(1)

AD-A254 893

REPORT DOCUMEN[illegible]

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT 1992	COVERED DISSERTATION
4. TITLE AND SUBTITLE Customer-Driven Reliability Models For Multistate Coherent Systems		5. FUNDING NUMBERS
6. AUTHOR(S) Ralph A. Boedigheimer, Captain		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Student Attending: University of Oklahoma		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/CI/CIA-92-013D
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFIT/CI Wright-Patterson AFB OH 45433-6583		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release IAW 190-1 Distributed Unlimited ERNEST A. HAYGOOD, Captain, USAF Executive Officer		12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words)		
14. SUBJECT TERMS		15. NUMBER OF PAGES 279
		16. PRICE CODE

012 92-23920

292 pp

92 8 28 023

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CUSTOMER-DRIVEN RELIABILITY MODELS FOR MULTISTATE COHERENT SYSTEMS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF PHILOSOPHY

By

RALPH ALAN BOEDIGHEIMER

Norman, Oklahoma

1992

CUSTOMER-DRIVEN RELIABILITY MODELS FOR MULTISTATE COHERENT SYSTEMS

A DISSERTATION

APPROVED FOR THE SCHOOL OF INDUSTRIAL ENGINEERING

BY

[illegible]

Dr. Kailash C. Kapur (Chairman)

[illegible]

Dr. A. Ravindran

Lawrence Leemis

Dr. Lawrence M. Leemis

[illegible]

Dr. Pakize S. Pulat

[illegible]

Dr. Robert E. Schlegel

[illegible]

Dr. Kevin A. Grasse

ACKNOWLEDGEMENTS

I thank God for the many precious gifts He has given me and dedicate this work to the glory of His name.

I would like to express my sincere appreciation to Dr. Kailash Kapur for inspiring and motivating me throughout this effort. This dissertation would have been impossible without his willingness to take over as the chairman of my committee and his profound insights into reliability. I would like to thank Dr. Kapur, Dr. Ravindran, Dr. Leemis, and Dr. Pulat for providing me with a strong foundation of courses. Doctors Leemis and Schlegel deserve special recognition for their detailed review and constructive criticism of my work. I also thank Dr. Grasse, who graciously agreed to participate on my committee.

Finally, I thank my family for the love and support they provided for the past three years. Becki gave me confidence and encouragement whenever I began to have doubts about my ability. David and Susan were understanding of their part-time Dad and a great source of energy and pride. No one could be blessed with a more perfect family.

TABLE OF CONTENTS

Chapter Page

Chapter Page

Chapter Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

ABSTRACT

The most commonly used reliability model is the binary model. However, the continuous model better represents items which degrade through a continuum of values. Unfortunately, the continuous model results in an overwhelming number of calculations. The multistate model is a sensible compromise between the binary and continuous models. The purpose of this dissertation is to develop multistate models based on the voice of the customer.

Structural, stochastic, and dynamic properties are reviewed for the binary, multistate, and continuous models. The multistate and continuous models are generalized to allow a different number of system and component states. Analogous properties are shown for the general multistate model and the general continuous model.

The general multistate model is developed and evaluated from the viewpoint of the customer. A method for state classification is presented that allows the customer to define the number of system and component states. A technique using the convolution of random variables is devised to estimate the component state probabilities after the customer specifies the desired system lifetime. An algorithm is designed to determine the customer's implicit structure function by having the customer specify a set of boundary points. A procedure to convert from one set of boundary points to the other is developed and implemented with a computer program. The conversion program limits the amount of information required from the customer. Two additional computer programs are written to implement existing techniques for estimating system state probabilities. Expected loss is introduced as a substitute characteristic for reliability.

1. INTRODUCTION

The nature of a product is defined by the inherent qualities that are particular to the product. The customer uses qualities to establish a preference for one product over another. Gitlow, Gitlow, Oppenheim, and Oppenheim [1989] define quality as the extent a product surpasses the needs and expectations of the customer.

Every quality has a desired direction for improvement. Selecting a product is easy when all qualities are at their best level. The trouble comes when forced to make a choice between products with conflicting qualities (i.e. one quality's improvement causes another quality's decline). For this case, some attempt must be made to convert the qualities to a common scale. Another problem occurs when the qualities are too broad or vague. The solution to ambiguous qualities is to replace them with substitute characteristics that are quantitative and more easily compared. However, generating several substitute characteristics confounds the problem of conflicting qualities.

As an example, suppose a customer is looking for a new home computer. He decides to use four qualities to compare various brands: cost, hard drive capacity, clock speed, and reliability. The other qualities are either not important or so important that only one level of the quality is acceptable. In general, the customer strives to decrease cost and increase the other three qualities. After further thought, the customer divides the cost quality into two substitute characteristics: original system price and average annual maintenance cost. Several of the qualities conflict. For example, a price decrease results in a slower computer with less permanent memory. Suppose the customer can choose from the 3 computer systems listed in Table 1.1.

Table 1.1 Computer System Qualities.

Quality	System A	System B	System C
System Price	$1500	$1600	$1400
Average Annual Maintenance Cost	$120	$105	$105
Hard Disk Capacity	40 Mega-Byte	80 Mega-Byte	60 Mega-Byte
Clock Speed	10 Mega-Hertz	12 Mega-Hertz	12 Mega-Hertz
Reliability (2 years)	.95	.98	.97

The customer would never choose System A since its qualities are inferior to those of System C. To make a final choice, the customer must decide whether the additional 20 Megabytes and higher reliability of System B is worth the extra $200 in system price. If not, he should purchase System C.

The previous discussion emphasizes that the essential element in product selection is the customer. In fact, only the customer can decide which qualities are important and how the qualities are weighted to discriminate between products. Stated in different terms, the quality of a product is defined and evaluated by the customer.

1.1 Background

Reliability is a quality. Therefore, reliability must be defined and evaluated from the viewpoint of the customer. Reliability models derived for their intuitive appeal or mathematical simplicity fail to satisfy this logical and important criterion.

1.1.1 Binary Model

Determining the reliability of a complex system from the structure of the system and the reliability of the components is a fundamental problem in reliability theory. For the binary model, reliability is defined as the probability that the product will perform its intended function adequately under stated environmental conditions for a specified interval of time [Kapur and Lamberson, 1977].

Unfortunately, a single measure of reliability does not always provide enough information for the customer to make an informed choice between products. The customer can often make a better choice by simultaneously exploring several substitute characteristics for reliability. For example, the failure variability, the p^{th} percentile, and the mean time to failure could all be included in the customer's evaluation. Each measure gives an indirect assessment of the reliability for the system. The simultaneous consideration of several conflicting substitute characteristics for reliability results in a multiobjective optimization problem that can be solved with existing techniques.

1.1.2 Multistate Model

The multistate model allows the customer to specify more than two discrete states for the components and the system. The previous definition for reliability is no longer valid since now there are different degrees of functioning. This forced the development of several substitute characteristics for reliability: El-Neweihi, Proschan, and Sethuraman [1978] suggested the expected system state; Butler [1979] divided the set of system states into an acceptable set and an unacceptable set; Griffith [1980] proposed the expected utility derived from the system states.

1.1.3 Continuous Model

The continuous model allows the performance of the system and components to be specified along some continuum. Baxter [1984,1986], Kaleva [1986], Baxter and Kim [1986], and Montero, Tejada, and Yáñez [1990] concentrated on continuum structures that map from the unit hypercube to the unit interval. For this special case, system reliability was defined as the expected system state or the probability that the system state exceeded some given value between 0 and 1.

1.2 Objectives

The primary objective of this research is to develop customer-driven reliability models for multistate coherent systems. The models will require customer interaction at every step. The customer will define the number of distinct component and system states. The customer will stipulate a desired system lifetime, allowing the estimation of the probability distribution for each component. The customer will define the system by specifying when a change in the state of any one of the components forces a change in the state of the system. The previous input characterizes the customer's implicit structure function and allows the calculation of system state probabilities. The customer will choose one or more substitute characteristics for reliability that summarize the system state probabilities. The best compromise solution will be found through interaction with the customer.

The second objective of this research is to develop a new substitute characteristic for multistate reliability based on expected loss to the customer. The new measure will be sensitive to the pattern of degradation about a specified lifetime. In other words, the measure will be a function of not only the number of state changes, but also the time of each state change relative to the specified lifetime.

1.3 Scope

The components and systems considered in this paper are assumed to be nonrepairable.

Also in this paper, the random variables representing the *n* component states are assumed to be mutually independent unless specifically stated otherwise. Mutually independent random variables are defined and described in the following paragraphs.

DEFINITION. Discrete random variables $X_1, X_2, \ldots, X_n$ are mutually independent if and only if

$$\Pr[X_1=x_1, X_2=x_2, \ldots, X_n=x_n] = p_1\ p_2 \cdots p_n$$

where $p_i = \Pr[X_i=x_i]$, $i=1,2,\ldots,n$. Continuous random variables are mutually independent if and only if

$$f(x_1, x_2, \ldots, x_n) = f(x_1)\ f(x_2) \cdots f(x_n)$$

for every $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and $f(x_i)$ is the marginal probability density function of X_i.

The assumption of mutual independence is stronger than the assumption of pairwise independence. Clark and Disney [1970] showed that a finite set of random variables can be pairwise independent without the whole set being mutually independent.

The independence assumption implies that the state of one component will have no effect on the states of the other components in the system. Obviously, independence can not and should not be assumed for every system. However, the case of dependent component states will not be pursued here.

1.4 Overview

Chapter 1 provided an introduction to quality and reliability, emphasizing the role of the customer in the definition and evaluation of reliability. The main objective of the dissertation is to develop customer-driven reliability models for multistate systems, including an innovative substitute characteristic for reliability based on the voice of the customer. The study was narrowed to systems composed of mutually independent components. Chapter 2 gives a review of the deterministic, stochastic, and dynamic properties for the binary model. Chapter 3 presents the same properties for the multistate model, making a distinction between the early multistate models where the system and component states were restricted to the same set and a new general multistate model which allows a different number of discrete states for the system and each component. Chapter 4 reviews the continuous model where the system and component states degrade through a continuum of values. For the first time, the continuous model is generalized to allow different ranges for the system and component states. Chapter 5 presents the customer-driven multistate reliability model and demonstrates how to get the customer involved at every step during the development and evaluation of the model. Chapter 6 contains several specific applications of the customer-driven multistate reliability model. Chapter 7 provides conclusions and recommendations for further research.

2. THE BINARY MODEL

This chapter presents a review of the structural and stochastic properties of the binary model most commonly used in reliability theory.

2.1 Structural Properties

Structural properties characterize the deterministic relationship between the state of the system and the states of the components at a fixed moment in time.

2.1.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components comprising the system.
x_i	state of component i; $x_i \in \{0,1\}$ for $i=1,2,\dots,n$.
$\mathbf{x}$	component state vector; $\mathbf{x} = (x_1,x_2,\dots,x_n)$.
$\phi(\mathbf{x})$	structure function; system state for $\mathbf{x}$.
ϕ	state of the system; $\phi \in \{0,1\}$.
$\prod_{i=1}^{n} x_i$	$\mathrm{Min}\{x_1,x_2,\dots,x_n\} = x_1\, x_2 \cdots x_n$.
$\coprod_{i=1}^{n} x_i$	$\mathrm{Max}\{x_1,x_2,\dots,x_n\} = 1-(1-x_1)(1-x_2)\cdots(1-x_n)$.
$\mathbf{0}$	$(0,0,\dots,0)$.
$\mathbf{1}$	$(1,1,\dots,1)$.
$(0_i,\mathbf{x})$	$(x_1,x_2,\dots,x_{i-1},0,x_{i+1},\dots,x_n)$.
$(1_i,\mathbf{x})$	$(x_1,x_2,\dots,x_{i-1},1,x_{i+1},\dots,x_n)$.
$(\cdot_i,\mathbf{x})$	$(x_1,x_2,\dots,x_{i-1},\cdot,x_{i+1},\dots,x_n)$.
$\mathbf{y} > \mathbf{x}$	$y_i \geq x_i \ \forall i$ and $y_i > x_i$ for some i.

$\mathbf{y} \geq \mathbf{x}$	$y_i \geq x_i \quad \forall i$.
$\mathbf{x} \amalg \mathbf{y}$	$(x_1 \amalg y_1, x_2 \amalg y_2, \ldots, x_n \amalg y_n)$ where $x_i \amalg y_i = \mathrm{Max}\{x_i, y_i\}$.
$\mathbf{x} \prod \mathbf{y}$	$(x_1 \prod y_1, x_2 \prod y_2, \ldots, x_n \prod y_n)$ where $x_i \prod y_i = \mathrm{Min}\{x_i, y_i\}$.
P_j	jth minimal path set; $j=1,2,\ldots,s$.
$\alpha_j(\mathbf{x})$	indicator variable; 1 if all components in P_j work.
C_j	jth minimal cut set; $j=1,2,\ldots,t$.
$\beta_j(\mathbf{x})$	indicator variable; 0 if all components in C_j fail.
ϕ^D	dual structure function.
$\mathbf{1} - \mathbf{x}$	$(1-x_1, 1-x_2, \ldots, 1-x_n)$.
$I_\phi(i)$	structural importance of component i in ϕ.
C	component set; $C = \{c_1, c_2, \ldots, c_n\}$.
A_j	component set of module j; $A_j \subset C$.
χ_j	structure function of module j.
ψ	organizing structure for a modular decomposition.

2.1.2 Introduction

Birnbaum, Esary, and Saunders [1961] introduced the binary model, which has served as the basis for the mathematical and statistical theory of reliability. For this model, the system and n components are assumed to be in one of two possible states: functioning or failed.

The order of a system, n, is the number of distinct components that make up the system. Along with the order, a system is characterized by its structure which describes how the components are configured. The structure of the system is represented by a function, known as the structure function, which determines the system state from the states of the components.

Suppose the state of the ith component is represented by the binary variable x_i, where

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is functioning} \\ 0 & \text{if component } i \text{ has failed} \end{cases}$$

for i=1,2,...,n. The binary component states are summarized with the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The structure function $\phi(\mathbf{x})$ determines the binary state of the system ϕ from the component state vector so that

$$\phi = \begin{cases} 1 & \text{if the system is functioning} \\ 0 & \text{if the system has failed.} \end{cases}$$

2.1.3 Special Structures

Birnbaum et al. [1961] defined three basic structures for the binary case: series, parallel, and k-out-of-n. A series system is defined so that the system functions if and only if each component functions. The structure function is

$$\phi(\mathbf{x}) = \prod_{i=1}^{n} x_i = \text{Min}\{x_1, x_2, \ldots, x_n\} = x_1 \; x_2 \cdots x_n .$$

A parallel system is defined so that the system fails if and only if all the components fail. The structure function is

$$\phi(\mathbf{x}) = \coprod_{i=1}^{n} x_i = \text{Max}\{x_1, x_2, \ldots, x_n\} = 1-(1-x_1)(1-x_2)\cdots(1-x_n) .$$

A k-out-of-n system is defined so that the system functions if and only if at least k-out-of-n components function. The structure function is

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq k \\ 0 & \text{if } \sum_{i=1}^{n} x_i < k. \end{cases}$$

Series and parallel systems are special cases of the k-out-of-n structure. A series system is an n-out-of-n structure while a parallel system is a 1-out-of-n structure.

2.1.4 Coherent Structures

From the beginning, the founders of reliability theory only considered the set of structures that were intuitively appealing. Birnbaum et al. [1961] coined the term coherent system and offered the following definition:

DEFINITION - A structure ϕ is <u>coherent</u> iff

i. $\phi(\mathbf{y}) \geq \phi(\mathbf{x})$ for all $\mathbf{y} > \mathbf{x}$ and

ii. $\phi(\mathbf{0}) = 0$ and $\phi(\mathbf{1}) = 1$.

They also defined the ith component to be essential or relevant if $\phi(0_i, \mathbf{x}) \neq \phi(1_i, \mathbf{x})$ for some $(\cdot_i, \mathbf{x})$.

Esary and Proschan [1963a] modified the definition of coherence given by Birnbaum et al. [1961] to include all increasing[1] functions:

DEFINITION - A structure ϕ is <u>coherent</u> iff

i. $\phi(\mathbf{y}) \geq \phi(\mathbf{x})$ for all $\mathbf{y} \geq \mathbf{x}$ and

ii. $\phi(\mathbf{0}) = 0$ and $\phi(\mathbf{1}) = 1$.

[1]The term increasing (decreasing) is used in place of nondecreasing (nonincreasing) in the reliability literature.

Barlow and Proschan [1981] combined the concepts of component relevance and increasing structure functions to give a commonly used definition of a binary coherent system.

DEFINITION - A system or structure ϕ is <u>coherent</u> iff

i. $\phi(\mathbf{y}) \geq \phi(\mathbf{x})$ for all $\mathbf{y} \geq \mathbf{x}$ and

ii. Each component is relevant to the system where the i^{th} component is <u>relevant</u> iff $\phi(1_i,\mathbf{x}) \neq \phi(0_i,\mathbf{x})$ for some $(\cdot_i,\mathbf{x})$.

Interpreting the last definition, repair of a failed component cannot cause system deterioration and failure of a working component cannot cause system improvement. Also, all components in a coherent system must influence the state of the system for at least one component state vector. All irrelevant components should be removed from the system to reduce cost.

EXAMPLE 2.1 Show that $\phi_1(\mathbf{x}) = 1 - x_1x_2$ and $\phi_2(\mathbf{x}) = 1 - (1-x_1)(1-x_1x_2)$ and are not coherent structure functions as defined by Barlow and Proschan.

$\phi_1(\mathbf{x})$ is a decreasing function since $\phi_1(0,0) = 1$ and $\phi_1(1,1) = 0$. Thus ϕ_1 degrades as components are repaired. For $\phi_2(\mathbf{x})$, component 2 is irrelevant since $\phi_2(0,0) = \phi_2(0,1) = 0$ and $\phi_2(1,0) = \phi_2(1,1) = 1$. The system can be improved by removing component 2.

Elimination of unrealistic structures allowed Esary, Marshall, and Proschan [1970] to present the next three theorems for every coherent system.

The first theorem implies that every coherent system can be bounded below by a series arrangement of the system's components and bounded above by a parallel arrangement of the system's components.

THEOREM 2.1 If ϕ is a coherent system, then

$$\prod_{i=1}^{n} x_i \le \phi(\mathbf{x}) \le \coprod_{i=1}^{n} x_i.$$

The second theorem states an important result about redundancy. Suppose that enough components exist to build two identical structures. The choices are to construct a system made from:

(1) Two identical structures arranged in parallel or

(2) One structure with each component in parallel.

The next theorem states the design principle that it is better to duplicate at the component level rather than at the system level, so choice (2) is better. Equality holds when ϕ is a parallel structure.

THEOREM 2.2 If ϕ is a coherent system, then

$$\phi(\mathbf{x} \amalg \mathbf{y}) \ge \phi(\mathbf{x}) \amalg \phi(\mathbf{y})$$

for any state vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x} \amalg \mathbf{y}$ is defined as $(x_1 \amalg y_1, x_2 \amalg y_2, \ldots, x_n \amalg y_n)$ where $x_i \amalg y_i = \text{Max}\{x_i, y_i\}$ and $\phi(\mathbf{x}) \amalg \phi(\mathbf{y}) = \text{Max}\{\phi(\mathbf{x}), \phi(\mathbf{y})\}$.

The final theorem has a similar interpretation. Again, suppose that enough components exist to build two identical structures. The choices are to construct a system made from:

(1) Two identical structures arranged in series or

(2) One structure with each component in series.

The next theorem states that placing the two structures in series is better than placing the components of the structure in series, so choice (1) is better. Equality holds when ϕ is a series structure.

THEOREM 2.3 If ϕ is a coherent system, then

$$\phi(\mathbf{x}\ \Pi\ \mathbf{y}) \le \phi(\mathbf{x})\ \Pi\ \phi(\mathbf{y})$$

for any state vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x}\ \Pi\ \mathbf{y}$ is defined as $(x_1\ \Pi\ y_1, x_2\ \Pi\ y_2, \ldots, x_n\ \Pi\ y_n)$ where $x_i\ \Pi\ y_i = \text{Min}\{x_i, y_i\}$ and $\phi(\mathbf{x})\ \Pi\ \phi(\mathbf{y}) = \text{Min}\{\phi(\mathbf{x}), \phi(\mathbf{y})\}$.

2.1.5 Equivalent Coherent Structures

Birnbaum et al. [1961] showed that any coherent structure can be represented by several alternate structures. The subsequent discussion and notation follow mainly from Ross [1989].

A minimal path set is a minimal set of functioning components that guarantees that the system functions.

DEFINITIONS. A component state vector $\mathbf{x}$ is called a path vector if $\phi(\mathbf{x}) = 1$. A path vector $\mathbf{x}$ is a minimal path vector if $\phi(\mathbf{y}) = 0$ for any $\mathbf{y} < \mathbf{x}$. If $\mathbf{x}$ is a minimal path vector, then $P(\mathbf{x}) = \{i \mid x_i = 1\}$ is called a minimal path set.

Let $P_1, P_2, \ldots, P_s$ denote the minimal path sets for a given system. Let $\alpha_j(\mathbf{x})$ be an indicator variable of the j^{th} minimal path set defined by

$$\alpha_j(\mathbf{x}) = \begin{cases} 1 & \text{if all components of } P_j \text{ are functioning} \\ 0 & \text{otherwise} \end{cases}$$
$$= \prod_{i \in P_j} x_i .$$

The system will function if and only if all the components in at least one minimal path set are functioning. Hence, the state of the system is given by

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha_j(\mathbf{x}) = 1 \text{ for some } j \\ 0 & \text{if } \alpha_j(\mathbf{x}) = 0 \text{ for all } j. \end{cases}$$

This equation is a parallel arrangement and can be rewritten in the following equivalent forms:

$$\begin{aligned} \phi(\mathbf{x}) &= \operatorname*{Max}_j \alpha_j(\mathbf{x}) \\ &= \operatorname*{Max}_j \prod_{i \in P_j} x_i \\ &= 1 - \prod_{j=1}^{s} (1 - \prod_{i \in P_j} x_i) \\ &= \coprod_{j=1}^{s} \prod_{i \in P_j} x_i . \end{aligned}$$

Therefore, for the binary case, minimal path sets can be used to represent any coherent system as a parallel arrangement of series structures.

A minimal cut set is a minimal set of failed components that guarantees that the system fails.

DEFINITIONS. A component state vector $\mathbf{x}$ is called a <u>cut vector</u> if $\phi(\mathbf{x}) = 0$. A cut vector $\mathbf{x}$ is a <u>minimal cut vector</u> if $\phi(\mathbf{y}) = 1$ for any $\mathbf{y} > \mathbf{x}$. If $\mathbf{x}$ is a minimal cut vector, then $C(\mathbf{x}) = \{i \mid x_i = 0\}$ is called a <u>minimal cut set</u>.

Let $C_1, C_2, \ldots, C_t$ denote the minimal cut sets for a given system. Let $\beta_j(\mathbf{x})$ be an indicator variable of the jth minimal cut set defined by

$$\beta_j(\mathbf{X}) = \begin{cases} 0 & \text{if all components of } C_j \text{ are not functioning} \\ 1 & \text{otherwise} \end{cases}$$
$$= \coprod_{i \in C_j} x_i .$$

The system will fail if and only if all the components in at least one minimal cut set are not functioning. Hence, the state of the system is given by

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \beta_j(\mathbf{x}) = 1 \text{ for all } j \\ 0 & \text{if } \beta_j(\mathbf{x}) = 0 \text{ for some } j. \end{cases}$$

This equation is a series arrangement and can be rewritten in the following equivalent forms:

$$\begin{aligned} \phi(\mathbf{x}) &= \operatorname*{Min}_j \beta_j(\mathbf{x}) \\ &= \operatorname*{Min}_j \coprod_{i \in C_j} x_i \\ &= \prod_{j=1}^{t} \left(1 - \prod_{i \in C_j} (1 - x_i)\right) \\ &= \prod_{j=1}^{t} \coprod_{i \in C_j} x_i . \end{aligned}$$

Therefore, for the binary case, minimal cut sets can be used to represent any coherent system as a series arrangement of parallel structures.

EXAMPLE 2.2 Write the equivalent structure functions using minimal path and cut sets for the coherent system given in Figure 2.1 and show that they are equivalent to the original structure function.

