies, so that $(r_1/p_1)=(r_2/p_2)$, then $\lambda=0$ and the distribution in equations (9.35) and (9.36) are uniform.

The normalization constant K_2 is found by noting that the sum of the probability mass functions and the integrals of the density functions must add up to 1. Thus,

$$\sum_{\alpha_1=0}^{1} \sum_{\alpha_2=0}^{1} \int_0^N f(x,\alpha_1,\alpha_2)\, dx + p[0,0,1] + p[0,1,1] + p[N,1,0] + p[N,1,1] = 1 \tag{9.37}$$

The probability distributions of a two-machine line where the storage level is a continuous variable are thus completely determined. These functions are summarized in table 9.1. Continuous transfer lines with more than two machines give rise to very complex systems of equations (Sevast'yanov[1962], Graves[1977], Gordon-Clark[1977]) which have not yet been solved.

Table 9.1. Steady-state probability distributions for two-stage continuous lines.

$$f(x,0,0) = \frac{p_1 + p_2}{r_1 + r_2} \quad K_2 \; e^{-\lambda x}$$

$$f(x,0,1) = K_2 \; e^{-\lambda x}$$

$$f(x,1,0) = K_2 \; e^{-\lambda x}$$

$$f(x,1,1) = \frac{r_1 + r_2}{p_1 + p_2} \quad K_2 \; e^{-\lambda x}$$

$$p[0,0,0] = 0$$

$$p[0,0,1] = \frac{p_1 + p_2}{r_1 p_2} \quad K_2$$

$$p[0,1,0] = 0$$

$$p[0,1,1] = \frac{1}{p_2} \; K_2$$

$$p[N,0,0] = 0$$

$$p[N,0,1] = 0$$

$$p[N,1,0] = \frac{p_1 + p_2}{r_2 p_1} \quad K_2 \; e^{-\lambda N}$$

$$p[N,1,1] = \frac{1}{p_1} \; K_2 \; e^{-\lambda N}$$

The constants K_2 and λ are given by equations (9.37) and (9.34) respectively.

9.2 The δ-Transformation and its Limit as $\delta \to 0$

In section 6.3, the δ-transformation is introduced as follows:

$$\left.\begin{aligned} \bar{r}_i &\triangleq r_i\delta \\ \bar{p}_i &\triangleq p_i\delta \\ \bar{N} &\triangleq N/\delta \end{aligned}\right\} \tag{9.38}$$

The resulting system is equivalent to the original system with cycle length equal to δ, and the efficiency is virtually unchanged by the transformation.

The steady-state probabilities of the transformed system are now computed. The resulting expressions are shown to approach the continuous results of section 9.1 as $\delta \to 0$.

From equation (3.25),

$$\bar{Y}_1 = \frac{r_1 + r_2 - (r_1r_2 + p_2r_1)\delta}{p_1 + p_2 - (p_1p_2 + p_2r_1)\delta} \tag{9.39}$$

and

$$\bar{Y}_2 = \frac{r_1 + r_2 - (r_1r_2 + p_1r_2)\delta}{p_1 + p_2 - (p_1p_2 + p_1r_2)\delta} \tag{9.40}$$

where $\bar{Y}_i$ are the parameters in equation (3.13) for the <u>transformed</u> system. Each of the expressions in section (9.39) and (9.40) may be written as first order Taylor expansions around $\delta=0$:

$$\bar{Y}_i = \bar{Y}_{i0} + \bar{Y}_{i1}\delta + o(\delta) \quad ; \quad i=1,2 \tag{9.41}$$

where

$$\begin{aligned} \bar{Y}_{i0} &= \left[\bar{Y}_i\right]_{\delta=0} \\ &= \frac{r_1 + r_2}{p_1 + p_2} \quad ; \quad i=1,2 \end{aligned} \tag{9.42}$$

and

$$\bar{Y}_{11} = \frac{d}{d\delta}\left[\bar{Y}_1\right]_{\delta=0}$$

$$= \frac{(r_1+r_2)(p_1p_2+r_2p_1)-(p_1+p_2)(r_1r_2+p_2r_1)}{(p_1+p_2)^2} \tag{9.43}$$

$$\bar{Y}_{21} = \frac{d}{d\delta}\left[\bar{Y}_2\right]_{\delta=0}$$

$$= \frac{(r_1+r_2)(p_1p_2+r_1p_2)-(p_1+p_2)(r_1r_2+p_1r_2)}{(p_1+p_2)^2} \tag{9.44}$$

Similarly, from equations (3.25) and (9.41),

$$\bar{X} = \bar{Y}_2 / \bar{Y}_1 \tag{9.45}$$

$$= \frac{\bar{Y}_{20} + \bar{Y}_{21}\delta}{\bar{Y}_{10} + \bar{Y}_{11}\delta} + o(\delta) \tag{9.46}$$

Using Taylor's theorem, (9.46) becomes

$$\bar{X} = \frac{\bar{Y}_{20}}{\bar{Y}_{10}} + \left[\frac{\bar{Y}_{10}\bar{Y}_{21} - \bar{Y}_{20}\bar{Y}_{11}}{\bar{Y}_{10}^{\,2}}\right]\delta + o(\delta) \tag{9.47}$$

which, by using equations (9.42)-(9.44) and simplifying considerably, gives

$$\bar{X} = 1 + \left((p_2r_1-p_1r_2)\left[\frac{1}{p_1+p_2} + \frac{1}{r_1+r_2}\right]\right)\delta + o(\delta) \tag{9.48}$$

$$= 1 - \lambda\,\delta \tag{9.49}$$

where λ is the same constant that is defined in equation (9.34). The definition of internal states for the modified system is derived from equation (2.20). Since the transformation has the effect of dividing parts into $1/\delta$ slices (See section 6.3), internal states are defined to be those for which the storage level n/δ obeys the relation

$$\frac{2}{\delta} \leq \frac{n}{\delta} \leq \frac{N-2}{\delta} \tag{9.50}$$

Equation (3.13) for the transformed system is

$$\bar{p}[\tfrac{n}{\delta},\alpha_1,\alpha_2] = C\,\bar{X}^{\frac{n}{\delta}}\,\bar{Y}_1^{\alpha_1}\,\bar{Y}_2^{\alpha_2} \tag{9.51}$$

$$= C\,(1-\lambda\delta)^{n/\delta}(\bar{Y}_{10}+\bar{Y}_{11}\delta)^{\alpha_1}(\bar{Y}_{20}+\bar{Y}_{21}\delta)^{\alpha_2} + o(\delta) \tag{9.52}$$

It is noted that

$$\lim_{\delta\to 0}(1-\lambda\delta)^{n/\delta} = e^{-n\lambda} \tag{9.53}$$

For the continuous system, the storage level is denoted by x, and the steady-state probability density function is defined as follows:

$$f(x,\alpha_1,\alpha_2) = \frac{\bar{p}[\frac{n+\delta}{\delta},\alpha_1,\alpha_2] - \bar{p}[\frac{n}{\delta},\alpha_1,\alpha_2]}{\delta} \tag{9.54}$$

For each combination of machine states α_i, equation (9.52) becomes, as $\delta\to 0$,

$$f(x,0,0) = C\,e^{-\lambda x} \tag{9.55}$$

$$f(x,0,1) = C\,e^{-\lambda x}\left[\frac{r_1+r_2}{p_1+p_2}\right] \tag{9.56}$$

$$f(x,1,0) = C\,e^{-\lambda x}\left[\frac{r_1+r_2}{p_1+p_2}\right] \tag{9.57}$$

$$f(x,1,1) = C\,e^{-\lambda x}\left[\frac{r_1+r_2}{p_1+p_2}\right]^2 \tag{9.58}$$

It is easy to verify that equations (9.55)-(9.58) agree with the probability density functions given in table 9.1, with

$$K_2 = C \frac{r_1 + r_2}{p_1 + p_2} \tag{9.59}$$

The boundary state probability expressions of section 3.2.1 are shown below to reduce to the results of section 9.1 as $\delta \to 0$. It is noted that in the discrete line, n=1 and n=N-1 are considered boundary storage levels. In the transformed system, this corresponds to $n/\delta = 1/\delta$ and $n/\delta = (N-1)/\delta$. Thus, as $\delta \to 0$, the boundary becomes x=0 and x=N only.

From table 3.1,

$$p[0,0,1] = C\, X \frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_2 r_1} \tag{9.60}$$

For the transformed system, equation (9.60) becomes

$$\bar{p}[0,0,1] = C\, \bar{X} \frac{\bar{r}_1 + \bar{r}_2 - \bar{r}_1 \bar{r}_2 - \bar{p}_2 \bar{r}_1}{\bar{p}_2 \bar{r}_1}$$

$$= C\, \bar{X} \frac{r_1 + r_2 - (r_1 r_2 + p_2 r_1)\delta}{p_2 r_1 \delta} \tag{9.61}$$

Noting that as $\delta \to 0$, $\bar{X} \to 1$, the limit as $\delta \to 0$ of (9.61) is

$$\bar{p}[0,0,1] = \frac{C}{\delta} \frac{r_1 + r_2}{p_2 r_1} \tag{9.62}$$

It is shown in section 6.3 that the normalizing constant C is of the order of δ. This follows from the fact that C is the reciprocal of the sums of the probability mass functions and the integrals of the density functions, and that all these functions are of the order of $1/\delta$. Thus, the expression in (9.62) is bounded as $\delta \to 0$; this applies to all the other probability mass functions as well.

Similarly, from table 3.1,

$$p[N,1,0] = C\,X^{N-1}\,\frac{r_1 + r_2 - r_1 r_2 - p_2 r_1}{p_1 r_2} \tag{9.63}$$

which gives, for the transformed system,

$$\bar{p}[N,1,0] = C\,\bar{X}^{\frac{N-\delta}{\delta}}\,\frac{r_1 + r_2 - (r_1 r_2 + p_1 r_2)\delta}{p_1 r_2 \delta} \tag{9.64}$$

As $\delta \to 0$, using (9.53) gives

$$\bar{p}[N,1,0] = \frac{C}{\delta}\,e^{-\lambda N}\,\frac{r_1 + r_2}{p_1 r_2} \tag{9.65}$$

Analogously, the limits of

$$\bar{p}[\delta,1,1] = \frac{C\,\bar{X}}{p_2 \delta}\;\frac{r_1 + r_2 - (r_1 r_2 + p_2 r_1)\delta}{p_1 + p_2 - (p_1 p_2 + p_2 r_1)\delta} \tag{9.66}$$

$$\bar{p}[\tfrac{N-\delta}{\delta},1,1] = \frac{C\,\bar{X}^{\frac{N-\delta}{\delta}}}{p_1 \delta}\;\frac{r_1 + r_2 - (r_1 r_2 + p_1 r_2)\delta}{p_1 + p_2 - (p_1 p_2 + p_1 r_2)\delta} \tag{9.67}$$

(from table 3.1, for the transformed system) become, as $\delta \to 0$, and using equation (9.53),

$$\bar{p}[0,1,1] = \frac{C}{p_2 \delta}\;\frac{r_1 + r_2}{p_1 + p_2} \tag{9.68}$$

$$\bar{p}[N,1,1] = \frac{C}{p_1 \delta}\,e^{-\lambda N}\;\frac{r_1 + r_2}{p_1 + p_2} \tag{9.69}$$

By using equation (9.59), it is easy to verify that equations (9.62), (9.65), (9.68), and (9.69) are identical to the corresponding expressions in table 9.1. Thus, the steady-state probabilities for a discrete two-machine line outlined in chapter 3 give results that are identical to those obtained by differential equations when the δ-transformation is applied and $\delta \to 0$, i.e. when the length of a machining cycle is allowed to approach zero.

9.3 The Production Rate of a Continuous Line

Happel[1967] notes that in some continuous industries, down times are not a major problem because it is possible to make up for the lost time once the repair is made. He adds, however, that in some large continuous systems, such as the petrochemical industry, down time is a more serious problem due to the unavailability of interstage storage capacity (Goff[1970]). This suggests that in some cases, particularly when up and down times are not excessively long compared to the service rates of the stages (See section 5.1.3), storage elements may make a contribution to the production rate of some actual continuous systems.

The production rate of a two-stage continuous line where both stages operate at unit service rates is given by

$$E = \sum_{\alpha_1=0}^{1} \int_0^N f(x,\alpha_1,1)\,dx \quad + \; p[N,1,1] \; + \; p[0,1,1]$$

$$= C \left[\frac{r_1+r_2}{p_1+p_2}\right]\left(\left[\frac{1}{p_2} + \frac{1}{\lambda}\left(1+ \frac{r_1+r_2}{p_1+p_2}\right)\right] + \left[\frac{1}{p_1} - \frac{1}{\lambda}\left(1+ \frac{r_1+r_2}{p_1+p_2}\right)\right] e^{-\lambda N}\right) \tag{9.70}$$

where C is obtained by normalizing the probability functions:

$$\frac{1}{C} = \sum_{\alpha_1=0}^{1} \sum_{\alpha_2=0}^{1} \int_0^N f(x,\alpha_1,\alpha_2)\,dx \quad + \; p[0,0,1] \; +$$

$$p[0,1,1] \; + \; p[N,1,0] \; + \; p[N,1,1] \tag{9.71}$$

$$= \left[\frac{r_1+r_2}{p_2}\left(\frac{1}{r_1} + \frac{1}{p_1+p_2}\right) + \frac{1}{\lambda}\left(1+ \frac{r_1+r_2}{p_1+p_2}\right)^2\right] +$$

$$\left[\frac{r_1+r_2}{p_1}\left(\frac{1}{r_2} + \frac{1}{p_1+p_2}\right) - \frac{1}{\lambda}\left(1+ \frac{r_1+r_2}{p_1+p_2}\right)^2\right] e^{-\lambda N} \tag{9.72}$$

A simple example is considered here. The system consists of a plug-flow reactor and a distillation column, separated by a holding tank of known finite capacity. The reactor and distillation column are taken as unreliable stages in a two-stage continuous line. The unreliable nature of these stages may be due to failure in pumps, in heating or cooling systems, or in other devices.

It is assumed that there is no volume change during the reaction. Thus, the flow rates through the stages are equal. Time is scaled so that this flow rate is 1 volume unit / time unit.

The system parameters are given in table 9.2. The volume of the holding tank is varied and the system production rate is computed by means of equation (9.70). Some values of efficiency for different storage capacities appear in table 9.3. The discrete line values corresponding to the same line parameters are also given for comparison.

It is seen that as in the discrete case, transfer line efficiency increases with storage capacity, but approaches an asymptote. From the discussion of the δ-transformation in section 6.3, it follows that the limiting efficiencies $E(0)$ and $E(\infty)$ are computed by using equations (5.51) and (5.53). It is easily verified that the results in table 9.3 (which were computed by setting $N=0$ and $N\to\infty$ in equation (9.70)) confirm this.

The results in table 9.3 are also an indication that the δ-transformation is a good approximation; since the continuous line results are equal to those for a transformed system as $\delta\to 0$, the good agreement between the discrete and continuous line results suggests that in many cases, it is possible to model a discrete line approximately by a continuous line, and obtain its efficiency with considerably less computation by using (9.70).

p_1	$= 0.01$	r_1	$= 0.09$
p_2	$= 0.02$	r_2	$= 0.1$

Table 9.2. System parameters for continuous line.

Storage Capacity	Continuous Efficiency	Discrete Efficiency
0	0.7627	0.7627
0.1	0.7634	-
1	0.7658	0.7695
10	0.7926	0.7899
100	0.8315	0.8319
∞	0.8333	0.8333

Table 9.3. Efficiency for continuous and discrete lines for several storage capacities.

10. SUMMARY, CONCLUSIONS AND FUTURE DIRECTIONS

The aim of the work presented here has been to obtain analytical and numerical methods in order to quantify the relationships between design parameters and performance measures in unreliable transfer lines with interstage buffer storages.

A Markov chain model is formulated (Chapter 2); the states of the system are defined as sets of numbers describing the operational conditions of the machines (up or down) and the number of parts waiting in the interstage queues. The steady-state probabilities of these states are sought in order to compute various system performance measures such as expected production rate, in-process inventory, and idle times.

A closed-form solution is guessed and used in the steady-state transition equations for obtaining expressions for state probabilities (Chapter 3). The solution of this system of equations for two-stage systems has been discussed in the literature. A method is developed to obtain solutions for longer transfer lines as well. The set of boundary transition equations is solved algebraically to give a set of expressions that are used in a sum-of-terms form solution. The number of expressions to be derived changes with the number of stages in the line, but not with the capacities of interstage buffer storages. Once these expressions are found, a small system of equations (whose dimensions are linear with storage capacities in three-machine lines) is solved to obtain the steady-state probabilities of the system. Nevertheless, this system of equations is ill-conditioned, and causes numerical problems.

Some numerical methods are derived for obtaining the steady-state probabilities by using the sparsity and structure of the transition matrix (Chapter 4). An iterative multiplication scheme is introduced and analyzed. A recursive algorithm for solving the large system of transition equations by making use of the nested block tri-diagonal structure of the transition matrix is developed and discussed. This algorithm is general and applies to any number of stages, although computer memory requirements are considerable.

The steady-state probabilities are used to compute exactly the expected production rate (efficiency) of the system (Chapter 5). Alternate ways to calculate efficiency are introduced; it is proved that in the finite storage capacity case, the steady-state rates of flow through both machines in a two-machine line are equal. The proof is not complete for longer lines. The effects of system transients on line efficiency are discussed; it is shown that the extent to which steady-state values represent the actual performance of the system depends strongly on the system parameters. The relationship between storage capacity and line efficiency is investigated. It is demonstrated that storages contribute most to the system production rate when the line is balanced. Furthermore, the rate at which storage capacity improves line production rate depends on the magnitudes of the failure and repair probabilities of the system. An inductive proof of the assertion that infinite storage efficiency is equal to the efficiency of the worst stage is presented.

The relationship between storage capacity and forced-down times is investigated. It is shown that the infinite storage production rate is such that the system bottleneck is saturated. Furthermore, the line production rate increases almost linearly with the efficiency in isolation of the system bottleneck until it ceases to be limiting; at that time, the line production rate reaches an asymptote. For a balanced line, it is shown that increasing storage capacity can be as beneficial as improving the efficiency in isolation of individual machines, although providing buffer capacity may be cheaper than improving the reliability of machines.

The effect of storage size and machine efficiencies on expected in-process inventory is studied. The asymptotic behavior of in-process inventory is demonstrated, and the effects of the relative efficiencies of upstream and downstream segments of the transfer line on a particular storage are discussed.

The system efficiency is also computed by approximate methods (Chapter 6). These consist in lumping machines and storages together in equivalent single stages, thereby reducing the number of stages in the model; and in lumping workpieces together, thereby reducing the capacities of storages in the model. Both approaches have the effect of reducing the dimensions of the state space and saving computation considerably.

The results obtained are discussed with relation to a roll products paper finishing line (Chapter 7). Possible discrepencies between the model and actual systems are discussed in the light of this example. Approximations or changes in the model to account for such discrepencies are proposed and investigated.

The model is extended to continuous-time systems. In discussing a batch chemical process in which batches require random processing times, a two-stage line with exponentially distributed service times is analyzed and the results are applied to a plant consisting of a batch reactor and a still, separated by unreliable pumps and parallel holding tanks (Chapter 8).

A differential equations approach is reviewed for obtaining the steady-state probability distributions in the case where the material traveling through the system is or can be modeled as a fluid (Chapter 9). A numerical transformation introduced in chapter 6 is taken to its limits, and these results are shown to agree with those of the differential equations solution. The expressions obtained are applied to a chemical plant consisting of a plug-flow reactor and a distillation column, separated by unreliable pumps and a finite holding tank.

Directions for future research include more complex, non-linear system topologies and random processing times. A flexible manufacturing system in which many types of parts flow through the system in a non-deterministic order may be modeled as a system where stages have random processing times. The value of the results presented here will greatly increase if they can serve as a basis for more work in complex flexible manufacturing systems.