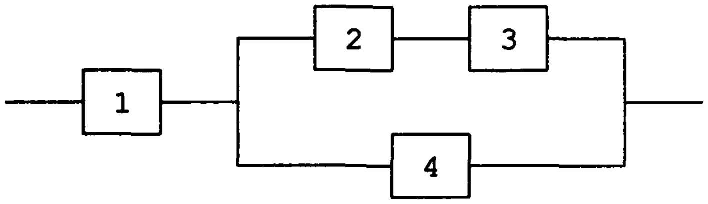


Figure 2.1 Coherent System of 4 Components.

The original structure function is given by

$$\begin{aligned}\phi(\mathbf{x}) &= x_1\ [1-(1-x_2x_3)(1-x_4)] \\ &= x_1\ [x_2x_3 + x_4 - x_2x_3x_4] \\ &= x_1x_2x_3 + x_1x_4 - x_1x_2x_3x_4.\end{aligned}$$

The minimal path sets are {1,2,3}, {1,4}. So

$$\begin{aligned}\phi(\mathbf{x}) &= 1-(1-x_1x_2x_3)(1-x_1x_4) \\ &= 1 - [1 - x_1x_2x_3 - x_1x_4 + x_1^2x_2x_3x_4] \\ &= x_1x_2x_3 + x_1x_4 - x_1x_2x_3x_4.\end{aligned}$$

The minimal cut sets are {1}, {2,4}, {3,4}. So

$$\begin{aligned}\phi(\mathbf{x}) &= x_1\ [1-(1-x_2)(1-x_4)]\ [1-(1-x_3)(1-x_4)] \\ &= x_1\ [x_2 + x_4 - x_2x_4]\ [x_3 + x_4 - x_3x_4] \\ &= x_1\ [x_2x_3+x_2x_4-x_2x_3x_4+x_3x_4+x_4^2-x_3x_4^2-x_2x_3x_4-x_2x_4^2+x_2x_3x_4^2] \\ &= x_1\ [x_2x_3+x_2x_4-x_2x_3x_4+x_3x_4+x_4-x_3x_4-x_2x_3x_4-x_2x_4+x_2x_3x_4] \\ &= x_1\ [x_2x_3 - x_2x_3x_4 + x_4] \\ &= x_1x_2x_3 + x_1x_4 - x_1x_2x_3x_4.\end{aligned}$$

Therefore, the three representations are equivalent.

As a final note, Birnbaum et al. [1961] presented the following expansion to reduce the order of the structure function by one:

$$\phi(\mathbf{x}) = x_i\ \phi(1_i,\mathbf{x}) + (1 - x_i)\ \phi(0_i,\mathbf{x}). \qquad (2.1)$$

The structure can be expanded with respect to any component.

2.1.6 Dual Structure Function

Birnbaum et al. [1961] introduced the dual structure function which has proven useful both for switching systems and systems with two different modes of failure.

DEFINITION. Given a structure ϕ, the <u>dual structure</u> ϕ^D is defined by

$$\phi^D(\mathbf{x}) = 1 - \phi(\mathbf{1} - \mathbf{x}).$$

The structure function and dual structure function have several important relationships which have been stated and proven by many authors.

THEOREM 2.4 The dual of a k-out-of-n system is an (n-k+1)-out-of-n system.

Proof: ϕ is a k-out-of-n system which is given as

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq k \\ 0 & \text{if } \sum_{i=1}^{n} x_i < k. \end{cases}$$

When evaluated at $(\mathbf{1} - \mathbf{x})$,

$$\phi(\mathbf{1} - \mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} (1 - x_i) \geq k \\ 0 & \text{if } \sum_{i=1}^{n} (1 - x_i) < k. \end{cases}$$

Therefore,

$$1 - \phi(\mathbf{1} - \mathbf{x}) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} x_i \leq n - k \\ 1 & \text{if } \sum_{i=1}^{n} x_i > n - k. \end{cases}$$

Changing to the form of a k-out-of-n system results in

$$1 - \phi(\mathbf{1} - \mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq n - k + 1 \\ 0 & \text{if } \sum_{i=1}^{n} x_i < n - k + 1 \end{cases}$$

which is an (n-k+1)-out-of-n system.

Since a series system is an n-out-of-n structure and a parallel system is a 1-out-of-n structure, the following two corollaries are immediately apparent:

COROLLARY 2.1 The dual of a series system of n components is a parallel system of n components.

COROLLARY 2.2 The dual of a parallel system of n components is a series system of n components.

Continuing with the relationships between ϕ and ϕ^D:

THEOREM 2.5 If $\mathbf{x}$ is a path vector of ϕ, then $(\mathbf{1} - \mathbf{x})$ is a cut vector of ϕ^D, and vice versa.

Proof: Let $\mathbf{x}$ be a path vector of ϕ. Then $\phi(\mathbf{x}) = 1$ and $1-\phi(\mathbf{x}) = 0$. But $1-\phi(\mathbf{x}) = \phi^D(\mathbf{1} - \mathbf{x})$. So $\phi^D(\mathbf{1} - \mathbf{x}) = 0$ and $(\mathbf{1} - \mathbf{x})$ is a cut vector of ϕ^D. Let $\mathbf{x}$ be a cut vector of ϕ. Then $\phi(\mathbf{x}) = 0$ and $1-\phi(\mathbf{x}) = 1$. Therefore, $\phi^D(\mathbf{1} - \mathbf{x}) = 1$ and $(\mathbf{1} - \mathbf{x})$ is a path vector of ϕ^D.

THEOREM 2.6 The minimal cut sets for ϕ are the minimal path sets for ϕ^D, and vice versa.

Proof: Suppose the minimal cut sets of ϕ are denoted by $C_1, C_2, \ldots, C_t$. Using the alternate form with minimal cut sets,

$$\phi(\mathbf{x}) = \prod_{j=1}^{t} \coprod_{i \in C_j} x_i .$$

Therefore, the dual can be written as

$$\begin{aligned} 1 - \phi(\mathbf{1} - \mathbf{x}) &= 1 - \prod_{j=1}^{t} \coprod_{i \in C_j} (1 - x_i) \\ &= 1 - \prod_{j=1}^{t} \left(1 - \prod_{i \in C_j} x_i\right) \\ &= \coprod_{j=1}^{t} \prod_{i \in C_j} x_i \, . \end{aligned}$$

which is an alternate form of ϕ^D with minimal path sets. Thus, $C_1, C_2, \ldots, C_t$ are the minimal path sets of ϕ^D. Now, suppose the minimal path sets of ϕ are denoted by $P_1, P_2, \ldots, P_s$. Using the form with minimal path sets,

$$\phi(\mathbf{x}) = \coprod_{j=1}^{s} \prod_{i \in P_j} x_i .$$

Therefore, the dual can be written as

$$\begin{aligned} 1 - \phi(\mathbf{1} - \mathbf{x}) &= 1 - \coprod_{j=1}^{s} \prod_{i \in P_j} (1 - x_i) \\ &= \prod_{j=1}^{s} \left(1 - \prod_{i \in P_j} (1 - x_i)\right) \\ &= \prod_{j=1}^{s} \coprod_{i \in P_j} x_i \end{aligned}$$

which is an alternate form of ϕ^D with minimal cut sets. Thus, $P_1, P_2, \ldots, P_s$ are the minimal cut sets of ϕ^D.

THEOREM 2.7 The dual of the dual structure function is the original structure function; that is, $(\phi^D)^D = \phi(\mathbf{x})$.

Proof: $(\phi^D)^D = 1 - \phi^D(\mathbf{1} - \mathbf{x})$

$$= 1 - [1 - \phi(\mathbf{1} - (\mathbf{1} - \mathbf{x}))]$$
$$= 1 - [1 - \phi(\mathbf{x})]$$
$$= \phi(\mathbf{x}).$$

2.1.7 Structural Importance

Some components play a more important role than others in determining whether or not the system will function based on their location in the system. For example, a component that is contained in every minimal path set is intuitively more important than a component contained in only one minimal path set. The number of minimal path sets that contain the i^{th} component can be counted by determining when the component's failure causes the system to fail; that is, when

$$\phi(1_i,\mathbf{x}) - \phi(0_i,\mathbf{x}) = 1.$$

If the state of component i is fixed, there are 2^{n-1} remaining component state vectors. Birnbaum [1969] showed that a good measure of structural importance for component i is

$$I_\phi(i) = \frac{1}{2^{n-1}} \sum_{\{\mathbf{x}|x_i=1\}} [\phi(1_i,\mathbf{x}) - \phi(0_i,\mathbf{x})]$$

where $\frac{1}{2^{n-1}}$ scales $I_\phi(i)$ so that $0 < I_\phi(i) \leq 1$ for $i=1,2,\ldots,n$. Note that $I_\phi(i)$ is strictly positive for any coherent system.

2.1.8 Modules and Modular Decomposition

Birnbaum et al. [1961] were first to explore the decomposition of large problems into smaller self-contained problems. A module is a group of components that can be treated as a single component, having only a single input and output from the rest of the system. Birnbaum and Esary [1965] formalized the definition and extended the result to three modules.

DEFINITION - Suppose (C,ϕ) is a coherent system where C is the set of components. Suppose that $A \subset C$. Let A' denote the subset of C complementary to A. The coherent system (A,χ) is a <u>module</u> of (C,ϕ) if

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^A,\mathbf{x}^{A'}) = \psi[\chi(\mathbf{x}^A),\mathbf{x}^{A'}]$$

where ψ is the organizing coherent structure function.

EXAMPLE 2.3 For the structure given in Figure 2.2, find the structure function χ for the module consisting of components 1 and 2 as well as for the organizing structure ψ.

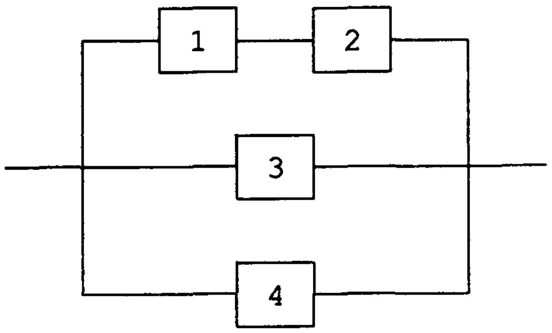


Figure 2.2 Structure for Example 2.3.

$\chi = x_1x_2$ and

$\psi = 1 - (1-\chi)(1-x_3)(1-x_4)$.

Barlow and Proschan [1981] generalized this result for a discrete number of disjoint modules.

DEFINITION - A <u>modular decomposition</u> of a coherent system (C,ϕ) is a set $\{(A_1,\chi_1),(A_2,\chi_2),\ldots,(A_k,\chi_k)\}$ along with the organizing structure ψ such that

i) $\{A_1, A_2, \ldots, A_k\}$ partition C into disjoint subsets and

ii) $\phi(\mathbf{x}) = \psi[\chi_1(\mathbf{x}^{A_1}), \chi_2(\mathbf{x}^{A_2}), \ldots, \chi_k(\mathbf{x}^{A_k})]$.

EXAMPLE 2.4 For the structure given in Figure 2.3, find the structure functions for module χ_1 consisting of components 1,2, and 3, for module χ_2 consisting of components 4 and 5 as well as for the organizing structure ψ.

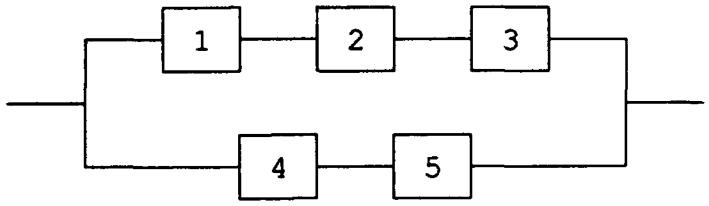


Figure 2.3 Structure for Example 2.4.

$\chi_1 = x_1x_2x_3$, $\chi_2 = x_4x_5$, and

$\psi = 1 - (1-\chi_1)(1-\chi_2)$.

In practice, a complex system is decomposed into major subsystems. Complex subsystems are further decomposed into assemblies. The decomposition process continues until each module's structure function is obvious. The organizing structure provides the means to determine the overall structure function for the system. In summary, modular decomposition of a system breaks up a complex problem into several smaller, more manageable problems.

2.2 Stochastic Properties

So far, only the deterministic properties of binary structure functions have been discussed. Stochastic properties characterize the probabilistic relationship between the state of the system and the states of the components at a fixed moment in time.

2.2.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components comprising the system.
X_i	random variable for the state of component i.
x_i	state of component i; $x_i \in \{0,1\}$ for $i=1,2,\ldots,n$.
$\mathbf{X}$	random component state vector; $\mathbf{X} = (X_1,X_2,\ldots,X_n)$.
$\mathbf{x}$	fixed component state vector; $\mathbf{x} = (x_1,x_2,\ldots,x_n)$.
p_i	reliability of component i; $p_i = Pr[X_i = 1]$.
$\mathbf{p}$	component reliability vector; $\mathbf{p} = (p_1,p_2,\ldots,p_n)$.
$\phi(\mathbf{X})$	random variable for the state of the system.
ϕ	fixed state of the system; $\phi = \phi(\mathbf{x})$.
$E[\phi(\mathbf{X})]$	the expected system state.
$r(\mathbf{p})$	reliability function; $r(\mathbf{p}) = E[\phi(\mathbf{X})]$.
$I_r(i)$	reliability importance of component i.
P_j	jth minimal path set; $j=1,2,\ldots,s$.
C_j	jth minimal cut set; $j=1,2,\ldots,t$.
A_j	component set of module j.
$\alpha_j(\mathbf{x})$	indicator variable; 1 if all components in P_j work.
$\beta_j(\mathbf{x})$	indicator variable; 0 if all components in C_j fail.
χ_j	structure of module j.
ψ	organizing structure for a modular decomposition.
Σ_j	jth term of the inclusion-exclusion principle.

2.2.2 The Reliability Function

As developed by Birnbaum et al. [1961], one of the most important problems of reliability theory is to determine the system reliability from the reliability of the components. Suppose a system consists of n components. Let X_i denote the random state of component i and x_i denote a specific state of component i. The random and specific states for all components are summarized by the random component state vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and the fixed component state vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. After defining the reliability of the i^{th} component as $p_i = \Pr[X_i = 1] = E[X_i]$, the vector of component reliabilities can be represented by $\mathbf{p} = (p_1, p_2, \ldots, p_n)$.

For the system, let $\phi(\mathbf{X})$ be the random system state and $\phi(\mathbf{x}) = \phi$ be a fixed system state. Suppose system reliability is defined as the probability that the system is functioning so that $r = \Pr[\phi(\mathbf{X}) = 1] = E[\phi(\mathbf{X})]$. If the random variables X_i, $i = 1, \ldots, n$ are mutually independent, then the system reliability is a function of the component reliabilities. The function $r(\mathbf{p})$ is called the reliability function and it is defined so that $r = r(\mathbf{p})$.

2.2.3 Reliability Importance

Section 2.1.7 discussed structural importance which was based on a component's location in the structure. This section discusses reliability importance, also taking into account the stochastic performance of each component in the system.

Birnbaum [1969] developed a measure of component importance which not only uses the structure ϕ but also the component reliabilities $\mathbf{p}$. The following expansion of the reliability function can be derived from the expected value of Equation 2.1:

$$r(\mathbf{p}) = p_i\, r(1_i,\mathbf{p}) + (1 - p_i)\, r(0_i,\mathbf{p}) \text{ for } i=1,2,\ldots,n.$$

Reliability importance is found by taking the partial derivative of $r(\mathbf{p})$ with respect to p_i.

$$I_r(i) = \frac{\partial r(\mathbf{p})}{\partial p_i} = r(1_i,\mathbf{p}) - r(0_i,\mathbf{p}).$$

The partial derivative determines the increase in system reliability per unit change in the reliability of component i. In this sense, the most important component will have the highest partial derivative. For a series system, the most important component has the lowest reliability, while for a parallel system, the most important component has the highest reliability. Prudent design engineers focus reliability improvement programs on components with high reliability importance.

2.2.4 Exact System Reliability

Enumeration, inclusion-exclusion, pivoting, and modular decomposition are four techniques used to determine the exact system reliability from the component reliabilities. Each technique assumes the components are mutually independent.

2.2.4.1 Enumeration

The enumeration technique delineates all possible component state vectors, $\mathbf{x} \in S$. The system state, $\phi(\mathbf{x})$, and probability, $\Pr[\mathbf{X} = \mathbf{x}]$, are determined fcr each component state vector. Reliability is the sum of the probabilities for all $\mathbf{x} \in S$ with $\phi(\mathbf{x}) = 1$.

EXAMPLE 2.5 Suppose that p_1=.9, p_2=.8, p_3=.7, and p_4=.6 for the structure given in Figure 2.4.

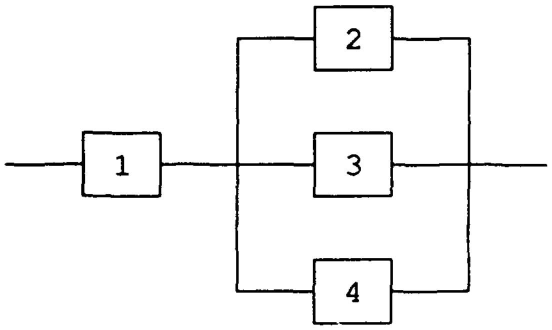


Figure 2.4 Structure for Enumeration.

Assuming components are mutually independent, find r.

$\mathbf{x}$	$\phi(\mathbf{x})$	$\Pr[\mathbf{X}=\mathbf{x}]$	$\mathbf{x}$	$\phi(\mathbf{x})$	$\Pr[\mathbf{X}=\mathbf{x}]$
0000	0	.0024	1000	0	.0216
0001	0	.0036	1001	1	.0324
0010	0	.0056	1010	1	.0504
0011	0	.0084	1011	1	.0756
0100	0	.0096	1100	1	.0864
0101	0	.0144	1101	1	.1296
0110	0	.0224	1110	1	.2016
0111	0	.0336	1111	1	.3024

The system reliability can be found by summing the probability of all $\mathbf{x} \in S$ where $\phi(\mathbf{x}) = 1$. Thus, r = .0324+.0504+.0756+.0864+.1296+.2016+.3024 = .8784. Note that $\Pr[\phi(\mathbf{X}) = 0] = .1216$.

The number of component state vectors is 2^n and the number of calculations needed to enumerate all $\mathbf{x}$ becomes unmanageable for large, complex systems. It seems more reasonable to calculate system reliability from a subset of all component state vectors such as the minimal path or minimal cut sets.

2.2.4.2 Inclusion-Exclusion

The inclusion-exclusion technique of Barlow and Proschan [1981] is based on the fact that the system functions if and only if all the components in at least one minimal path set function. Suppose ϕ is a binary coherent system with minimal path sets $P_1, P_2, \ldots, P_s$. Let E_j be the event that all components in P_j are functioning for $j=1,\ldots,s$. Then the system reliability is given by $r = \Pr[\phi(\mathbf{X}) = 1] = \Pr[\bigcup_{j=1}^{s} E_j]$.

The probability of the union of a finite number of events can be found from Feller's inclusion-exclusion principle [1968].

THEOREM 2.8 For any s events $E_1, \ldots, E_s$,

$$\Pr[\bigcup_{j=1}^{s} E_j] = \sum_{j=1}^{s} \Pr[E_j] - \sum_{j<k} \Pr[E_j E_k] + \sum_{j<k<l} \Pr[E_j E_k E_l]$$
$$- \sum_{j<k<l<m} \Pr[E_j E_k E_l E_m] + \cdots + (-1)^{s+1} \Pr[E_1 E_2 \cdots E_s].$$

Exact reliability can also be derived from minimal cut sets. Recall that the system will fail if and only if all the components in at least one minimal cut set have failed. Suppose ϕ is a binary coherent system with minimal cut sets $C_1, C_2, \ldots, C_t$. Let E_j be the event that all components in C_j have failed for $j=1,\ldots,t$. Then the system reliability is given by

$$r = 1 - \Pr[\phi(\mathbf{X}) = 0] = 1 - \Pr[\bigcup_{j=1}^{t} E_j]$$

where the union of events is again derived from Theorem 2.8.

EXAMPLE 2.6 For the structure given in example 2.5, calculate the system reliability using minimal path and minimal cut sets.

The minimal path sets for example 2.5 are {1,2}, {1,3}, and {1,4}. Therefore,

$$\begin{aligned} r &= \Pr[(X_1,X_2=1) \cup (X_1,X_3=1) \cup (X_1,X_4=1)] \\ &= \Pr[X_1,X_2=1] + \Pr[X_1,X_3=1] + \Pr[X_1,X_4=1] \\ &\quad - \Pr[X_1,X_2,X_3=1] - \Pr[X_1,X_3,X_4=1] - \Pr[X_1,X_2,X_4=1] \\ &\quad + \Pr[X_1,X_2,X_3,X_4=1] \\ &= (.9)(.8) + (.9)(.7) + (.9)(.6) - (.9)(.8)(.7) \\ &\quad - (.9)(.7)(.6) - (.9)(.8)(.6) + (.9)(.8)(.7)(.6) \\ &= .72+.63+.54-.504-.378-.432+.3024 = .8784 \end{aligned}$$

which agrees with example 2.5.

The minimal cut sets for example 2.5 are {1}, and {2,3,4}. Therefore,

$$\begin{aligned} r &= 1 - \Pr[(X_1=0) \cup (X_2,X_3,X_4=0)] \\ &= 1 - \{\Pr[X_1=0] + \Pr[X_2,X_3,X_4=0] - \Pr[X_1,X_2,X_3,X_4=0]\} \\ &= 1 - \{.1 + (.2)(.3)(.4) - (.1)(.2)(.3)(.4)\} \\ &= 1 - \{.1 + .024 -.0024\} = 1 - .1216 = .8784 \end{aligned}$$

which also agrees with example 2.5.

Note that the number of calculations needed to calculate exact reliability from s minimal path sets is

$$\binom{s}{1} + \binom{s}{2} + \ldots + \binom{s}{s} = 2^s - 1$$

and from t minimal cut sets is

$$\binom{t}{1} + \binom{t}{2} + \ldots + \binom{t}{t} = 2^t - 1.$$

Since enumeration requires 2^n calculations, inclusion-exclusion is superior to enumeration when the number of minimal path sets or minimal cut sets is less than the number of components and the minimal path or minimal cut sets are known.

2.2.4.3 Pivotal Decomposition

The pivotal decomposition technique developed by Birnbaum et al. [1961] uses the following reliability expansion formula:

$$r(\mathbf{p}) = p_i \, r(1_i,\mathbf{p}) + (1 - p_i) \, r(0_i,\mathbf{p}) \text{ for } i=1,2,\ldots,n.$$

The state of any component is fixed and reliability calculations are made from the simplified systems.

EXAMPLE 2.7 For the structure in Figure 2.5, show that the reliability functions found from minimal path sets and pivotal decomposition are the same.

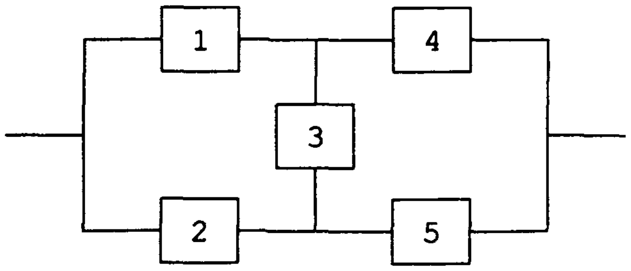


Figure 2.5 Structure for Pivotal Decomposition.

The minimal path sets are {1,4}, {1,3,5}, {2,3,4}, and

{2,5}. The structure function is

$\phi(\mathbf{X}) = 1-(1-x_1x_4)(1-x_1x_3x_5)(1-x_2x_3x_4)(1-x_2x_5)$

$= 1-(1-x_1x_4-x_1x_3x_5+x_1x_3x_4x_5)(1-x_2x_3x_4-x_2x_5+x_2x_3x_4x_5)$

$= 1-(1-x_2x_3x_4-x_2x_5+x_2x_3x_4x_5-x_1x_4+x_1x_2x_3x_4+x_1x_2x_4x_5-x_1x_2x_3x_4x_5$

$-x_1x_3x_5+x_1x_2x_3x_4x_5+x_1x_2x_3x_5-x_1x_2x_3x_4x_5+x_1x_3x_4x_5-x_1x_2x_3x_4x_5$

$-x_1x_2x_3x_4x_5+x_1x_2x_3x_4x_5)$

$= x_1x_4+x_2x_5+x_1x_3x_5+x_2x_3x_4-x_1x_2x_3x_4-x_1x_2x_3x_5-x_1x_2x_4x_5-x_1x_3x_4x_5$

$-x_2x_3x_4x_5+2x_1x_2x_3x_4x_5.$

$r = Pr[\phi(\mathbf{X})=1] = E[\phi(\mathbf{X})] = p_1p_4+p_2p_5+p_1p_3p_5+p_2p_3p_4$

$-p_1p_2p_3p_4-p_1p_2p_3p_5-p_1p_2p_4p_5-p_1p_3p_4p_5-p_2p_3p_4p_5+2p_1p_2p_3p_4p_5.$

Suppose pivotal decomposition is performed on the third component. When the third component functions, the structure shown in Figure 2.5 changes to the structure shown in Figure 2.6.

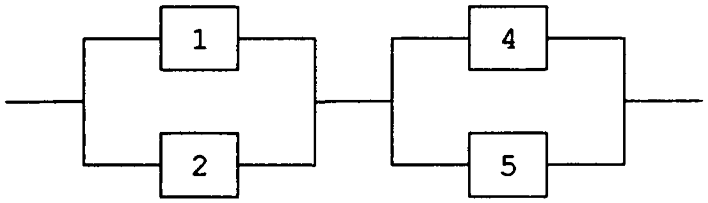


Figure 2.6 New Structure With $x_3 = 1$.

The new structure function is

$$\phi_1(\mathbf{X}) = [1-(1-x_1)(1-x_2)][1-(1-x_4)(1-x_5)]$$

and reliability is given by

$$r(1_3,\mathbf{p}) = E[\phi_1(\mathbf{X})] = [1-(1-p_1)(1-p_2)][1-(1-p_4)(1-p_5)].$$

When the third component fails, the structure in Figure 2.5 changes to the structure shown in Figure 2.7.

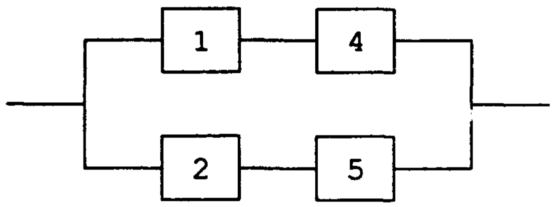


Figure 2.7 New Structure With $x_3 = 0$.

The new structure function is

$$\phi_0(\mathbf{X}) = [1-(1-x_1x_4)(1-x_2x_5)]$$

and reliability is given by

$$r(0_3,\mathbf{p}) = E[\phi_0(\mathbf{X})] = [1-(1-p_1p_4)(1-p_2p_5)].$$

The overall system reliability can be found from the reliability expansion about component 3.

$$\begin{aligned}
r(\mathbf{p}) &= p_3\ r(1_3,\mathbf{p}) + (1 - p_3)\ r(0_3,\mathbf{p}) \\
&= p_3\ [1-(1-p_1)(1-p_2)]\ [1-(1-p_4)(1-p_5)] \\
&\quad + (1-p_3)\ [1-(1-p_1p_4)(1-p_2p_5)] \\
&= p_3\ [p_1+p_2-p_1p_2]\ [p_4+p_5-p_4p_5] \\
&\quad + (1-p_3)\ [p_1p_4+p_2p_5-p_1p_2p_4p_5] \\
&= p_1p_4+p_2p_5+p_1p_3p_5+p_2p_3p_4-p_1p_2p_3p_4-p_1p_2p_3p_5-p_1p_2p_4p_5 \\
&\quad -p_1p_3p_4p_5-p_2p_3p_4p_5+2p_1p_2p_3p_4p_5
\end{aligned}$$

which matches the result found using minimal path sets.

2.2.4.4 Modular Decomposition

In practical reliability analysis, systems are often divided into disjoint subsystems (modules) and evaluated separately [Birnbaum and Esary, 1965]. The reliability of each subsystem is determined from the component reliabilities using any of the previously mentioned techniques. Then the overall system reliability is found from the subsystem reliabilities. The following example demonstrates the computational savings potential of modular decomposition:

EXAMPLE 2.8 Consider the structure of 7 binary components given in Figure 2.8.

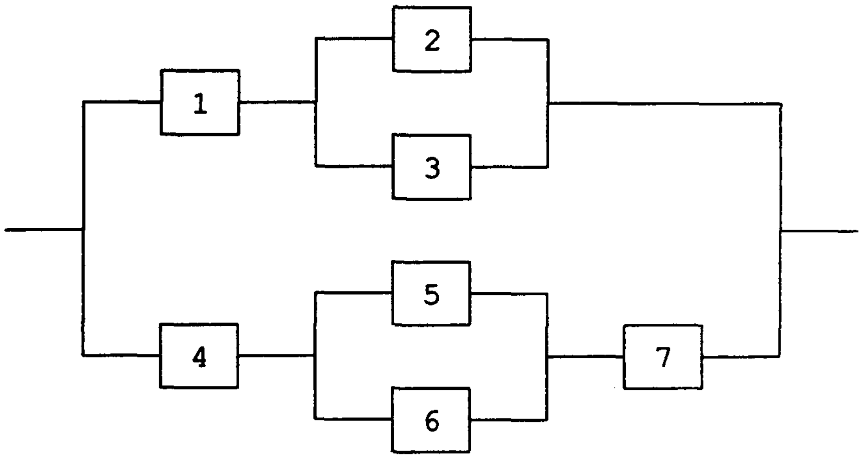


Figure 2.8 Structure for Modular Decomposition.

With enumeration, there are 2^7 or 128 possible component state vectors that must be evaluated to determine system reliability. Evaluate the effectiveness of the following modular decomposition:

$$A_1 = \{c_1, c_2, c_3\}$$
$$A_2 = \{c_4\}$$
$$A_3 = \{c_5, c_6\}$$
$$A_4 = \{c_7\}.$$

All 4 modules will also be binary systems. Modules A_1, A_2, A_3, and A_4 have 8, 2, 4, and 2 component state vectors, respectively. The organizing structure must be evaluated for 2^4 or 16 component state vectors. Thus, the modular decomposition has reduced the total number of component state vectors from 128 to 32.

Unfortunately, each of the 4 techniques for calculating exact system reliability is burdensome for large, complex systems. As an alternative, system reliability can be approximated with lower and upper bounds.

2.2.5 Bounding System Reliability

Up to this point, the discussion has been restricted to the case where the random variables for the n component states are mutually independent. Esary and Proschan [1970] were first to discuss the less restrictive case of associated components.

DEFINITION - The vector of random component states $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ are associated if $\mathrm{Cov}[f(\mathbf{X}), g(\mathbf{X})] \geq 0$ for all increasing functions f and g.

Lug nuts that share a common load are a good example of associated components.

Reliability bounds have been constructed for mutually independent and associated random variables. The bounds are more explicit if the random variables are independent. Each of the bounds is based on the following commonly known result:

THEOREM 2.9 If $X_1, X_2, \ldots, X_n$ are binary associated random variables representing the n component states, then

$$\Pr[\prod_{i=1}^{n} X_i = 1] \geq \prod_{i=1}^{n} \Pr[X_i = 1]$$

and

$$\Pr[\coprod_{i=1}^{n} X_i = 1] \leq \coprod_{i=1}^{n} \Pr[X_i = 1].$$

2.2.5.1 Trivial Bounds

Trivial bounds were obtained by comparing any coherent system with the worst possible structure (series) and the best possible structure (parallel) for the components.

THEOREM 2.10 Let ϕ be a coherent system composed of associated components with reliabilities given by $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. Then

$$\prod_{i=1}^{n} p_i \le r(\mathbf{p}) \le \coprod_{i=1}^{n} p_i .$$

The lower and upper bounds are derived by taking the expected value of the bounds for ϕ given in Theorem 2.1 and applying Theorem 2.9.

2.2.5.2 Path/Cut Bounds

Path/Cut Bounds were developed by Esary and Proschan [1963a] from the minimal path and minimal cut sets. The lower bound comes from the minimal cut sets while the upper bound comes from the minimal path sets.

THEOREM 2.11 Let ϕ be a coherent system of associated components. As before, let $\alpha_j(\mathbf{x})$ be the indicator variable of the j^{th} minimal path set and $\beta_j(\mathbf{x})$ be the indicator variable of the j^{th} minimal cut set. Then

$$\prod_{j=1}^{t} \Pr[\beta_j(X) = 1] \le r(\mathbf{p}) \le \coprod_{j=1}^{s} \Pr[\alpha_j(X) = 1] .$$

The bounds come from the relationship between the structure function and the indicator variables and the application of Theorem 2.9.

When the components are independent, the bounds of Theorem 2.11 can be explicitly derived from the component reliabilities.

THEOREM 2.12 Let ϕ be a coherent system of independent components with minimal path sets $P_1, P_2, \ldots, P_s$ and minimal cut sets $C_1, C_2, \ldots, C_t$. Then

$$\prod_{j=1}^{t} \coprod_{i \in C_j} p_i \leq r(\mathbf{p}) \leq \coprod_{j=1}^{s} \prod_{i \in P_j} p_i .$$

2.2.5.3 Min/Max Bounds

Min/Max Bounds were developed by Barlow and Proschan [1981]. The lower bound comes from the minimal path sets while the upper bound comes from the minimal cut sets.

THEOREM 2.13 Let ϕ be a coherent system with minimal path sets $P_1, P_2, \ldots, P_s$ and minimal cut sets $C_1, C_2, \ldots, C_t$. Then the following bounds always hold:

$$\operatorname*{Max}_{j=1,2,\ldots,s} \Pr[\operatorname*{Min}_{i \in P_j} X_i = 1] \leq r(\mathbf{p}) \leq \operatorname*{Min}_{j=1,2,\ldots,t} \Pr[\operatorname*{Max}_{i \in C_j} X_i = 1] .$$

If the components are associated, then

$$\operatorname*{Max}_{j=1,2,\ldots,s} \{\prod_{i \in P_j} p_i\} \leq r(\mathbf{p}) \leq \operatorname*{Min}_{j=1,2,\ldots,t} \{\coprod_{i \in C_j} p_i\} .$$

It has been shown that the Path/Cut Bounds of Theorems 2.11 and 2.12 are not always tighter than the trivial bounds of Theorem 2.10. On the other hand, the Min/Max Bounds of Theorem 2.13 are always tighter than the trivial bounds.

2.2.5.4 Combining Bounds

Unfortunately, no bound superiority can be established between the Path/Cut Bounds and the Min/Max Bounds when the components are mutually independent. However, minimal cut sets generally provide tighter bounds for mutually independent components with high reliabilities and minimal path sets generally provide tighter bounds for mutually independent components with low reliabilities. Thus, a combination of the bounds in Theorems 2.12 and 2.13 seems appropriate for mutually independent components.

THEOREM 2.14 Let ϕ be a coherent system of independent components with minimal path sets $P_1, P_2, \ldots, P_s$ and minimal cut sets $C_1, C_2, \ldots, C_t$. Then

$$\mathrm{Max}\{\prod_{j=1}^{t} \coprod_{i \in C_j} p_i, \max_{j=1,2,\ldots,s} \{\prod_{i \in P_j} p_i\}\} \le r(\mathbf{p}) \le \mathrm{Min}\{\coprod_{j=1}^{s} \prod_{i \in P_j} p_i, \min_{j=1,2,\ldots,t} \{\coprod_{i \in C_j} p_i\}\}.$$

2.2.5.5 Improved Path/Cut Bounds

Bodin [1970] developed better Path/Cut Bounds with modular decomposition. Path/Cut Bounds are determined for each module. The bounds are then used to determine Path/Cut Bounds for the system. Bodin [1970] showed that these bounds were always tighter than the Path/Cut Bounds found directly from the entire system.

EXAMPLE 2.9 Determine the trivial, Path/Cut, Min/Max, and Improved Path/Cut Bounds for the structure in Figure 2.9 given that all components are mutually independent with common reliability p.

The minimal cut sets are {1}, {2}, {3,4}, and {3,5}.

The minimal path sets are {1,2,3} and {1,2,4,5}.

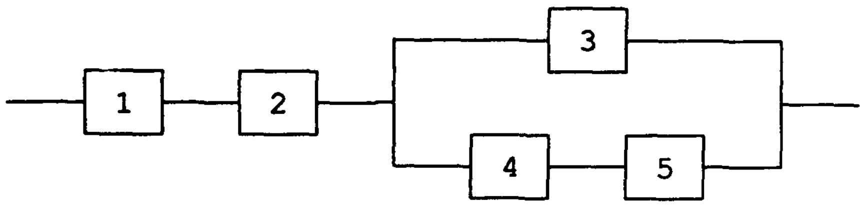


Figure 2.9 Structure for Reliability Bounds.

The trivial bounds are $p^5 \le r(\mathbf{p}) \le 1 - (1 - p)^5$.

The Path/Cut Bounds are

$p^2[1-(1-p)^2]^2 \le r(\mathbf{p}) \le [1-(1-p^3)(1-p^4)]$.

The Min/Max Bounds are

$\text{Max}\{p^3,p^4\} \le r(\mathbf{p}) \le \text{Min}\{p,p,1-(1-p)^2,1-(1-p)^2\}$.

Consider the following modular decomposition:

$A_1 = \{x_1,x_2\}$, $A_2 = \{x_3\}$, and $A_3 = \{x_4,x_5\}$.

Then $\chi_1 = x_1 \prod x_2$, $\chi_2 = x_3$, and $\chi_3 = x_4 \prod x_5$.

The organizing structure is $\psi = \chi_1 \prod (\chi_2 \amalg \chi_3)$.

The Path/Cut Bounds for χ_1 and χ_3 are $p^2 \le r(\mathbf{p}) \le p^2$.

The Path/Cut Bounds for χ_2 are $p \le r(\mathbf{p}) \le p$.

The minimal cut sets of ψ are $\{\chi_1\}$ and $\{\chi_2,\chi_3\}$.

The minimal path sets of ψ are $\{\chi_1,\chi_2\}$ and $\{\chi_1,\chi_3\}$.

The Improved Path/Cut Bounds are

$p^2[1-(1-p)(1-p^2)] \le r(\mathbf{p}) \le 1-(1-p^2p)(1-p^2p^2)$.

The exact reliability function is given by

$r(\mathbf{p}) = p^2[1-(1-p)(1-p^2)] = p^3 + p^4 - p^5$.

Tables 2.1 and 2.2 compare the lower and upper boundary points for various levels of p. Notice the lower Improved Path/Cut Bounds give the exact reliability. The dominance of some of the bounds can be seen from the tables. Min/Max Bounds are better than trivial bounds. As shown in Table 2.1, Path/Cut Bounds are better for high p and Min/Max Bounds are better for low p. In Table 2.2, the opposite is true. Improved Path/Cut Bounds are at least as good as Path/Cut Bounds.

Table 2.1 Lower Bound Comparison.

p	Trivial	Path/Cut	Min/Max	Improved
.99	.95099	.979904	.97030	.979905
.95	.77378	.89799	.85738	.89810
.90	.59049	.79388	.72900	.79461
.75	.23730	.49438	.42188	.50098
.5	.03125	.14062	.12500	.15625
.25	9.76 E -4	.01196	.01563	.01855
.1	1.00 E -5	3.61 E -4	1.00 E -3	1.09 E -3
.05	3.12 E -7	2.38 E -5	1.25 E -4	1.31 E -4
.01	1.00 E -10	3.96 E -8	1.00 E -6	1.01 E -6

Table 2.2 Upper Bound Comparison.

p	Trivial	Path/Cut	Min/Max	Improved
.99	.99999	.99883	.99000	.99883
.95	.99999	.97354	.95000	.97354
.90	.99999	.90680	.90000	.90680
.75	.99902	.60480	.75000	.60480
.5	.96875	.17969	.50000	.17969
.25	.76270	.01947	.25000	.01947
.1	.40951	1.10 E -3	.10000	1.10 E -3
.05	.22622	1.31 E -4	.05000	1.31 E -4
.01	.04901	1.01 E -6	.01000	1.01 E -6

2.2.5.6 Inclusion-Exclusion Bounds

Barlow and Proschan [1981] also developed bounds using the inclusion-exclusion principle of Feller [1968]. Let Σ_j be the j^{th} summation term in Theorem 2.8. The reliability function can be bounded as follows:

$$\begin{aligned} r(\mathbf{p}) &\leq \Sigma_1 \\ r(\mathbf{p}) &\geq \Sigma_1 - \Sigma_2 \\ &\vdots \\ r(\mathbf{p}) &= \Sigma_1 - \Sigma_2 + \Sigma_3 - \ldots + (-1)^{s+1} \Sigma_s. \end{aligned}$$

Unfortunately, the bounds do not consistently improve as more terms are added. In fact, Inclusion-Exclusion Bounds are not restricted between 0 and 1. The only guarantee is that exact reliability will be found after including every summation. Still, only a few terms may be necessary to provide narrow bounds for system reliability.

2.3 Dynamic Properties

The structural and stochastic properties describe the deterministic and probabilistic relationships between the system state and the states of the components at a fixed moment in time. The next logical step is to consider the relationship between the lifetime distribution of the system and the lifetime distributions of the components. Barlow and Proschan [1981] summarized the dynamic properties for the binary model.

2.3.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

T	system lifetime.
t	fixed time.
$R(t)$	survivor function; $R(t) = \Pr[T > t]$.
$F(t)$	cumulative distribution function; $F(t) = \Pr[T \le t]$.
$f(t)$	probability density function.
$h(t)$	hazard function; $h(t) = f(t)/R(t)$.
$H(t)$	cumulative hazard function; $H(t) = \int_0^t h(\tau)d\tau$.
$X_i(t)$	state of component i at time t, $i=1,2,\ldots,n$.
T_i	lifetime for component i, $i=1,2,\ldots,n$.
$\mathbf{X}(t)$	vector of component states at time t; $\mathbf{X}(t) = (X_1(t),X_2(t),\ldots,X_n(t))$.
$\phi(\mathbf{X}(t))$	system state at time t.
$R_i(t)$	survivor function for component i at time t; $R_i(t) = \Pr[X_i(t) = 1] = E[X_i(t)] = \Pr[T_i > t]$ for $i=1,2,\ldots,n$.
$R(t)$	survivor function for the system at time t; $R(t) = \Pr[\phi(\mathbf{X}(t)) = 1] = E[\phi(\mathbf{X}(t))] = \Pr[T > t]$.
$\mathbf{R}(t)$	vector of component survivor functions at time t; $\mathbf{R}(t) = (R_1(t),R_2(t),\ldots,R_n(t))$.
$r(\mathbf{R}(t))$	reliability function at time t; $r(\mathbf{R}(t)) = E[\phi(\mathbf{X}(t))]$.
μ_i	mean of the lifetime distribution for component i.

2.3.2 Lifetime Distribution Functions

Suppose that the nonnegative random variable T denotes the lifetime of the system. There are several ways to completely specify the system's lifetime distribution. The survivor function R(t) gives the probability that the lifetime exceeds a given time t so that $R(t) = \Pr[T > t]$ for $t \geq 0$. Since $R(t) = 1 - F(t)$, where F(t) is the cumulative distribution function, $R(0) = 1$, $\lim_{t \to \infty} R(t) = 0$, and R(t) is decreasing. The probability density function f(t) indicates the likelihood of failure during a time period Δt so that $\Pr[t \leq T \leq t+\Delta t] = \int_t^{t+\Delta t} f(\tau)\, d\tau$. As with any probability density function, $\int_{-\infty}^{\infty} f(\tau)\, d\tau = 1$ and $f(t) \geq 0$ for $t \geq 0$. When the derivative exists, $f(t) = -R'(t)$. The hazard function h(t) measures the degree of risk to failure at time t. It is defined as the instantaneous failure rate given the component or system has survived to time t so that

$$\begin{aligned} h(t) &= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \Pr[t < T \leq t+\Delta t \mid T > t] \\ &= \lim_{\Delta t \to 0} \frac{R(t) - R(t+\Delta t)}{R(t)\, \Delta t} \\ &= \frac{-R'(t)}{R(t)} = \frac{f(t)}{R(t)} \quad \text{for } t \geq 0. \end{aligned}$$

The hazard function is useful for determining how the risk of a component or system changes with time.

Knowing any one of these representations allows the others to be generated. If R(t) is known, then

$$f(t) = -R'(t) \quad \text{and} \quad h(t) = \frac{-R'(t)}{R(t)}.$$

If f(t) is known, then

$$R(t) = \int_t^{\infty} f(\tau)\, d\tau \quad \text{and} \quad h(t) = \frac{f(t)}{\int_t^{\infty} f(\tau)\, d\tau}.$$

If h(t) is known, then

$$f(t) = h(t)\exp\left[-\int_0^t h(\tau)\, d\tau\right] \quad \text{and} \quad R(t) = \exp\left[-\int_0^t h(\tau)\, d\tau\right].$$

There are several other functions that can be used to completely specify the lifetime distribution.

2.3.3 Lifetime Distribution Classes

Lifetime distributions can be classified by the shape of R(t), f(t), or h(t). Since it is useful to study the way risk changes with time, lifetime distributions are most often grouped together according to the shape of h(t).

DEFINITIONS. A lifetime distribution belongs to the increasing failure rate (IFR) class if h(t) is an increasing function. A lifetime distribution belongs to the decreasing failure rate (DFR) class if h(t) is a decreasing function.

EXAMPLE 2.10 Classify the exponential and Weibull distributions according to the shape of h(t).
For the exponential distribution with $f(t) = \lambda e^{-\lambda t}$, and $R(t) = e^{-\lambda t}$, $h(t) = \lambda$ for $t \geq 0$. Therefore, the failure rate for the exponential distribution is constant and the distribution belongs to both the IFR and DFR class for all λ. For the Weibull

distribution with $f(t) = \alpha\lambda^{\alpha}t^{\alpha-1}\exp[-(\lambda t)^{\alpha}]$ and $R(t) = \exp[-(\lambda t)^{\alpha}]$, $h(t) = \alpha\lambda^{\alpha}t^{\alpha-1} = \alpha\lambda(\lambda t)^{\alpha-1}$ for $t \geq 0$. Therefore, the Weibull distribution belongs to the DFR class if $0 < \alpha < 1$, the IFR class if $\alpha > 1$, and both classes if $\alpha = 1$ (exponential case).

Esary and Proschan [1963b] gave an example where independent components with lifetime distributions belonging to the IFR class did not result in a system with a lifetime distribution in the IFR class. This led to the definition of a larger class of lifetime distributions known as the increasing failure rate on the average (IFRA) class. This class has been shown to be the smallest class where a coherent system of IFRA components remains IFRA.

DEFINITIONS. A lifetime distribution belongs to the _increasing failure rate on the average (IFRA) class_ if $\frac{1}{t}\int_0^t h(\tau)\,d\tau = \frac{H(t)}{t}$ is increasing in $t \geq 0$.

A lifetime distribution belongs to the _decreasing failure rate on the average (DFRA) class_ if $\frac{H(t)}{t}$ is decreasing in $t \geq 0$.

Equivalent definitions state that a lifetime distribution belongs to the IFRA class if $-(1/t)\log R(t)$ is increasing in $t \geq 0$ or $[R(t)]^{1/t}$ is decreasing in $t \geq 0$.

The IFRA and DFRA classes of lifetime distributions can be enlarged further. The following classes are important when considering different component replacement policies.

DEFINITIONS. A lifetime distribution belongs to the new better than used (NBU) class if $R(t+x) \le R(t)R(x)$ for any $t \ge 0$ and $x \ge 0$. A lifetime distribution belongs to the new worse than used (NWU) class if $R(t+x) \ge R(t)R(x)$ for any $t \ge 0$ and $x \ge 0$.

The NBU (NWU) class says that the lifetime of a new component is stochastically greater (less) than the remaining lifetime of a component still working at time t.