11. APPENDIX: PROGRAM LISTINGS, I/O INFORMATION AND SAMPLE OUTPUTS

A.1 Two-Machine Line, Analytical Solution

This program computes the steady-state probability distribution for a two-machine transfer line by using the closed-form expressions obtained in chapter 3.

The input is as follows:

First Card:	Columns 1-20:	Failure probabilities (p_i, i=1,2) (Format F10.5)
	Columns 21-40:	Repair probabilities (r_i, i=1,2) (Format F10.5)
	Columns 41-70:	Lower bound, upper bound, and step size for incrementing storage capacity.

```
FORTRAN IV G1  RELEASE 2.0              MAIN              DATE = 78191           17/29/43

                   C  ANALYTIC SOLUTION TO TWO-STATION PROBLEM
 0001                  IMPLICIT REAL*8 (A-H,O-Z)
 0002                  DIMENSION P2(2004)
                     1    ,I2(501)
                     1,PZ(501,2,2)   ,  PY(501,2,2)
                     1    ,PN(3)
 0003                  INTEGER AA,BB,A1,B1,A2,B2
 0004              199 WRITE (6,97)
 0005               97 FORMAT (1H1)
 0006                  READ(5,6,END=1000) P,Q,R,S,IN,JN,KN
 0007                6 FORMAT (4F10.5,3I10)
 0008                  UP=1./P
 0009                  UQ=1./Q
 0010                  UR=1./R
 0011                  US=1./S
 0012                  WRITE (6,4) P, UP, Q, UQ, R, UR, S, US
 0013                4 FORMAT (1H ,   20HPARAMETERS                              /
                     150H P = PROBABILITY HEAD 1 GOES DOWN WHILE IT IS UP = F9.5      ,
                     135H  --  AVERAGE UP-TIME OF HEAD 1    = F12.5/
                     150H Q = PROBABILITY HEAD 2 GOES DOWN WHILE IT IS UP = F9.5      ,
                     135H  --  AVERAGE UP-TIME OF HEAD 2    = F12.5/
                     150H R = PROBABILITY HEAD 1 GOES UP WHILE IT IS DOWN = F9.5      ,
                     135H  --  AVERAGE DOWN-TIME OF HEAD 1 = F12.5/
                     150H S = PROBABILITY HEAD 2 GOES UP WHILE IT IS DOWN = F9.5      ,
                     135H  --  AVERAGE DOWN-TIME OF HEAD 2 = F12.5/
                     1)
 0014                  A0 = 1./(1. + P/R + Q/S)
 0015                  AI=P/R
 0016                  IF(Q/S.GT.P/R) AI=Q/S
 0017                  AI = 1./(1. + AI)
 0018                  WRITE  (6,44)  A0,AI
 0019               44 FORMAT ( 28H EFFICIENCY WITH NO BUFFER = F9.5/
                     1 34H EFFICIENCY WITH INFINITE BUFFER = F9.5)
 0020                  DO 9999 NN=IN,JN,KN
 0021                  WRITE (6,98)
 0022                  WRITE (6,45) NN
 0023               45 FORMAT (
                     1  19H STORAGE CAPACITY = I5,   7H PIECES)
 0024                  NN1 = NN + 1
 0025                  NQ=NN-1
 0026                  WRITE (6,98)
 0027               98 FORMAT (////)
 0028                  K=4*NN+4
 0029                  AK=K
 0030                  RS=R+S-R*S
 0031                  PQ=P+Q-P*Q
 0032                  RSQ=RS-R*Q
 0033                  RSP=RS-S*P
 0034                  PQR=PQ-Q*R
 0035                  PQS=PQ-P*S
 0036                  Y=RSQ/PQS
 0037                  Z=RSP/PQR
 0038                  X=Z/Y
```

```
FORTRAN IV G1  RELEASE 2.0              MAIN               DATE = 78191          17/29/43

0039                IF (NN .GT. IN) GO TO 200
0040                WRITE (6,98)
0041                WRITE (6,19) X,Y,Z
0042             19 FORMAT (    5H X =     F8.5/
                   1            5H Y =     F8.5/
                   1            5H Z =     F8.5)
0043           200 CONTINUE
0044                DO 101 I1 = 3,NQ
0045                II1=I1-1
0046                DO 101 I2=1,2
0047                II2=I2-1
0048                DO 101 I3=1,2
0049                II3=I3-1
0050           101 PZ(I1,I2,I3)=X**II1*Y**II2*Z**II3
0051                PZ(1,1,1) = 0.
0052                PZ(1,2,1) = 0.
0053                PZ(1,2,2) = 0.
0054                PZ(2,2,1) = 0.
0055                PZ(2,1,1) = X
0056                PZ(2,1,2) = X*Z
0057                PZ(2,2,2) = X* (S+(1.-Q)*Z) /Q
0058                PZ(1,1,2) = PZ(2,2,2)*PQR/R
0059                PZ(NN+1,1,1) = 0.
0060                PZ(NN+1,1,2) = 0.
0061                PZ(NN+1,2,2) = 0.
0062                PZ(NN  ,1,2) = 0.
0063                XN = X**NQ
0064                PZ(NN  ,1,1) = XN
0065                PZ(NN  ,2,1) = XN*Y
0066                PZ(NN  ,2,2) = XN* (R+(1.-P)*Y) /P
0067                PZ(NN+1,2,1) = PZ(NN,2,2)*PQS/S
0068                C=0
0069                DO 102 I1=1,NN1
0070                IQ(I1) = I1-1
0071                DO 102 I2=1,2
0072                DO 102 I3=1,2
0073           102 C=C+PZ(I1,I2,I3)
0074                DO 103 I1=1,NN1
0075                DO 103 I2=1,2
0076                DO 103 I3=1,2
0077           103 PZ(I1,I2,I3)=PZ(I1,I2,I3)/C
0078                WRITE(6,98)
0079                WRITE (6,31) ((((PZ(I1,I2,I3),I3=1,2),I2=1,2), IQ(I1),
                   1    I1=1,NN1))
0080            31 FORMAT (30H PROBABILITY DISTRIBUTION                /
                   1  / 10X, 3H0 0, 12X, 3H0 1,  12X,  3H1 0,  12X,  3H1 1,
                   1    15X, 1HN//
                   1    (6X,4E13.6,I15/))
             C   Z1 = PROB HEAD 1 OPERATING
             C   Z2 = PROB HEAD 2 OPERATING
             C   Z3 = EXPECTED NUMBER PIECES IN QUEUE
             C   Z4 = PROB QUEUE EMPTY
             C   Z5 = PROB QUEUE FULL
```

```
FORTRAN IV G1  RELEASE 2.0           MAIN           DATE = 78191        17/29/43

                   C  Z6 = PROB 0   0
                   C  Z7 = PROB 0   1
                   C  Z8 = PROB 1   0
                   C  Z9 = PROB 1   1
 0081                    Z1 = 0.
 0082                    Z2 = 0.
 0083                    Z3 = 0.
 0084                    Z4 = 0.
 0085                    Z5 = 0.
 0086                    Z6 = 0.
 0087                    Z7 = 0.
 0088                    Z8 = 0.
 0089                    Z9 = 0.
 0090                    DO 7 J=1,K
 0091                    BB=(J-1)/2
 0092                    B1=J-1-2*BB
 0093                    N1=BB/2
 0094                    AN=N1
 0095                    A1=BB-2*N1
 0096                    NP = N1 + 1
 0097                    P2(J)  = PZ(NP,A1+1,B1+1)
 0098                    IF (A1 .EQ. 0 .AND. N1 .LT. NN)  Z1 = Z1 + PZ(NP,A1+1,B1+1)*R
 0099                    IF (A1 .EQ. 1 .AND. N1 .LT. NN)  Z1 = Z1 + PZ(NP,A1+1,B1+1)*(1.-P)
 0100                    IF (B1 .EQ. 0 .AND. N1 .GT. 0 )  Z2 = Z2 + PZ(NP,A1+1,B1+1)*S
 0101                    IF (B1 .EQ. 1 .AND. N1 .GT. 0 )  Z2 = Z2 + PZ(NP,A1+1,B1+1)*(1.-Q)
 0102                    Z3 = Z3+AN*P2(J)
 0103                    IF (N1 .EQ. 0 ) Z4 = Z4 +P2(J)
 0104                    IF (N1 .EQ. NN) Z5 = Z5 +P2(J)
 0105                    IF (A1 .EQ. 0 .AND. B1 .EQ. 0) Z6 = Z6 + P2(J)
 0106                    IF (A1 .EQ. 0 .AND. B1 .EQ. 1) Z7 = Z7 + P2(J)
 0107                    IF (A1 .EQ. 1 .AND. B1 .EQ. 0) Z8 = Z8 + P2(J)
 0108                    IF (A1 .EQ. 1 .AND. B1 .EQ. 1) Z9 = Z9 + P2(J)
 0109                  7 CONTINUE
 0110                    WRITE (6,98)
 0111                    WRITE (6,20)
 0112                 20 FORMAT (  20H  TOTALS                            )
 0113                    WRITE (6,15) Z6,Z7,Z8,Z9
 0114                 15 FORMAT (4F15.5)
 0115                    WRITE (6,98)
 0116                    WRITE (6,98)
 0117                    Z10=Z3/AN*100.
 0118                    WRITE(6,8) Z1,Z2,Z3,Z10,Z4,Z5
 0119                  8 FORMAT (
                        1 29 H EFFICIENCY E1                =     F9.5/
                        1 29 H EFFICIENCY E2                =     F9.5/
                        1 29 H AVERAGE STORAGE FILL         =     F9.5/
                        1 29 H AVERAGE STORAGE FILL (%)     =          F9.5/
                        1 29 H PROBABILITY STORAGE EMPTY =     F9.5/
                        1 29 H PROBABILITY STORAGE FULL  =     F9.5/
                        1)
 0120             9999 CONTINUE
 0121                    GO TO 199
 0122             1000 CONTINUE
 0123                    END
```

A.2 The Boundary Transition Equations Generator

This program is written in the IBM FORMAC Symbolic Mathematics Interpreter language (See Tobey et.al.[1969], Trufyn[n.d.]), a superset of PL/I.

A sample output which includes only the lower boundary corner and edge state transition equations for a three-machine system with storage capacities $N_1=N_2=10$ appears on pages 264- 266.

The input to this program is not formatted. The following data is required, in this order:

K, the number of machines in the line;

i, p_i, r_i, the machine indices and failure and repair probabilities for i=1,..,K.

(Note that i must be an integer, but the failure and repair probabilities are declared and hence treated by the program as characters. A sample input may thus be: 1, "P1", "R1".)

i, N_i the storage indices and maximum capacities for i=1,..,K-1.