DEFINITIONS. A lifetime distribution belongs to the new better than used in expectation (NBUE) class if the distribution has a finite mean μ and

$\int_t^\infty R(\tau)\,d\tau \le \mu R(t)$ for $t \ge 0$. A lifetime distribution belongs to the new worse than used in expectation (NWUE) class if the distribution has a finite mean μ and $\int_t^\infty R(\tau)\,d\tau \ge \mu R(t)$ for $t \ge 0$.

The NBUE (NWUE) class says that the expected lifetime of a new component is greater (less) than the expected remaining lifetime of a component still working at time t. The relationships between the various classes of lifetime distributions is given below:

$$\text{IFR} \Rightarrow \text{IFRA} \Rightarrow \text{NBU} \Rightarrow \text{NBUE}$$

$$\text{DFR} \Rightarrow \text{DFRA} \Rightarrow \text{NWU} \Rightarrow \text{NWUE}.$$

2.3.4 Distribution Class Closure

A lifetime distribution class is said to be closed when an operation on the lifetime distribution of the components always results in a lifetime distribution belonging to the same class. Closure has been studied with respect to the following operations: forming coherent systems, the addition of lifetimes (convolutions), and the linear combination of lifetimes (arbitrary mixtures).

Table 2.3 summarizes the results given by Barlow and Proschan [1981] for the eight lifetime distribution classes and the three reliability operations given above. Closure has not been proved or disproved for the NWUE class under the operation of arbitrary mixtures.

Table 2.3 Closure of Lifetime Distribution Classes for Various Reliability Operations.

Lifetime Distribution Classes	Reliability Operations		
	Coherent Systems	Convolutions	Arbitrary Mixtures
IFR	Not	Closed	Not
IFRA	Closed	Closed	Not
DFR	Not	Not	Closed
DFRA	Not	Not	Closed
NBU	Closed	Closed	Not
NBUE	Not	Closed	Not
NWU	Not	Not	Not
NWUE	Not	Not	?

2.3.5 Exact System Lifetime Distribution

For the dynamic situation, the problem changes to finding the lifetime distribution of the system from the lifetime distributions of the n independent components that comprise the system. Let $X_i(t)$ be the random state of component i at time t and let T_i be the component lifetime. Let $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$ be the vector of random component states at time t. Let $\phi(\mathbf{X}(t))$ be the system state at time t and let T be the system lifetime. The survivor functions for the components and the system are given by $R_i(t) = \Pr[X_i(t) = 1] = E[X_i(t)] = \Pr[T_i > t]$ for $i=1,2,\ldots,n$ and $R(t) = \Pr[\phi(\mathbf{X}(t)) = 1] = E[\phi(\mathbf{X}(t))] = \Pr[T > t]$.

Let $\mathbf{R}(t) = (R_1(t), R_2(t), \ldots, R_n(t))$. Since $X_i(t)$, $i=1,2,\ldots,n$ are mutually independent, R(t) is a function of $\mathbf{R}(t)$. The relationship is given by the reliability function $r(\cdot)$ which is derived from $\phi(\mathbf{x})$ so that $R(t) = r(\mathbf{R}(t))$.

EXAMPLE 2.11 Find R(t), f(t), and h(t) for a series system of n independent components with $R_i(t) = \exp[-\lambda_i t]$, $i=1,2,\ldots,n$.

$\phi(\mathbf{x}) = \prod_{i=1}^{n} x_i$ and $E[\phi(\mathbf{X}(t))] = \prod_{i=1}^{n} E[X_i(t)]$ so that

$$R(t) = \prod_{i=1}^{n} e^{-\lambda_i t} = \exp[-\sum_{i=1}^{n} \lambda_i t]$$

$$f(t) = -R'(t) = \left(\sum_{i=1}^{n} \lambda_i\right)\exp\left[-\sum_{i=1}^{n} \lambda_i t\right] \text{ and}$$

$$h(t) = \frac{f(t)}{R(t)} = \sum_{i=1}^{n} \lambda_i .$$

Of course, the lifetime distribution of the system becomes intractable for all but the simplest structures.

2.3.6 Bounding System Reliability

Barlow and Proschan [1981] used the closure theorems to develop bounds on system reliability. Let $r(\mathbf{R}(t))$ be the reliability function of a coherent system of n mutually independent components. Suppose that T_i have unknown lifetime distributions with known means μ_i and that each of the distributions belongs to the IFR class. Barlow and Proschan [1981] showed that if the lifetime distribution for component i belongs to the IFR class, then the largest lower bound on $R_i(t)$ is $\exp[-t/\mu_i]$ for $t < \mu_i$. Using the fact that the reliability function is increasing in each argument,

$R(t) = r(R_1(t), R_2(t), \ldots, R_n(t)) \geq r(e^{-t/\mu_1}, e^{-t/\mu_2}, \ldots, e^{-t/\mu_n})$ for $t < \text{Min}\{\mu_1, \mu_2, \ldots, \mu_n\}$.

EXAMPLE 2.12 Find a lower bound on R(t) for the system of 3 mutually independent components given in Figure 2.10. Assume the lifetime distributions are in the IFR class and $\mu_1 = 10$ $\mu_2 = 5$ and $\mu_3 = 8$.

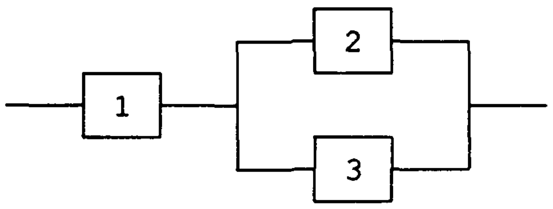


Figure 2.10 Structure for Bounding Reliability.

$\phi(\mathbf{x}) = x_1[1-(1-x_2)(1-x_3)]$ so

$R(t) \geq e^{-t/10}[1 - (1 - e^{-t/5})(1 - e^{-t/8})]$ for $t < 5$.

2.4 Summary

This chapter presented a broad review of the structural, stochastic, and dynamic properties for the binary reliability model. It was designed to provide a convenient reference for comparing the corresponding properties of the multistate and continuous models. No original material was contributed.

3. THE MULTISTATE MODEL

This chapter presents the structural and stochastic properties of the most general multistate coherent model. The chapter makes comparisons to some of the more restrictive multistate models developed by other authors.

3.1 Structural Properties

Structural properties characterize the deterministic relationship between the state of the system and the states of the components at a fixed moment in time.

3.1.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components comprising the system.
x_i	state of component i; $x_i \in \{0,1,\ldots,M_i\}$, $i=1,2,\ldots,n$.
M_i	best state of component i; $M_i \in \{0,1,2,\ldots\}$ $\forall i$.
Ω_i	state space of component i; $\Omega_i = \{0,1,\ldots,M_i\}$ $\forall i$.
$\mathbf{x}$	component state vector; $\mathbf{x} = (x_1,x_2,\ldots,x_n)$.
S	component state space; $S = \{\mathbf{x} \mid x_i \in \Omega_i \ \forall i\}$.
ϕ	state of the system; $\phi \in \{0,1,\ldots,M\}$.
M	best state of the system; $M \in \{0,1,2,\ldots\}$.
Ω	state space of the system; $\Omega = \{0,1,\ldots,M\}$.
$\phi(\mathbf{x})$	structure function; system state for $\mathbf{x} \in S$.
S_k	kth equivalence class; $S_k = \{\mathbf{x} \in S \mid \phi(\mathbf{x}) = k\}$.
$(j_i,\mathbf{x})$	$(x_1,x_2,\ldots,x_{i-1},j,x_{i+1},\ldots,x_n)$, $i=1,2,\ldots,n$ and $j \in \Omega_i$.
$\mathbf{y} > \mathbf{x}$	$y_i \geq x_i$ $\forall i$ and $y_i > x_i$ for at least one i.
$\mathbf{y} \geq \mathbf{x}$	$y_i \geq x_i$ $\forall i$.

$\mathbf{x}_M$	$\mathbf{x}$ with all components at best state; $(M_1, M_2, \ldots, M_n)$.
$\mathbf{x}_0$	$\mathbf{x}$ with all components at worst state; $(0, 0, \ldots, 0)$.
P_j	j^{th} minimal path set; $j=1,2,\ldots,s$.
$\mathbf{J}$	$(j, j, \ldots, j)$.
L_k	the set of lower boundary points to level k, $k=1,2,\ldots,M$.
L_{kj}	the j^{th} lower boundary point to level k, $k=1,2,\ldots,M$ and $j=1,2,\ldots,s_k$.
U_k	the set of upper boundary points to level k, $k=0,1,\ldots,M-1$.
U_{kj}	the j^{th} upper boundary point to level k, $k=0,1,\ldots,M-1$ and $j=1,2,\ldots,t_k$.
$\mathbf{x} \vee \mathbf{y}$	$(x_1 \vee y_1, x_2 \vee y_2, \ldots, x_n \vee y_n)$.
$\mathbf{x} \wedge \mathbf{y}$	$(x_1 \wedge y_1, x_2 \wedge y_2, \ldots, x_n \wedge y_n)$.
$\mathbf{x} \amalg \mathbf{y}$	$(x_1 \amalg y_1, x_2 \amalg y_2, \ldots, x_n \amalg y_n)$ where $x_i \amalg y_i = \text{Max}\{x_i, y_i\}$.
$\mathbf{x} \Pi \mathbf{y}$	$(x_1 \Pi y_1, x_2 \Pi y_2, \ldots, x_n \Pi y_n)$ where $x_i \Pi y_i = \text{Min}\{x_i, y_i\}$.
y_{ij}	indicator variable; $y_{ij} = 1$ if $x_i \geq j$.
$\phi^k(\mathbf{x})$	indicator variable; $\phi^k(\mathbf{x}) = 1$ if $\phi(\mathbf{x}) \geq k$.
$\mathcal{P}_k(\mathbf{x})$	minimal path set generator function.
$\mathcal{U}_k(\mathbf{x})$	minimal cut set generator function.
ϕ^D	dual structure function.
C	set of components; $C = \{c_1, c_2, \ldots, c_n\}$.
x_i^D	state of component i in ϕ^D.
$I_\phi(i)$	structural importance of component i in ϕ.

A_j	set of components in module j.
χ_j	structure function of module j.
ψ	organizing structure function.

3.1.2 Introduction

Hudson [1981] introduced the most general multistate model. For this model, each component and the system are allowed to have a different number of discrete states.

For a multistate system with n components, the state of the ith component is given by the discrete variable x_i where

$$x_i = \begin{cases} 0 & \text{if component } i \text{ is in the worst state} \\ 1 \\ \vdots & \text{intermediate states of degradation} \\ M_i - 1 \\ M_i & \text{if component } i \text{ is in the best state} \end{cases}$$

for $i=1,2,\ldots,n$ and $M_i < \infty$. The state space for component i has $M_i + 1$ elements and is designated by Ω_i. The component state vector, $\mathbf{x} = (x_1,x_2,\ldots,x_n)$, is the vector of component states and the component state space, $S = \{\mathbf{x} \mid x_i \in \Omega_i, \forall i\}$, is the set of possible component state vectors.

The state of the system is given by the variable ϕ where

$$\phi = \begin{cases} 0 & \text{if the system is in the worst state} \\ 1 \\ \vdots & \text{intermediate states of degradation} \\ M-1 \\ M & \text{if the system is in the best state.} \end{cases}$$

The state space for the system has $M + 1$ elements and is designated by $\Omega = \{0,1,2,\ldots,M\}$. The model assumes that the state of the system is completely determined by the states of the n components. The relationship is described by the structure function $\phi(\mathbf{x})$ which can be concisely written as $\phi\colon \{0,1,\ldots,M_1\}^n \rightarrow \{0,1,\ldots,M\}$ or $\phi\colon \Omega_1^n \rightarrow \Omega$ or $\phi\colon S \rightarrow \Omega$.

Hudson and Kapur [1983b] noted that customers do not always wish to make a distinction between the system states for different component state vectors. For example, the customer may define a multistate structure function so that $\phi(1,1,1) = \phi(1,1,2)$, implying that the increase of x_3 is not significant from the customer's perspective.

The same logic is implicit in binary models. For a series system consisting of n binary components, 2^n-1 component state vectors are equivalent, each resulting in a failed system. Only the component state vector $(1,1,\ldots,1)$ is significantly different from the others, causing the system to function.

Component state vectors with the same system state are said to belong to the same equivalence class.

DEFINITION. The <u>k^{th} equivalence class S_k</u> is given by

$$S_k = \{\mathbf{x} \in S \mid \phi(\mathbf{x}) = k\},\ k=0,1,\ldots,M.$$

$S_0, S_1, \ldots, S_M$ are disjoint sets that partition S into M+1 equivalence classes. Therefore, $S = \bigcup_{k=0}^{M} S_k$. For the binary case, S_1 contains the path vectors, while S_0 contains the cut vectors.

3.1.3 Special Structures

The earliest articles in multistate reliability [Barlow and Wu, 1978; El-Neweihi, Proschan, and Sethuraman, 1978] defined a series structure so that $\phi(\mathbf{x}) = \text{Min}\{x_1, x_2, \ldots, x_n\}$, a parallel structure so that $\phi(\mathbf{x}) = \text{Max}\{x_1, x_2, \ldots, x_n\}$, and a k-out-of-n structure so that $\phi(\mathbf{x}) = x_{(n-k+1)}$ where $x_{(1)}, \ldots, x_{(n)}$ is an increasing sequence of $x_1, x_2, \ldots, x_n$. These definitions have two drawbacks. First, they implicitly assume that the component state spaces use the same scale. It seems more reasonable to allow a series system where $\phi(2,3,3) < \phi(3,2,3)$ if the first component is more important to the customer. Second, the definitions restrict the state space of the system to the state space of the component with the highest number of states. It seems more reasonable to allow the customer to specify the number of system states based on his interpretation of the system.

Hudson [1981] defined a series, parallel, and k-out-of-n structure intuitively by characterizing the set of component state vectors in the lowest and highest equivalence classes.

DEFINITION. ϕ is a <u>series structure function</u> iff

i. $S_0 = \{\mathbf{x} \in S \mid \mathbf{x} = (0_i, \mathbf{x})$ for some $i=1,2,\ldots,n\}$ and

ii. $S_M = \{(M_1, M_2, \ldots, M_n)\} = \{\mathbf{x}_M\}$.

DEFINITION. ϕ is a <u>parallel structure function</u> iff

i. $S_0 = \{(0,0,\ldots,0)\} = \{\mathbf{x}_0\}$ and

ii. $S_M = \{\mathbf{x} \in S \mid \mathbf{x} = ((M_i)_i, \mathbf{x})$ for some $i=1,2,\ldots,n\}$.

DEFINITION. ϕ is a <u>k-out-of-n structure function</u> iff

i. $S_0 = \{\mathbf{x} \in S \mid n-k+1$ or more of the components of $\mathbf{x}$ are at their minimum value $0\}$ and

ii. $S_M = \{\mathbf{x} \in S \mid k$ or more of the components of $\mathbf{x}$ are at their maximum value $M_1\}$.

Hudson's definitions eliminate the drawbacks of the earlier definitions. However, the definitions create a new problem because they do not specify which vectors are contained in the equivalence classes between S_0 and S_M. Therefore, several structure functions exist for each category. This leads to problems interpreting theorems that involve the concepts of series and parallel systems.

For the binary case, a series system of n components has one minimal path vector and n minimal cut vectors. Parallel systems of n components have n minimal path vectors and one minimal cut vector. A k-out-of-n system has $\binom{n}{k}$ minimal path vectors and $\binom{n}{n-k+1}$ minimal cut vectors.

Many authors have generalized the concepts of minimal path and minimal cut vectors for multistate systems. Hudson [1981] referred to the minimal path (cut) vectors as the lower (upper) boundary points to level k. Janan [1985] gave definitions closely resembling the following:

DEFINITION. $\mathbf{x}$ is a <u>lower boundary point to level k</u> if $\phi(\mathbf{x}) \geq k$ and $\mathbf{y} < \mathbf{x}$ implies that $\phi(\mathbf{y}) < k$, $k=1,2,\ldots,M$.

DEFINITION. $\mathbf{x}$ is an <u>upper boundary point to level k</u> if $\phi(\mathbf{x}) \leq k$ and $\mathbf{y} > \mathbf{x}$ implies that $\phi(\mathbf{y}) > k$, $k=0,\ldots,M-1$.

For the binary model, the lower boundary points to level 1 are the minimal path vectors, while the upper boundary points to level 0 are the minimal cut vectors.

Using the logic of the binary model, the following new definitions are offered for series, parallel, and k-out-of-n structure functions:

DEFINITION. ϕ is a <u>series structure function</u> iff ϕ has one lower boundary point to level j, j=1,2,...,M and n upper boundary points to level j, j=0,1,...,M-1.

DEFINITION. ϕ is a <u>parallel structure function</u> iff ϕ has n lower boundary points to level j, j=1,2,...,M and one upper boundary point to level j, j=0,1,...,M-1.

DEFINITION. ϕ is a <u>k-out-of-n structure function</u> iff ϕ has $\binom{n}{k}$ lower boundary points to level j, j=1,2,...,M and $\binom{n}{n-k+1}$ upper boundary points to level j, j=0,...,M-1.

Series and parallel systems are special cases of the k-out-of-n structure. A series system is an n-out-of-n structure while a parallel system is a 1-out-of-n structure. The definitions reduce to the binary concepts of series, parallel, and k-out-of-n when M = 1.

EXAMPLE 3.1 Suppose that n=2, M_1=3, M_2=2, M=2, and $\phi(\mathbf{x})$ is enumerated by the customer in Table 3.1. Show that $\phi(\mathbf{x})$ does not meet the early definitions for a parallel structure, but that it does meet the new definition.

Table 3.1 $\phi(\mathbf{x})$ for Example 3.1.

	$\phi(\mathbf{x})$	x_2 = 0	1	2
x_1	0	0	1	2
	1	0	1	2
	2	1	1	2
	3	2	2	2

$\phi(\mathbf{x}) \neq \text{Max}\{x_1, x_2\}$, $\forall$ $\mathbf{x}$. For example, $\phi(2,1) = 1$. Using Hudson's definition, $S_0 \neq \{(0,0)\}$. For example, $\mathbf{x} = (1,0)$ is also an element of S_0. The lower boundary points to level 1 are (2,0) and (0,1). The lower boundary points to level 2 are (3,0) and (0,2). The upper boundary points to levels 0 and 1 are (1,0) and (2,1), respectively. Therefore, $\phi(\mathbf{x})$ meets the new definition for a parallel structure function.

3.1.4 Coherent Structures

The authors who extended reliability theory to the multistate model wished to eliminate unrealistic structure functions. As in the binary model, the term "coherent system" was used. However, there is no generally accepted definition of coherence for a multistate system. Ohi and Nishida [1984] presented a summary of the definitions from many authors. Let Ω_i represent the state space of the ith component, Ω represent the state space of the system, and ϕ represent the structure function.

DEFINITION [Barlow and Wu, 1978]. ϕ is BW-coherent iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$ and

ii. $\phi(\mathbf{x}) = \underset{j=1,2,\ldots,s}{\text{Max}} \; \underset{i\in P_j}{\text{Min}} \; x_i$ where P_j is the j^{th} minimal path set defined as if the components were binary.

In words, ϕ is defined as the maximum of the worst component in each minimal path set.

DEFINITION [El-Neweihi, Proschan, Sethuraman, 1978]. ϕ is <u>EPS-coherent</u> iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$,

ii. $\phi(\mathbf{x})$ is increasing,

iii. For every state j of every component i, there exists a vector $(\cdot_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) = j$ while $\phi(k_i,\mathbf{x}) \neq j$ for all states $k \neq j$, and

vi. $\phi(\mathbf{J}) = j$ for $j=0,1,\ldots,M$.

Griffith [1980] gave definitions for a multistate monotone system (MMS) and for three types of coherence: strongly coherent, coherent, and weakly coherent.

DEFINITION [Griffith, 1980]. ϕ is a <u>multistate monotone system</u> (MMS) iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$,

ii. $\phi(\mathbf{x})$ is increasing, and

iii. $\underset{i=1,2,\ldots,n}{\text{Min}} \; x_i \leq \phi(\mathbf{x}) \leq \underset{i=1,2,\ldots,n}{\text{Max}} \; x_i$.

DEFINITION [Griffith, 1980]. ϕ is <u>G-strongly coherent</u> iff ϕ is a MMS and for any component i and any state j, there exists a vector $(\cdot_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) = j$ while $\phi(k_i,\mathbf{x}) \neq j$ for all states $k \neq j$.

DEFINITION [Griffith, 1980]. ϕ is <u>G-coherent</u> iff ϕ is a MMS and for any component i and state $j \geq 1$, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi((j-1)_i, \mathbf{x}) < \phi(j_i, \mathbf{x})$.

DEFINITION [Griffith, 1980]. ϕ is <u>G-weakly coherent</u> iff ϕ is a MMS and for any component i and any state j, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi(j_i, \mathbf{x}) \neq \phi(k_i, \mathbf{x})$ for some state $k \neq j$.

As interpreted by Block and Savits [1982], G-strong coherence implies that every state of each component is relevant to the same system state. G-coherence implies that every state of each component is relevant to the system. G-weak coherence implies that each component is relevant to the system. Each of Griffith's definitions is progressively less restrictive.

Butler [1982] gave a definition that was similar to the G-weakly coherent system proposed by Griffith.

DEFINITION [Butler, 1982]. ϕ is <u>B-coherent</u> iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$,

ii. $\phi(\mathbf{x})$ is increasing,

iii. For every component i, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi(M_i, \mathbf{x}) > \phi(0_i, \mathbf{x})$, and

iv. $\phi(\mathbf{x}_0) = 0$ and $\phi(\mathbf{x}_M) = M$.

Natvig [1982] gave definitions for two categories of coherence which he called type 1 and type 2:

DEFINITION [Natvig, 1982]. ϕ is <u>N-type1 coherent</u> iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$,

ii. $\phi(\mathbf{x})$ is increasing,

iii. For every state j of every component i, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi(j_i, \mathbf{x}) \geq j$ and $\phi((j-1)_i, \mathbf{x}) \leq j-1$, and

iv. $\phi(\mathbf{J}) = j$ for $j=0,1,\ldots,M$.

DEFINITION [Natvig, 1982]. ϕ is <u>N-type2 coherent</u> iff

i. $\Omega_i = \Omega = \{0,1,\ldots,M\}$, $i=1,2,\ldots,n$, and

ii. There exist binary coherent structures ϕ_j, $j=1,\ldots,M$ such that ϕ satisfies

$$\phi(\mathbf{x}) \geq j \Leftrightarrow \phi_j(I_j(\mathbf{x})) = 1$$

for any vector $\mathbf{x}$ and any state $j \geq 1$. The indicator vector is given by

$I_j(\mathbf{x}) = (I_j(x_1),\ldots,I_j(x_n))$, for $j=1,2,\ldots,M$ where

$$I_j(x_j) = \begin{cases} 1 & \text{if } x_i \geq j \\ 0 & \text{if } x_i < j. \end{cases}$$

The N-type2 model and the BW model transform the multistate system into several binary systems. Natvig [1982] showed that N-type2 coherent systems are BW-coherent when the binary structures are the minimal path sets.

Ohi and Nishida [1984] gave definitions for five types of coherence: strongly-coherent, coherent, sub-coherent, pseudo-coherent, and weakly-coherent. As with the definition first given by Hudson [1981], a distinction was made between the state spaces of each component and the system. Let s and t be distinct states in Ω. Let j and k be distinct states in Ω_i.

DEFINITION [Ohi and Nishida, 1984]. ϕ is a multistate monotone system (MMS) iff

i. $\Omega_i = \{0,1,\ldots,M_i\}$, $i=1,2,\ldots,n$,

ii. $\Omega = \{0,1,\ldots,M\}$, and

iii. $\phi(\mathbf{x})$ is increasing.

DEFINITION [Ohi and Nishida, 1984]. ϕ is ON-strongly coherent iff

i. ϕ is a MMS and

ii. For every component i and all system states s and t, there exist vectors $(j_i,\mathbf{x})$ and $(k_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) = s$ and $\phi(k_i,\mathbf{x}) = t$.

DEFINITION [Ohi and Nishida, 1984]. ϕ is ON-coherent iff

i. ϕ is a MMS and

ii. For every component i and all system states s, there exist vectors $(j_i,\mathbf{x})$ and $(k_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) = s-1$ and $\phi(k_i,\mathbf{x}) = s$.