(Note that i and N_i must be integers. Because N_i are used as upper limits on several loops in the program, these can not be given as characters.)

```
***EQUATION NO.              1              ***
ZERO =  - P.( 0, 0, 0, 1, 1 ) + (  - R1 + 1 ) P.( 0, 1, 0, 1, 0 ) R3 + (  - R1 + 1 ) P.( 0, 0, 0, 1, 1 ) + (  - P3 + 1
) (  - R1 + 1 ) P.( 0, 1, 0, 1, 1 )
***EQUATION NO.              2              ***
ZERO =  - P.( 0, 1, 0, 1, 0 ) + (  - R1 + 1 ) P.( 0, 1, 0, 1, 1 ) P3 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 0, 1, 0, 1, 0 )
***EQUATION NO.              3              ***
ZERO =  - P.( 0, 1, 0, 1, 1 ) + (  - R1 + 1 ) P.( 1, 1, 0, 0, 0 ) R2 R3 + (  - P2 + 1 ) P.( 1, 1, 1, 1, 0 ) P1 R3 + (
 - R1 + 1 ) P.( 0, 2, 0, 1, 0 ) R3 + (  - R1 + 1 ) P.( 1, 0, 0, 0, 1 ) R2 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 0
, 1 ) R2 + (  - P2 + 1 ) P.( 1, 0, 1, 1, 1 ) P1 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 1, 1, 1, 1, 1 ) P1 + (  - P3 + 1 ) (
 - R1 + 1 ) P.( 0, 2, 0, 1, 1 ) + (  - P2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 1, 1 )
***EQUATION NO.              4              ***
ZERO =  - P.( 0, 2, 0, 1, 0 ) + (  - R1 + 1 ) P.( 1, 1, 0, 0, 1 ) R2 P3 + (  - P2 + 1 ) P.( 1, 1, 1, 1, 1 ) P1 P3 + (
 - R1 + 1 ) P.( 0, 2, 0, 1, 1 ) P3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 1, 1 ) P3 + (  - R3 + 1 ) (  - R1 + 1 ) P
.( 1, 1, 0, 0, 0 ) R2 + (  - P2 + 1 ) (  - R3 + 1 ) P.( 1, 1, 1, 1, 0 ) P1 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 0, 2, 0, 1
, 0 )
***EQUATION NO.              5              ***
ZERO =  - P.( 0, 2, 0, 1, 1 ) + (  - R1 + 1 ) P.( 1, 2, 0, 0, 0 ) R2 R3 + (  - P2 + 1 ) P.( 1, 2, 1, 1, 0 ) P1 R3 + (
 - R1 + 1 ) P.( 0, 3, 0, 1, 0 ) R3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 1, 2, 0, 1, 0 ) R3 + (  - P3 + 1 ) (  - R1 + 1 ) P
.( 1, 2, 0, 0, 1 ) R2 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 1, 2, 1, 1, 1 ) P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 0, 3, 0, 1
, 1 ) + (  - P2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 2, 0, 1, 1 )
***EQUATION NO.              6              ***
ZERO =  - P.( 1, 0, 0, 0, 1 ) + P.( 1, 1, 1, 1, 0 ) P2 P1 R3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 0, 0 ) R3 + P.
( 1, 0, 1, 1, 1 ) P2 P1 + (  - P3 + 1 ) P.( 1, 1, 1, 1, 1 ) P2 P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 1, 1 ) P2
 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 1, 0, 0, 0, 1 ) + (  - R2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 1, 0, 0, 1 )
***EQUATION NO.              7              ***
ZERO =  - P.( 1, 0, 1, 1, 1 ) + P.( 0, 1, 0, 1, 0 ) R3 R1 + P.( 0, 0, 0, 1, 1 ) R1 + (  - P3 + 1 ) P.( 0, 1, 0, 1, 1 )
R1
***EQUATION NO.              8              ***
ZERO =  - P.( 1, 1, 0, 0, 0 ) + P.( 1, 1, 1, 1, 1 ) P2 P1 P3 + (  - R1 + 1 ) P.( 1, 1, 0, 1, 1 ) P2 P3 + (  - R2 + 1 )
(  - R1 + 1 ) P.( 1, 1, 0, 0, 1 ) P3 + (  - R3 + 1 ) P.( 1, 1, 1, 1, 0 ) P2 P1 + (  - R2 + 1 ) (  - R3 + 1 ) (  - R1 +
1 ) P.( 1, 1, 0, 0, 0 )
***EQUATION NO.              9              ***
ZERO =  - P.( 1, 1, 0, 0, 1 ) + P.( 1, 2, 1, 1, 0 ) P2 P1 R3 + (  - R1 + 1 ) P.( 1, 2, 0, 1, 0 ) P2 R3 + (  - R2 + 1 )
(  - R1 + 1 ) P.( 1, 2, 0, 0, 0 ) R3 + (  - P3 + 1 ) P.( 1, 2, 1, 1, 1 ) P2 P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 2,
0, 1, 1 ) P2 + (  - R2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 2, 0, 0, 1 )
***EQUATION NO.              10             ***
ZERO =  - P.( 1, 1, 0, 1, 1 ) + P.( 2, 1, 1, 0, 0 ) P1 R2 R3 + (  - R1 + 1 ) P.( 2, 1, 0, 0, 0 ) R2 R3 + P.( 2, 0, 1, 0
, 1 ) P1 R2 + (  - P3 + 1 ) P.( 2, 1, 1, 0, 1 ) P1 R2 + (  - R1 + 1 ) P.( 2, 0, 0, 0, 1 ) R2 + (  - P3 + 1 ) (  - R1 +
1 ) P.( 2, 1, 0, 0, 1 ) R2 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 2, 1, 1, 1, 1 ) P1 + (  - P2 + 1 ) (  - P3 + 1 ) (  - R1
 + 1 ) P.( 2, 1, 0, 1, 1 )
***EQUATION NO.              11             ***
ZERO =  - P.( 1, 1, 1, 1, 0 ) + P.( 0, 1, 0, 1, 1 ) P3 R1 + (  - R3 + 1 ) P.( 0, 1, 0, 1, 0 ) R1
***EQUATION NO.              12             ***
ZERO =  - P.( 1, 1, 1, 1, 1 ) + P.( 1, 1, 0, 0, 0 ) R2 R3 R1 + P.( 0, 2, 0, 1, 0 ) R3 R1 + P.( 1, 0, 0, 0, 1 ) R2 R1 +
(  - P3 + 1 ) P.( 1, 1, 0, 0, 1 ) R2 R1 + (  - P3 + 1 ) P.( 0, 2, 0, 1, 1 ) R1 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 1, 1,
0, 1, 1 ) R1 + (  - P2 + 1 ) (  - P1 + 1 ) P.( 1, 1, 1, 1, 0 ) R3 + (  - P2 + 1 ) (  - P1 + 1 ) P.( 1, 0, 1, 1, 1 ) +
(  - P2 + 1 ) (  - P1 + 1 ) (  - P3 + 1 ) P.( 1, 1, 1, 1, 1 )
***EQUATION NO.              13             ***
ZERO =  - P.( 1, 2, 0, 0, 0 ) + P.( 1, 2, 1, 1, 1 ) P2 P1 P3 + (  - R1 + 1 ) P.( 1, 2, 0, 1, 1 ) P2 P3 + (  - R2 + 1 )
(  - R1 + 1 ) P.( 1, 2, 0, 0, 1 ) P3 + (  - R3 + 1 ) P.( 1, 2, 1, 1, 0 ) P2 P1 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 1, 2,
0, 1, 0 ) P2 + (  - R2 + 1 ) (  - R3 + 1 ) (  - R1 + 1 ) P.( 1, 2, 0, 0, 0 )
```

```
***EQUATION NO.          14          ***
ZERO =  - P.( 1, 2, 0, 0, 1 ) + P.( 1, 3, 1, 1, 0 ) P2 P1 R3 + (  - R1 + 1 ) P.( 1, 3, 0, 1, 0 ) P2 R3 + (  - R2 + 1 )
(  - R1 + 1 ) P.( 1, 3, 0, 0, 0 ) R3 + (  - P3 + 1 ) P.( 1, 3, 1, 1, 1 ) P2 P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 3,
0, 1, 1 ) P2 + (  - R2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 1, 3, 0, 0, 1 )
***EQUATION NO.          15          ***
ZERO =  - P.( 1, 2, 0, 1, 0 ) + P.( 2, 1, 1, 0, 1 ) P1 R2 P3 + (  - R1 + 1 ) P.( 2, 1, 0, 0, 1 ) R2 P3 + (  - P2 + 1 )
P.( 2, 1, 1, 1, 1 ) P1 P3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 1, 1 ) P3 + (  - R3 + 1 ) P.( 2, 1, 1, 0, 0 ) P1
R2 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 0, 0 ) R2
***EQUATION NO.          16          ***
ZERO =  - P.( 1, 2, 0, 1, 1 ) + P.( 2, 2, 1, 0, 0 ) P1 R2 R3 + (  - R1 + 1 ) P.( 2, 2, 0, 0, 0 ) R2 R3 + (  - P2 + 1 )
P.( 2, 2, 1, 1, 0 ) P1 R3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 1, 0 ) R3 + (  - P3 + 1 ) P.( 2, 2, 1, 0, 1 ) P1
R2 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 0, 1 ) R2 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 2, 2, 1, 1, 1 ) P1 + (  - P2
 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 1, 1 )
***EQUATION NO.          17          ***
ZERO =  - P.( 1, 2, 1, 1, 0 ) + P.( 1, 1, 0, 0, 1 ) R2 P3 R1 + P.( 0, 2, 0, 1, 1 ) P3 R1 + (  - P2 + 1 ) P.( 1, 1, 0, 1
, 1 ) P3 R1 + (  - R3 + 1 ) P.( 1, 1, 0, 0, 0 ) R2 R1 + (  - R3 + 1 ) P.( 0, 2, 0, 1, 0 ) R1 + (  - P2 + 1 ) (  - P1 +
1 ) P.( 1, 1, 1, 1, 1 ) P3 + (  - P2 + 1 ) (  - P1 + 1 ) (  - R3 + 1 ) P.( 1, 1, 1, 1, 0 )
***EQUATION NO.          18          ***
ZERO =  - P.( 1, 2, 1, 1, 1 ) + P.( 1, 2, 0, 0, 0 ) R2 R3 R1 + P.( 0, 3, 0, 1, 0 ) R3 R1 + (  - P2 + 1 ) P.( 1, 2, 0, 1
, 0 ) R3 R1 + (  - P3 + 1 ) P.( 1, 2, 0, 0, 1 ) R2 R1 + (  - P3 + 1 ) P.( 0, 3, 0, 1, 1 ) R1 + (  - P2 + 1 ) (  - P3 +
1 ) P.( 1, 2, 0, 1, 1 ) R1 + (  - P2 + 1 ) (  - P1 + 1 ) P.( 1, 2, 1, 1, 0 ) R3 + (  - P2 + 1 ) (  - P1 + 1 ) (  - P3
 + 1 ) P.( 1, 2, 1, 1, 1 )
***EQUATION NO.          19          ***
ZERO =  - P.( 2, 0, 0, 0, 1 ) + (  - R2 + 1 ) P.( 2, 1, 1, 0, 0 ) P1 R3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 0, 0
 ) R3 + (  - P3 + 1 ) P.( 2, 1, 1, 1, 1 ) P2 P1 + (  - R2 + 1 ) P.( 2, 0, 1, 0, 1 ) P1 + (  - R2 + 1 ) (  - P3 + 1 ) P.
( 2, 1, 1, 0, 1 ) P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 1, 1 ) P2 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 0, 0, 0
, 1 ) + (  - R2 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 0, 1 )
***EQUATION NO.          20          ***
ZERO =  - P.( 2, 0, 1, 0, 1 ) + (  - R2 + 1 ) P.( 1, 1, 0, 0, 0 ) R3 R1 + (  - P3 + 1 ) P.( 1, 1, 0, 1, 1 ) P2 R1 + (
 - R2 + 1 ) P.( 1, 0, 0, 0, 1 ) R1 + (  - R2 + 1 ) (  - P3 + 1 ) P.( 1, 1, 0, 0, 1 ) R1 + (  - P1 + 1 ) P.( 1, 1, 1, 1
, 0 ) P2 R3 + (  - P1 + 1 ) P.( 1, 0, 1, 1, 1 ) P2 + (  - P1 + 1 ) (  - P3 + 1 ) P.( 1, 1, 1, 1, 1 ) P2
***EQUATION NO.          21          ***
ZERO =  - P.( 2, 1, 0, 0, 0 ) + P.( 2, 1, 1, 1, 1 ) P2 P1 P3 + (  - R2 + 1 ) P.( 2, 1, 1, 0, 1 ) P1 P3 + (  - R1 + 1 )
P.( 2, 1, 0, 1, 1 ) P2 P3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 0, 1 ) P3 + (  - R2 + 1 ) (  - R3 + 1 ) P.( 2, 1,
1, 0, 0 ) P1 + (  - R2 + 1 ) (  - R3 + 1 ) (  - R1 + 1 ) P.( 2, 1, 0, 0, 0 )
***EQUATION NO.          22          ***
ZERO =  - P.( 2, 1, 0, 0, 1 ) + P.( 2, 2, 1, 1, 0 ) P2 P1 R3 + (  - R2 + 1 ) P.( 2, 2, 1, 0, 0 ) P1 R3 + (  - R1 + 1 )
P.( 2, 2, 0, 1, 0 ) P2 R3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 0, 0 ) R3 + (  - P3 + 1 ) P.( 2, 2, 1, 1, 1 ) P2
P1 + (  - R2 + 1 ) (  - P3 + 1 ) P.( 2, 2, 1, 0, 1 ) P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 1, 1 ) P2 + (  - R2
 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 0, 1 )
***EQUATION NO.          23          ***
ZERO =  - P.( 2, 1, 0, 1, 1 ) + P.( 3, 1, 1, 0, 0 ) P1 R2 R3 + (  - R1 + 1 ) P.( 3, 1, 0, 0, 0 ) R2 R3 + P.( 3, 0, 1, 0
, 1 ) P1 R2 + (  - P3 + 1 ) P.( 3, 1, 1, 0, 1 ) P1 R2 + (  - R1 + 1 ) P.( 3, 0, 0, 0, 1 ) R2 + (  - P3 + 1 ) (  - R1 +
1 ) P.( 3, 1, 0, 0, 1 ) R2 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 3, 1, 1, 1, 1 ) P1 + (  - P2 + 1 ) (  - P3 + 1 ) (  - R1
 + 1 ) P.( 3, 1, 0, 1, 1 )
***EQUATION NO.          24          ***
ZERO =  - P.( 2, 1, 1, 0, 0 ) + P.( 1, 1, 0, 1, 1 ) P2 P3 R1 + (  - R2 + 1 ) P.( 1, 1, 0, 0, 1 ) P3 R1 + (  - R2 + 1 )
(  - R3 + 1 ) P.( 1, 1, 0, 0, 0 ) R1 + (  - P1 + 1 ) P.( 1, 1, 1, 1, 1 ) P2 P3 + (  - P1 + 1 ) (  - R3 + 1 ) P.( 1, 1,
1, 1, 0 ) P2
```

```
***EQUATION NO.            25          ***
ZERO =  - P.( 2, 1, 1, 0, 1 ) + P.( 1, 2, 0, 1, 0 ) P2 R3 R1 + (  - R2 + 1 ) P.( 1, 2, 0, 0, 0 ) R3 R1 + (  - P3 + 1 )
P.( 1, 2, 0, 1, 1 ) P2 R1 + (  - R2 + 1 ) (  - P3 + 1 ) P.( 1, 2, 0, 0, 1 ) R1 + (  - P1 + 1 ) P.( 1, 2, 1, 1, 0 ) P2
R3 + (  - P1 + 1 ) (  - P3 + 1 ) P.( 1, 2, 1, 1, 1 ) P2
***EQUATION NO.            26          ***
ZERO =  - P.( 2, 1, 1, 1, 1 ) + P.( 2, 1, 0, 0, 0 ) R2 R3 R1 + P.( 2, 0, 0, 0, 1 ) R2 R1 + (  - P3 + 1 ) P.( 2, 1, 0, 0
, 1 ) R2 R1 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 2, 1, 0, 1, 1 ) R1 + (  - P1 + 1 ) P.( 2, 1, 1, 0, 0 ) R2 R3 + (  - P1 +
1 ) P.( 2, 0, 1, 0, 1 ) R2 + (  - P1 + 1 ) (  - P3 + 1 ) P.( 2, 1, 1, 0, 1 ) R2 + (  - P2 + 1 ) (  - P1 + 1 ) (  - P3
 + 1 ) P.( 2, 1, 1, 1, 1 )
***EQUATION NO.            27          ***
ZERO =  - P.( 2, 2, 0, 0, 0 ) + P.( 2, 2, 1, 1, 1 ) P2 P1 P3 + (  - R2 + 1 ) P.( 2, 2, 1, 0, 1 ) P1 P3 + (  - R1 + 1 )
P.( 2, 2, 0, 1, 1 ) P2 P3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 0, 1 ) P3 + (  - R3 + 1 ) P.( 2, 2, 1, 1, 0 ) P2
P1 + (  - R2 + 1 ) (  - R3 + 1 ) P.( 2, 2, 1, 0, 0 ) P1 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 1, 0 ) P2 + (  - R2
 + 1 ) (  - R3 + 1 ) (  - R1 + 1 ) P.( 2, 2, 0, 0, 0 )
***EQUATION NO.            28          ***
ZERO =  - P.( 2, 2, 0, 0, 1 ) + P.( 2, 3, 1, 1, 0 ) P2 P1 R3 + (  - R2 + 1 ) P.( 2, 3, 1, 0, 0 ) P1 R3 + (  - R1 + 1 )
P.( 2, 3, 0, 1, 0 ) P2 R3 + (  - R2 + 1 ) (  - R1 + 1 ) P.( 2, 3, 0, 0, 0 ) R3 + (  - P3 + 1 ) P.( 2, 3, 1, 1, 1 ) P2
P1 + (  - R2 + 1 ) (  - P3 + 1 ) P.( 2, 3, 1, 0, 1 ) P1 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 3, 0, 1, 1 ) P2 + (  - R2
 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 2, 3, 0, 0, 1 )
***EQUATION NO.            29          ***
ZERO =  - P.( 2, 2, 0, 1, 0 ) + P.( 3, 1, 1, 0, 1 ) P1 R2 P3 + (  - R1 + 1 ) P.( 3, 1, 0, 0, 1 ) R2 P3 + (  - P2 + 1 )
P.( 3, 1, 1, 1, 1 ) P1 P3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 3, 1, 0, 1, 1 ) P3 + (  - R3 + 1 ) P.( 3, 1, 1, 0, 0 ) P1
R2 + (  - R3 + 1 ) (  - R1 + 1 ) P.( 3, 1, 0, 0, 0 ) R2
***EQUATION NO.            30          ***
ZERO =  - P.( 2, 2, 0, 1, 1 ) + P.( 3, 2, 1, 0, 0 ) P1 R2 R3 + (  - R1 + 1 ) P.( 3, 2, 0, 0, 0 ) R2 R3 + (  - P2 + 1 )
P.( 3, 2, 1, 1, 0 ) P1 R3 + (  - P2 + 1 ) (  - R1 + 1 ) P.( 3, 2, 0, 1, 0 ) R3 + (  - P3 + 1 ) P.( 3, 2, 1, 0, 1 ) P1
R2 + (  - P3 + 1 ) (  - R1 + 1 ) P.( 3, 2, 0, 0, 1 ) R2 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 3, 2, 1, 1, 1 ) P1 + (  - P2
 + 1 ) (  - P3 + 1 ) (  - R1 + 1 ) P.( 3, 2, 0, 1, 1 )
***EQUATION NO.            31          ***
ZERO =  - P.( 2, 2, 1, 0, 0 ) + P.( 1, 2, 0, 1, 1 ) P2 P3 R1 + (  - R2 + 1 ) P.( 1, 2, 0, 0, 1 ) P3 R1 + (  - R3 + 1 )
P.( 1, 2, 0, 1, 0 ) P2 R1 + (  - R2 + 1 ) (  - R3 + 1 ) P.( 1, 2, 0, 0, 0 ) R1 + (  - P1 + 1 ) P.( 1, 2, 1, 1, 1 ) P2
P3 + (  - P1 + 1 ) (  - R3 + 1 ) P.( 1, 2, 1, 1, 0 ) P2
***EQUATION NO.            32          ***
ZERO =  - P.( 2, 2, 1, 0, 1 ) + P.( 1, 3, 0, 1, 0 ) P2 R3 R1 + (  - R2 + 1 ) P.( 1, 3, 0, 0, 0 ) R3 R1 + (  - P3 + 1 )
P.( 1, 3, 0, 1, 1 ) P2 R1 + (  - R2 + 1 ) (  - P3 + 1 ) P.( 1, 3, 0, 0, 1 ) R1 + (  - P1 + 1 ) P.( 1, 3, 1, 1, 0 ) P2
R3 + (  - P1 + 1 ) (  - P3 + 1 ) P.( 1, 3, 1, 1, 1 ) P2
***EQUATION NO.            33          ***
ZERO =  - P.( 2, 2, 1, 1, 0 ) + P.( 2, 1, 0, 0, 1 ) R2 P3 R1 + (  - P2 + 1 ) P.( 2, 1, 0, 1, 1 ) P3 R1 + (  - R3 + 1 )
P.( 2, 1, 0, 0, 0 ) R2 R1 + (  - P1 + 1 ) P.( 2, 1, 1, 0, 1 ) R2 P3 + (  - P2 + 1 ) (  - P1 + 1 ) P.( 2, 1, 1, 1, 1 )
P3 + (  - P1 + 1 ) (  - R3 + 1 ) P.( 2, 1, 1, 0, 0 ) R2
***EQUATION NO.            34          ***
ZERO =  - P.( 2, 2, 1, 1, 1 ) + P.( 2, 2, 0, 0, 0 ) R2 R3 R1 + (  - P2 + 1 ) P.( 2, 2, 0, 1, 0 ) R3 R1 + (  - P3 + 1 )
P.( 2, 2, 0, 0, 1 ) R2 R1 + (  - P2 + 1 ) (  - P3 + 1 ) P.( 2, 2, 0, 1, 1 ) R1 + (  - P1 + 1 ) P.( 2, 2, 1, 0, 0 ) R2
R3 + (  - P2 + 1 ) (  - P1 + 1 ) P.( 2, 2, 1, 1, 0 ) R3 + (  - P1 + 1 ) (  - P3 + 1 ) P.( 2, 2, 1, 0, 1 ) R2 + (  - P2
 + 1 ) (  - P1 + 1 ) (  - P3 + 1 ) P.( 2, 2, 1, 1, 1 )
```

INPUT TO FORMAC PREPROCESSOR

```
BOUNDRY: PROCEDURE OPTIONS(MAIN);
      FORMAC_OPTIONS;
      GET LIST(N);
TWO: BEGIN; DECLARE NSTOR(N+1),MAXST(N+1),INSTO(N+1),MACHI(N),INMAC(N);
      DECLARE (REPR(N),FAIL(N)) CHARACTER (N/10+2) VARYING;
      DECLARE (FINAL,PROB) CHARACTER(10*N) VARYING;
      DECLARE CHR CHARACTER (5*N) VARYING;
LOOP01: DO J=1 TO N;
      GET LIST (I,FAIL(I),REPR(I));
      END LOOP01;
LOOP02: DO J=1 TO N-1;
      GET LIST (I,MAXST(I+1));
      END LOOP02;
      MAXST(1)=0; MAXST(N+1)=0;
      NSTA1=2**N; NSTA2=PROD(MAXST+1);
                  MAXST(N+1)=1;
      NSTOR(1)=1; NSTOR(N+1)=0;
      INSTO(1)=1; INSTO(N+1)=0;
      ITER=1;
LOOPA: DO IND1=1 TO NSTA2; LOOP1: DO IND2=1 TO NSTA1;
      NDUM=IND1-1;
LOOP2: DO IND3=2 TO N-1;
      NSTOR(IND3)=NDUM/(MAXST(IND3+1)+1);
      NDUM=NDUM-((MAXST(IND3+1)+1)*NSTOR(IND3));
      END LOOP2;
      NSTOR(N)=NDUM;
      NDUM=IND2-1;
LOOP3: DO IND3=1 TO N;
      MACHI(IND3)=NDUM/(2**(N-IND3));
      NDUM=NDUM-(MACHI(IND3)*(2**(N-IND3)));
      END LOOP3;
      INDEX=0;
LOOP35: DO IND3=2 TO N;
      IF NSTOR(IND3) > 3 & (MAXST(IND3)-3) > NSTOR(IND3)
      THEN GO TO ENDLOOP1;
      IF 2 >=NSTOR(IND3) THEN INDEX=1;
      IF NSTOR(IND3) >=(MAXST(IND3)-2) THEN INDEX=1;
      END LOOP35;
      IF INDEX = 0 THEN GO TO ENDLOOP1;
LOOP11: DO IND3=2 TO N;
      IF NSTOR(IND3) ¬= 0 THEN GO TO S102;
      IF MACHI(IND3) = 0 THEN GO TO ENDLOOP1;
S100: IF NSTOR(IND3-1) = 0 THEN GO TO ENDLOOP11;
      IF IND3=2 THEN GO TO S101;
      IF NSTOR(IND3-1)=1 & MACHI(IND3-2)=1 THEN GO TO ENDLOOP11;
S101: IF MACHI(IND3-1)=1 THEN GO TO ENDLOOP1;
      ELSE GO TO ENDLOOP11;
S102: IF NSTOR(IND3) ¬= 1 THEN GO TO S1025;
      IF MACHI(IND3) = 1 THEN GO TO ENDLOOP11;
      ELSE GO TO S100;
S1025: IF NSTOR(IND3) ¬= MAXST(IND3) THEN GO TO S105;
      IF MACHI(IND3-1) = 0 THEN GO TO ENDLOOP1;
S103: IF NSTOR(IND3+1)=MAXST(IND3+1) THEN GO TO ENDLOOP11;
      IF IND3=N THEN GO TO S104;
      IF NSTOR(IND3+1)=(MAXST(IND3+1)-1) & MACHI(IND3+1)=1 THEN GO TO
      ENDLOOP11;
S104: IF MACHI(IND3) = 1 THEN GO TO ENDLOOP1;
      ELSE GO TO ENDLOOP11;
S105: IF NSTOR(IND3) ¬= (MAXST(IND3)-1) THEN GO TO ENDLOOP11;
```

```
     IF MACHI(IND3-1) = 0 THEN GO TO S103;
ENDLOOP11: END LOOP11;
     FINAL='P.(';
LOOP91: DO IND3=2 TO N;
      INT=0;
S150:          I=NSTOR(IND3)/(10**(INT+1)); IF I=0 THEN GO TO S151;
     INT=INT+1; GO TO S150;
S151: CHR=NSTOR(IND3)||',';
     CHR=SUBSTR(CHR,9-INT,2+INT);
     FINAL=FINAL||CHR;
     END LOOP91;
LOOP101: DO IND3=1 TO N-1;
     CHR=MACHI(IND3)||',';
     CHR=SUBSTR(CHR,9,2);
     FINAL=FINAL||CHR;
     END LOOP101;
     CHR=MACHI(N)||')';
     CHR=SUBSTR(CHR,9,2);
     FINAL=FINAL||CHR;
     LET(SUM=0);
LOOPB: DO IDD1=1 TO NSTA2; LOOP4: DO IDD2=1 TO NSTA1;
     NDUM=IDD1-1;
LOOP5: DO IND3=2 TO N-1;
     INSTO(IND3)=NDUM/(MAXST(IND3+1)+1);
     NDUM=NDUM-((MAXST(IND3+1)+1)*INSTO(IND3));
     END LOOP5;
     INSTO(N)=NDUM;
     NDUM=IDD2-1;
LOOP6: DO IND3=1 TO N;
     INMAC(IND3)=NDUM/(2**(N-IND3));
     NDUM=NDUM-(INMAC(IND3)*(2**(N-IND3)));
     END LOOP6;
LOOP61: DO IND3=2 TO N;
     IF ABS(NSTOR(IND3)-INSTO(IND3)) > 1 THEN GO TO ENDLOOP4;
     END LOOP61;
LOOP12: DO IND3=2 TO N;
     IF INSTO(IND3) ¬= 0 THEN GO TO S202;
     IF INMAC(IND3) = 0 THEN GO TO ENDLOOP4;
S200: IF INSTO(IND3-1) = 0 THEN GO TO ENDLOOP12;
     IF IND3 = 2 THEN GO TO S201;
     IF INSTO(IND3-1)=1 & INMAC(IND3-2)=1 THEN GO TO ENDLOOP12;
S201: IF INMAC(IND3-1) = 1 THEN GO TO ENDLOOP4;
     ELSE GO TO ENDLOOP12;
S202: IF INSTO(IND3) ¬= 1 THEN GO TO S2025;
     IF INMAC(IND3) = 1 THEN GO TO ENDLOOP12;
     ELSE GO TO S200;
S2025: IF INSTO(IND3) ¬= MAXST(IND3) THEN GO TO S205;
     IF INMAC(IND3-1) = 0 THEN GO TO ENDLOOP4;
S203: IF INSTO(IND3+1)=MAXST(IND3+1) THEN GO TO ENDLOOP12;
     IF IND3 = N THEN GO TO S204;
     IF INSTO(IND3+1)=(MAXST(IND3+1)-1) & INMAC(IND3+1)=1
     THEN GO TO ENDLOOP12;
S204: IF INMAC(IND3) = 1 THEN GO TO ENDLOOP4;
     ELSE GO TO ENDLOOP12;
S205: IF INSTO(IND3) ¬= (MAXST(IND3)-1) THEN GO TO ENDLOOP12;
     IF INMAC(IND3-1) = 0 THEN GO TO S203;
ENDLOOP12: END LOOP12;
     LET(TRN=1);
LOOP7: DO IND3=1 TO N;
     IF INMAC(IND3) ¬= 0 THEN GO TO S608;
```

```
     LET(TRN=("MACHI(IND3)"*"REPR(IND3)"+(1-"MACHI(IND3)")*(1-"REPR(
     IND3)"))*TRN);
     GO TO ENDLOOP7;
S608: IF INSTO(IND3) ¬= 0 & INSTO(IND3+1) ¬= MAXST(IND3+1) THEN GO TO
     S609;
     IF MACHI(IND3)=1 THEN GO TO ENDLOOP7; ELSE GO TO ENDLOOP4;
S609: LET(TRN=((1-"MACHI(IND3)")*"FAIL(IND3)"+"MACHI(IND3)"*(1-"FAIL(
     IND3)"))*TRN);
ENDLOOP7: END LOOP7;
LOOP8: DO IND3=2 TO N;
     IF INSTO(IND3) ¬= 0 THEN GO TO S613;
     IF INSTO(IND3-1) ¬= 0 THEN GO TO S612;
     IF NSTOR(IND3) = 0 THEN GO TO ENDLOOP8; ELSE GO TO ENDLOOP4;
S612: IF NSTOR(IND3) = MACHI(IND3-1) THEN GO TO ENDLOOP8; ELSE GO TO
     ENDLOOP4;
S613: IF INSTO(IND3) ¬= MAXST(IND3) THEN GO TO S615;
     IF INSTO(IND3+1) ¬= MAXST(IND3+1) THEN GO TO S614;
     IF NSTOR(IND3) = MAXST(IND3) THEN GO TO ENDLOOP8; ELSE GO TO
     ENDLOOP4;
S614: IF NSTOR(IND3) = (MAXST(IND3)-MACHI(IND3)) THEN GO TO ENDLOOP8;
     ELSE GO TO ENDLOOP4;
S615: IF INSTO(IND3-1) = 0 THEN GO TO S617;
     IF INSTO(IND3+1) = MAXST(IND3+1) THEN GO TO S616;
     IF NSTOR(IND3) = (INSTO(IND3)+MACHI(IND3-1)-MACHI(IND3)) THEN GO
     TO ENDLOOP8; ELSE GO TO ENDLOOP4;
S616: IF NSTOR(IND3) = (INSTO(IND3)+MACHI(IND3-1)) THEN GO TO ENDLOOP8;
     ELSE GO TO ENDLOOP4;
S617: IF INSTO(IND3+1) = MAXST(IND3+1) THEN GO TO S618;
     IF NSTOR(IND3) = (INSTO(IND3)-MACHI(IND3)) THEN GO TO ENDLOOP8;
     ELSE GO TO ENDLOOP4;
S618: IF NSTOR(IND3) ¬= INSTO(IND3) THEN GO TO ENDLOOP4;
ENDLOOP8: END LOOP8;
     PROB='P.(';
LOOP9: DO IND3=2 TO N;
      INT=0;
S152:         I=INSTO(IND3)/(10**(INT+1)); IF I=0 THEN GO TO S153;
     INT=INT+1; GO TO S152;
S153: CHR=INSTO(IND3)||',';
     CHR=SUBSTR(CHR,9-INT,2+INT);
     PROB=PROB||CHR;
     END LOOP9;
LOOP10: DO IND3=1 TO N-1;
     CHR=INMAC(IND3)||',';
     CHR=SUBSTR(CHR,9,2);
     PROB=PROB||CHR;
     END LOOP10;
     CHR=INMAC(N)||')';
     CHR=SUBSTR(CHR,9,2);
     PROB=PROB||CHR;
     LET(SUM=SUM+(TRN*"PROB"));
ENDLOOP4: END LOOP4; END LOOPB;
     PUT LIST('  ***EQUATION NO.  ',ITER,' ***') ;
     PRINT_OUT(ZERO=SUM-"FINAL");
     ITER=ITER+1;
ENDLOOP1: END LOOP1; END LOOPA;
     END BOUNDRY;
```

A.3 The Power Method Iterative Multiplication Program

This program computes the steady-state probability distribution for a three-machine line by the power method. It is possible to speed convergence by first solving a small problem and then performing the δ-transformation on the results in order to get an accurate initial guess for the larger problem.

The input is as follows:

First Card:	Columns 1-30:	Failure probabilities (p_i, i=1,2,3) (Format F10.5)
	Columns 31-60:	Repair probabilities (r_i, i=1,2,3) (Format F10.5)
	Columns 61-64:	Storage capacities (N_i, i=1,2) (Format I2)
	Columns 65-67:	Value of $1/\delta$ (>1, since smaller problems are to be solved first.) (Format I3)
	Columns 68-70:	Number of times the δ-transformation is to be performed. (Format I3)
	Columns 71-80:	Convergence criterion (Note that $\varepsilon \rightarrow \varepsilon\delta^2$ when the transformation is applied) (Format F10.5)

```
C  ---THREE STATIONS---SOLUTION BY SPARSE MATRIX ITERATION
C --- AND BOOTSTRAPPING
C -- ACCELERATE EVERY TENTH STEP
C -- SCALE MODIFICATION 2
          COMMON               P,Q,P3,R,S,R3,NN1,NN2,K
         1    ,NP(2) ,AP(3),AR(3), NN11,NN21
         2 ,   ERR,ERR2,NQ
         1   ,IX
         4     ,IM,IFACT,AFACT,KK
          COMMON /CS/ C(10000)
          COMMON /BS/ IY(10000,2)
          COMMON / PRTY / Y(1000)
          DOUBLE PRECISION Y
          INTEGER AA,BB,A1,B1,A2,B2
          DO  999 I=1,1000
  999     Y(I)=0
          NQ = 10000
          ERR2 = 1.E-6
  199     CONTINUE
          READ (5,6) P,Q,P3,R,S,R3,NN1,NN2,IFACT,KK,ERR
    6     FORMAT (6F10.5,2I2,I3,I3 ,F10.5)
          IF (NN1 .EQ. 0) GO TO 199
  997     IF (NN2 .EQ. 0) GO TO 199
          AFACT = IFACT
          DO 9000 IM = 1,KK
          WRITE (6,97)
   97     FORMAT (1H1)
          IF (IM .EQ. 1) GO TO 9001
          ERR = ERR/AFACT      /AFACT
          P  =  P  /AFACT
          Q  =  Q  /AFACT
          R  =  R  /AFACT
          S  =  S  /AFACT
          P3 =  P3 /AFACT
          R3 =  R3 /AFACT
          NN1=  NN1*IFACT
          NN2=  NN2*IFACT
 9001     CONTINUE
          NN11 = NN1+ 1
          NN21 = NN2+ 1
          NP(1) = NN11
          NP(2) = NN21
          AP(1) = P
          AP(2) = Q
          AP(3) = P3
          AR(1) = R
          AR(2) = S
          AR(3) = R3
          WRITE (6,4) P,Q,P3,R,S,R3,NN1,NN2,ERR
    4     FORMAT (1H ,    20HPARAMETERS
         1    4H P =F15.5/
         1    4H Q =F15.5/
         1    4H P3=F15.5/
```

```
                     1    4H R =F15.5/
                     1    4H S =F15.5/
                     1    4H R3=F15.5/
                     1    4H N1=I10/
                     1    4H N2=I10/
                     1    4H E =E15.5)
0043                  WRITE (6,98)
0044               98 FORMAT (////)
0045                  K=8*NN11*NN21
0046                  IF (K .GT.1000) GO TO 199
0047                  CALL AMAT
0048             9000 CONTINUE
0049                  GO TO 199
0050                  END
```

FORTRAN IV G1 RELEASE 2.0 AMAT DATE = 77159 18/58/39

```
      SUBROUTINE AMAT
      COMMON          P,Q,P3,R,S,R3,NN1,NN2,K
     1    ,NP(2) ,AP(3),AR(3), NN11,NN21
     2 ,  ERR,ERR2,NQ          ,IX
     4    ,IM,IFACT,AFACT
      COMMON /NS/ JN1,JN2
      COMMON /CS/ C(10000)
      COMMON /BS/  B(10000,2)
      INTEGER AA,BB,A1,B1,A2,B2
     1    ,B
      IX = 0
      DO 8001 I1 = 1,2
      DO 8002 I2 = 1,2
      DO 8003 JN1 = 1,NN11
      DO 8004 JN2 = 1,NN21
      CALL NTRANS (I1,I2,JN1,IN1,1)
      DO 8005 I3 = 1,2
      CALL NTRANS (I2,I3,JN2,IN2,2)
      DO 8006 J1 = 1,2
      CALL ATRANS (1,J1,I1,PP1,JN1,JN2)
      IF (PP1 .LT. ERR2) GO TO 8006
      DO 8007 J2 = 1,2
      CALL ATRANS (2,J2,I2,PP2,JN1,JN2)
      IF (PP2 .LT. ERR2) GO TO 8007
      DO 8008 J3 = 1,2
      CALL ATRANS (3,J3,I3,PP3,JN1,JN2)
      AX       = PP1*PP2*PP3
      IF (AX .LT. ERR2) GO TO 8008
      A2 = I3+2*I2+4*I1+8*IN2+8*NN21*(IN1-1)-14
      A1 = J3+2*J2+4*J1+8*JN2+8*NN21*(JN1-1)-14
      IX = IX + 1
      B(IX,1) = A1
      B(IX,2) = A2
      C(IX) = AX
 8008 CONTINUE
 8007 CONTINUE
 8006 CONTINUE
 8005 CONTINUE
 8004 CONTINUE
 8003 CONTINUE
 8002 CONTINUE
 8001 CONTINUE
      CALL ITER
      RETURN
      END
```

```
      SUBROUTINE ITER
      COMMON             P,Q,P3,R,S,R3,NN1,NN2,K
     1    ,NP(2) ,AP(3),AR(3),  NN11,NN21
     2 ,   ERR,ERR2,NQ          ,IX
     4     ,IM,IFACT,AFACT,KK
      DIMENSION RR1(1000),RR2(1000)
      DOUBLE PRECISION RR1,RR2
      K2 = K*K
      WRITE (6,9) IX,K2
    9 FORMAT (// ' THERE ARE',I5,' NON-ZERO ELEMENTS OUT OF A POSSIBLE',
     1    I10/' IN THE TRANSITION MATRIX'//)
      ICOUNT = 0
      IF (IM .NE. 1) GO TO 9000
      AK = K
      AK = 1./AK
      DO 6 I = 1,K
    6 RR1(I) = AK
    8 CALL MATMLT(RR1,RR2)
      ICOUNT = ICOUNT + 1
      AX = 0.
      BX = 0.
      DO 7 I = 1,K
      AX = AX +DABS(RR1(I) - RR2(I))
      RR11 = RR1(I)
    7 BX = AMAX1(BX,RR11)
      IF (AX .LE. BX*ERR) GO TO 10
      TX = AX
      DO 3 I = 1,K
      IF (RR2(I) .LT. ERR2) GO TO 32
      RR1(I) = RR2(I)
      GO TO 3
   32 RR1(I) = 0.
    3 CONTINUE
      AX = 0.
      DO 33 I = 1,K
   33 AX = AX + RR1(I)
      AX = 1./AX
      DO 34 I = 1,K
   34 RR1(I) = RR1(I) * AX
      IF (ICOUNT .EQ. (ICOUNT/10)*10)
     1 CALL PRINT (RR1,NN11,NN21,ICOUNT,TX,BX,0)
      GO TO 8
   10 J=0
      IF(IM.EQ.KK) J=1
      CALL PRINT (RR2,NN11,NN21,ICOUNT,AX,BX,J)
      RETURN
    5 FORMAT (60X,E20.8)
   31 FORMAT (30H1PROBABILITY DISTRIBUTION             ///
     1     (8E15.5/))
    4 FORMAT   (6I10)
 9000 CALL SCALE (RR1,RR2)
      GO TO 8
      END
```

FORTRAN IV G1 RELEASE 2.0   ATRANS   DATE = 77159   18/58/39

```
      SUBROUTINE ATRANS (L1,L2,L3,PP,JN1,JN2)
      COMMON             P,Q,P3,R,S,R3,NN1,NN2,K
     1   ,NP(2) ,AP(3),AR(3), NN11,NN21
      AL3 = L3 - 1
      IF (L2 .NE. 1) GO TO 10
      PP = AR(L1)*AL3 + (1.-AR(L1))*(1.-AL3)
      RETURN
   10 IF (L1-2) 1,2,3
    1 IF (JN1 .EQ. NN11) GO TO 4
    5 PP = AP(L1)*(1.-AL3) + (1.-AP(L1))*AL3
      RETURN
    4 PP = AL3
      RETURN
    2 IF (JN1 .EQ. 1 .OR. JN2 .EQ. NN21) GO TO 4
      GO TO 5
    3 IF (JN2-1) 4,4,5
      END
```

```
FORTRAN IV G1  RELEASE 2.0            NTRANS            DATE = 77159          18/58/39

0001                  SUBROUTINE NTRANS (A,B,N1,N2,III)
                C   MODIFIED TO ACCOUNT FOR STORAGE BACKUP
0002                  COMMON             P,Q,P3,R,S,R3,NN1,NN2,K
                     1    ,NP(2) ,AP(3),AR(3), NN11,NN21
0003                  COMMON /NS/ IN(2)
0004                  INTEGER A,B
0005                  N2 = N1      +
                     1      (A-1)*IU(III)    -    (B-1)*ID(III)
0006                  RETURN
0007                  END
```

```
FORTRAN IV G1  RELEASE 2.0              IU                  DATE = 77159          18/58/39

 0001                    FUNCTION IU(IX)
 0002                    COMMON            P,Q,P3,R,S,R3,NN1,NN2,K
                        1    ,NP(2) ,AP(3),AR(3), NN11,NN21
 0003                    COMMON /NS/ IN(2)
 0004                    IF (IX .EQ. 1) GO TO 1
 0005                    IF (IN(IX - 1) .NE.    1       ) GO TO 1
 0006                  2 IU = 0
 0007                    RETURN
 0008                  1 IF (IN(IX) .EQ. NP(IX)) GO TO 2
 0009                    IU = 1
 0010                    RETURN
 0011                    END
```

```
FORTRAN IV G1  RELEASE 2.0              ID                DATE = 77159          18/58/39

 0001                  FUNCTION ID(IX)
 0002                  COMMON            P,Q,P3,R,S,R3,NN1,NN2,K
                      1   ,NP(2) ,AP(3),AR(3), NN11,NN21
 0003                  COMMON /NS/ IN(2)
 0004                  IF (IX .EQ. 2) GO TO 1
 0005                  IF (IN(IX + 1) .NE. NP(IX + 1)) GO TO 1
 0006                2 ID = 0
 0007                  RETURN
 0008                1 IF (IN(IX) .EQ.   1   ) GO TO 2
 0009                  ID = 1
 0010                  RETURN
 0011                  END
```

```
FORTRAN IV G1  RELEASE 2.0              MATMLT              DATE = 77159          18/58/39

0001                SUBROUTINE MATMLT (Z1, Z2)
0002                COMMON             P,Q,P3,R,S,R3,NN1,NN2,K
                   1    ,NP(2) ,AP(3),AR(3), NN11,NN21
                   2 ,   ERR,ERR2,NQ       ,IX
0003                COMMON /CS/ C(10000)
0004                COMMON /BS/  B(10000,2)
0005                INTEGER B
0006                DIMENSION Z1(1000),Z2(1000)
0007                DOUBLF PRFCISION Z1,Z2
0008                DO 1 I1=1,K
0009              1 Z2 (I1)=0.
0010                DO 2 IZ = 1,IX
0011              2 Z2(B(IZ,2)) = Z2(B(IZ,2)) + C(IZ)*Z1(B(IZ,1))
0012                RETURN
0013                END
```

FORTRAN IV G1 RELEASE 2.0 PRINT DATE = 77159 18/58/39

```
      SUBROUTINE PRINT (X,N1,N2,II,AA,BB,IFLG)
      COMMON / PRTY / Y(1000)
      DIMENSION X(1000)
      DOUBLE PRECISION X
     1    ,Y
      WRITE (6,1) II
    1 FORMAT (1H1, 10X, 'PROBABILITY DISTRIBUTION -- ITERATION ',
     1   I4/////  4X,'000', 12X, '001',12X, '010', 12X, '011',
     2   12X,'100', 12X, '101', 12X, '110', 12X,'111'/////)
      N28 = N2*8
      DO 2 I = 1,N1
      I1 = I - 1
      N28I = N28*I1
      WRITE (6,3) I1
    3 FORMAT ( //10X, 'N1 =',I3//)
      IF(IFLG.EQ.1) WRITE(7,99)(X(N28I+J),J=1,N28)
   99 FORMAT(8E10.5)
    2 WRITE (6,4)(X(N28I+J),J=1,N28)
    4 FORMAT (8E15.5/)
      CC = AA/BB
      WRITE (6,5) AA,BB,CC
    5 FORMAT (///' STOP CRITERION: AX =', E17.8, '  BX =', E17.8,2X,
     1    2X,
     1    'AX/BX ='   ,  E17.8)
      EE = 0.
      K = N28*N1
      DD = AX
      AX = 0.
      DO 1002 I = 1,K
 1002 AX = AX + DABS (X(I) - Y(I))
      IF (II .LE. 30) GO TO 1000
      IF (II .NE. (II/20)*20) GO TO 1000
      ALAM = AX/DD
      EE = ALAM/(1.-ALAM)
 1000 CONTINUE
      DO 1001 I = 1,K
      Y(I) = X(I) - EE*(Y(I) - X(I))
 1001 X(I) = Y(I)
      RETURN
      END
```

FORTRAN IV G1 RELEASE 2.0 SCALE DATE = 77159 18/58/39

```
      SUBROUTINE SCALE (RR1,RR2)
      COMMON            P,Q,P3,R,S,R3,NN1,NN2,K
     1   ,NP(2) ,AP(3),AR(3), NN11,NN21
     2 ,  ERR,ERR2,NO          ,IX
     4    ,IM,IFACT,AFACT
      INTEGER A1, A2
      DIMENSION RR1(1000),RR2(1000)
      DOUBLE PRECISION RR1,RR2
     2    ,RAT(3)
      RAT(1) = AFACT*AFACT
      RAT (2) = AFACT
      RAT(3) = 1.
      NM11 = (NN11-1)/IFACT + 1
      NM21 = (NN21-1)/IFACT + 1
      DO 2 JN1 = 1,NN11
      IN1 = (JN1-1)/IFACT + 1
      IF (JN1 .EQ. 2) IN1 = 2
      IF (JN1 .EQ. 3 .OR. JN1 .EQ. 4) IN1 = 3
      IF (JN1 .EQ. NN1) IN1 = NM11 - 1
      IF (JN1 .EQ. NN11-2 .OR. JN1 .EQ. NN11-3) IN1 = NM11-2
      DO 2 JN2 = 1,NN21
      IN2 = (JN2-1)/IFACT + 1
      IF (JN2 .EQ. 2) IN2 = 2
      IF (JN2 .EQ. 3 .OR. JN2 .EQ. 4) IN2 = 3
      IF (JN2 .EQ. NN2) IN2 = NM21 - 1
      IF (JN2 .EQ. NN21-2 .OR. JN2 .EQ. NN21-3) IN2 = NM21-2
      DO 2 J1 = 1,2
      DO 2 J2 = 1,2
      DO 2 J3 = 1,2
      A1 = J3+2*J2+4*J1+8*JN2+8*NN21*(JN1-1)-14
      A2 = J3+2*J2+4*J1+8*IN2+8*NM21*(IN1-1)-14
      II = 1
      IF (JN1 .LT. 2) II = II + 1
      IF (JN2 .LT. 2) II = II + 1
      IF (JN1 .GT. NN1) II = II + 1
      IF (JN2 .GT. NN2) II = II + 1
    2 RR1(A1) = RR2(A2)/RAT(II)
      CALL PRINT (RR1,NN11,NN21,0,1.,1.,0)
      RETURN
      END
```

A.4 Block Tri-Diagonal Equation System Solver

The present version of this program is for three-machine lines only. The program can be rewritten in a recursive language (e.g. PL/I) in order to solve general k-machine lines. The necessary change is indicated by the dotted line on the flow-chart on page 283.