DEFINITION [Ohi and Nishida, 1984]. ϕ is ON-sub-coherent iff

i. ϕ is a MMS and

ii. For every component i and all system states s, there exist vectors $(j_i,\mathbf{x})$ and $(k_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) \neq s$ and $\phi(k_i,\mathbf{x}) = s$.

DEFINITION [Ohi and Nishida, 1984]. ϕ is ON-pseudo-coherent iff

i. ϕ is a MMS and

ii. For every component i and all system states s, there exist vectors $(j_i,\mathbf{x})$ and $(k_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) \leq s-1$ and $\phi(k_i,\mathbf{x}) \geq s$.

DEFINITION [Ohi and Nishida, 1984]. ϕ is <u>ON-weakly-coherent</u> iff

i. ϕ is a MMS and

ii. For every component i, there exist vectors $(j_i,\mathbf{x})$ and $(k_i,\mathbf{x})$ such that $\phi(j_i,\mathbf{x}) \neq \phi(k_i,\mathbf{x})$.

Ohi and Nishida [1984] and Abouammoh and Al-Kadi [1991] have shown the relationships between the various definitions.

There are two aspects of the previous definitions for coherence that are too restrictive. First, the state spaces of the components and system should not be restricted to the same set. For example, most of the previous definitions require that $\Omega = \Omega_i$, i=1,2,...,n. Instead, the model should permit $\Omega \neq \Omega_1 \neq \Omega_2 \neq \ldots \neq \Omega_n$. Second, the system state should not be restricted for specific $\mathbf{x} \in S$. For example, an EPS-coherent system requires that $\phi(\mathbf{J}) = j$. Instead, the model should only require that at least one component state vector belong to the lowest and highest equivalence classes.

The previous definitions for coherence contain two desired aspects. First, the system should not improve with the deterioration of a component and the system should not deteriorate with the improvement of a component. Therefore, $\phi(\mathbf{x})$ must be an increasing function. Second, the system should only contain relevant components. The component relevance condition is the main difference in the previous definitions. An equivalent form of the least restrictive component relevance condition that uses information readily supplied by the customer is developed next.

The least restrictive relevance condition was given in the definition for G-weakly coherent. Griffith [1980] proved that this condition can be replaced with another equivalent condition that is much easier to check: for any component i, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi(0_i, \mathbf{x}) < \phi(M_i, \mathbf{x})$. Extending the condition for a general multistate system: for any component i, there exists a vector $(\cdot_i, \mathbf{x})$ such that $\phi(0_i, \mathbf{x}) < \phi((M_i)_i, \mathbf{x})$.

One serious disadvantage of all the previous relevance conditions is that the structure function must be known to check for component relevance. Most of the time, the customer will not know $\phi(\mathbf{x})$ explicitly. However, the customer should be able to describe the structure with either the lower or upper boundary points to level k. Therefore, it would be best to define component relevance in terms of either the lower or the upper boundary points to level k.

Using Griffith's equivalent condition, suppose that component i is not relevant. Then, for every vector $(\cdot_i, \mathbf{x})$, $\phi(0_i, \mathbf{x}) \geq \phi((M_i)_i, \mathbf{x})$. But since ϕ is increasing, $\phi(0_i, \mathbf{x}) = \phi(1_i, \mathbf{x}) = \ldots = \phi((M_i)_i, \mathbf{x})$. By definition, only $(0_i, \mathbf{x})$ is a potential lower boundary point to level k and only $((M_i)_i, \mathbf{x})$ is a potential upper boundary point to level k. Thus, when component i is irrelevant, all lower boundary points to level k have $x_i = 0$ and all upper boundary points to level k have $x_i = M_i$. From this, a relevant component is defined.

DEFINITION. Component i is <u>relevant</u> if there exists a lower boundary point to level k such that $x_i \neq 0$ for some $k=1,2,\dots,M$ or an upper boundary point to level k such that $x_i \neq M_i$ for some $k=0,1,\dots,M-1$.

For the binary model, this definition says that component i is relevant if $x_i = 1$ in some minimal path vector or $x_i = 0$ in some minimal cut vector.

Considering the undesirable and desirable aspects of the previous definitions and the new definition for component relevance, this dissertation uses the following definition for a general multistate coherent system:

DEFINITION. ϕ is a <u>general multistate coherent system</u> (general MCS) iff

i. $\Omega_i = \{0,1,\dots,M_i\}$, $i=1,2,\dots,n$,

ii. $\Omega = \{0,1,\dots,M\}$,

iii. S_0 and S_M are not empty,

iv. $\phi(\mathbf{x})$ is increasing, and

v. For every component i $(i=1,2,\dots,n)$, there exists a lower boundary point to level k such that $x_i \neq 0$ for some $k=1,2,\dots,M$ or an upper boundary point to level k such that $x_i \neq M_i$ for some $k=0,1,\dots,M-1$.

EXAMPLE 3.2 Suppose that $n=2$, $M_1 = M_2 = M = 2$, and the customer specifies the following lower and upper boundary points:

Level	Lower Boundary Point	Level	Upper Boundary Point
1	(1,0)	0	(0,2)
2	(2,0)	1	(1,2)

Determine if the components are relevant.

Component 2 in not relevant since $x_2 = 0$ for all lower boundary points and $x_2 = 2$ for all upper boundary points. This checks with the actual general MCS of $\phi(\mathbf{x}) = \text{Max}\{x_1, \text{Min}\{x_1, x_2\}\}$ which was used to generate the boundary points.

The next theorem states that the worst $\mathbf{x} \in S$ will always be an member of the lowest equivalence class and the best $\mathbf{x} \in S$ will always be a member of the highest equivalence class.

THEOREM 3.1 If ϕ is a general MCS, then $\mathbf{x}_0 \in S_0$ and $\mathbf{x}_M \in S_M$.

Proof: From condition (iii), S_0 is not empty. Let $\mathbf{y} \in S_0$. Clearly, $\mathbf{x}_0 \leq \mathbf{y}$. From condition (iv), $\phi(\mathbf{x}_0) \leq \phi(\mathbf{y}) = 0$. But $\phi(\mathbf{x}_0) \geq 0$, so $\phi(\mathbf{x}_0) = 0$ or $\mathbf{x}_0 \in S_0$. The same logic can be used to show that $\mathbf{x}_M \in S_M$.

As with binary systems, the elimination of unrealistic structures allows important results to be developed for every general MCS. The next theorem gives bounds on the structure function similar to the binary bounds developed from the best and worst arrangement of the components (Theorem 2.1).

THEOREM 3.2 Suppose that ϕ is a general MCS with s_k lower boundary points to level k, k=1,2,...,M given by $L_k = \{L_{k1}, L_{k2}, \ldots, L_{k,s_k}\}$ and t_k upper boundary points to level k, k=0,1,...,M-1 given by $U_k = \{U_{k1}, U_{k2}, \ldots, U_{k,t_k}\}$. Suppose that ϕ_l is a general MCS with one lower boundary point to level k, k=1,2,...,M given by L_k° and $L_k^\circ \in L_k$. Suppose that ϕ_u is a general MCS with one upper boundary point to level k, k=0,1,...,M-1 given by U_k° and $U_k^\circ \in U_k$. Then $\phi_l(\mathbf{x}) \leq \phi(\mathbf{x}) \leq \phi_u(\mathbf{x})$.

Proof: The proof will be delayed until after the discussion in section 3.1.5.

Barlow and Wu [1978] proved the following results for any increasing multistate coherent system. The theorems are equally valid for the general MCS.

THEOREM 3.3 If ϕ is a general MCS, then

$$\phi(\mathbf{x} \vee \mathbf{y}) \geq \phi(\mathbf{x}) \vee \phi(\mathbf{y})$$

for any component state vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x} \vee \mathbf{y}$ is defined as $(x_1 \vee y_1, x_2 \vee y_2, \ldots, x_n \vee y_n)$ where $x_i \vee y_i = \text{Max}\{x_i, y_i\}$ and $\phi(\mathbf{x}) \vee \phi(\mathbf{y}) = \text{Max}\{\phi(\mathbf{x}), \phi(\mathbf{y})\}$.

Proof: For any two component state vectors, $\mathbf{x} \vee \mathbf{y} \geq \mathbf{x}$ and $\mathbf{x} \vee \mathbf{y} \geq \mathbf{y}$. Since ϕ is increasing, $\phi(\mathbf{x} \vee \mathbf{y}) \geq \phi(\mathbf{x})$ and $\phi(\mathbf{x} \vee \mathbf{y}) \geq \phi(\mathbf{y})$. Thus, $\phi(\mathbf{x} \vee \mathbf{y}) \geq \text{Max}\{\phi(\mathbf{x}), \phi(\mathbf{y})\}$ and the result follows.

THEOREM 3.4 If ϕ is a general MCS, then

$$\phi(\mathbf{x} \wedge \mathbf{y}) \leq \phi(\mathbf{x}) \wedge \phi(\mathbf{y})$$

for any component state vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x} \wedge \mathbf{y}$ is defined as $(x_1 \wedge y_1, x_2 \wedge y_2, \ldots, x_n \wedge y_n)$ where $x_i \wedge y_i = \text{Min}\{x_i, y_i\}$ and $\phi(\mathbf{x}) \wedge \phi(\mathbf{y}) = \text{Min}\{\phi(\mathbf{x}), \phi(\mathbf{y})\}$.

Proof: For any two component state vectors, $\mathbf{x} \wedge \mathbf{y} \leq \mathbf{x}$

and $\mathbf{x} \wedge \mathbf{y} \leq \mathbf{y}$. Since ϕ is increasing, $\phi(\mathbf{x} \wedge \mathbf{y}) \leq \phi(\mathbf{x})$ and $\phi(\mathbf{x} \wedge \mathbf{y}) \leq \phi(\mathbf{y})$. Thus, $\phi(\mathbf{x} \wedge \mathbf{y}) \leq \text{Min}\{\phi(\mathbf{x}),\phi(\mathbf{y})\}$ and the result follows.

These two theorems were enough when a series structure function was defined so that $\phi(\mathbf{x}) = \text{Min}\{x_1,x_2,\ldots,x_n\}$ and a parallel structure so that $\phi(\mathbf{x}) = \text{Max}\{x_1,x_2,\ldots,x_n\}$. The change in the definition of series and parallel structures requires that these two theorems be reevaluated for the general MCS.

Let $\coprod$ be defined as a parallel general MCS and $\prod$ be defined as a series general MCS. The next two examples show that $\phi(\mathbf{x} \coprod \mathbf{y}) \geq \phi(\mathbf{x}) \coprod \phi(\mathbf{y})$ and $\phi(\mathbf{x} \prod \mathbf{y}) \leq \phi(\mathbf{x}) \prod \phi(\mathbf{y})$ are <u>not</u> true for all component state vectors $\mathbf{x}$ and $\mathbf{y}$.

EXAMPLE 3.3 Suppose that $\coprod(\mathbf{x})$ and $\phi(\mathbf{x})$ are enumerated as shown in Table 3.2.

Table 3.2 $\coprod(\mathbf{x})$ and $\phi(\mathbf{x})$ for Example 3.3.

			x_2	
	$\coprod(\mathbf{x})$	0	1	2
	0	0	0	1
x_1	1	0	0	1
	2	1	1	1

			x_2	
	$\phi(\mathbf{x})$	0	1	2
	0	0	0	1
x_1	1	1	1	2
	2	1	2	2

Show that $\coprod(\mathbf{x})$ is a parallel general MCS and $\phi(\mathbf{x} \coprod \mathbf{y}) < \phi(\mathbf{x}) \coprod \phi(\mathbf{y})$ for some $\mathbf{x}$ and $\mathbf{y}$.

$\coprod(\mathbf{x})$ is a parallel general MCS because it has two lower boundary points to level 1 $\{(2,0),(0,2)\}$ and one upper boundary point to level 0 $\{(1,1)\}$. Let

$\mathbf{x} = (1,2)$ and $\mathbf{y} = (1,1)$. Then $\mathbf{x} \amalg \mathbf{y} = (0,1)$ and $\phi(\mathbf{x} \amalg \mathbf{y}) = 0$. $\phi(\mathbf{x}) = 2$, $\phi(\mathbf{y}) = 1$, and $\phi(\mathbf{x}) \amalg \phi(\mathbf{y}) = 1$. Thus, $\phi(\mathbf{x} \amalg \mathbf{y}) < \phi(\mathbf{x}) \amalg \phi(\mathbf{y})$.

EXAMPLE 3.4 Suppose that $\Pi(\mathbf{x})$ and $\phi(\mathbf{x})$ are enumerated as shown in Table 3.3.

Table 3.3 $\Pi(\mathbf{x})$ and $\phi(\mathbf{x})$ for Example 3.4.

		x_2		
	$\Pi(\mathbf{x})$	0	1	2
x_1	0	0	0	0
	1	0	1	1
	2	0	1	1

		x_2		
	$\phi(\mathbf{x})$	0	1	2
x_1	0	0	1	2
	1	0	2	2
	2	1	2	2

Show that $\Pi(\mathbf{x})$ is a series general MCS and $\phi(\mathbf{x} \Pi \mathbf{y}) > \phi(\mathbf{x}) \Pi \phi(\mathbf{y})$ for some $\mathbf{x}$ and $\mathbf{y}$.

$\Pi(\mathbf{x})$ is a series general MCS because it has one lower boundary point to level 1 $\{(1,1)\}$ and two upper boundary points to level 0 $\{(2,0)(0,2)\}$. Let $\mathbf{x} = (1,1)$ and $\mathbf{y} = (1,2)$. Then $\mathbf{x} \Pi \mathbf{y} = (1,1)$ and $\phi(\mathbf{x} \Pi \mathbf{y}) = 2$. $\phi(\mathbf{x}) = 2$, $\phi(\mathbf{y}) = 2$, and $\phi(\mathbf{x}) \Pi \phi(\mathbf{y}) = 1$. Thus, $\phi(\mathbf{x} \Pi \mathbf{y}) > \phi(\mathbf{x}) \Pi \phi(\mathbf{y})$.

3.1.5 Equivalent Coherent Structures

Block and Savits [1982] developed a technique for generating an equivalent structure function by decomposing the multistate system into several binary structures. They based the binary structure functions for level k on either the lower or upper boundary points to level k. Their technique is expanded for the general MCS.

Suppose the customer specifies the lower boundary points to level k, k=1,2,...,M. Define the following two indicator variables:

$$y_{ij} = \begin{cases} 0 & \text{if } x_i < j \\ 1 & \text{if } x_i \geq j \end{cases}$$

for i=1,2,...,n and j=1,2,...,M_i.

$$\phi^k(\mathbf{y}) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) < k \\ 1 & \text{if } \phi(\mathbf{x}) \geq k \end{cases} \qquad (3.1)$$

for k=1,2,...,M. As the multistate structure function is a function of a vector of multistate variables $(x_1, x_2, ..., x_n)$, the binary structure function is a function of a vector of binary variables $(y_{11}, ..., y_{1,M_1}, y_{21}, ..., y_{2,M_2}, ..., y_{n1}, ..., y_{n,M_n})$.

Suppose the s_k lower boundary points to level k are given by $L_{k1}, L_{k2}, ..., L_{k,s_k}$. Let the set of all lower boundary points to level k be designated L_k. If $\mathbf{x} \in L_k$, then let

$$\mathcal{L}_k(\mathbf{x}) = \{(i, x_i) \text{ for all } x_i \neq 0\}.$$

In the binary case, the ordered pairs of $\mathcal{L}_k(\mathbf{x})$ form a minimal path set for each $\mathbf{x} \in L_k$. Block and Savits [1982] wrote the binary structure function for level k as

$$\phi^k(\mathbf{y}) = \operatorname*{Max}_{\mathbf{x} \in L_k} \ \operatorname*{Min}_{(i,j) \in \mathcal{L}_k(\mathbf{x})} y_{ij}.$$

From Equation 3.1, $\phi(\mathbf{x})$ can be found from the sum

$$\phi(\mathbf{x}) = \phi^1(\mathbf{y}) + \phi^2(\mathbf{y}) + \ldots + \phi^M(\mathbf{y}).$$

A similar derivation uses the upper boundary points to level k. Suppose the customer specifies the upper boundary points to level k, k=0,1,...,M-1. Define the following two indicator variables:

$$y_{ij} = \begin{cases} 0 & \text{if } x_i \leq j \\ 1 & \text{if } x_i > j \end{cases}$$

for i=1,2,...,n and j=0,1,...,M_i-1.

$$\phi^k(\mathbf{y}) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) \leq k \\ 1 & \text{if } \phi(\mathbf{x}) > k \end{cases} \qquad (3.2)$$

for k=0,1,...,M-1. As the multistate structure function is a function of a vector of multistate variables $(x_1, x_2, \ldots, x_n)$, the binary structure function is a function of the binary variables $(y_{10}, \ldots, y_{1,M_1-1}, y_{20}, \ldots, y_{2,M_2-1}, \ldots, y_{n0}, \ldots, y_{n,M_n-1})$.

Suppose the t_k upper boundary points to level k are given by $U_{k1}, U_{k2}, \ldots, U_{k,t_k}$. Let the set of all upper boundary points to level k be designated U_k. If $\mathbf{x} \in U_k$, then let

$$\mathfrak{U}_k(\mathbf{x}) = \{(i, x_i) \text{ for all } x_i \neq M_i\}.$$

In the binary case, the ordered pairs of $\mathfrak{U}(\mathbf{x})$ form a minimal cut set for each $\mathbf{x} \in U_k$. Block and Savits [1982] wrote the binary structure function for level k as

$$\phi^k(\mathbf{y}) = \underset{\mathbf{x} \in U_k}{\text{Min}} \; \underset{(i,j) \in \mathfrak{U}(\mathbf{x})}{\text{Max}} \; y_{ij}.$$

From Equation 3.2, $\phi(\mathbf{x})$ can be found from the sum

$$\phi(\mathbf{x}) = \phi^0(\mathbf{y}) + \phi^1(\mathbf{y}) + \ldots + \phi^{M-1}(\mathbf{y}).$$

The procedure is demonstrated in the following example:

EXAMPLE 3.5 Suppose a general MCS of 3 mutually independent components is defined by the customer with the following lower boundary points:

$L_{41}=(310)$ $L_{42}=(211)$

$L_{31}=(310)$ $L_{32}=(120)$ $L_{33}=(111)$

$L_{21}=(300)$ $L_{22}=(201)$ $L_{23}=(110)$ $L_{24}=(020)$ $L_{25}=(011)$

$L_{11}=(200)$ $L_{12}=(101)$ $L_{13}=(010)$

Write an expression for the system structure function.

$\mathfrak{L}_4(L_{41}) = \{(1,3),(2,1)\}$ $\mathfrak{L}_4(L_{42}) = \{(1,2),(2,1),(3,1)\}$

$\mathfrak{L}_3(L_{31}) = \{(1,3),(2,1)\}$ $\mathfrak{L}_3(L_{32}) = \{(1,1),(2,2)\}$

$\mathfrak{L}_3(L_{33}) = \{(1,1),(2,1),(3,1)\}$ $\mathfrak{L}_2(L_{21}) = \{(1,3)\}$

$\mathfrak{L}_2(L_{22}) = \{(1,2),(3,1)\}$ $\mathfrak{L}_2(L_{23}) = \{(1,1),(2,1)\}$

$\mathfrak{L}_2(L_{24}) = \{(2,2)\}$ $\mathfrak{L}_2(L_{25}) = \{(2,1),(3,1)\}$ $\mathfrak{L}_1(L_{11}) = \{(1,2)\}$

$\mathfrak{L}_1(L_{12}) = \{(1,1),(3,1)\}$ $\mathfrak{L}_1(L_{13}) = \{(2,1)\}$

$\phi^4(\mathbf{y}) = \text{Max}\{y_{13}y_{21},\ y_{12}y_{21}y_{31}\}$

$\phi^3(\mathbf{y}) = \text{Max}\{y_{13}y_{21},\ y_{11}y_{22},\ y_{11}y_{21}y_{31}\}$

$\phi^2(\mathbf{y}) = \text{Max}\{y_{13},\ y_{12}y_{31},\ y_{11}y_{21},\ y_{22},\ y_{21}y_{31}\}$

$\phi^1(\mathbf{y}) = \text{Max}\{y_{12},\ y_{11}y_{31},\ y_{21}\}$

The system structure function can be calculated from

$\phi(\mathbf{x}) = \phi^1(\mathbf{y}) + \phi^2(\mathbf{y}) + \phi^3(\mathbf{y}) + \phi^4(\mathbf{y}).$

The transformation of Block and Savits [1982] and Wood [1985] allows the proof of Theorem 3.2.

Proof of Theorem 3.2: Using the lower boundary points to level k, the structure function for level k is

$$\phi^k(\mathbf{y}) = \operatorname*{Max}_{\mathbf{x}\in L_k} \operatorname*{Min}_{(i,j)\in \mathfrak{L}_k(\mathbf{x})} y_{ij}.$$

Then it is clear that $\phi^k(\mathbf{y}) \geq \phi^k{}_l(\mathbf{y})$ for k=1,2,...,M since ϕ^k maximizes over all lower boundary points and $L_k^o \in L_k$. The system structure function comes from

$$\phi(\mathbf{x}) = \sum_{k=1}^{M} \phi^k(\mathbf{y})$$

so it must be true that $\phi(\mathbf{x}) \geq \phi_l(\mathbf{x})$. Using the upper boundary points to level k, the structure function is

$$\phi^k(\mathbf{y}) = \operatorname*{Min}_{\mathbf{x}\in U_k} \operatorname*{Max}_{(i,j)\in \mathfrak{U}_k(\mathbf{x})} y_{ij}.$$

Then it is clear that $\phi^k(\mathbf{y}) \leq \phi^k{}_u(\mathbf{y})$ for k=0,1,,...,M-1 since ϕ^k minimizes over all upper boundary points and $U_k^o \in U_k$. The system structure function comes from

$$\phi(\mathbf{x}) = \sum_{k=0}^{M-1} \phi^k(\mathbf{y})$$

so it must be true that $\phi(\mathbf{x}) \leq \phi_u(\mathbf{x})$.

The transformation also provides some insight into the new definitions for parallel and series structure functions. In the multistate case, it is no longer meaningful to represent a series or a parallel structure in a functional block diagram. The new definitions isolate the cases when the alternate representation given by Block and Savits [1982] can be simplified. A series structure has one lower boundary point to level k, L_{k1}, k=1,2,...,M. Therefore, the alternate representation for a series structure is given by

$$\phi(\mathbf{x}) = \sum_{k=1}^{M} \phi^{k}(\mathbf{y}) = \sum_{k=1}^{M} \operatorname*{Min}_{(i,j)\in\mathcal{L}_k(L_{k1})} y_{ij} \,.$$

Likewise, since a parallel structure has only one upper boundary point to level k, U_{k1}, k=0,1,...,M-1, an alternate representation for parallel structure is

$$\phi(\mathbf{x}) = \sum_{k=0}^{M-1} \phi^{k}(\mathbf{y}) = \sum_{k=0}^{M-1} \operatorname*{Max}_{(i,j)\in\mathcal{U}_k(U_{k1})} y_{ij} \,.$$

El-Neweihi et al. [1978] developed an expansion to reduce the order of a multistate structure function by one. The following expansion, given by Hudson and Kapur [1983], extends the result to the general MCS:

$$\phi(\mathbf{x}) = \sum_{j=0}^{M_i} \phi(j_i,\mathbf{x})\, I_{ij} \quad \text{for } i=1,2,\ldots,n \tag{3.3}$$

where

$$I_{ij} = \begin{cases} 1 & \text{if } x_i=j \\ 0 & \text{if } x_i\neq j. \end{cases}$$

The expansion can be performed about any component. It is a generalization of Equation 2.1 given for the binary model in section 2.1.5.

3.1.6 Dual Structure Function

The definition of the dual structure function was first extended to multistate systems by El-Neweihi et al. [1978] and to a general MCS by Hudson [1981].

DEFINITION. Let ϕ be a structure function of a general MCS. The dual structure function ϕ^D is defined by

$$\phi^D(\mathbf{x}) = M - \phi(M_1-x_1, M_2-x_2, \ldots, M_n-x_n)$$
$$= M - \phi(\mathbf{x}_M - \mathbf{x}).$$

For a general MCS (C,ϕ), the dual is (C^D,ϕ^D). Note that the sets of components (C and C^D) are the same for both systems. However, the notation C^D is used to clarify that when the primal component C_i is in state x_i, the corresponding dual component C_i^D is in state $x_i^D = M_i - x_i$.