The program uses the IBM IMSL subroutine LINV2F to invert the lowest-level main-diagonal blocks. The closed-form solutions for the inverses of these blocks may be incorporated in the program (See section 4.2.2).

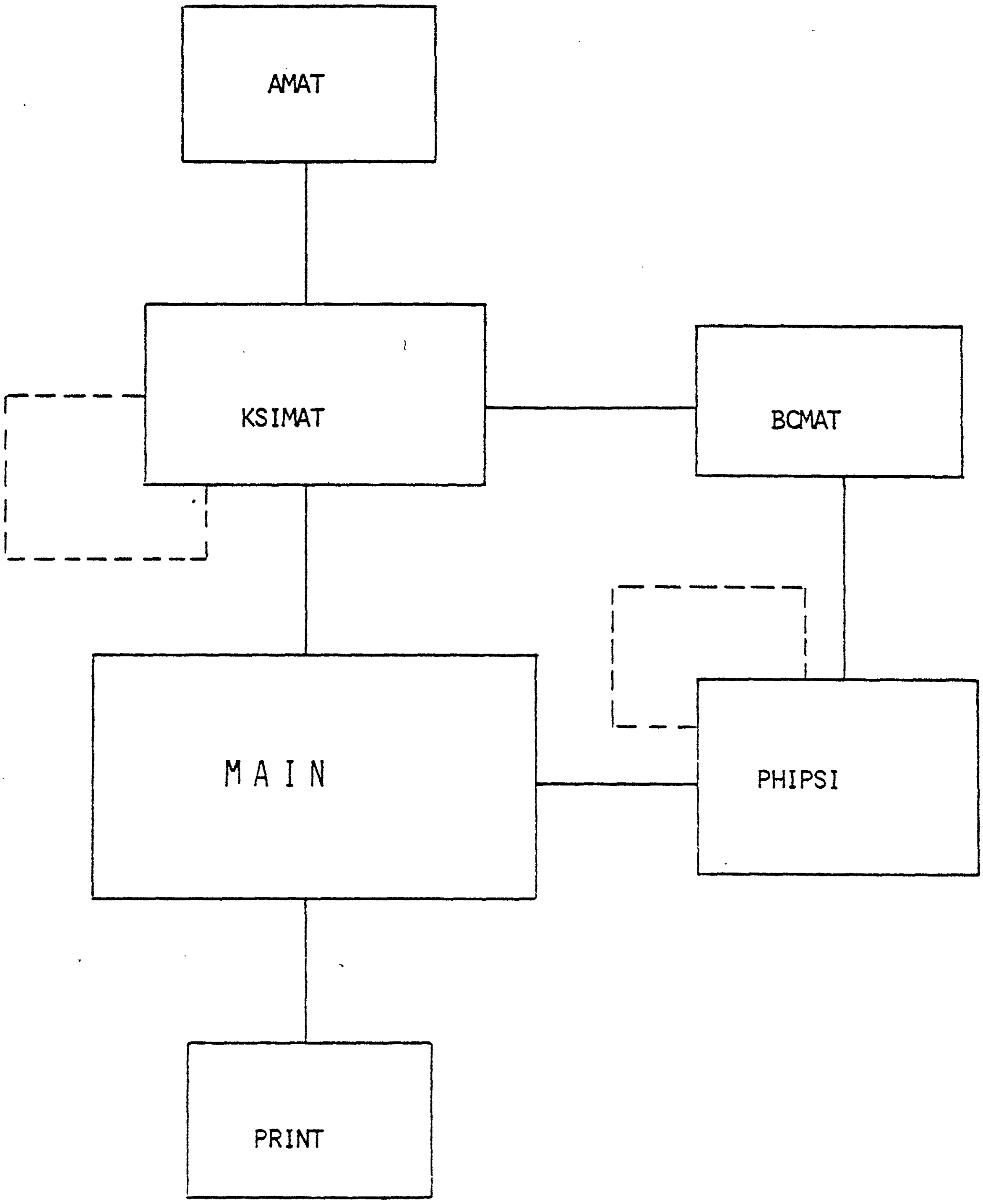
The input is as follows:

```
First Card :  Columns 1-3: Number of machines (in the present
                           vesrion of the program, this must be 3)
              Column 4:    An asterisk (*) in this column supresses
                           the printing of the probability distribution)

Next K Cards: Columns 1-13: Repair probability of machine (in order)
                            (Format E13.6)
              Columns 14-26:Failure probability of machine (in order)
                            (Format E13.6)

Next K-1 Cards: Columns 1-5: Storage capacity (in order) (Format I5)
```

A sample of the output of this program appears on page 284.



Flow-chart of the Block tri-diagonal equation system solver. (The dotted lines indicate the recursions necessary for solving systems of transition equations for transfer lines with more than three machines.)

3 MACHINES AND 2 STORAGES.

```
MACHINE  1  FAILURE PROBABILITY:  0.100000D+00, MEAN UP-TIME:    0.100000D+02
            REPAIR PROBABILITY:   0.200000D+00, MEAN DOWN-TIME:  0.500000D+01
            EFFICIENCY (IN ISOLATION):  0.666667D+00

MACHINE  2  FAILURE PROBABILITY:  0.500000D-01, MEAN UP-TIME:    0.200000D+02
            REPAIR PROBABILITY:   0.200000D+00, MEAN DOWN-TIME:  0.500000D+01
            EFFICIENCY (IN ISOLATION):  0.800000D+00

MACHINE  3  FAILURE PROBABILITY:  0.500000D-01, MEAN UP-TIME:    0.200000D+02
            REPAIR PROBABILITY:   0.150000D+00, MEAN DOWN-TIME:  0.666667D+01
            EFFICIENCY (IN ISOLATION):  0.750000D+00

STORAGE  1 HAS MAXIMUM CAPACITY:     4

STORAGE  2 HAS MAXIMUM CAPACITY:     4
```

PROBABILITY DISTRIBUTION :

N1 = 0

000	001	010	011	100	101	110	111	N2
0.0	0.0	0.0	0.13183D+00	0.0	0.0	0.0	0.0	0
0.0	0.0	0.42525D-02	0.34020D-01	0.0	0.0	0.0	0.0	1
0.0	0.0	0.51995D-02	0.88745D-02	0.0	0.0	0.0	0.0	2
0.0	0.0	0.47515D-02	0.76938D-02	0.0	0.0	0.0	0.0	3
0.0	0.0	0.43149D-02	0.0	0.0	0.0	0.0	0.0	4

N1 = 1

000	001	010	011	100	101	110	111	N2
0.0	0.44718D-02	0.0	0.0	0.0	0.0	0.0	0.32957D-01	0
0.17467D-03	0.38044D-03	0.0	0.10063D-01	0.0	0.0	0.10631D-02	0.17133D+00	1
0.16458D-03	0.39255D-03	0.51171D-03	0.10548D-02	0.0	0.0	0.91719D-02	0.15987D-01	2
0.15691D-03	0.0	0.49270D-03	0.79309D-02	0.0	0.0	0.83329D-02	0.14086D-01	3
0.0	0.0	0.44590D-02	0.0	0.0	0.0	0.75519D-02	0.0	4

N1 = 2

000	001	010	011	100	101	110	111	N2
0.0	0.43394D-02	0.0	0.0	0.0	0.96877D-02	0.0	0.0	0
0.15678D-03	0.37092D-03	0.0	0.10009D-01	0.45799D-03	0.81976D-03	0.0	0.20728D-01	1
0.15010D-03	0.40145D-03	0.50743D-03	0.93399D-03	0.41720D-03	0.73827D-03	0.10647D-02	0.25207D-02	2
0.14425D-03	0.0	0.45617D-03	0.81363D-02	0.37992D-03	0.0	0.10488D-02	0.14810D-01	3
0.0	0.0	0.45398D-02	0.0	0.0	0.0	0.81445D-02	0.0	4

N1 = 3

000	001	010	011	100	101	110	111	N2
0.0	0.42293D-02	0.0	0.0	0.0	0.93210D-02	0.0	0.0	0
0.14505D-03	0.36955D-03	0.0	0.0	0.38573D-03	0.73924D-03	0.0	0.10052D+00	1
0.13796D-03	0.40984D-03	0.0	0.0	0.35670D-03	0.76266D-03	0.47802D-02	0.86948D-02	2
0.13474D-03	0.0	0.0	0.67640D-03	0.33351D-03	0.0	0.42945D-02	0.76627D-01	3
0.0	0.0	0.38330D-02	0.0	0.0	0.0	0.14430D-01	0.0	4

N1 = 4

000	001	010	011	100	101	110	111	N2
0.0	0.0	0.0	0.0	0.0	0.79784D-01	0.0	0.0	0
0.0	0.0	0.0	0.0	0.21587D-02	0.44816D-02	0.0	0.0	1
0.0	0.0	0.0	0.0	0.19977D-02	0.42264D-02	0.0	0.0	2
0.0	0.0	0.0	0.0	0.18557D-02	0.0	0.0	0.13754D-01	3
0.0	0.0	0.0	0.0	0.0	0.0	0.77937D-01	0.0	4

```
LINE EFFICIENCY =     0.54254D+00

AVERAGE STORAGE FILLS  :
STORAGE 1  :      0.19110D+01
STORAGE 2  :      0.15007D+01

AVERAGE STORAGE FILLS (FRACTION OF MAXIMUM CAPACITY) :
STORAGE 1  :      0.47774D+00
STORAGE 2  :      0.37516D+00

TOTAL IN-PROCESS INVENTORY  :      0.34116D+01
```

```
FORTRAN IV G1  RELEASE 2.0             AMAT               DATE = 78177            16/48/48

0001                 SUBROUTINE AMAT(INDEX,FINV)
            C*****BUILDS AND INVERTS LOWEST LEVEL MAIN DIAGONAL BLOCK
0002                 IMPLICIT REAL*8 (A-H,O-Z)
0003                 COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IDUM,NNN,ASTER
0004                 DIMENSION F(8,8),NREGN(4),MAC(3),MACIN(3),FINV(8,8),WKAREA(100)
0005                 N1=N-1
0006                 NREGN(1)=2
0007                 NREGN(N+1)=2
0008                 J=INDEX-1
0009                 DO 1 I=1,N1
0010                 NREGN(N1-I+2)=(J/(3**(N1-I)))+1
0011               1 J=J-((NREGN(N1-I+2)-1)*(3**(N1-I)))
            C*****NREGN(I)=1/2/3 IF STORAGE (I-1) IS EMPTY/INTERNAL/FULL
0012                 DO 2 I=1,NN
0013                 DO 2 J=1,NN
0014               2 F(I,J)=0.D0
0015                 DO 115 L=1,NN
0016                 J=L-1
0017                 DO 3 K=1,N
0018                 MAC(K)=J/(2**(N-K))
0019               3 J=J-(MAC(K)*(2**(N-K)))
0020                 DO 7 J=1,N1
0021                 IN=MAC(J)
0022                 IF(NREGN(J).EQ.1) IN=0
0023                 IOUT=-MAC(J+1)
0024                 IF(NREGN(J+2).EQ.3) IOUT=0
0025                 K=NREGN(J+1)
0026                 GO TO (4,5,6),K
0027               4 IF(IN.NE.0)  GO TO 115
0028                 GO TO 7
0029               5 IF((IN+IOUT).NE.0) GO TO 115
0030                 GO TO 7
0031               6 IF(IOUT.NE.0) GO TO 115
0032               7 CONTINUE
            C*****NOW BUILD ROW OF MATRIX
0033                 DO 11 I=1,NN
0034                 J=I-1
0035                 DO 9 K=1,N
0036                 MACIN(K)=J/(2**(N-K))
0037               9 J=J-(MACIN(K)*(2**(N-K)))
            C*****MACIN(I)=0/1 IS THE INITIAL STATE OF MACHINE I
0038                 F(L,I)=1.D0
0039                 DO 11 J=1,N
0040                 F(L    ,I)=F(L    ,I)*((1.D0-R(J))**((1-MACIN(J))*(1-MAC(J))))*
                    1 (R(J)**((1-MACIN(J))*MAC(J)))
0041                 IF(NREGN(J).EQ.1.OR.NREGN(J+1).EQ.3) GO TO 10
0042                 F(L    ,I)=F(L    ,I)*(P(J)**(MACIN(J)*(1-MAC(J))))*
                    1                                   ((1.D0-P(J))**(MACIN(J)*MAC(J)))
0043                 GO TO 11
0044              10 IF((MACIN(J)*(1-MAC(J))).NE.0) F(L    ,I)=0.D0
0045              11 CONTINUE
0046             115 CONTINUE
0047                 DO 12 I=1,NN
```