Intuitively, the dual has the following interpretation. For the binary case, the dual system functions iff the primal system fails. For the general MCS, the primal system is in state k iff the dual system is in state M - k. The boundary points of the primal and dual have a special relationship.

THEOREM 3.5 $\mathbf{x}$ is a lower boundary point to level k for the general MCS ϕ iff $(\mathbf{x}_M - \mathbf{x})$ is an upper boundary point to level $(M - k)$ for the dual general MCS ϕ^D.

Proof: Suppose $\mathbf{x}$ is a lower boundary point to level k for ϕ. Then $\phi(\mathbf{x}) \geq k$ and if $\mathbf{y} < \mathbf{x}$, then $\phi(\mathbf{y}) < k$. From the definition of the dual

$\phi^D(\mathbf{x}_M - \mathbf{x}) = M - \phi(\mathbf{x}_M - (\mathbf{x}_M - \mathbf{x})) = M - \phi(\mathbf{x})$. Therefore, $\phi(\mathbf{x}) = M - \phi^D(\mathbf{x}_M - \mathbf{x}) \geq k$ and $\phi^D(\mathbf{x}_M - \mathbf{x}) \leq M - k$. Now suppose that $\mathbf{y} > \mathbf{x}_M - \mathbf{x}$. Rearranging terms, $\mathbf{x}_M - \mathbf{y} < \mathbf{x}$ and $\phi(\mathbf{x}_M - \mathbf{y}) < k$. So $\phi^D(\mathbf{y}) = M - \phi(\mathbf{x}_M - \mathbf{y}) > M - k$. In summary, $\phi^D(\mathbf{x}_M - \mathbf{x}) \le M - k$ and $\mathbf{y} > \mathbf{x}_M - \mathbf{x}$ implies that $\phi^D(\mathbf{y}) > M - k$. By definition, $\mathbf{x}_M - \mathbf{x}$ is an upper boundary point to level $M - k$ for the dual structure ϕ^D. Suppose $(\mathbf{x}_M - \mathbf{x})$ is an upper boundary point to level $(M-k)$ for ϕ^D. Then $\phi^D(\mathbf{x}_M - \mathbf{x}) \le M - k$ and if $\mathbf{y} > \mathbf{x}_M - \mathbf{x}$, then $\phi^D(\mathbf{y}) > M - k$. From the definition of the dual, $\phi^D(\mathbf{x}_M - \mathbf{x}) = M - \phi(\mathbf{x}) \le M - k$ and $\phi(\mathbf{x}) \ge k$. Now suppose that $\mathbf{y} < \mathbf{x}$. Thus, $\mathbf{x}_M - \mathbf{y} > \mathbf{x}_M - \mathbf{x}$ and $\phi^D(\mathbf{x}_M - \mathbf{y}) > M-k$. So $\phi(\mathbf{y}) = M - \phi^D(\mathbf{x}_M - \mathbf{y}) < k$. In summary, $\phi(\mathbf{x}) \ge k$ and $\mathbf{y} < \mathbf{x}$ implies that $\phi(\mathbf{y}) < k$. Therefore, $\mathbf{x}$ is a lower boundary point to level k for the structure ϕ.

EXAMPLE 3.6 Suppose that $n=2$, $M_1=3$, $M_2=2$, $M=4$, and $\phi(\mathbf{x})$ is enumerated as in Table 3.4. Enumerate the dual structure, $\phi^D(\mathbf{x})$, and demonstrate Theorem 3.5.

Table 3.4 $\phi(\mathbf{x})$ for Example 3.6.

		x_2		
	$\phi(\mathbf{x})$	0	1	2
x_1	0	0	0	0
	1	1	2	2
	2	1	3	3
	3	1	3	4

$$\phi^D(\mathbf{x}) = M - \phi(M_1 - x_1, M_2 - x_2)$$
$$= 4 - \phi(3 - x_1, 2 - x_2).$$

$\phi^D(\mathbf{x})$ is enumerated in Table 3.5.

Table 3.5 $\phi^D(\mathbf{x})$ for Example 3.6.

		x_2		
	$\phi^D(\mathbf{x})$	0	1	2
x_1	0	0	1	3
	1	1	1	3
	2	2	2	3
	3	4	4	4

The lower boundary point to level 1 for ϕ is (1,0). By Theorem 3.5, the upper boundary point to level 3 for ϕ^D is (2,2) which can be seen from Table 3.5. The upper boundary points to level 3 for ϕ are (3,1) and (2,2). By Theorem 3.5, the lower boundary points to level 1 for ϕ^D are (0,1) and (1,0) which checks with Table 3.5.

The remainder of this section is dedicated to stating and proving some of the more common theorems that relate to the dual structure function. The proofs are necessary because different definitions have been used for a general MCS and for k-out-of-n structures.

THEOREM 3.6 If the primal is a general MCS, then the dual is also a general MCS.

Proof:

i. Suppose that $\Omega_i = \{0,1,\ldots,M_i\}$, $i=1,2,\ldots,n$ for ϕ. When the state of component i in ϕ is x_i, then the state of component i in ϕ^D is $x_i^D = M_i - x_i$. Thus, $\Omega_i^D = \{0,1,\ldots,M_i\}$, $i=1,2,\ldots,n$ for ϕ^D.

ii. Suppose that $\Omega = \{0,1,\ldots,M\}$ for ϕ. When the system state of ϕ is Ω^p, then the system state of ϕ^D is $M - \Omega^p$. Thus, $\Omega^p = \{0,1,\ldots,M\}$ for ϕ^D.

iii. Suppose S_0 and S_M are not empty. Then the vectors $\mathbf{y}$ and $\mathbf{z}$ exist such that $\phi(\mathbf{y}) = 0$ and $\phi(\mathbf{z}) = M$. From the definition of the dual,

$$\phi^D(\mathbf{x}_M - \mathbf{x}) = M - \phi(\mathbf{x}_M - (\mathbf{x}_M - \mathbf{x})) = M - \phi(\mathbf{x}).$$

Substituting, $\phi^D(\mathbf{x}_M - \mathbf{y}) = M$ and $\phi^D(\mathbf{x}_M - \mathbf{z}) = 0$. Thus, $\mathbf{x}_M - \mathbf{y} \in S^D_M$ and $\mathbf{x}_M - \mathbf{z} \in S^D_0$.

iv. Let $\mathbf{x}, \mathbf{y} \in S$ such that $\mathbf{x} \leq \mathbf{y}$. Then $\mathbf{x}_M - \mathbf{x} \geq \mathbf{x}_M - \mathbf{y}$. Since ϕ satisfies (vi), $\phi(\mathbf{x}_M - \mathbf{x}) \geq \phi(\mathbf{x}_M - \mathbf{y})$. Thus, $M - \phi(\mathbf{x}_M - \mathbf{x}) \leq M - \phi(\mathbf{x}_M - \mathbf{y})$. From the definition of the dual, $\phi^D(\mathbf{x}) \leq \phi^D(\mathbf{y})$. Thus ϕ^D is increasing.

v. Suppose ϕ^D does not satisfy (v). Then there exists an i such that $x_i^D = 0$ for all lower boundary points of ϕ^D and $x_i^D = M$ for all upper boundary points of ϕ^D. By the previous theorem, $x_i = M - 0$ for all upper boundary points of ϕ and $x_i = M - M$ for all lower boundary points of ϕ. But this contradicts the fact that ϕ satisfies (v). Thus ϕ^D satisfies (v).

THEOREM 3.7 The dual of a k-out-of-n general MCS is an (n-k+1)-out-of-n general MCS.

Proof: Suppose that ϕ is a k-out-of-n general MCS. By definition, ϕ has $\binom{n}{k}$ lower boundary points to level j, j=1,2,...,M and $\binom{n}{n-k+1}$ upper boundary points to level j,

$j=0,1,\ldots,M-1$. By theorem 3.7, ϕ^D has $\binom{n}{k}$ upper boundary points to level $M-j$, $j=1,2,\ldots,M$ and $\binom{n}{n-k+1}$ lower boundary points to level $M-j$, $j=0,\ldots,M-1$. By letting $j' = M-j$, ϕ^D has $\binom{n}{k}$ upper boundary points to level j', $j'=0,1,\ldots,M-1$ and $\binom{n}{n-k+1}$ lower boundary points to level j', $j'=1,2,\ldots,M$. Therefore, ϕ^D is an $(n-k+1)$-out-of-n general MCS.

Since a series system is an n-out-of-n structure and a parallel system is a 1-out-of-n structure, the following two corollaries are immediately apparent.

COROLLARY 3.1 The dual of a series general MCS of n components is a parallel general MCS of n components.

COROLLARY 3.2 The dual of a parallel general MCS of n components is a series general MCS of n components.

Finally, Janan [1985] showed that the dual is idempotent.

THEOREM 3.8 The dual of the dual is the primal.

Proof:
$$\begin{aligned}[\phi^D]^D &= M - \phi^D(\mathbf{x}_M - \mathbf{x}) \\ &= M - [M - \phi(\mathbf{x}_M - (\mathbf{x}_M - \mathbf{x}))] \\ &= M - [M - \phi(\mathbf{x})] = \phi(\mathbf{x}).\end{aligned}$$

3.1.7 Structural Importance

Block and Savits [1982] discussed a connection between the concepts of component relevance, system coherence, and structural importance. In general, component i is relevant if there exists a component state vector that satisfies the model's relevancy condition. For the binary model, component i is relevant if there is an $\mathbf{x}$ such that $\phi(1_i,\mathbf{x}) \neq \phi(0_i,\mathbf{x})$. For the G-weakly-coherent multistate model, component i is relevant if there is an $\mathbf{x}$ such that $\phi(0_i,\mathbf{x}) < \phi(M_i,\mathbf{x})$. For the general MCS, component i is relevant if there is an $\mathbf{x}$ such that $\phi(0_i,\mathbf{x}) < \phi((M_i)_i,\mathbf{x})$. The system is coherent if, among other conditions, all components are relevant.

Finally, the structural importance of component i can be defined as the proportion of the component state vectors where the relevance condition holds. For the binary model, the structural importance for component i was given in section 2.1.7. For the G-weakly-coherent system, the structural importance for component i can be generated by

$$I_\phi(i) = \frac{1}{(M+1)^{n-1}} \sum_{\{\mathbf{x} | x_i = M\}} N(\mathbf{x}) \quad \text{where}$$

$$N(\mathbf{x}) = \begin{cases} 1 & \text{if } \phi(0_i,\mathbf{x}) < \phi(M_i,\mathbf{x}) \\ 0 & \text{if } \phi(0_i,\mathbf{x}) = \phi(M_i,\mathbf{x}). \end{cases}$$

Extending to the general MCS, the structural importance for component i can be calculated from

$$I_\phi(i) = \frac{1}{\prod_{j \neq i} (M_j+1)} \sum_{\{\mathbf{x} | x_i = M_i\}} N(\mathbf{x}) \quad \text{where}$$

$$N(\mathbf{x}) = \begin{cases} 1 & \text{if } \phi(0_i,\mathbf{x}) < \phi((M_i)_i,\mathbf{x}) \\ 0 & \text{if } \phi(0_i,\mathbf{x}) = \phi((M_i)_i,\mathbf{x}). \end{cases}$$

3.1.8 Modules and Modular Decomposition

Fardis and Cornell [1981] used modular decomposition to simplify the calculation of reliability for multistate systems composed of duplicate components. Butler [1982] developed bounds on system reliability with modular decomposition for the multistate case. Hudson and Kapur [1983a] extended the definitions for modules and modular decomposition to the general MCS.

DEFINITION - Suppose (C,ϕ) is a general MCS where C is the set of components. Suppose that $A \subset C$. Let A' denote the subset of C complementary to A. The general MCS (A,χ) is a <u>module</u> of (C,ϕ) if

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^A, \mathbf{x}^{A'}) = \psi[\chi(\mathbf{x}^A), \mathbf{x}^{A'}]$$

where ψ is a MCS called the organizing structure.

DEFINITION - A <u>modular decomposition</u> of a general MCS (C,ϕ) is a set $\{(A_1,\chi_1),(A_2,\chi_2),\ldots,(A_k,\chi_k)\}$ of general MCSs along with the organizing structure ψ such that

i) $\{A_1,A_2,\ldots,A_k\}$ partition C into disjoint subsets and

ii) $\phi(\mathbf{x}) = \psi[\chi_1(\mathbf{x}^{A_1}),\chi_2(\mathbf{x}^{A_2}),\ldots,\chi_k(\mathbf{x}^{A_k})]$.

Using modules for a multistate system is especially valuable when modeling a physical system that can be divided into distinct subsystems. In essence, several smaller multistate models are generated and solved separately. The results of each of the smaller problems are combined with the organizing structure to analyze the entire system.

3.2 Stochastic Properties

So far, only the deterministic properties of the general multistate model have been discussed. Stochastic properties characterize the probabilistic relationship between the state of the system and the states of the components at a fixed moment in time.

3.2.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components.
X_i	random variable for the state of component i.
x_i	fixed state of component i; $x_i \in \Omega_i$.
$\mathbf{X}$	random component state vector; $\mathbf{X} = (X_1, X_2, \ldots, X_n)$.
$\mathbf{x}$	fixed component state vector; $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.
$\phi(\mathbf{X})$	random variable for the state of the system.
ϕ	fixed state of the system; $\phi = \phi(\mathbf{x})$.
P_k	$\Pr[\phi(\mathbf{X}) = k]$, $k=0,1,\ldots,M$.
P_{ij}	$\Pr[X_i = j]$, $i=1,2,\ldots,n$ and $j=0,1,\ldots,M_i$.
Q_k	$\Pr[\phi(\mathbf{X}) \geq k]$, $k=1,2,\ldots,M$.
Q_{ij}	$\Pr[X_i \geq j]$, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$.
$\mathbf{q}_i$	$(Q_{i1}, Q_{i2}, \ldots, Q_{i,M_i})$.
$\mathbf{q}$	$(Q_1, Q_2, \ldots, Q_M)$.
$r(\mathbf{q})$	performance function.
L_{kj}	jth lower boundary point to level k, $k=1,2,\ldots,M$ and $j=1,2,\ldots,s_k$.
E_{kj}	event that $\mathbf{X} \geq L_{kj}$, $k=1,2,\ldots,M$ and $j=1,2,\ldots,s_k$.
M_{ij}	best state of the ith component in module j.
n_j	number of components in module j.
I_ψ	index of efficiency of modular decomposition.

$\mathbf{x} << \mathbf{y}$ $x_i < y_i$ $\forall i$.

3.2.2 The Performance Function

For binary systems, the main problem in reliability theory is to determine the system reliability from the reliability of the components. Knowing the system reliability allows us to find the system unreliability since only two system states are possible.

El-Neweihi et al. [1978] explored the same problem for a multistate system and Hudson [1981] extended the problem to the general MCS. For a system of n components, let X_i denote the random state of component i and x_i denote a specific state of component i. The random and specific states for all components are summarized in the random component state vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and the fixed component state vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Let $\phi(\mathbf{X})$ be the random system state and $\phi(\mathbf{x})$ or ϕ be a fixed system state.

For the general MCS, the problem changes to finding the system's probability distribution, P_k, where

$$P_k = \Pr[\phi(\mathbf{X}) = k], \; k=0,1,\ldots,M$$

from every component's probability distribution, P_{ij}, where

$$P_{ij} = \Pr[X_i = j], \; i=1,2\ldots,n \text{ and } j=0,1,\ldots,M_i.$$

An equivalent statement of the problem is finding the performance distribution of the system, Q_k, where

$$Q_k = \Pr[\phi(\mathbf{X}) \geq k], \; k=1,2,\ldots,M$$

from every component's performance distribution, Q_{ij}, where

$$Q_{ij} = \Pr[X_i \geq j], \; i=1,2,\ldots,n \text{ and } j=1,2,\ldots,M_i.$$

The second problem formulation works better for the general MCS for several reasons. First, less calculation is required since $Q_0 = 1$. Second, the most efficient techniques available make use of boundary points and naturally result in a performance distribution. Finally, Griffith [1980] stated that the second formulation sometimes allows for direct comparison of the system performance of two systems. Suppose that $\mathbf{q}_i = (Q_{i1}, Q_{i2}, \ldots, Q_{i,M_i})$ is the performance vector for component i and $\mathbf{q} = (Q_1, Q_2, \ldots, Q_M)$ is the performance vector for the system. Let $\mathbf{q}^1$ and $\mathbf{q}^2$ be the performance vectors for two different systems. System one is superior to system two if $\mathbf{q}^1 \geq \mathbf{q}^2$.

For binary models, reliability was defined as the probability that the system functions. For multistate models, there are different degrees of functioning so a new measure of system performance is required. El-Neweihi et al. [1978] suggested $E[\phi(\mathbf{X})]$ or the expected system state. Butler [1979] promoted $Pr[\phi(\mathbf{X}) \geq k]$, especially when the customer was willing to divide system states into two categories ($\geq$ k or < k). $E[\phi(\mathbf{X})]$ and $Pr[\phi(\mathbf{X}) \geq k]$ are equivalent measures for the binary model. Griffith [1980] used $E[u(\phi(\mathbf{X}))]$ or the expected utility of the system state.

Each of these definitions provides a measure of the performance for multistate systems. However, it is the customer that evaluates the system performance, so it must be the customer that indicates the most appropriate definition. If the customer wants to measure the center and spread of the distribution, then $E[\phi(\mathbf{X})]$ and $Var[\phi(\mathbf{X})]$ seem appropriate. If the customer can separate the system's probability distribution into "good" and "bad" states, then $Pr[\phi(\mathbf{X}) \geq k]$ works well. If the customer wants to evaluate efficient performance distributions, then $E[u(\phi(\mathbf{X}))]$ allows the customer to weigh the different possibilities.

The second objective of this research is to develop a new substitute characteristic for multistate reliability based on the expected loss to the customer. The new measure will be sensitive to the pattern of degradation about a desired system lifetime.

If the random variables X_i, $i=1,\ldots,n$ are mutually independent, then $\mathbf{q}$ may be expressed as a function of $\mathbf{q}_i$, $i=1,2,\ldots,n$. Each of the given measures of performance are defined as a function of $\mathbf{q}$ and therefore, they are also a function of $\mathbf{q}_i$, $i=1,2,\ldots,n$. The relationship between the system's measure of performance and the component performance vectors is given by the performance function:

$$r = r(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n).$$

3.2.3 Performance Importance

The definition for performance importance depends on the definition chosen for the performance function. Suppose that $E[\phi(\mathbf{X})]$ is used. The following expansion of the performance function can be derived using the assumption of mutual independence and taking the expected value of Equation 3.3:

$$r(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = \sum_{j=0}^{M_i} P_{ij}\, E[\phi(j_i, \mathbf{X})].$$

Proceeding in a manner similar to El-Neweihi et al. [1978],

$$r(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = \sum_{j=1}^{M_i} P_{ij}\, E[\phi(j_i, \mathbf{X})] + P_{i0}\, E[\phi(0_i, \mathbf{X})].$$

Since $P_{i0} = 1 - \sum_{j=1}^{M_i} P_{ij}$,

$$r(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = \sum_{j=1}^{M_i} P_{ij}\, \{E[\phi(j_i, \mathbf{X})] - E[\phi(0_i, \mathbf{X})]\} + E[\phi(0_i, \mathbf{X})].$$

The performance importance of component i at state j is

$$\frac{\partial r}{\partial P_{ij}} = E[\phi(j_i, \mathbf{X}) - \phi(0_i, \mathbf{X})]$$

for $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$.

3.2.4 Exact System Performance

Enumeration, inclusion-exclusion, pivoting, and modular decomposition are four techniques used to determine the exact probability distribution from the probability distributions of the components. Each technique assumes the components are mutually independent. The computer program given in Appendix A implements the first three techniques directly and the fourth technique indirectly. It works well for moderately large problems of about 10 components, each with 4 states. Of course, the program will also work for binary systems since they are a special case of the general MCS.

3.2.4.1 Enumeration

The enumeration technique determines the system state and probability for every possible component state vector. It tallies the probabilities for all component state vectors in the same equivalence class.

EXAMPLE 3.7 Suppose that $n=2$, $M_1=3$, $M_2=2$, $M=4$, and $\phi(\mathbf{x})$ is enumerated by the customer as shown in Table 3.6.

Table 3.6 $\phi(\mathbf{x})$ for Example 3.7.

		x_2		
		0	1	2
x_1	0	0	1	2
	1	0	1	3
	2	1	2	3
	3	2	4	4

Suppose that the probability distributions for the components are determined to be:

$P_{10} = .05$ $P_{11} = .1$ $P_{12} = .15$ $P_{13} = .7$

$P_{20} = .1$ $P_{21} = .3$ $P_{22} = .6$

Assuming components are mutually independent, find P_k, $k=0,1,2,3,4$. The probability of each component state vector is found by enumeration:

$\mathbf{x}$	$\phi(\mathbf{x})$	$\Pr[\mathbf{X}=\mathbf{x}]$	$\mathbf{x}$	$\phi(\mathbf{x})$	$\Pr[\mathbf{X}=\mathbf{x}]$
00	0	.005	20	1	.015
01	1	.015	21	2	.045
02	2	.030	22	3	.090
10	0	.010	30	2	.070
11	1	.030	31	4	.210
12	3	.060	32	4	.420

The system probability distribution, P_k, is found by summing the probability of all component states where $\phi(\mathbf{x}) = k$, $k=0,1,2,3,4$. Therefore, $P_0 = .015$, $P_1 = .06$, $P_2 = .145$, $P_3 = .15$, and $P_4 = .63$.

Note that the number of component state vectors is

$$\prod_{i=1}^{n} (M_i + 1)$$

and the number of calculations needed to enumerate all component state vectors becomes unmanageable for large systems. It seems more reasonable to calculate probability distribution from a subset of $\mathbf{x} \in S$.

3.2.4.2 Inclusion-Exclusion

Lower boundary points to level k can be used to calculate P_k, $k=0,1,\ldots,M$ given P_{ij}, $i=1,2,\ldots,n$ and $j=0,1,\ldots,M_i$ [Natvig, 1982]. The method is based on the following theorem proven by Borges and Rodrigues [1983]:

THEOREM 3.9 $\phi(\mathbf{x}) \geq k$ if and only if $\mathbf{x} \geq \mathbf{y}$ for some $\mathbf{y} \in L_k$.

Proof: Suppose $\mathbf{x} \geq \mathbf{y}$ for some $\mathbf{y} \in L_k$. Since $\mathbf{y} \in L_k$, $\phi(\mathbf{y}) \geq k$. Since ϕ is increasing, $\phi(\mathbf{x}) \geq \phi(\mathbf{y}) \geq k$. For necessity, suppose that $\phi(\mathbf{x}) \geq k$. Consider the procedure of decreasing the values of the elements of $\mathbf{x}$ while keeping $\phi(\mathbf{x}) \geq k$. Eventually, it will no longer be possible to continue without decreasing the system state. Let this vector be denoted by $\mathbf{y}$. At this point in the procedure, $\mathbf{y} \in L_k$ by definition. Since the procedure only decreased the elements of $\mathbf{x}$, then $\mathbf{x} \geq \mathbf{y}$ for some $\mathbf{y} \in L_k$.

Suppose ϕ is a multistate coherent system with s_k lower boundary points to level k, designated $L_{k1}, L_{k2}, \ldots, L_{k,s_k}$. Let

E_{kj} be the event that $\mathbf{x} \geq L_{kj}$ for $j=1,\ldots,s_k$. Then

$$\Pr[\phi(X) \geq k] = \begin{cases} \Pr\left[\bigcup_{j=1}^{s_k} E_{kj}\right] & k=1,2,\ldots,M \\ 1 & k=0. \end{cases}$$

The union of events is evaluated with Theorem 2.8. The system probability distribution can be found from

$$P_k = \Pr[\phi(\mathbf{X}) \geq k] - \Pr[\phi(\mathbf{X}) \geq k+1] = Q_k - Q_{k+1},\ k=0,1,\ldots,M$$

and the fact that $Q_{M+1} = 0$. The next example demonstrates the technique using lower boundary points.