FORTRAN IV G1 RELEASE 2.0 AMAT DATE = 78177 16/48/48

```
      12 F(I,I)=F(I,I)-1.D0                                           MAT00540
         IF(INDEX.NE.1) GO TO 14                                      MAT00550
         DO 13 I=1,NN                                                 MAT00560
      13 F(1,I)=1.D0                                                  MAT00570
      14 CALL LINV2F(F,NN,NN,FINV,6,WKAREA,IER)                       MAT00580
         RETURN                                                       MAT00590
         END                                                          MAT00600
```

```
FORTRAN IV G1  RELEASE 2.0          BCMAT              DATE = 78177          16/48/48

                  SUBROUTINE BCMAT(MATNM,INDEX,F)
            C*****BUILDS LOWEST LEVEL BLOCKS EXCEPT MAIN DIAGONAL
                  IMPLICIT REAL*8(A-H,O-Z)
                  COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IDUM,NNN,ASTER
                  DIMENSION F(8,8),NREGN(4),MAC(3),MACIN(3),NSTO(2)
                  N1=N-1
                  NM=NN-1
                  NREGN(1)=2
                  NREGN(N+1)=2
                  J=INDEX-1
                  DO 1 I=1,N1
                  NREGN(N1-I+2)=(J/(3**(N1-I)))+1
                1 J=J-((NREGN(N1-I+2)-1)*(3**(N1-I)))
            C*****NREGN(I)=1/2/3 IF STORAGE (I-1) IS EMPTY/INTERNAL/FULL
                  J=MATNM
                  DO 2 I=1,N
                  MAC(I)=J/(2**(N-I))
                2 J=J-(MAC(I)*(2**(N-I)))
            C*****MAC(I)=0/1 IF FINAL MACHINE I IS DOWN/UP IN INTERNAL CASE MATRIX
                  DO 3 I=1,N1
                3 NSTO(I)=MAC(I)-MAC(I+1)
            C*****NSTO(I)=-1/0/1 IF STORAGE I GOES DOWN/CONSTANT/UP IN THIS MATRIX
                  DO 4 I=1,NN
                  DO 4 J=1,NN
                4 F(I,J)=0.D0
                  DO 135 L=1,NM
                  J=L
                  DO 5 K=1,N
                  MAC(K)=J/(2**(N-K))
                5 J=J-(MAC(K)*(2**(N-K)))
                  DO 9 J=1,N1
                  IN=MAC(J)
                  IF(NREGN(J).EQ.1) IN=0
                  IOUT=-MAC(J+1)
                  IF(NREGN(J+2).EQ.3) IOUT=0
                  K=NREGN(J+1)
                  GO TO (6,7,8),K
                6 IF(IN.NE.NSTO(J)) GO TO 135
                  GO TO 9
                7 IF((IN+IOUT).NE.NSTO(J)) GO TO 135
                  GO TO 9
                8 IF(IOUT.NE.NSTO(J)) GO TO 135
                9 CONTINUE
            C*****NOW BUILD ROW OF MATRIX
                  DO 13 I=1,NN
                  J=I-1
                  DO 11 K=1,N
                  MACIN(K)=J/(2**(N-K))
               11 J=J-(MACIN(K)*(2**(N-K)))
            C*****MACIN(I)=0/1 IS THE INITIAL STATE OF MACHINE I
                  F( L+1,I)=1.D0
                  DO 13 J=1,N
                  F(L+1 ,I)=F(L+1 ,I)*((1.D0-R(J))**((1-MACIN(J))*(1-MAC(J))))*
```

```
FORTRAN IV G1  RELEASE 2.0              BCMAT                DATE = 78177           16/48/48

                        1 (R(J)**((1-MACIN(J))*MAC(J)))                                       MAT01140
 0048                     IF(NREGN(J).EQ.1.OR.NREGN(J+1).EQ.3) GO TO 12                       MAT01150
 0049                     F(L+1 ,I)=F(L+1 ,I)*(P(J)**(MACIN(J)*(1-MAC(J))))*                  MAT01160
                        1                          ((1.D0-P(J))**(MACIN(J)*MAC(J)))           MAT01170
 0050                     GO TO 13                                                            MAT01180
 0051                  12 IF((MACIN(J)*(1-MAC(J))).NE.0) F(L+1 ,I)=0.D0                       MAT01190
 0052                  13 CONTINUE                                                            MAT01200
 0053                 135 CONTINUE                                                            MAT01210
 0054                     RETURN                                                              MAT01220
 0055                     END                                                                 MAT01230
```

```
      SUBROUTINE KSIMAT(IND,KSINV)
C*****CONSTRUCTS INVERSE OF SECOND-LEVEL MAIN DIAGONAL BLOCK
C*****THE PROGRAM ASSUMES THAT THERE ARE NO PATHOLOGICAL CASES
C*****WITH STORAGE SIZES LESS THAN 2
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 KSINV
      COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IOUT,NNN,ASTER
      DIMENSION B(8,8),C(8,8),AINV(8,8)
      DIMENSION XINV(8,8,11),D(8,8,11),KSINV(88,88)
      DIMENSION DUM1(8,8),DUM2(8,8),DUM3(8,8),DUM4(8)
      INDEX=IND
      CALL AMAT(INDEX,AINV)
      DO 10 I=1,NN
      DO 10 J=1,NN
   10 XINV(I,J,1)=AINV(I,J)
      JC=0
      DO 105 I=2,N
  105 JC=JC+(2**(N-I))
      JB=2**(N-1)
      DO 20 LOOP=2,LIMIT
      IF(LOOP.LE.3) CALL BCMAT(JC,INDEX,C)
      IF(LOOP.EQ.2.OR.LOOP.EQ.LIMIT) INDEX=INDEX-1
      IF(LOOP.EQ.2.OR.LOOP.EQ.LIMIT) CALL BCMAT(JB,INDEX,B)
      DO 11 I=1,NN
      DO 11 J=1,NN
      DUM1(I,J)=0.D0
      DO 11 K=1,NN
   11 DUM1(I,J)=DUM1(I,J)+C(I,K)*XINV(K,J,LOOP-1)
      DO 12 I=1,NN
      DO 12 J=1,NN
      DUM2(I,J)=0.D0
      DO 12 K=1,NN
   12 DUM2(I,J)=DUM2(I,J)+DUM1(I,K)*B(K,J)
C*****DUM2 IS A SPARSE MATRIX WITH SOME NONZERO ROWS
      IF(LOOP.EQ.2.OR.LOOP.EQ.LIMIT) CALL AMAT(INDEX,AINV)
      DO 13 I=1,NN
      DO 13 J=1,NN
   13 XINV(I,J,LOOP)=AINV(I,J)
      DO 19 I=1,NN
      DO 18 J=1,NN
      IF(DUM2(I,J).EQ.0.D0) GO TO 18
      SCAL=-1.D0
      DO 14 K=1,NN
   14 SCAL=SCAL+DUM2(I,K)*XINV(K,I,LOOP)
      DO 15 K=1,NN
      DUM4(K)=0.D0
      DO 15 L=1,NN
   15 DUM4(K)=DUM4(K)+DUM2(I,L)*XINV(L,K,LOOP)
      DO 16 K=1,NN
      DO 16 L=1,NN
   16 DUM3(K,L)=XINV(K,I,LOOP)*DUM4(L)
      DO 17 K=1,NN
      DO 17 L=1,NN
```

```
      17 XINV(K,L,LOOP)=XINV(K,L,LOOP)-DUM3(K,L)/SCAL
         GO TO 19
      18 CONTINUE
      19 CONTINUE
      20 CONTINUE
C*****X'S DEFINED, COMPUTING D'S
         DO 35 IDENT=1,LIMIT
         INDEX=IND
         L=LIMIT-IDENT
         IF(L.EQ.0) GO TO 22
         DO 21 I=1,L
         DO 21 J=1,NN
         DO 21 K=1,NN
      21 D(J,K,I)=0.D0
      22 L=L+1
         DO 27 I=L,LIMIT
         IF(I.NE.L) GO TO 24
         DO 23 J=1,NN
         DO 23 K=1,NN
      23 D(J,K,I)=XINV(J,K,I)
         IF(I.EQ.LIMIT) GO TO 27
         IF(I.EQ.1) CALL BCMAT(JC,INDEX,C)
         INDEX=INDEX-1
         GO TO 27
      24 IF(I.EQ.(L+1).AND.I.NE.2) CALL BCMAT(JC,INDEX,C)
         IF(I.EQ.(L+2).AND.I.EQ.3) CALL BCMAT(JC,INDEX,C)
         DO 25 J=1,NN
         DO 25 K=1,NN
         DUM1(J,K)=0.D0
         DO 25 M=1,NN
      25 DUM1(J,K)=DUM1(J,K)-C(J,M)*D(M,K,I-1)
         DO 26 J=1,NN
         DO 26 K=1,NN
         D(J,K,I)=0.D0
         DO 26 M=1,NN
      26 D(J,K,I)=D(J,K,I)+XINV(J,M,I)*DUM1(M,K)
      27 CONTINUE
C*****D'S DEFINED, COMPUTING KSI INVERSE
         INDEX=IND-2
         DO 34 I=1,LIMIT
         IF(I.NE.1) GO TO 30
         DO 28 J=1,NN
         DO 28 K=1,NN
      28 DUM3(J,K)=D(J,K,LIMIT)
         DC 29 J=1,NN
         DO 29 K=1,NN
      29 KSINV(J,(IDENT-1)*NN+K)=DUM3(J,K)
         CALL BCMAT(JB,INDEX,B)
         INDEX=INDEX+1
         GO TO 34
      30 IF(I.EQ.3) CALL BCMAT(JB,INDEX,B)
         DO 31 J=1,NN
         DO 31 K=1,NN
```

```
FORTRAN IV G1  RELEASE 2.0              KSIMAT              DATE = 78177          16/46/48

 0101                 DUM1(J,K)=0.D0
 0102                 DO 31 L=1,NN
 0103              31 DUM1(J,K)=DUM1(J,K)+B(J,L)*DUM3(L,K)
 0104                 DO 32 J=1,NN
 0105                 DO 32 K=1,NN
 0106                 DUM2(J,K)=0.D0
 0107                 DO 32 L=1,NN
 0108              32 DUM2(J,K)=DUM2(J,K)+XINV(J,L,LIMIT+1-I)*DUM1(L,K)
 0109                 DO 33 J=1,NN
 0110                 DO 33 K=1,NN
 0111                 DUM3(J,K)=D(J,K,LIMIT+1-I)-DUM2(J,K)
 0112              33 KSINV((I-1)*NN+J,(IDENT-1)*NN+K)=DUM3(J,K)
 0113              34 CONTINUE
                C*****BLOCK-COLUMN 'IDENT' OF KSI INVERSE COMPUTED
 0114              35 CONTINUE
                C*****KSI INVERSE COMPUTED
 0115                 RETURN
 0116                 END
```

```
FORTRAN IV G1  RELEASE 2.0              PHIPSI              DATE = 78177          16/48/48

0001              SUBROUTINE PHIPSI(MATNM,IND,F)
            C*****BUILDS SECOND LEVEL OFF-DIAGONAL BLOCKS
            C*****PRESENT VERSION FOR THREE MACHINES
0002              IMPLICIT REAL*8(A-H,O-Z)
0003              COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IOUT,NNN,ASTER
0004              DIMENSION BC(8,8)
0005              DIMENSION F(88,88)
0006              DO 1 I=1,NNN
0007              DO 1 K=1,NNN
0008            1 F(I,K)=0.D0
0009              K=0
0010              J=1
0011              IF(MATNM.EQ.2) J=6
0012              IFLAG=0
0013            2 INDEX=IND
0014              DO 6 I=1,LIMIT
0015              IF(IFLAG.EQ.0) GO TO 3
0016              IF(MATNM.EQ.1.AND.I.EQ.1) GO TO 5
0017              IF(MATNM.EQ.2.AND.I.EQ.LIMIT) GO TO 6
0018            3 IF(I.LE.2.OR.I.EQ.LIMIT) CALL BCMAT(J,INDEX,BC)
0019              DO 4 L=1,NN
0020              DO 4 M=1,NN
0021            4 F((LIMIT-I+K)*NN+L,(LIMIT-I)*NN+M)=BC(L,M)
0022            5 IF(I.EQ.1.OR.I.EQ.(LIMIT-1)) INDEX=INDEX-1
0023            6 CONTINUE
0024              IF(IFLAG.NE.0) RETURN
0025              IFLAG=1
0026              K=1
0027              IF(MATNM.EQ.2) K=-1
0028              J=5
0029              IF(MATNM.EQ.2) J=2
0030              GO TO 2
0031              END
```

```
FORTRAN IV G1  RELEASE 2.0              PRINT                DATE = 78177           16/48/49

0001            SUBROUTINE PRINT(STATE)
          C*****OUTPUTS RESULTS WITH THREE-MACHINE FORMAT
          C*****CALCULATES AND OUTPUTS EFFICIENCY
          C*****NOTE---SLOWEST MOVING INDEX IS *LAST* STORAGE
0002            IMPLICIT REAL*8(A-H,O-Z)
0003            COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IOUT,NNN,ASTER
0004            DIMENSION STATE(3080)
0005            DATA STAR/'*'/
0006            IF(ASTER.NE.STAR) WRITE(IOUT,1)
0007          1 FORMAT(1H1,2X,'PROBABILITY DISTRIBUTION :',//   )
0008            N1=NSTOR(1)+1
0009            N2=NSTOR(2)+1
0010            EFF=0.D0
0011            AVG1=0.0D0
0012            AVG2=0.0D0
0013            DO 6 I=1,N1
0014            NN1=I-1
0015            NBEGN=NN1*8
0016            IF(ASTER.NE.STAR) WRITE(IOUT,2) NN1
0017          2 FORMAT(1H0,/,10X,'N1 = ',I5,/,3X,'000',10X,'001',10X,'010',10X,
               1  '011',10X,'100',10X,'101',10X,'110',10X,'111',21X,'N2',/)
0018            DO 6 J=1,N2
0019            NN2=J-1
0020            IF(NN2.EQ.0) GO TO 4
0021            DO 3 K=2,8,2
0022          3 EFF=EFF+STATE(NBEGN+K)
0023          4 DO 45 K=1,8
0024            AVG1=AVG1+STATE(NBEGN+K)*NN1
0025            AVG2=AVG2+STATE(NBEGN+K)*NN2
0026         45 IF(STATE(NBEGN+K).LT.1.D-12) STATE(NBEGN+K)=0.D0
0027            IF(ASTER.NE.STAR) WRITE(IOUT,5) (STATE(NBEGN+K),K=1,8),NN2
0028          5 FORMAT(1H ,8E13.5,11X,I4)
0029          6 NBEGN=NBEGN+N1*8
0030            WRITE(IOUT,7) EFF
0031          7 FORMAT(1H0,////,' LINE EFFICIENCY = ',E15.5)
0032            WRITE(IOUT,75) AVG1,AVG2
0033         75 FORMAT(1H0,'AVERAGE STORAGE FILLS  :  ',/,' STORAGE 1  :  '
               1  ,E15.5,/,' STORAGE 2  :  ',E15.5)
0034            AGG1=AVG1/NSTOR(1)
0035            AGG2=AVG2/NSTOR(2)
0036            WRITE(IOUT,8) AGG1,AGG2
0037          8 FORMAT(1H0,'AVERAGE STORAGE FILLS (FRACTION OF MAXIMUM CA'
               1,'PACITY) : ',/,' STORAGE 1  :  ',E15.5,/,' STORAGE 2  :  '
               2,E15.5)
0038            TINVRY=AVG1+AVG2
0039            WRITE(IOUT,9) TINVRY
0040          9 FORMAT(1H0,'TOTAL IN-PROCESS INVENTORY  :  ',E15.5)
0041            RETURN
0042            END
```

```
FORTRAN IV G1  RELEASE 2.0            MAIN              DATE = 78177            16/48/48

      C*****MAIN PROGRAM - THREE MACHINE TWO STORAGE CASE
      C*****HIGHEST LEVEL - COMPUTATION OF STATE VECTOR
      C*****THE PROGRAM ASSUMES THAT THERE ARE NO PATHOLOGICAL CASES
      C*****WITH STORAGE SIZES LESS THAN 2
            IMPLICIT REAL*8 (A-H,O-Z)
            REAL*8 KSINV
      C*****WARNING - DIMENSIONS MUST BE READJUSTED FOR DIFFERENT DATA
      C*****MAXIMUM DIMENSION - 3 MACHINES AND 2 STORAGES
      C*****MAXIMUM DIMENSION - STORAGES N1=10,N2=34
            COMMON P(3),R(3),NSTOR(2),N,NN,LIMIT,NST,IOUT,NNN,ASTER
            DIMENSION KSINV(88,88),PHI(88,88),PSI(88,88),STATE(3080)
            DIMENSION DUM1(88,88),DUM2(88,88),DUM3(88,88),DUM4(88)
            DIMENSION XINV(88,88,35)
            IN=5
            IOUT=6
          1 READ(IN,2,END=999) N,ASTER
          2 FORMAT(I3,A1)
      C*****ASTERISK IN COL.4 SUPPRESSES PRINTING OF PROB. DIST.
            IF(N.EQ.0) GO TO 91
      C*****N=0 - INCREMENT SECOND STORAGE
            NLAST=N
            IFLAG=0
            NN=2**N
            ISTO=N-1
          3 FORMAT(1H1,I3,' MACHINES AND ',I3,' STORAGES.',//)
            DO 310 J=1,N
        310 READ(IN,4) R(J),P(J)
          4 FORMAT(2E13.6)
            DO 320 J=1,ISTO
        320 READ(IN,7) NSTOR(J)
            WRITE(IOUT,330)
        330 FORMAT(1H0,/////////)
            WRITE(IOUT,3) N,ISTO
          7 FORMAT(I5)
            DO 5 J=1,N
            TIMUP=1.D0/P(J)
            TIMDN=1.D0/R(J)
            EFISOL=R(J)/(R(J)+P(J))
          5 WRITE(IOUT,6) J,P(J),TIMUP,R(J),TIMDN,EFISOL
          6 FORMAT(1H ,'MACHINE ',I3,'  FAILURE PROBABILITY: ',E13.6,', MEAN U
           1P-TIME:',3X,E13.6,/,14X,'REPAIR PROBABILITY:  ',E13.6,', MEAN DOWN
           2-TIME: ',E13.6,/,14X,'EFFICIENCY (IN ISOLATION): ',E13.6,/)
            NST=2**N
            DO 8 J=1,ISTO
            NST=NST*(NSTOR(J)+1)
          8 WRITE(IOUT,9) J,NSTOR(J)
          9 FORMAT(1H ,'STORAGE ',I3,' HAS MAXIMUM CAPACITY: ',I5,/)
            INDEX=3**ISTO
      C*****THERE ARE 'INDEX' COMBINATIONS OF 'LOWER BOUNDARY', 'INTERNAL',
      C*****AND 'UPPER BOUNDARY' REGIONS FOR EACH STORAGE.
            LIMIT=NSTOR(1)+1
            NNN=LIMIT*NN
            CALL KSIMAT(INDEX,KSINV)
```

```
      DO 10 I=1,NNN
      DO 10 J=1,NNN
   10 XINV(I,J,1)=KSINV(I,J)
      L1=NSTOR(2)+1
      LOOP=1
      GO TO 100
   91 READ(IN,92) NUP,NTIM
   92 FORMAT(2I5)
      WRITE(IOUT,330)
      N=NLAST
      N1ND=1
   93 INDEX=2*(3**(ISTO-1))
      IFLAG=1
      LOOP=L1-1
      L1=L1+NUP
      NSTOR(ISTO)=NSTOR(ISTO)+NUP
      NST=2**N
      DO 94 I=1,ISTO
   94 NST=NST*(NSTOR(I)+1)
      WRITE(IOUT,3) N,ISTO
      DO 95 I=1,N
      TIMUP=1.D0/P(I)
      TIMDN=1.D0/R(I)
      EFISOL=R(I)/(R(I)+P(I))
   95 WRITE(IOUT,6) I,P(I),TIMUP,R(I),TIMDN,EFISOL
      DO 96 I=1,ISTO
   96 WRITE(IOUT,9) I,NSTOR(I)
  100 LOOP=LOOP+1
      IF(LOOP.GT.L1) GO TO 20
      IF(LOOP.LE.3) CALL PHIPSI(1,INDEX,PHI)
      DO 11 I=1,NNN
      DO 11 J=1,NNN
      DUM1(I,J)=0.D0
      DO 11 K=1,NNN
   11 DUM1(I,J)=DUM1(I,J)+PHI(I,K)*XINV(K,J,LOOP-1)
      IF(LOOP.EQ.2.OR.LOOP.EQ.L1) INDEX=INDEX-(3**(ISTO-1))
      IF(LOOP.EQ.2.OR.LOOP.EQ.L1) CALL PHIPSI(2,INDEX,PSI)
      DO 12 I=1,NNN
      DO 12 J=1,NNN
      DUM2(I,J)=0.D0
      DO 12 K=1,NNN
   12 DUM2(I,J)=DUM2(I,J)+DUM1(I,K)*PSI(K,J)
C*****DUM2 IS A SPARSE MATRIX WITH SOME NONZERO ROWS
      IF(IFLAG.EQ.1) CALL KSIMAT(INDEX,KSINV)
      IF(IFLAG.EQ.1) IFLAG=2
      IF(LOOP.EQ.2.OR.LOOP.EQ.L1) CALL KSIMAT(INDEX,KSINV)
      DO 13 I=1,NNN
      DO 13 J=1,NNN
   13 XINV(I,J,LOOP)=KSINV(I,J)
      DO 19 I=1,NNN
      DO 18 J=1,NNN
      IF(DUM2(I,J).EQ.0.D0) GO TO 18
      SCAL=-1.D0
```

```
      DO 14 K=1,NNN
   14 SCAL=SCAL+DUM2(I,K)*XINV(K,I,LOOP)
      DO 15 K=1,NNN
      DUM4(K)=0.D0
      DO 15 L=1,NNN
   15 DUM4(K)=DUM4(K)+DUM2(I,L)*XINV(L,K,LOOP)
      DO 16 K=1,NNN
      DO 16 L=1,NNN
   16 DUM3(K,L)=XINV(K,I,LOOP)*DUM4(L)
      DO 17 K=1,NNN
      DO 17 L=1,NNN
   17 XINV(K,L,LOOP)=XINV(K,L,LOOP)-DUM3(K,L)/SCAL
      GO TO 19
   18 CONTINUE
   19 CONTINUE
      GO TO 100
   20 CONTINUE
C*****X'S DEFINED, NOW COMPUTING THE FIRST COLUMN OF T' INVERSE
      INDEX=3
      DO 34 I=1,L1
      IF(I.NE.1) GO TO 30
      DO 28 J=1,NNN
      DO 28 K=1,NNN
   28 DUM3(J,K)=XINV(J,K,L1)
      CALL PHIPSI(2,INDEX,PSI)
      INDEX=INDEX+(3**(ISTO-1))
      GO TO 325
   30 IF(I.EQ.3 ) CALL PHIPSI(2,INDEX,PSI)
      DO 31 J=1,NNN
      DO 31 K=1,NNN
      DUM1(J,K)=0.D0
      DO 31 L=1,NNN
   31 DUM1(J,K)=DUM1(J,K)+PSI(J,L)*DUM3(L,K)
      DO 32 J=1,NNN
      DO 32 K=1,NNN
      DUM3(J,K)=0.D0
      DO 32 L=1,NNN
   32 DUM3(J,K)=DUM3(J,K)-XINV(J,L,L1+1-I)*DUM1(L,K)
  325 DO 33 J=1,NNN
   33 STATE((I-1)*NNN+J)=DUM3(J,1)
   34 CONTINUE
C*****FIRST COLUMN OF T' INVERSE COMPUTED.
C*****APPLYING MATRIX INVERSION LEMMA FOR ROW OF ONES
      SUM=1.D0
      I=NN+1
      DO 35 J=I,NST
   35 SUM=SUM+STATE(J)
      DO 36 I=1,NST
   36 STATE(I)=STATE(I)/SUM
C*****STATE VECTOR COMPUTED.
      CALL PRINT(STATE)
      WRITE(IOUT,330)
      IF(IFLAG.EQ.0) GO TO 1
```

```
FORTRAN IV G1  RELEASE 2.0            MAIN            DATE = 78177         16/48/48

                   NIND=NIND+1
                   IF(NIND.LE.NTIM) GO TO 93
                   GO TO 1
               999 CONTINUE
                   END
```

A.5 The Transfer Line Simulator

This program uses the IBM IMSL function GGUB to generate a random number; it makes the stochastic decisions by comparing the magnitude of the random number with the predetermined failure and repair probabilities.

The input is as follows:

```
First Card:   Columns 1-2:  Number of machines (K ≤ 9)
              Columns 3-9:  Time limit on simulation run
                            (Number of cycles)

Next K Cards: Columns 1-2: Index of the machine (i)
              Columns 3-7: Probability of failure (p_i) (Format F5.3)
              Columns 8-13: Probability of repair (r_i) (Format F5.3)

Next K-1 Cards: Columns 1-2: Index of storage (i)
                Columns 3-5: Capacity of storage (N_i)

Next Card:  Columns 1-2: Option parameter
                         (0: Transient analysis
                          1: State frequency ratios
                          2: Frequencies of producing/not producing
                             for n consecutive cycles)

Next Card: (Only if Option parameter = 0)
           Columns 1-13: Steady-state efficiency (Format E13.6)
```

```
      DIMENSION NSTAT( 121, 8),MACHI(3),NSTOR( 4),MAXST( 4),
     1          FAIL(3),REPR(3),MSTOR(3),PI( 5),IND( 5),
     2          SUM( 5),SQR( 5),RAT( 5),AVG( 5)
      DIMENSION FULL(1000),EMPT(1000)
      DIMENSION FT(22),FNUM(9)
      DATA FNUM/'1','2','3','4','5','6','7','8','9'/
      DATA SLASH/' /, '/
      DATA FT/'(1H ',',2(2','X,I3',')',3X',',    ',','    ','(2X,','I4),',
     1      '2X, ','    ','    ','(2X,','I2),',',','5X,I','7,5X',',F9.','6,  ',
     2      '    ','    ','    ','(2X,','F9.6','))  '/
      LIT=1000
      IN=5
      IOUT=6
   99 READ(IN,1,END=98) N,LIMIT
    1 FORMAT(I2,I7)
      TOTAL=LIMIT
      IF(N.GT.0.AND.N.LE.9) GO TO 401
      WRITE(IOUT,400) N,LIMIT
  400 FORMAT(1H ,'INCORRECT DATA ',I2,2X,I7)
      CALL EXIT
  401 IX=767
      DO 402 I=1,LIT
      FULL(I)=0
  402 EMPT(I)=0
      FT(6)=FNUM(N-1)
      FT(10)=FNUM(N)
      IF(N.GT.3) FT(17)=SLASH
      IF(N.GT.5) GO TO 801
      FT(19)=FNUM(N2)
      GO TO 802
  801 FT(18)=FNUM(1)
      IND1=2*N-11
      FT(19)=FNUM(IND1)
  802 CONTINUE
      NN=N-1
      WRITE(IOUT,101) N,NN,LIMIT
  101 FORMAT(1H1,I2,'  MACHINES, ',I2,'  STORAGES.  TIME LIMIT :  ',I7/)
      DO 5 IND1=1,N
      READ(IN,2) I,DUM1,DUM2
    2 FORMAT(I2,2F5.3)
      WRITE(IOUT,205) I,DUM1,DUM2
  205 FORMAT(1H ,I2,2(2X,F5.3) )
      IF(0.LT.I.AND.I.LE.N) GO TO 4
      WRITE(IOUT,3) I,DUM1,DUM2
    3 FORMAT(1H ,'INCORRECT DATA  ',I2,2F5.3)
      CALL EXIT
    4 FAIL(I)=DUM1
    5 REPR(I)=DUM2
      DO 8 IND1=1,NN
      READ(IN,6) I,NDUM
    6 FORMAT(I2,I3)
      WRITE(IOUT,65) I,NDUM
   65 FORMAT(1H ,I2,I3)
```

```
FORTRAN IV G1  RELEASE 2.0                MAIN               DATE = 78192          14/39/32

0050                   IF(0.LT.I.AND.I.LE.NN) GO TO 8
0051                   WRITE(IOUT,7) I,NDUM
0052                 7 FORMAT(1H ,'INCORRECT DATA  ',I2,I3)
0053                   CALL EXIT
0054                 8 MAXST(I+1)=NDUM
0055                   MAXST(N+1)=1
0056                   READ(IN,209) IOP
0057               209 FORMAT(I2)
0058                   IF (IOP.NE.0) GO TO 84
0059                   READ(IN,210) EPIC
0060               210 FORMAT(E13.6)
0061                84 CONTINUE
0062                   WRITE(IOUT,85)
0063                85 FORMAT(1H0,/)
0064                   NSTA1=2**N
0065                   NSTA2=1
0066                   DO 9 IND1=2,N
0067                 9 NSTA2=NSTA2*(MAXST(IND1)+1)
0068                   DO 10 IND1=1,NSTA2
0069                   DO 10 IND2=1,NSTA1
0070                10 NSTAT(IND1,IND2)=0
0071                   DO 11 IND1=1,N
0072                11 MACHI(IND1)=1
0073                   LAST=1
0074                   MAXF=0
0075                   MAXP=0
0076                   MMPT=1
0077                   NPART=0
0078                   NSTEP=LIMIT/100
0079                   CALC=0.
0080                   FLAST=0.
0081                   SUMSQ=0.
0082                   STEP=NSTEP
0083                   IF(IOP.EQ.0) WRITE(IOUT,115) EPIC
0084               115 FORMAT(1H ,'  TIME:  PIECES PRODUCED:  SAMPLE AVERAGE:  CUMULATIVE
                      1 AVERAGE:  EFFICIENCY=',E13.6,/)
0085                   NSTOR(1)=1
0086                   NSTOR(N+1)=0
0087                   DO 12 IND1=2,N
0088                12 NSTOR(IND1)=1
0089                   NTIME=0
0090                13 NTIME=NTIME+1
0091                   DO 15 IND1=1,N
0092                   IF(MACHI(IND1).EQ.1) GO TO 14
0093                   CALL GGUB(IX,1,RNUM)
0094                   IF(RNUM.GT.REPR(IND1)) GO TO 15
0095                   MACHI(IND1)=1
0096                   GO TO 15
0097                14 IF(NSTOR(IND1).EQ.0) GO TO 15
0098                   IF(NSTOR(IND1+1).EQ.MAXST(IND1+1)) GO TO 15
0099                   CALL GGUB(IX,1,RNUM)
0100                   IF(RNUM.GT.FAIL(IND1)) GO TO 15
0101                   MACHI(IND1)=0
```

```
FORTRAN IV G1   RELEASE 2.0              MAIN              DATE = 78192          14/39/02

      15 CONTINUE
         DO 20 IND1=2,N
         IF(NSTOR(IND1).NE.0) GO TO 16
         IF(NSTOR(IND1-1).EQ.0) GO TO 20
         MSTOR(IND1)=MACHI(IND1-1)
         GO TO 20
      16  IF(NSTOR(IND1).NE.MAXST(IND1)) GO TO 17
         IF(NSTOR(IND1+1).EQ.MAXST(IND1+1)) GO TO 20
         MSTOR(IND1)=NSTOR(IND1)-MACHI(IND1)
         GO TO 20
      17 IF(NSTOR(IND1-1).EQ.0) GO TO 19
         IF(NSTOR(IND1+1).EQ.MAXST(IND1+1)) GO TO 18
         MSTOR(IND1)=NSTOR(IND1)-MACHI(IND1)+MACHI(IND1-1)
         GO TO 20
      18 MSTOR(IND1)=NSTOR(IND1)+MACHI(IND1-1)
         GO TO 20
      19 IF(NSTOR(IND1+1).EQ.MAXST(IND1+1)) GO TO 20
         MSTOR(IND1)=NSTOR(IND1)-MACHI(IND1)
      20 CONTINUE
         IF(LAST.EQ.1) GO TO 204
         IF(NSTOR(N).NE.0.AND.MACHI(N).EQ.1) GO TO 201
         MMPT=MMPT+1
         GO TO 208
     201 IF(MMPT.LE.LTT) GO TO 203
         WRITE(IOUT,202) NTIME,LTT
     202 FORMAT(1H ,'RUN STOPPED AT TIME = ',I7,/' SYSTEM DID NOT PRODUCE F
        1OR LONGER THAN ',I7)
         GO TO 235
     203 IF(MMPT.GT.MAXP) MAXP=MMPT
         EMPT(MMPT)=EMPT(MMPT)+1
         LAST=1
         MMPT=1
         NPART=NPART+1
         GO TO 208
     204 IF(NSTOR(N).EQ.0.OR.MACHI(N).EQ.0) GO TO 2045
         MMPT=MMPT+1
         NPART=NPART+1
         GO TO 208
    2045 IF(MMPT.LE.LTT) GO TO 207
         WRITE(IOUT,206) NTIME,LTT
     206 FORMAT(1H ,'RUN STOPPED AT TIME = ',I7,/' SYSTEM PRODUCED FOR LONG
        1ER THAN ',I7)
         GO TO 235
     207 IF(MMPT.GT.MAXF) MAXF=MMPT
         FULL(MMPT)=FULL(MMPT)+1
         LAST=0
         MMPT=1
     208 CONTINUE
         DO 21 IND1=2,N
      21 NSTOR(IND1)=MSTOR(IND1)
         IND1=1
         IF(NN.LT.2) GO TO 225
         DO 22 IND3=2,NN
```

```
   22 IND1=IND1+(NSTOR(IND3)*(MAXST(IND3+1)+1))
  225 IND1=IND1+NSTOR(N)
      IND2=1
      DO 23 IND3=1,N
   23 IND2=IND2+((2**(N-IND3))*MACHI(IND3))
      NSTAT(IND1,IND2)=NSTAT(IND1,IND2)+1
      IF(IOP.NE.0) GO TO 234
      IF(((NTIME/NSTEP)*NSTEP).NE.NTIME) GO TO 234
      CALC=CALC+1.
      DUM1=NPART
      DUM2=NTIME
      DUM2=DUM1/DUM2
      DUM1=DUM1-FLAST
      FLAST=NPART
      DUM1=DUM1/STEP
      SUMSQ=SUMSQ+(DUM1**2)
      WRITE(IOUT,233) NTIME,NPART,DUM1,DUM2
  233 FORMAT(1H ,I7,10X,I7,4X,E13.6,6X,E13.6)
  234 CONTINUE
      IF(NTIME.LT.LIMIT) GO TO 13
      IF(IOP.NE.0) GO TO 235
      DUM1=(SUMSQ/CALC)-(DUM2**2)
      DUM2=(SUMSQ/CALC)-(EFIC**2)
      WRITE(IOUT,2345) CALC,DUM1,DUM2
 2345 FORMAT(1H0,/' VARIANCE OF PRODUCTION RATE (',F5.0,' SAMPLES)',/,
     1 ' WITH EXPECTED VALUE=CUMULATIVE AVERAGE :',E13.6,/,' WITH EXPEC
     2TED VALUE=ANALYTICAL EFFICIENCY :',E13.6,//)
  235 CONTINUE
      WRITE(IOUT,85)
      NNN=(2*N)-1
      DO 901 IND1=1,NN
  901 IND(IND1)=1
      DO 902 IND1=1,N
  902 IND(NN+IND1)=2**(N-IND1)
      IND3=N-2
      IF(IND3.EQ.0) GO TO 9035
      DO 903 IND1=1,IND3
      DO 903 IND2=IND1,IND3
  903 IND(IND1)=IND(IND1)*(MAXST(IND2+2)+1)
 9035 DO 904 I=1,NNN
      FI(I)=0
      SUM(I)=0
  904 SQR(I)=0
      DO 28 IND1=1,NSTA2
      DO 28 IND2=1,NSTA1
      NDUM=IND1-1
      IF(NN.LT.2) GO TO 255
      DO 25 IND3=2,NN
      NSTOR(IND3)=NDUM/(MAXST(IND3+1)+1)
   25 NDUM=NDUM-((MAXST(IND3+1)+1)*NSTOR(IND3))
  255 NSTOR(N)=NDUM
      NDUM=IND2-1
      DO 26 IND3=1,N
```

```
                MACHI(IND3)=NDUM/(2**(N-IND3))
             26 NDUM=NDUM-(MACHI(IND3)*(2**(N-IND3)))
                STATE=NSTAT(IND1,IND2)
                PROB=STATE/TOTAL
                DO 905 I=1,NNN
            905 RAT(I)=0
                DO 906 I=2,N
                IF(NSTOR(I).LE.1.OR.NSTOR(I).GE.(MAXST(I)-1)) GO TO 911
            906 CONTINUE
                DO 910 I=1,NNN
                DUM1=NSTAT(IND1,IND2)
                IF(I.LE.NN) GO TO 907
                IF(MACHI(I-NN).EQ.0) GO TO 910
                DUM2=NSTAT(IND1,IND2-IND(I))
                IF(DUM2.EQ.0.) GO TO 910
                RAT(I)=DUM1/DUM2
                SUM(I)=SUM(I)+RAT(I)
                SQR(I)=SQR(I)+(RAT(I)**2)
                FI(I)=FI(I)+1
                GO TO 910
            907 IF(NSTOR(I+1).EQ.2) GO TO 910
                N3=NSTOR(I+1)-1
                DO 950 IND3=2,N3
                N4=(N3-IND3+1)*IND(I)
                DUM2=NSTAT(IND1-N4,IND2)
                IF(DUM2.EQ.0.) GO TO 950
                DUM3=NSTOR(I+1)-IND3
                DUM3=1./DUM3
                RAT(I)=(DUM1/DUM2)**DUM3
                SUM(I)=SUM(I)+RAT(I)
                SQR(I)=SQR(I)+(RAT(I)**2)
                FI(I)=FI(I)+1
            950 CONTINUE
            910 CONTINUE
            911 WRITE(IOUT,FT) IND1,IND2,(NSTOR(I),I=2,N),(MACHI(I),I=1,N),
              1  NSTAT(IND1,IND2),PROB,(RAT(I),I=1,NNN)
             28 CONTINUE
                IF(IOP.NE.1) GO TO 8124
                DO 282 I=1,NN
                IF(FI(I).NE.0.) GO TO 282
                WRITE(IOUT,281)
            281 FORMAT(1H0,'LESS THAN 2 FULLY INTERNAL STORAGE STATES.')
                GO TO 8125
            282 CONTINUE
                DO 914 I=1,NNN
                AVG(I)=SUM(I)/FI(I)
                VAR=(SQR(I)/FI(I))-(AVG(I)**2)
                IF(I.GT.NN) GO TO 912
                WRITE(IOUT,29) I,FI(I),AVG(I),VAR
             29 FORMAT(1H ,'  STORAGE ',I2,'  (',F8.0,' POINTS)    AVERAGE : ',
              1  F9.6,'  VARIANCE : ',F9.6)
                GO TO 914
            912 IND1=I-NN
```

```
      WRITE(IOUT,913) IND1,FI(I),AVG(I),VAR
  913 FORMAT(1H ,' MACHINE ',I2,' (',F8.0,' POINTS)    AVERAGE : ',
     1 F9.6,' VARIANCE : ',F9.6)
  914 CONTINUE
      N1=3
      DO 805 IND1=3,N
      IND3=2
      DO 804 IND2=IND1,N
  804 IND3=IND3*(MAXST(IND2)+1)
  805 N1=N1+IND3
      N2=NSTA2-N1+1
      FI(1)=0
      SUM(1)=0
      SQR(1)=0
      DO 811 IND1=N1,N2
      DO 811 IND2=1,NSTA1
      DUM1=NSTAT(IND1,IND2)
      DUM1=DUM1/TOTAL
      NDUM=IND1-1
      IF(NN.LT.2) GO TO 807
      DO 806 IND3=2,NN
      NSTOR(IND3)=NDUM/(MAXST(IND3+1)+1)
  806 NDUM=NDUM-((MAXST(IND3+1)+1)*NSTOR(IND3))
  807 NSTOR(N)=NDUM
      NDUM=IND2-1
      DO 808 IND3=1,N
      MACHI(IND3)=NDUM/(2**(N-IND3))
  808 NDUM=NDUM-(MACHI(IND3)*(2**(N-IND3)))
      DUM3=1
      DO 809 IND3=2,N
      DUM2=AVG(IND3-1)**NSTOR(IND3)
  809 DUM3=DUM3*DUM2
      DO 810 IND3=1,N
      DUM2=AVG(IND3+NN)**MACHI(IND3)
  810 DUM3=DUM3*DUM2
      DUM2=DUM1/DUM3
      SUM(1)=SUM(1)+DUM2
      SQR(1)=SQR(1)+(DUM2**2)
  811 FI(1)=FI(1)+1
      DUM1=SUM(1)/FI(1)
      DUM2=(SQR(1)/FI(1))-(DUM1**2)
      WRITE(IOUT,812) FI(1),DUM1,DUM2
  812 FORMAT(1H ,'NORMALIZING CONSTANT (',F8.0,' POINTS)  AVERAGE : ',
     1   E12.6,'   VARIANCE : ',F9.6)
 8124 IF(IOP.NE.2) GO TO 99
 8125 SSM=0
      EFF=0
      DO 813 I=1,MAXF
      EFF=EFF+(I*FULL(I))
  813 SSM=SSM+FULL(I)
      DO 814 I=1,MAXF
  814 FULL(I)=FULL(I)/SSM
      SSM=0
```

```
0306                DO 815 I=1,MAXP
0307            815 SSM=SSM+EMPT(I)
0308                DO 816 I=1,MAXP
0309            816 EMPT(I)=EMPT(I)/SSM
0310                WRITE(IOUT,817)
0311            817 FORMAT(1H ,'PROBABILITY OF PRODUCING N PIECES CONSECUTIVELY :')
0312                DO 818 I=1,MAXP
0313            818 WRITE(IOUT,819) I,FULL(I)
0314            819 FORMAT(1H ,I7,3X,E13.6)
0315                WRITE(IOUT,820)
0316            820 FORMAT(1H ,'PROBABILITY OF NOT PRODUCING FOR N CONSECUTIVE TIME ST
               1EPS :')
0317                DO 821 I=1,MAXP
0318            821 WRITE(IOUT,819) I,EMPT(I)
0319                FLOG=EFF/TOTAL
0320                IND1=EFF
0321                WRITE(IOUT,822) IND1,LIMIT,FLOG
0322            822 FORMAT(1H ,'THE SYSTEM PRODUCED ',I7,' PIECES IN ',I7,' TIME STEPS
               1.   EFFICIENCY = ',E13.6)
0323                GO TO 99
0324             98 CONTINUE
0325                END
```

12. BIBLIOGRAPHY AND REFERENCES

Aleksandrov, A.M.[1968], "Output Flows of a Class of Queueing Systems", Tekhnicheskaya Kibernetika, 6, 4 (In Russian).(Eng.tr. Engineering Cybernetics, 6,4,1-8 (1968)).

Anderson, D.R.[1968], "Transient and Steady-State Minimum Cost In-Process Inventory Capacities for Production Lines", unpublished Ph.D. Thesis, Department of Industrial Engineering, Purdue University.

Anderson, D.R. and Moodie, C.L.[1969], "Optimal Buffer Storage Capacity in Production Line Systems", Int. J. Prod. Res., 7,3,233-240.

Artamonov, G.T.[1976], "Productivity of a Two-Instrument Discrete Processing Line in the Presence of Failures", Kibernetika, 3,126-130 (In Russian).(Eng.tr. Cybernetics, 12,3,464-468 (1977)).

Avi-Itzhak, B. and Naor, P.[1963], "Some Queueing Problems with the Service Station Subject to Breakdown", Op. Res., 11,3,303-320.

Avi-Itzhak, B. and Yadin, M.[1965], "A Sequence of Two Servers with No Intermediate Queue", Manag. Sci., 11,5,553-564.

Bagchi, T.P. and Templeton, J.G.C.[1972], Numerical Methods in Markov Chains and Bulk Queues, Lecture Notes in Economical and Mathematical Systems 72, Springer-Verlag.

Barlow, R.E. and Proschan, F.[1975], Statistical Theory of Reliability and Life Testing: Probability Models, Holt, Rinehart and Winston.

Barten, K.[1962], "A Queueing Simulator for Determining Optimum Inventory Levels in a Sequential Process", J. Ind. Eng., 13,4,247-252.

Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G.[1975], "Open, Closed and Mized Networks of Queues with Different Classes of Customers", J. ACM, 22,2,248-260.

Benson, F. and Cox, D.R.[1951], "The Productivity of Machines Requiring Attention at Random Intervals", J. Royal Stat. Soc. B, 13,65-82.

Boyce, W.E. and DiPrima, R.C.[1969], Elementary Differential Equations and Boundary Value Problems, John Wiley.

Brockmeyer, E., Halstrøm, H.L., and Jensen, A.[1960], The Life and Works of A.K. Erlang, Acta Polytechnica Scandinavia No. 287.

Burke, P.J.[1956], "The Output of a Queueing System", Op. Res., 4,699-704.

Burke, P.J.[1972], "Output Processes and Tandem Queues", in J.Fox ed., Proceedings of the Symposium on Computer-Communication Networks and Teletraffic (New York, April 4-6, 1972), Microwave Research Institute Symposia Series XXII, Polytechnic Press.

Buxey, G.M., Slack, N.D. and Wild, R.[1973], "Production Flow Line System Design - A Review", AIIE Trans., 5,1,37-48.

Buzacott, J.A.[1967a], "Markov Chain Analysis of Automatic Transfer Line with Buffer Stock", unpublished Ph.D. Thesis, Department of Engineering Production, University of Birmingham.