EXAMPLE 3.8 Determine the lower boundary points for the MCS given in example 3.7 and calculate the exact system probability distribution.

$\Pr[\phi(\mathbf{X}) \geq 0] = 1.0$

The lower boundary points to level 1 are (0,1) and (2,0). Therefore,

$$\Pr[\phi(\mathbf{X}) \geq 1] = \Pr[X_1 \geq 0,\ X_2 \geq 1] + \Pr[X_1 \geq 2,\ X_2 \geq 0] - \Pr[X_1 \geq 2,\ X_2 \geq 1]$$

$$= (1)(.9) + (.85)(1) - (.85)(.9) = .985$$

The lower boundary points to level 2 are (0,2), (2,1), and (3,0). Therefore,

$$\Pr[\phi(\mathbf{X}) \geq 2] = \Pr[X_1 \geq 0,\ X_2 \geq 2] + \Pr[X_1 \geq 2,\ X_2 \geq 1] + \Pr[X_1 \geq 3,\ X_2 \geq 0] - \Pr[X_1 \geq 2,\ X_2 \geq 2] - \Pr[X_1 \geq 3,\ X_2 \geq 2] - \Pr[X_1 \geq 3,\ X_2 \geq 1] + \Pr[X_1 \geq 3,\ X_2 \geq 2]$$

$$= (1)(.6) + (.85)(.9) + (.7)(1) - (.85)(.6) - (.7)(.6) - (.7)(.9) + (.7)(.6) = .925$$

The lower boundary points to level 3 are (1,2) and (3,1). Therefore,

$Pr[\phi(\mathbf{X}) \geq 3] = Pr[X_1 \geq 1,\ X_2 \geq 2] + Pr[X_1 \geq 3,\ X_2 \geq 1]$

$- Pr[X_1 \geq 3,\ X_2 \geq 2]$

$= (.95)(.6) + (.7)(.9) - (.7)(.6) = .78$

The lower boundary point to level 4 is (3,1). Thus,

$Pr[\phi(\mathbf{X}) \geq 4] = Pr[X_1 \geq 3,\ X_2 \geq 1] = (.7)(.9) = .63$

Therefore, $P_4 = Pr[\phi(\mathbf{X}) = 4] = .63 - 0.0 = .63$,

$P_3 = Pr[\phi(\mathbf{X}) = 3] = .78 - .63 = .15$,

$P_2 = Pr[\phi(\mathbf{X}) = 2] = .925 - .78 = .145$,

$P_1 = Pr[\phi(\mathbf{X}) = 1] = .985 - .925 = .06$, and

$P_0 = Pr[\phi(\mathbf{X}) = 0] = 1.0 - .985 = .015$.

The results match the solution found in Example 3.7. Note that a similar technique can be developed using the upper boundary points to level k, k=0,1,...,M-1.

3.2.4.3 Pivotal Decomposition

Iyer [1989] has shown that pivotal decomposition can be used to calculate exact system reliability for the multistate case. In contrast to the binary case, it is not likely that the customer can completely specify the structure function for multistate systems. However, the customer should be capable of *specifying the boundary points* to each level of the system. Once this is done, the state of one of the components is fixed and reliability calculations are made from the boundary points with reduced dimension.

Suppose the probability distribution, $P_{ij} = Pr[X_i = j]$

for $j=0,1,\ldots,M_i$, is known for each component in the system. Also suppose that the customer can determine the s_k lower boundary points to each level k for $k=1,\ldots,M$ or the t_k upper boundary points to each level k for $k=0,\ldots,M-1$. With this information, pivotal decomposition can be used to determine $\Pr[\phi(\mathbf{X}) = k]$ for $k=0,1,\ldots M$.

The following algorithm was developed to implement the pivotal decomposition strategy to determine $\Pr[\phi(\mathbf{X}) \geq k]$ from the lower boundary points (LBPs) to level k:

<u>Pivotal Decomposition Algorithm</u>

1. Determine the probability distribution for each component in the system.
2. Let the customer specify the LBPs to level k.
3. Choose a component to pivot on. Iyer [1989] has discussed several pivot selection rules.
4. Fix the component state at the lowest level of the chosen component obtained by any LBP. List all LBPs with the chosen component at this level. When only one LBP remains, the branching is fathomed (go to step 8). Otherwise, go to step 5.
5. Fix the component state at the next highest level. If the highest component state is exceeded, go to 7. Otherwise, list all LBPs with the chosen component at this higher level. In addition, create temporary LBPs by listing all LBPs from the previous level with the fixed component state changed to the next highest

level. These temporary LBPs include the probabilities of vectors summarized by the LBPs with the component state at lower levels.

6. Remove any of the temporary LBPs that are greater than the original group of LBPs. This prevents any probabilities from being counted twice. When only one LBP remains, the branching is fathomed (go to step 8). Otherwise, go to step 5.

7. Determine if all branches have been fathomed. If so, STOP. The tallied probability is $\Pr[\phi(\mathbf{X}) \geq k]$. If not, continue with step 3 for all unfathomed branches.

8. Calculate the probability of the fathomed LBP using $\Pr[X_i = j]$ for fixed components and $\Pr[X_i \geq j]$ for components not fixed. Tally the probability of all fathomed LBPs. Return to step 5.

The following example is given to demonstrate the pivotal decomposition technique. Figure 3.1 shows the solution pictorially. Removed temporary LBPs are marked with an asterisk (*). Components fixed at a given state are underlined. The branches are labeled to correspond with the discussion of the solution.

EXAMPLE 3.9 Suppose a system of 3 mutually independent components is defined such that

$P_{10} = .1$ $P_{11} = .2$ $P_{12} = .3$ $P_{13} = .4$

$P_{20} = .1$ $P_{21} = .3$ $P_{22} = .6$

$P_{30} = .2$ $P_{31} = .8$

Let the s_k lower boundary points to level k be designated $L_{k1}, L_{k2}, \ldots, L_{k,s_k}$. Suppose the customer is able to specify the following lower boundary points:

L_{41}=(310) L_{42}=(211)

L_{31}=(310) L_{32}=(120) L_{33}=(111)

L_{21}=(300) L_{22}=(201) L_{23}=(110) L_{24}=(020) L_{25}=(011)

L_{11}=(200) L_{12}=(101) L_{13}=(010)

Determine the $Pr[\phi(\mathbf{X}) \geq 2]$ from the LBPs to level 2.

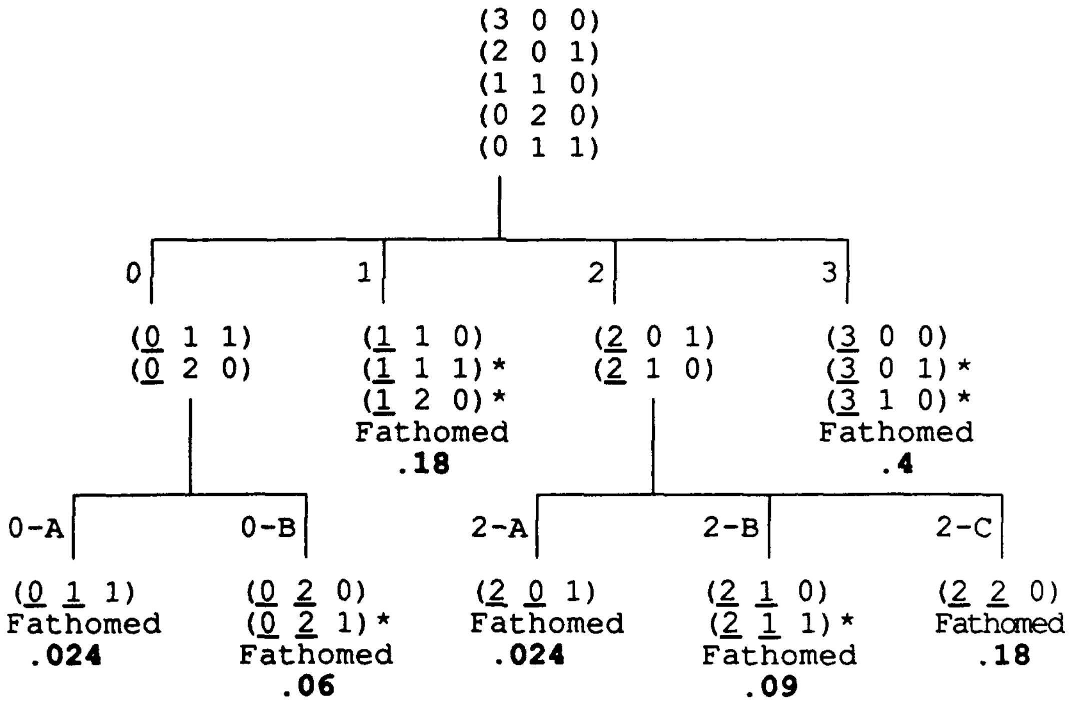


Figure 3.1 Pivotal Decomposition Diagram.

Suppose the pivot is on component 1. BRANCH ZERO: the state of component 1 is fixed at level 0. The two LBPs with x_1=0 are (011) and (020). BRANCH ONE: the state of component 1 is fixed at level 1. The one LBP with x_1=1 is (110). Temporary LBPs are established at (111) and (120). Both are eliminated since they are greater than (110). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=1]$ $Pr[X_2\geq1]$ $Pr[X_3\geq0]$ = (.2)(.9)(1) = .18. BRANCH TWO: the state of component 1 is fixed at level 2. The one LBP with $x_1=2$ is (201). A temporary LBP is established at (210). BRANCH THREE: the state of component 1 is fixed at level 3. The one LBP with $x_1=3$ is (300). Temporary LBPs are established at (301) and (310). Both are eliminated since they are greater that (300). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=3]$ $Pr[X_2\geq0]$ $Pr[X_2\geq0]$ = (.4)(1)(1) = .4. Component 1 is at its maximum state, but two branches remain unfathomed! Let the next pivot be on component 2. BRANCH ZERO-A: the state of component 2 is fixed at level 1. The one LBP off branch one with $x_2=1$ is (011). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=0]$ $Pr[X_2=1]$ $Pr[X_3\geq1]$ = (.1)(.3)(.8) = .024. BRANCH ZERO-B: the state of component 2 is fixed at level 2. The one LBP off branch one with $x_2=2$ is (020). A temporary LBP is established at (021). It is eliminated since it is greater than (020). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=0]$ $Pr[X_2=2]$ $Pr[X_3\geq0]$ = (.1)(.6)(1) = .06. BRANCH TWO-A: the state of component 2 is fixed at level 0. The one LBP off branch three with $x_2=0$ is (201). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=2]\ Pr[X_2=0]\ Pr[X_3 \geq 1] = (.3)(.1)(.8) = .024$. BRANCH TWO-B: the state of component 2 is fixed at level 1. The one LBP off branch three with $x_2=1$ is (210). A temporary LBP is established at (211). It is eliminated since it is greater than (210). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=2]\ Pr[X_2=1]\ Pr[X_3 \geq 0] = (.3)(.3)(1) = .09$. BRANCH TWO-C: the state of component 2 is fixed at level 2. There are no LBPs off branch three with $x_2=2$. A temporary LBP is established at (220). Since only one LBP remains, the branch is fathomed and the probability is determined as $Pr[X_1=2]\ Pr[X_2=2]\ Pr[X_3 \geq 0] = (.3)(.6)(1) = .18$. All the branches are fathomed. The tallied probability is $.18 + .4 + .024 + .06 + .024 + .09 + .18 = .958$.

3.2.4.4 Modular Decomposition

The main purpose for modular decomposition is to reduce the number of calculations necessary to determine the system's probability distribution. Hudson and Kapur [1983a] developed a measure to evaluate the effectiveness of each given decomposition. Let M_{ij} denote the maximum value of component i in module j and n_j represent the number of components in module j. Let M^j denote the maximum value of module j and k denote the number of modules. Let $\{(A_1,\chi_1),(A_2,\chi_2),\ldots,(A_k,\chi_k)\},\psi$ be a modular decomposition of (C,ϕ). Let $n(\phi)$, $n(\psi)$, and $n(\chi_j)$ be the domains of the original structure, the organizing structure, and the jth module of the system. Then $n(\phi) = \prod_{i=1}^{n} (M_i + 1)$, $n(\psi) = \prod_{j=1}^{k} (M^j + 1)$ and $n(\chi_j) = \prod_{i=1}^{n_j} (M_{ij} + 1)$. The index of modular efficiency is defined by the following ratio:

$$I_\psi = \frac{n(\psi) + \sum_{j=1}^{k} n(\chi_j)}{n(\phi)}$$

$$= \frac{\prod_{j=1}^{k} (M^j + 1) + \sum_{j=1}^{k} \prod_{i=1}^{n_j} (M_{ij} + 1)}{\prod_{i=1}^{n} (M_i + 1)}$$

The next example demonstrates the calculation of the index of efficiency for a given modular decomposition.

EXAMPLE 3.10 Consider the system in Figure 3.2.

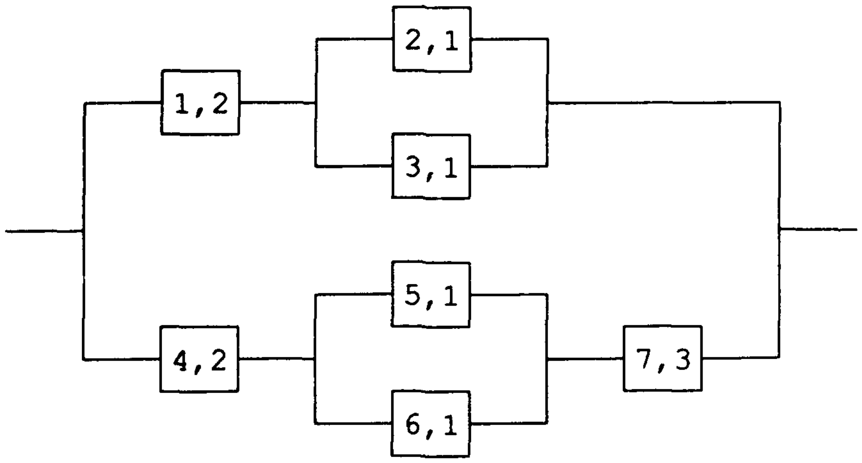


Figure 3.2 Seven Component Coherent Structure.

The numbers in each box represent the component number and maximum state of the component. Determine the index of modular efficiency considering the following modular decomposition

$A_1 = \{c_1\}$	$M^1 = 2$
$A_2 = \{c_2, c_3\}$	$M^2 = 2$
$A_3 = \{c_4\}$	$M^3 = 2$
$A_4 = \{c_5, c_6\}$	$M^4 = 2$
$A_5 = \{c_7\}$	$M^5 = 3$

For the original system, $n(\phi) = 3^2 \cdot 2^4 \cdot 4 = 576$. For the decomposition, $n(\psi) = 3^4 \cdot 4 = 324$, $n(\chi_1) = 3$, $n(\chi_2) = 4$, $n(\chi_3) = 3$, $n(\chi_4) = 4$, and $n(\chi_5) = 4$. The index $I_\psi = (324 + 3 + 4 + 3 + 4 + 4)/576 = 342/576 = .59375$

The efficiency indicator estimates that finding the probability distribution using the given modular decomposition only requires 60% of the calculations required using the original structure.

3.2.5 Bounding System Performance

The applicable system performance bounds will depend on the substitute characteristics for reliability chosen by the customer. Block and Savits [1982] developed several bounds for $\Pr[\phi(\mathbf{X}) \geq k]$. Bounds on $E[\phi(\mathbf{X})]$ can be derived using the following relationship between the two performance measures:

THEOREM 3.10 $E[\phi(\mathbf{X})] = \Pr[\phi(\mathbf{X}) \geq 1] + \Pr[\phi(\mathbf{X}) \geq 2] + \ldots + \Pr[\phi(\mathbf{X}) \geq M]$.

Proof: $E[\phi(\mathbf{X})] = \sum_{k=0}^{M} k \Pr[\phi(\mathbf{X}) = k]$

$= P_1 + (2)\, P_2 + \ldots + (M)\, P_M$

$= (P_1 + P_2 + \ldots + P_M) + (P_2 + P_3 + \ldots + P_M) + \ldots + P_M$

$= Q_1 + Q_2 + \ldots + Q_M.$

Therefore, the bounds found for $\Pr[\phi(\mathbf{X}) \geq k]$, $k=1,2,\ldots,M$ allow similar bounds to be constructed for $E[\phi(\mathbf{X})]$.

Performance bounds can be constructed for independent and associated random variables. Of course, the bounds are more explicit if the random variables are independent. The bounds are based on a commonly known theorem.

THEOREM 3.11 If $X_1, X_2, \ldots, X_n$ are associated random variables, then

$$\Pr[X_1 > x_1, \ldots, X_n > x_n] \geq \prod_{i=1}^{n} \Pr[X_i > x_i]$$

and

$$\Pr[X_1 \leq x_1, \ldots, X_n \leq x_n] \geq \prod_{i=1}^{n} \Pr[X_i \leq x_i].$$

The next six sections describe some of the bounds on $\Pr[\phi(\mathbf{X}) \geq k]$ derived by other authors. For the case of mutually independent components, the bounds are implemented in the computer program found in Appendix B.

3.2.5.1 Trivial Bounds

Trivial bounds similar to Theorem 2.10 were developed and proven by Hudson [1981]. These bounds are based on a single lower boundary point to level k.

THEOREM 3.12 Let $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in L_k$, $k=1,2,\ldots,M$.

Then $$\prod_{i=1}^{n} Q_{i,y_i} \leq Q_k \leq 1 - \prod_{i=1}^{n} (1 - Q_{i,y_i}).$$

Proof: Suppose $\mathbf{y} \in L_k$. Then $\phi(\mathbf{y}) \geq k$ by definition. Since ϕ is increasing, if $\mathbf{x} \geq \mathbf{y}$, then $\phi(\mathbf{x}) \geq \phi(\mathbf{y})$. Thus, $\Pr[\mathbf{X} \geq \mathbf{y}] \leq \Pr[\phi(\mathbf{X}) \geq \phi(\mathbf{y}) \geq k]$, or

$\Pr[X_1 \geq y_1, X_2 \geq y_2, \ldots, X_n \geq y_n] \leq \Pr[\phi(\mathbf{X}) \geq k]$, or

$\Pr[X_1 \geq y_1] \Pr[X_2 \geq y_2] \cdots \Pr[X_n \geq y_n] \leq \Pr[\phi(\mathbf{X}) \geq k]$, or

$\prod_{i=1}^{n} \Pr[X_i \geq y_i] \leq \Pr[\phi(\mathbf{X}) \geq k]$. Therefore, $\prod_{i=1}^{n} Q_{i,y_i} \leq Q_k$.

Since $Y \in L_k$, if $\mathbf{x} < \mathbf{y}$, then $\phi(\mathbf{x}) < k$. Thus,

$\Pr[\mathbf{X} < \mathbf{y}] \leq \Pr[\phi(\mathbf{X}) < k]$. By definition,

$\Pr[\mathbf{X} << \mathbf{y}] \leq \Pr[\mathbf{X} < \mathbf{y}] \leq \Pr[\phi(\mathbf{X}) < k]$, or

$\prod_{i=1}^{n} \Pr[X_i < y_i] \leq \Pr[\phi(\mathbf{X}) < k]$, or

$\prod_{i=1}^{n} \{1 - \Pr[X_i \geq y_i]\} \leq \{1 - \Pr[\phi(\mathbf{X}) \geq k]\}$, or

$\prod_{i=1}^{n} (1 - Q_{i,y_i}) \leq 1 - Q_k$. Thus , $Q_k \leq 1 - \prod_{i=1}^{n} (1 - Q_{i,y_i})$.

EXAMPLE 3.11 Suppose that $M_1 = 3$ and $M_2 = 2$ for a two-component structure with $M = 4$. Suppose the component probability distributions for the two components are:

$P_{10} = .1 \quad P_{11} = .1 \quad P_{12} = .1 \quad P_{13} = .7$

$P_{20} = .1 \quad P_{21} = .3 \quad P_{22} = .6$

Construct the bounds on Q_3 if $\mathbf{x} = (2,2)$ is a lower boundary point to level 3.

$Q_3 \geq \Pr[X_1 \geq 2] \Pr[X_2 \geq 2] = (.8)(.6) = .48$.

$Q_3 \leq 1 - (1 - \Pr[X_1 \geq 2])(1 - \Pr[X_2 \geq 2])$

$= 1 - (1 - .8)(1 - .6) = 1 - .08 = .92$.

Construct the bounds on Q_1 if $\mathbf{x} = (0,1)$ is a lower boundary point to level 1.

$Q_1 \geq \Pr[X_1 \geq 0] \Pr[X_2 \geq 1] = (1)(.9) = .9.$

$Q_1 \leq 1 - (1 - \Pr[X_1 \geq 0])(1 - \Pr[X_2 \geq 1])$

$= 1 - (1 - 1)(1 - .9) = 1 - 0 = 1.0.$

Similar bounds were derived using a single upper boundary point to level k, k=0,1,...,M-1.

THEOREM 3.13 Let $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in U_k$, $k=0,1,\ldots,M-1$.

Then $\prod_{i=1}^{n} Q_{i,y_i+1} \leq Q_{k+1} \leq 1 - \prod_{i=1}^{n} (1 - Q_{i,y_i+1})$.

Proof: Similar to Theorem 3.12.

3.2.5.2 Path/Cut Bounds

Path/Cut Bounds were developed by Block and Savits [1982] from the lower and upper boundary points for the structure. Suppose the s_k lower boundary points to level k are given by $L_{k1}, L_{k2}, \ldots, L_{k,s_k}$. Let the set of all lower boundary points to level k be L_k. If $\mathbf{x} \in L_k$, then let

$$\mathfrak{L}_k(\mathbf{x}) = \{(i, x_i) \text{ for all } x_i \neq 0\}.$$

Suppose the t_k upper boundary points to level k are given by $U_{k1}, U_{k2}, \ldots, U_{k,t_k}$. Let the set of all upper boundary points to level k be designated U_k. If $\mathbf{x} \in U_k$, then let

$$\mathfrak{U}_k(\mathbf{x}) = \{(i, x_i) \text{ for all } x_i \neq M_i\}.$$

THEOREM 3.14 Let ϕ be a general MCS with associated components. Then

$$\prod_{\mathbf{x}\in U_{k-1}} \Pr\left[\bigcup_{(i,j)\in\mathfrak{U}_{k-1}(\mathbf{x})} \{X_i>j\}\right] \le Q_k \le \coprod_{\mathbf{x}\in L_k} \Pr\left[\bigcap_{(i,j)\in\mathfrak{L}_k(\mathbf{x})} \{X_i>j-1\}\right]$$

for k=1,2,...,M.

The lower bound comes from the upper boundary points while the upper bound comes from the lower boundary points.

When the components are independent, the bounds of Theorem 3.14 can be explicitly derived from the performance distributions of the components.

THEOREM 3.15 Let ϕ be a general MCS with independent components. Then

$$\prod_{\mathbf{x}\in U_{k-1}} \coprod_{(i,j)\in\mathfrak{U}_{k-1}(\mathbf{x})} Q_{i,j+1} \le Q_k \le \coprod_{\mathbf{x}\in L_k} \prod_{(i,j)\in\mathfrak{L}_k(\mathbf{x})} Q_{i,j}$$

for k=1,2,...,M.

3.2.5.3 Min/Max Bounds

Min/Max Bounds were developed by Block and Savits [1982] so that the lower bound comes from the minimal path sets while the upper bound comes from the minimal cut sets.

THEOREM 3.16 Let ϕ be a general MCS. Then the following bounds always hold for k=1,2,...,M:

$$\operatorname*{Max}_{\mathbf{x}\in L_k} \Pr\left[\bigcap_{(i,j)\in\mathfrak{L}_k(\mathbf{x})} \{X_i>j-1\}\right] \le Q_k \le \operatorname*{Min}_{\mathbf{x}\in U_{k-1}} \Pr\left[\bigcup_{(i,j)\in\mathfrak{U}_{k-1}(\mathbf{x})} \{X_i>j\}\right].$$

If the components are associated, then

$$\operatorname*{Max}_{\mathbf{x}\in L_k} \left\{\prod_{(i,j)\in\mathfrak{L}_k(\mathbf{x})} Q_{i,j}\right\} \le Q_k \le \operatorname*{Min}_{\mathbf{x}\in U_{k-1}} \left\{\coprod_{(i,j)\in\mathfrak{U}_{k-1}(\mathbf{x})} Q_{i,j+1}\right\}.$$

3.2.5.4 Combining Bounds

The upper boundary points generally provide a tighter bound for mutually independent components with large probabilities in the higher states and the lower boundary points generally provide a tighter bound for mutually independent components with large probabilities in the lower states. Thus, a combination of the bounds in Theorems 3.15 and 3.16 is appropriate for mutually independent components.