Buzacott, J.A.[1967b], "Automatic Transfer Lines with Buffer Stocks", Int. J. Prod. Res., 5,3,183-200.

Buzacott, J.A.[1968], "Prediction of the Efficiency of Production Systems without Internal Storage", Int. J. Prod. Res., 6,3,173-188.

Buzacott, J.A.[1969], "Methods of Reliability Analysis of Production Systems Subject to Breakdowns", in D.Grouchko ed., Operations Research and Reliability, Proc. of a NATO Conf. (Turin, Italy, June 24 - July 4, 1969), 211-232.

Buzacott, J.A.[1971], "The Role of Inventory Banks in Flow-Line Production Systems", Int. J. Prod. Res., 9,4,425-436.

Buzacott, J.A.[1972], "The Effect of Station Breakdowns and Random Processing Times on the Capacity of Flow Lines with In-Process Storage", AIIE Trans., 4,4,308-312.

Buzacott, J.A.[1976], "The Production Capacity of Job Shops with Limited Storage Space", Int. J. Prod. Res., 14,5,597-605.

Buzen, J.P.[1971], "Queueing Network Models of Multiprogramming", unpublished Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University.

Buzen, J.P.[1973], "Computational Algorithms for Closed Queueing Networks with Exponential Servers", Comm. ACM, 16,9,527-231.

Chandy, K.M.[1972], "The Analysis and Solutions for General Queueing Networks", Proc. of the Sixth Ann. Princeton Conf. on Information Sciences and Systems (March 23-24, 1972), 224-228.

Chandy, K.M., Herzog, U., and Woo, L.[1975a], "Parametric Analysis of Queueing Networks", IBM J. Res. Dev., 19,1,36-42.

Chandy, K.M., Herzog, U., and Woo, L.[1975b], "Approximate Analysis of General Queueing Networks", IBM J. Res. Dev., 19,1,43-49.

Chang, W.[1963], "Output Distribution of a Single-Channel Queue", Op. Res., 11,4,620-623.

Chu, W.W.[1970], "Buffer Behavior for Poisson Arrivals and Multiple Synchronous Constant Outputs", IEEE Trans. Computers, C-19,6,530-534.

Çinlar, E. and Disney, R.L.[1967], "Stream of Overflows from a Finite Queue", Op. Res., 15,1,131-134.

Cook, N.H., Gershwin, S.B., Kanellakis, P.C., Schick, I.C., and Ward, J.E. [1977], "Complex Materials Handling and Assembly Systems, First Interim Progress Report", Report ESL-IR-740, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Cox, D.R. and Smith, W.L.[1974], Queues, Chapman and Hall.

Denning, P.J. and Buzen, J.P.[1977], "Operational Analysis of Queueing Networks", Preprint of the Third International Symposium of Modeling and Performance Evaluation of Computer Systems (Bonn, W.Germany, October 3 - 5, 1977).

Disney, R.L.[1962], "Some Multichannel Queueing Problems with Ordered Entry", J. Ind. Eng., 13,1,46-48.

Disney, R.L.[1963], "Some Results of Multichannel Queueing Problems with Ordered Entry - an Application to Conveyor Theory", J. Ind. Eng., 14,2,105-108.

Disney, R.L.[1972], "A Matrix Solution for the Two Server Queue with Overflow", Manag. Sci., 19,3,254-265.

Duff, I.S.[1976], "A Survey of Sparse Matrix Research", Computer Science and Systems Division, Report CSS 28, AERE Harwell.

Eilon, S. and Lampkin, W.[1968], Inventory Control Abstracts (1953-1965), Oliver and Boyd.

Elmaghraby, S.E.[1966], The Design of Production Systems, Reinhold.

Erpsher, Y.B.[1952], "Losses of Working Time and Division of Automatic Lines into Sections", Stanki i Instrument, 23,7,7-16 (In Russian). (Eng.tr. DSIR CTS 631, CTS 634).

Esary, J.D., Marshall, A.W., and Proschan, F.[1969], "Determining the Approximate Constant Failure Rate for a System Whose Components Have Constant Failure Rates", in D. Grouchko ed., Operations Research and Reliability, Proc. of a NATO Conf. (Turin, Italy, June 24 - July 4, 1969).

Evans, R.V.[1967], "Geometric Distribution in Some Two-Dimensional Queueing Systems", Op. Res., 15, 830-846.

Feller, W.[1966], An Introduction to Probability Theory and its Applications, vol. II, John Wiley.

Finch, P.D.[1959], "Cyclic Queues with Feedback", J. Royal Stat. Soc. B, 21,153-157.

Freeman, M.C.[1964], "The Effects of Breakdowns and Interstage Storage on Production Line Capacity", J. Ind. Eng., 15,4,194-200.

Galliher, H.P.[1962], "Queueing", in Notes on Operations Research, Operations Research Center, Massachusetts Institute of Technology, M.I.T. Press.

Gaver, D.P.[1954], "The Influence of Service Times in Queueing Processes", J. Op. Res. Soc. America, 2,2,139-149.

Gelenbe, E. and Muntz, R.R.[1976], "Probabilistic Models of Computer Systems - Part I (Exact Results)", Acta Informatica, 7,35-60.

Gershwin, S.B.[1973a], "Reliability and Storage Size, Part I", unpublished memorandum No. FM 44700-107A, The Charles Stark Draper Laboratory.

Gershwin, S.B.[1973b], "Reliability and Storage Size, Part II", unpublished memorandum No. FM 44700-110, The Charles Stark Draper Laboratory.

Gershwin, S.B. and Berman, O.[1978], "Analysis of Transfer Lines Consisting of Two Unreliable Stages with Random Processing Times and Finite Storage Buffers", "Complex Materials Handling and Assembly Systems, Volume VII", Report ESL-FR-834-7, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Gershwin, S.B. and Schick, I.C.[1977], "Reliability and Storage Size", in Athans et.al., "Complex Materials Handling and Assembly Systems, Second Interim Progress Report", Report ESL-IR-771, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Gershwin, S.B. and Schick, I.C.[1978], "Analysis of Transfer Lines Consisting of Three Unreliable Stages and Finite Storage Buffers", "Complex Materials Handling and Assembly Systems, Volume IX", Report ESL-FR-834-9, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Giammo, T.[1976], "Validation of a Computer Performance Model of the Exponential Queueing Network Family", Acta Informatica, 7,137-152.

Goff, R.E. [1970], "An Investigation of the Effect of Interstage Storage on Production Line Efficiency", unpublished M.S. Thesis, Department of Industrial Engineering, Texas Tech University.

Golub, G.H. [1969], "Matrix Decomposition and Statistical Calculations", Technical Report CS 124, Computer Science Department, Stanford University.

Golub, G.H. and Kahan, W. [1965], "Calculating the Singular Values and Pseudo-Inverse of a Matrix", J. SIAM Numerical Analysis B, 2,2, 205-224.

Goode, H.P. and Saltzman, S. [1962], "Estimating Inventory Limits in a Station Grouped Production Line", J. Ind. Eng., 13,6,484-490.

Gordon, W.J. and Newell, G.F. [1967a], "Closed Queueing Systems with Exponential Servers", Op. Res.,15,2,254-265.

Gordon, W.J. and Newell, G.F. [1967b], "Cyclic Queueing Systems with Restricted Length Queues", Op. Res., 15,2,266-277.

Gordon-Clark, M.R. [1977], Private Communication.

Graves, S.G. [1977], Private Communication.

Grenander, U. and Szegö, G. [1958], Toeplitz Forms and their Applications, University of California.

Groover, M.P. [1975], "Analyzing Automatic Transfer Machines", Ind. Eng., 7,11,26-31.

Hanifin, L.E., Liberty, S.G., and Taraman, K. [1975], "Improved Transfer Line Efficiency Utilizing Systems Simulation", Technical Paper MR75-169, Society of Manufacturing Engineers.

Happel, J. [1967], Chemical Process Economics, John Wiley.

Hatcher, J.M. [1969], "The Effect of Internal Storage on the Production Rate of a Series of Stages Having Exponential Service Times", AIIE Trans., 1,2,150-156.

Haydon, B.J. [1972], "System States for a Series of Finite Queues", Op. Res., 20,6,1137-1141.

Herzog, U., Woo, R., and Chandy, K.M. [1974], "Solution of Queueing Problems by a Recursive Technique", IBM Res. Rep. RC-4957.

Hildebrand, D.K. [1967], "Stability of Finite Queue, Tandem Server Systems", J. Appl. Prob.,4,571-583.

Hildebrand, D.K.[1968], "On the Capacity of Tandem Server, Finite Queue, Service Systems", Op.Res., 16,72-82.

Hillier, F.S. and Boling, R.W.[1966], "The Effect of Some Design Factors on the Efficiency of Production Lines with Variable Operation Times", J. Ind. Eng., 17,12,651-658.

Hillier, F.S. and Boling, R.W.[1967], "Finite Queues in Series with Exponential or Erlang Service Times - a Numerical Approach", Op. Res., 15,2,286-303.

Hindmarsh, A.C.[1977], "Solution of Block-Tridiagonal Systems of Linear Algebraic Equations", Lawrence Livermore Laboratories.

Householder, A.S.[1965], The Theory of Matrices in Numerical Analysis, Blaisdell.

Howard, R.A.[1971], Dynamic Probabilistic Systems: volume I, Markov Models, John Wiley.

Hunt, G.C.[1956], "Sequential Arrays of Waiting Lines", Op. Res., 4,6,674-683.

IBM Virtual Machine Facility/370 : Introduction, Order No. GC20-1800.

Ignall, E. and Silver, A.[1977], "The Output of a Two-Stage System with Unreliable Machines and Limited Storage", AIIE Trans., 9,2,183-188.

Jackson, J.R.[1963], "Job-Shop-Like Queueing Systems", Manag. Sci., 10,1,131-142.

Jackson, R.R.P.[1954], "Queueing Systems with Phase Type Service", Op. Res. Quart., 5,4,109-120.

Jackson, R.R.P.[1956], "Random Queueing Processes with Phase Type Service", J. Royal Stat. Soc. B, 18,129-132.

Karlin, S.[1968], A First Course in Stochastic Processes, Academic Press.

Kay, E.[1972], "Buffer Stocks in Automatic Transfer Lines", Int. J. Prod. Res., 10,2,155-165.

Kemeny, J.G. and Snell, J.L.[1967], Finite Markov Chains, Van Nostrand.

Kimemia, J. and Gershwin, S.B.[1978], "Network Flow Optimization of a Flexible Manufacturing System", "Complex Materials Handling and Assembly Systems, Volume II", Report ESL -FR-834-2, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Knott, A.D.[1970a], "The Inefficiency of a Series of Work Stations - a Simple Formula", Int. J. Prod. Res., 8,2,109-119.

Knott, A.D.[1970b], "The Effect of Internal Storage on the Production Rate of a Series of Stages Having Exponential Service Times", AIIE Trans. (letters), 2,3,273.

Koenigsberg, E.[1958], "Cyclic Queues", Op. Res. Quart., 9,1,22-35.

Koenigsberg, E.[1959], "Production Lines and Internal Storage - a Review", Manag. Sci., 5,410-433.

Konheim, A.G. and Reiser, H.[1974], "A Queueing Model with Finite Waitingroom and Blocking", IBM Res. Rep. RC-5066. (Also J. ACM, 23,2,328-341 (1976)).

Kraemer, I.A. and Love, R.F.[1970], "A Model for Optimizing the Buffer Inventory Storage Size in a Sequential Production System", AIIE Trans., 2,1,64-69.

Kwo, T.T.[1958], "A Theory of Conveyors", Manag. Sci., 5,1,51-71.

Lam, S.S.[1977], "Queueing Networks with Population Size Constraints", IBM J. Res. Dev., 21,4,370-378.

Lavenberg, S.S.[1975], "Stability and Maximum Departure Rate of Certain Open Queueing Networks Having Finite Capacity Constraints", IBM Res. Rep. RJ-1625.

Lavenberg, S.S., Traiger, I.L., and Chang, A.[1973], "Bounds on Work-Rates for Closed Queueing Networks with Non-Exponential Stages", IBM Res. Rep. RJ-1182.

Love, R.F.[1967], "A Two-Station Stochastic Inventory Model with Exact Methods of Computing Optimal Policies", Naval Res. Logistics Quart., 14,2,185-217.

Masso, J.A.[1973], "Stochastic Lines and Inter-Stage Storages", unpublished M.S. Thesis, Department of Industrial Engineering, Texas Tech University.

Masso, J. and Smith, M.L.[1974], "Interstage Storages for Three Stage Lines Subject to Stochastic Failures", AIIE Trans., 6,4,354-358.

Moore, F.R.[1972], "Computational Model of a Closed Queueing Network with Exponential Servers", IBM J. Res. Dev., 16,6,567-572.

Moran, P.A.P.[1961], The Theory of Storage, Methuen.

Morse, P.M.[1965], Queues, Inventories and Maintenance, John Wiley.

Muth, E.J.[1973], "The Production Rate of a Series of Work Stations with Variable Service Times", Int. J. Prod. Res., 11,2,155-169.

Navon, J.M. [1977], "Algorithms for Solving Scalar and Block-Cyclic Tridiagonal Systems of Linear Algebraic Equations", National Research Institute for Mathematical Sciences, CSIR Special Report WISK 265.

Neuts, M.F. [1968], "Two Queues in Series with a Finite, Intermediate Waitingroom", J. Appl. Prob., 5,123-142.

Neuts, M.F. [1970], "Two Servers in Series, Studied in Terms of a Markov Renewal Branching Process", Adv. Appl. Prob., 2,110-149.

Newell, G.F. [1971], Applications of Queueing Theory, Chapman and Hall.

Okamura, K. and Yamashina, H. [1977], "Analysis of the Effect of Buffer Storage Capacity in Transfer Line Systems", AIIE Trans., 9,2,127-135.

Patterson, R.L. [1964], "Markov Processes Occurring in the Theory of Traffic Flow Through an N-Stage Stochastic Service System", J. Ind. Eng., 15,4,188-193.

Pritsker, A.A.B. [1966], "Application of Multichannel Queueing Results to the Analysis of Conveyor Systems", J. Ind. Eng., 17,1,14-21.

Purdue, P. [1972], "On the Use of Analytic Matric Functions in Queueing Theory", unpublished Ph.D. Thesis, Department of Statistics, Purdue University.

Rao, N.P. [1975a], "On the Mean Production Rate of a Two-Stage Production System of the Tandem Type", Int. J. Prod. Res., 13,2,207-217.

Rao, N.P. [1975b], "Two-Stage Production Systems with Intermediate Storage", AIIE Trans., 7,4,414-421.

Reiser, M. and Konheim, A.G. [1974], "Blocking in a Queueing Network with Two Exponential Servers", IBM Res. Rep. RJ-1360.

Richman, E. and Elmaghraby, S. [1957], "The Design of In-Process Storage Facilities", J. Ind. Eng., 8,1,7-9.

Sandler, G.H. [1963], System Reliability Engineering, Prentice-Hall.

Sarma, V.V.S. and Alam, M. [1975], "Optimal Maintenance Policies for Machines Subject to Deterioration and Intermittent Breakdowns", IEEE Trans. Systems, Man and Cybernetics, SMC-4, 396-398.

Schweitzer, P.J. [1976], "Maximum Throughput in Finite Capacity Open Queueing Networks", IBM Res. Rep. RC-5996.

Schweitzer, P.J. and Sam, S.S. [1976], "Buffer Overflow in a Store-and-Forward Network Node", IBM J. Res. Dev., 20,6,542-550.

Secco-Suardo, G.M.[1978], "Flow Optimization in a Closed Network of Queues", "Complex Materials Handling and Assembly Systems, Volume III", Report ESL-FR-834-3, Electronic Systems Laboratory, Massachusetts Institute of Technology.

Sevast'yanov, B.A.[1962], "Influence of Storage Bin Capacity on the Average Standstill Time of a Production Line", Teoriya Veroyatnostey i ee Primeneniya, 7,4,438-447 In Russian). (Eng.tr. Theory of Probability and its Applications, 7,4,429-438 (1962)).

Shedler, G.S.[1971], "A Queueing Model of a Two Level Storage System", IBM Res. Rep. RJ-802.

Shedler, G.S.[1973], "A Queueing Model of a Multiprogrammed Computer with a Two-Level Storage System", Comm. ACM, 16,1,3-10.

Sheskin, T.J.[1974], "Allocation of Interstage Storage Along an Automatic Transfer Production Line with Discrete Flow", unpublished Ph.D. Thesis, Department of Industrial and Management Systems Engineering, Pennsylvania State University.

Sheskin, T.J.[1976], "Allocation of Interstage Storage Along an Automatic Production Line", AIIE Trans., 8,1,146-152.

Simpson, K.F.Jr.[1958], "In-Process Inventories", Op.Res., 6,6,863-873.

Solberg, J.J.[1977], "A Mathematical Model of Computerized Manufacturing Systems", Conference on Optimal Planning of Computerized Manufacturing Systems Project (Purdue University, November 21 - 22, 1977)

Soyster, A.L. and Toof, D.I.[1976], "Some Comparative and Design Aspects of Fixed Cycle Production Systems", Naval Res. Logistics Quart., 23,3,437-454.

Stover, A.M.[1956], "Application of Queueing Theory to Operation and Expansion of a Chemical Plant", unpublished manuscript, Naugatuck Chemical Company (November 2, 1956).

Suzuki, T.[1964], "On a Tandem Queue with Blocking", J. Op. Res. Soc. Japan, 6,137-257.

Taylor, J. and Jackson, R.R.P.[1954], "An Application of the Birth and Death Process to the Provision of Spare Machines", Op. Res. Quart., 5,4,95-108.

Temperton, C.[1975], "Algorithms for the Solution of Cyclic Tridiagonal Systems", J. Comp. Physics, 19,317-323.

Tewarson, R.P.[1973], Sparse Matrices, Academic Press.

Tobey, R. et.al.[1969], "PL/I - FORMAC Symbolic Mathematics Interpreter", IBM Contributed Program Library, 360D-03.3.004.

Trufyn, N.[n.d.], "Guide to FORMAC", Institute of Computer Science, University of Toronto.

Varah, J.M.[1972], "On the Solution of Block-Tridiagonal Systems Arising from Certain Finite-Difference Equations", Math. Comp., 26,120,859-868.

Vladzievskii, A.P.[1952], "Probabilistic Law of Operation and Internal Storage of Automatic Lines", Avtomatika i Telemekhanika, 13,3,227-281 (In Russian).

Vladzievskii, A.P.[1953], "Losses of Working Time and the Division of Automatic Lines into Sections", Stanki i Instrument, 24,10 (In Russian).(Eng.tr. DSIR CTS 632).

Wallace, V.L.[1969], "The Solution of Quasi Birth and Death Processes Arising from Multiple Access Computer Systems", unpublished Ph.D. Thesis, Department of Computer Science, University of Michigan.

Wallace, V.L.[1972], "Toward an Algebraic Theory of Markovian Networks", in J. Fox ed., Proceedings of the Symposium on Computer-Communication Networks and Teletraffic (New York, April 4-6, 1972), Microwave Research Institute Symposia Series XXII, Polytechnic Press.

Wallace, V.L.[1973], "Algebraic Techniques for Numerical Solution of Queueing Networks", in A.B. Clarke ed., Mathematical Methods in Queueing Theory, Proc. of a Conf. at Western Michigan University (May 10-12, 1973), Lecture Notes in Economics and Mathematical Systems 98, Springer-Verlag.

Wallace, V.L. and Rosenberg, R.S.[1966], "Markovian Models and Numerical Analysis of Computer System Behavior", AFIPS Conf. Proc., 28,141-148.

Wyner, A.D.[1974], "On the Probability of Buffer Overflow under an Arbitrary Bounded Input-Output Distribution", SIAM J. Appl. Math., 27,4,544-570.

Young, H.H.[1967], "Optimization Models for Production Lines", J. Ind. Eng., 18,1,70-78.