THEOREM 3.17 Let ϕ be a coherent system of independent components. Then use the maximum lower bound and minimum upper bound found with Theorems 3.15 and 3.16.

3.2.5.5 Improved Path/Cut Bounds

Butler [1982] showed that improving Path/Cut Bounds with modular decomposition was also applicable to the multistate model. Path/Cut Bounds are determined for each module. The bounds are then used to determine Path/Cut Bounds for the system. Butler [1982] proved that these bounds were always tighter than the Path/Cut Bounds found from the system.

3.2.5.6 Inclusion-Exclusion Bounds

Natvig [1982] developed bounds using the inclusion-exclusion principle of Feller [1968]. Let Σ_j be the jth summation term in Theorem 2.8. Using Theorem 3.9, the probability distribution can be bounded as follows:

$$\Pr[\phi(\mathbf{X}) \geq k] \leq \Sigma_1$$

$$\Pr[\phi(\mathbf{X}) \geq k] \geq \Sigma_1 - \Sigma_2$$

$$\Pr[\phi(\mathbf{X}) \geq k] \leq \Sigma_1 - \Sigma_2 + \Sigma_3$$

$$\vdots$$

$$\Pr[\phi(\mathbf{X}) \geq k] = \Sigma_1 - \Sigma_2 + \Sigma_3 - \dots + (-1)^{s+1} \Sigma_s.$$

Unfortunately, the upper and lower bounds do not consistently improve as more terms are added. In fact, Inclusion-Exclusion Bounds are not restricted between 0 and 1. The only guarantee is that exact probability distribution will be found after determining every summation. Still, only a few terms may be needed to bound $\Pr[\phi(\mathbf{X}) \geq k]$ tightly.

3.3 Dynamic Properties

In the last two sections, the structural and stochastic properties of the general MCS were examined at a fixed moment in time. As in Chapter 2, the next step is to consider dynamic models, where the state of the components and the system vary with time. Barlow and Wu [1978], El-Neweihi et al. [1978], and Ross [1979] developed dynamic models for multistate coherent systems. Hudson [1981] extended the development of dynamic models for a general MCS.

3.3.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

t	fixed time; $t \geq 0$.
$X(t)$	state of stochastic process for a given t.
$X_i(t)$	state of component i at time t, $i=1,2,\ldots,n$.
$\mathbf{X}(t)$	vector of random component states at time t; $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$.
$\phi(\mathbf{X}(t))$	random system state at time t.
T^j	at first, time for stochastic process reach or go below state j; $T^j = \inf\{t \mid X(t) \leq j\}$;

	later, time for stochastic process to go below state j; $T^j = \inf\{t \mid X(t) < j\}$.
$R(t)$	survivor function for system at time t; $R(t) = \Pr[\phi(\mathbf{X}(t)) = 1] = E[\phi(\mathbf{X}(t))] = \Pr[T > t]$.
T_i^j	time for state of component i to go below state j; $T_i^j = \inf\{t \mid X_i(t) < j\}$.
T_{ij}	length of time component i spends in state j.
$Q_k(t)$	performance distribution of the system; $Q_k(t) = \Pr[\phi(\mathbf{X}(t)) \geq k]$, $k=1,2,\ldots,M$.
$Q_{ij}(t)$	performance distribution for component i; $Q_{ij}(t) = Q_{ij}(t) = \Pr[X_i(t) \geq j]$, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$.
$\mathbf{q}_i(t)$	performance vector for component i; $\mathbf{q}_{ij}(t) = (Q_{i1}(t), Q_{i2}(t), \ldots, Q_{i,M_i}(t))$.
$\mathbf{q}(t)$	performance vector for the system; $\mathbf{q}(t) = (Q_1(t), Q_2(t), \ldots, Q_M(t))$.
$r(t)$	system performance function; $r(t) = r(\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_n(t))$.
μ_{ij}	mean of distribution for $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{i,j}$.

3.3.2 Distribution Representations

In general, a stochastic process $\{X(t),\ t \in T\}$ is a collection of random variables $X(t)$ representing the state of the process for a given value of t. The state space of a stochastic process is defined as the set of all possible values that $X(t)$ can assume.

Let $\{X_i(t),\ t \geq 0\}$ for $i=1,2,\ldots,n$ be the decreasing and right continuous stochastic process representing the state of component i at time t, where t ranges over the nonnegative real numbers. The components and associated stochastic processes are assumed to be mutually independent. Let the vector of random component states at time t be designated by $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$. Let $\{\phi(\mathbf{X}(t)),\ t \geq 0\}$ denote the decreasing and right continuous stochastic process that represents the system state at time t.

3.3.3 Distribution Classes and Closure

Recall that Barlow and Wu [1978] defined a multistate system so that $\mathbf{\Omega} = \mathbf{\Omega}_1 = \{0,1,\ldots,M\}$. Suppose $\mathbf{\Omega}$ and $\mathbf{\Omega}_1$ can be divided into "bad" states $\{0,1,\ldots,j-1\}$ and "good" states $\{j,j+1,\ldots,M\}$. Essentially, this converts the multistate problem to a binary model. Using this transformation, Barlow and Wu [1978] applied the binary definition for IFRA.

> **DEFINITION.** Let j be fixed at the lowest "good" state. The distribution of time for component i to leave the "good" states starting from state M is an <u>IFRA random variable</u> if $\Pr[X_1(t) \geq j]^{1/t}$ is decreasing in $t \geq 0$ for some fixed j.

They proved IFRA closure with respect to the formation of coherent systems. That is, Barlow and Wu [1978] showed that if the length of time spent by each component in the "good" states is an IFRA random variable, then the corresponding length of time spent by the multistate system in the "good" states is also an IFRA random variable.

Ross [1979] basically followed the same strategy, but instead of fixing j, the length of time for a stochastic process to reach or go below state j must be an IFRA random variable for every possible j.

DEFINITION. The stochastic process $\{X(t),\ t \geq 0\}$ is an <u>IFRA process</u> if $T^j = \inf\{t \mid X(t) \leq j\}$ is an IFRA random variable for every j.

Ross [1979] proved IFRA closure with respect to the formation of coherent systems. That is, if $\{X_i(t),\ t \geq 0\}$, $i=1,2,\ldots,n$ are increasing independent IFRA processes, then $\{\phi(\mathbf{X}(t)),\ t \geq 0\}$ is also an IFRA process whenever ϕ is decreasing.

El-Neweihi et al. [1978] used the binary definition for a NBU random variable to define a NBU stochastic process.

DEFINITION. The stochastic process $\{X(t),\ t \geq 0\}$ is a <u>NBU stochastic process</u> if T^j is a NBU random variable for $j=0,1,\ldots,M-1$.

El-Neweihi et al. [1978] proved NBU closure with respect to the formation of coherent systems. So, if ϕ is an MCS and $\{X_i(t),\ t \geq 0\}$, $i=1,2,\ldots,n$ are independent NBU stochastic processes, then $\{\phi(\mathbf{X}(t)),\ t \geq 0\}$ is a NBU stochastic process.

Hudson [1981] proved IFRA and NBU closure for a general MCS. His definitions and theorems are modified slightly so that the random variable T^j is the first time the process goes below state j.

DEFINITIONS. T is an <u>IFRA random variable</u> if $(-1/t) \log R(t)$ is increasing in $t \geq 0$. T is an

NBU random variable if $R(t+x) \le R(t)R(x)$ for all $t \ge 0$ and $x \ge 0$.

DEFINITIONS. The stochastic process $\{X_i(t),\ t \ge 0\}$ is an IFRA (NBU) process if $T_i^j = \inf\{t \mid X_i(t) < j\}$ is an IFRA (NBU) random variable for $j=1,2,\ldots,M_i$.

DEFINITIONS. The stochastic process $\{\phi(\mathbf{X}(t)),\ t \ge 0\}$ is an IFRA (NBU) process if $T^j = \inf\{t \mid \phi(\mathbf{X}(t)) < j\}$ is an IFRA (NBU) random variable for $j=1,2,\ldots,M$.

THEOREM 3.18 Let $\{X_i(t),\ t \ge 0\}$, $i=1,2,\ldots,n$ be mutually independent stochastic IFRA (NBU) processes. If ϕ is a general MCS, then $\{\phi(\mathbf{X}(t)),\ t \ge 0\}$ is a stochastic IFRA (NBU) process.

There is one special case when it is simple to prove that T_i^j, $j=1,2,\ldots,M_i$ are IFRA (NBU) random variables. Let T_{ij} be the length of time component i spends in state j. Then $T_i^j = T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{i,j}$. If $T_{i1}, T_{i2}, \ldots, T_{i,M_i}$ are independent IFRA (NBU) random variables, then T_i^j, $j=1,2,\ldots,M_i$ are IFRA (NBU) random variables because the IFRA (NBU) class is closed with respect to the convolution of independent random variables.

3.3.4 Exact System Performance

For the general MCS, the problem changes to finding the performance distribution of the system, $Q_k(t)$, where

$$Q_k(t) = \Pr[\phi(\mathbf{X}(t)) \geq k],\ k=1,2,\ldots,M$$

from each component's performance distribution, $Q_{ij}(t)$, where

$$Q_{ij}(t) = \Pr[X_i(t) \geq j],\ i=1,2,\ldots,n \text{ and } j=1,2,\ldots,M_i.$$

In terms of previously defined variables for time, $Q_{ij}(t) = \Pr[X_i(t) \geq j] = \Pr[T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{i,j} > t] = \Pr[T_i^j > t]$.

$Q_{i0}(t) = 1.0$ for $i=1,2,\ldots,n$.

Let $\mathbf{q}_i(t) = (Q_{i1}(t), Q_{i2}(t), \ldots, Q_{i,M_i}(t))$ be the performance vector for component i and $\mathbf{q}(t) = (Q_1(t), Q_2(t), \ldots, Q_M(t))$ be the performance vector for the system. If the stochastic processes $\{X_i(t),\ t \geq 0\}$, $i=1,2,\ldots,n$ are independent, then $\mathbf{q}(t)$ may be expressed as a function of $\mathbf{q}_i(t)$, $i=1,2,\ldots,n$. $E[\phi(\mathbf{X}(t))]$, $\Pr[\phi(\mathbf{X}(t)) \geq k]$, and $E[u(\phi(\mathbf{X}(t)))]$ are measures of performance that can be defined as functions of $\mathbf{q}(t)$ and therefore, they may also be expressed as a function $\mathbf{q}_i(t)$, $i=1,2,\ldots,n$. The relationship between the system's measure of performance and the component performance vectors is given by the performance function:

$$r(t) = r(\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_n(t)).$$

EXAMPLE 3.12 Suppose the T_{1j}, $j=1,2,3$ are mutually independent exponential variables with $\lambda_1 = .2$ and T_{2j}, $j=1,2$ are mutually independent exponential variables with $\lambda_2 = .5$. Find $\mathbf{q}(t)$ and $E[\phi(\mathbf{X}(t))]$ for a system with $n = 2$, $M_1 = 3$, $M_2 = 2$, and $M = 2$ if the lower boundary points to level 1 are $L_1 = \{(2,0),\ (0,1)\}$ and the lower boundary points to level 2 are $L_2 = \{(3,1),\ (1,2)\}$.

The distribution of $T_{i,M_i} + T_{i,M_i-1} + \dots + T_{ij}$ is the convolution of (M_i-j+1) exponential distributions each with parameter λ_i which is an Erlang distribution with shape parameter (M_i-j+1) and scale parameter λ_i. In general, for Erlang (η,λ),

$$R(t) = \Pr[T > t] = 1 - F(t) = \sum_{k=0}^{\eta-1} \frac{(\lambda t)^k \exp(-\lambda t)}{k!}.$$

Therefore, $Q_{ij}(t) = \displaystyle\sum_{k=0}^{M_i-j} \frac{(\lambda_i t)^k \exp(-\lambda_i t)}{k!}$.

$Q_{13}(t) = e^{-.2t}$,

$Q_{12}(t) = e^{-.2t} + .2t\ e^{-.2t}$,

$Q_{11}(t) = e^{-.2t} + .2t\ e^{-.2t} + \dfrac{(.2t)^2\ e^{-.2t}}{2!}$,

$Q_{10}(t) = 1.0$,

$Q_{23}(t) = e^{-.5t}$,

$Q_{22}(t) = e^{-.5t} + .5t\ e^{-.5t}$,

$Q_{21}(t) = e^{-.5t} + .5t\ e^{-.5t} + \dfrac{(.5t)^2\ e^{-.5t}}{2!}$, and

$Q_{20}(t) = 1.0$.

Using the lower boundary points to level 1,

$Q_1(t) = Q_{12}(t)\ Q_{20}(t) + Q_{10}(t)\ Q_{21}(t) - Q_{12}(t)\ Q_{21}(t)$.

Using the lower boundary points to level 2,

$Q_2(t) = Q_{13}(t)\ Q_{21}(t) + Q_{11}(t)\ Q_{22}(t) - Q_{13}(t)\ Q_{22}(t)$.

From Theorem 3.10, $E[\phi(\mathbf{X}(t))] = Q_1(t) + Q_2(t)$. Table 3.7 summarizes the calculations for $E[\phi(\mathbf{X}(t))]$ and $\Pr[\phi(\mathbf{X}(t)) \geq k]$ at various times.

3.3.5 Bounding System Performance

For the dynamic situation, finding the exact performance distribution of the system from the performance distributions of the n independent components is a difficult problem. As in Chapter 2, the closure theorems can be used to develop a lower bound on system performance.

Table 3.7 System Performance at Various Times.

Random Variable	Time			
	1	2	3	4
$Q_{13}(t)$	.8187	.6703	.5488	.4493
$Q_{12}(t)$	.9825	.9384	.8781	.8088
$Q_{11}(t)$	.9989	.9921	.9769	.9526
$Q_{10}(t)$	1.000	1.000	1.000	1.000
$Q_{23}(t)$	.6065	.3679	.2231	.1353
$Q_{22}(t)$	.9098	.7358	.5578	.4060
$Q_{21}(t)$	.9856	.9197	.8088	.6767
$Q_{20}(t)$	1.000	1.000	1.000	1.000
$\Pr[\phi(\mathbf{X}(t)) \geq 1]$	.9997	.9951	.9767	.9382
$\Pr[\phi(\mathbf{X}(t)) \geq 2]$	.9708	.8532	.6827	.5084
$E[\phi(\mathbf{X}(t))]$	1.971	1.848	1.659	1.447

Let $r(\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_n(t))$ be the performance function of a coherent system of n mutually independent components. Generalizing a result from the binary model, suppose that $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{i,j}$ have unknown IFR distributions with known means μ_{ij}. Since the exponential distribution is the limiting distribution for the IFR class, $Q_{ij}(t) \geq \exp[-t/\mu_{ij}]$ for $t < \mu_{ij}$. Hudson [1981] was able to develop the lower bound on the performance function for a general MCS:

THEOREM 3.19 Let $\mathbf{s}_i(t) = (S_{11}(t), S_{12}(t), \ldots, S_{i,M_i}(t))$ for $i=1,2,\ldots,n$. Let $S_{ij}(t) = \exp[-t/\mu_{ij}]$ for $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$. Suppose that T_i^j have unknown IFR distributions with known means μ_{ij}, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$, then $r(t) = r(\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_n(t)) \geq r(\mathbf{s}_1(t), \mathbf{s}_2(t), \ldots, \mathbf{s}_n(t))$ for $t < \text{Min}\{\mu_{ij}\}$.

EXAMPLE 3.13 In the previous example, suppose that the distributions of T_{1j} and T_{2j} are unknown but the means are $\mu_1 = 5$ and $\mu_2 = 2$. Find a lower bound for $Q_1(t)$, $Q_2(t)$, and $E[\phi(\mathbf{X}(t))]$ given only that T_{1j}, $j=1,2,3$ and T_{2j}, $j=1,2$ are mutually independent IFR random variables.

The distribution of $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ij}$ is IFR and the convolution of (M_i-j+1) distributions.

Therefore, $\mu_{13} = 5$, $\mu_{12} = 10$, and $\mu_{11} = 15$.

$\mu_{23} = 2$, $\mu_{22} = 4$, and $\mu_{21} = 6$.

From Theorem 3.19,

$Q_{13}(t) \geq e^{-t/5}$, $Q_{12}(t) \geq e^{-t/10}$, and $Q_{11}(t) \geq e^{-t/15}$.

$Q_{23}(t) \geq e^{-t/2}$, $Q_{22}(t) \geq e^{-t/4}$, and $Q_{21}(t) \geq e^{-t/6}$.

Using the lower boundary points to level 1,

$Q_1(t) \geq e^{-t/10} + e^{-t/6} - e^{-t/10}\, e^{-t/6}$.

Using the lower boundary points to level 2,

$Q_2(t) \geq e^{-t/5}\, e^{-t/6} + e^{-t/15}\, e^{-t/4} - e^{-t/5}\, e^{-t/4}$.

From Theorem 3.10, $E[\phi(\mathbf{X}(t))] = Q_1(t) + Q_2(t) \geq$ $e^{-.1t} + e^{-.167t} - e^{-.267t} + e^{-.367t} + e^{-.317t} - e^{-.45t}$.

Table 3.8 shows calculations for lower bounds on $E[\phi(\mathbf{X}(t))]$ and $\Pr[\phi(\mathbf{X}(t)) \geq k]$ at various times.

Table 3.8 Bounds on System Performance at Various Times.

Variable	Time			
	1	2	3	4
$Q_{13}(t)$	.8187	.6703	.5488	.4493
$Q_{12}(t)$	.9048	.8187	.7408	.6703
$Q_{11}(t)$	.9355	.8752	.8187	.7659
$Q_{10}(t)$	1.000	1.000	1.000	1.000
$Q_{23}(t)$	.6065	.3679	.2231	.1353
$Q_{22}(t)$	.7788	.6065	.4724	.3679
$Q_{21}(t)$	.8465	.7165	.6065	.5134
$Q_{20}(t)$	1.000	1.000	1.000	1.000
$\Pr[\phi(\mathbf{X}(t)) \geq 1]$	.9854	.9486	.8980	.8396
$\Pr[\phi(\mathbf{X}(t)) \geq 2]$	.7840	.6046	.4604	.3472
$E[\phi(\mathbf{X}(t))]$	1.769	1.553	1.358	1.188

3.4 Boundary Point Conversion

A customer can completely describe a general MCS by specifying when a change in the state of any one of the components forces a change in the state of the system. Some customers may prefer to relate system and component state changes in terms of deterioration by specifying the lower boundary points. Other customers may wish to describe the system in terms of state improvement by specifying the upper boundary points.

In either case, getting the boundary points from the customer can be a long and tedious process. In fact, no efficient method has been developed for finding all the boundary points. To make this process easier, Wood [1985] developed the concept of multistate block diagrams and Janan [1985] proposed two algorithms that take advantage of the modularity of a system. Because of the difficulties involved with soliciting boundary points, it is not always reasonable to obtain both the upper and lower boundary points from the customer.

The boundary points are essential for calculating the exact probability distribution using inclusion-exclusion or pivotal decomposition. For large, complex systems, it is not always possible to find the exact probability distribution. In this case, the performance bounds given in section 3.2.5 must be used.

Almost all the performance bounds require both upper and lower boundary points. Therefore, it would be useful to develop a procedure to find the upper boundary points from the lower boundary points and vice versa. The program that accomplishes this task can be found in Appendix C. It is based on the following two algorithms. The first algorithm converts the upper boundary points to level k (U_k) to the lower boundary points to level k+1 (L_{k+1}).

U_k to L_{k+1} Conversion Algorithm

1. Set $k = 0$.

2. Stop if $k = M$. List all upper boundary points to level k ($\mathbf{x} \in U_k$).

3. For each $\mathbf{x} \in U_k$, list the potential lower boundary points to level $k+1$. The potential lower boundary points to level $k+1$ ($\mathbf{y}$) for an upper boundary point to level k ($\mathbf{x} \in U_k$) are defined as all $\mathbf{y} \in S$ such that $y_i = x_i + 1$ for one i and $y_j=0,1,\ldots,x_j \; \forall \; j \neq i$.

4. Eliminate from the list any $\mathbf{y}$ dominated by other $\mathbf{x} \in U_k$. $\mathbf{y}$ is dominated by $\mathbf{x} \in U_k$ if $\mathbf{y} \leq \mathbf{x}$.

5. Eliminate from the new list any $\mathbf{y}^1$ overcome by $\mathbf{y}^2$ where $\mathbf{y}^2$ overcomes $\mathbf{y}^1$ if $\mathbf{y}^1 \geq \mathbf{y}^2$. If $\mathbf{y}^1 = \mathbf{y}^2$, then only eliminate one vector from the list.

6. The remaining $\mathbf{y}$ on the list are the lower boundary points to level $k+1$.

7. Clear the list, set $k = k + 1$, and return to step 2.

A potential lower boundary points to level $k+1$ cannot have two x_i increased by 1 because of the following theorem.

THEOREM 3.20 If $\mathbf{x} = (x_1,x_2,x_3,\ldots,x_n) \in U_k$, then $\mathbf{x}' = (x_1+1,x_2+1,x_3,\ldots,x_n) \notin L_{k+1}$.

Proof: Suppose that $\mathbf{x}' \in L_{k+1}$. Then $\phi(x_1+1,x_2,\ldots,x_n) < k+1$. Since k is discrete, $\phi(x_1+1,x_2,\ldots,x_n) \leq k$. Now since $\mathbf{x} \in U_k$, $\phi(x_1+1,x_2,\ldots,x_n) > k$. Combining the last two inequalities results in a contradiction proving that the supposition is false.

Similar theorems could be proven when more than two x_i are increased by 1 or when any number of x_i are increased by 2 or more.

EXAMPLE 3.14 Consider a three-component general MCS with $M=4$, $M_1=3$, $M_2=2$, and $M_3=1$. Suppose the upper boundary points to level 1 are given by the customer as $U_1 = \{(2,0,0), (1,0,1), (0,1,0)\}$. Determine the lower boundary points to level 2.

At this point in the algorithm, k=1. The upper boundary points to level 1 are listed across the top of Table 3.9. The potential lower boundary points to level 2 for each $\mathbf{x} \in U_1$ are listed below each upper boundary point. The potential lower boundary points dominated by other $\mathbf{x} \in U_1$ are marked by an asterisk (*). The potential lower boundary points overcome by other potential lower boundary points are marked by a check mark (✓).

Table 3.9 Potential Lower Boundary Points.

200	101	010
300	201 ✓	110 ✓
210 ✓	200 *	100 *
110	111 ✓	020
010 *	110 ✓	011 ✓
201	011	001 *
101 *	010 *	
001 *		

Therefore, $L_2 = \{(3,0,0), (1,1,0), (2,0,1), (0,1,1), (0,2,0)\}$.

In a similar manner, the second algorithm converts the lower boundary points to level k (L_k) to the upper boundary points to level k-1 (U_{k-1}).

L_k to U_{k-1} Conversion Algorithm

1. Set $k = M$.
2. Stop if $k = 0$. List all lower boundary points to level k ($\mathbf{x} \in L_k$).
3. For each $\mathbf{x} \in L_k$, list the potential upper boundary points to level k-1. The potential upper boundary points to level k-1 ($\mathbf{z}$) for a lower boundary point to level k ($\mathbf{x} \in U_k$) are defined as all $\mathbf{z} \in S$ such that $z_i = x_i - 1$ for one i and $z_j = x_j, x_j+1, \ldots, M_j \ \forall \ j \neq i$.
4. Eliminate from the list any $\mathbf{z}$ dominated by other $\mathbf{x} \in L_k$. $\mathbf{z}$ is dominated by $\mathbf{x} \in L_k$ if $\mathbf{z} \geq \mathbf{x}$.
5. Eliminate from the new list any $\mathbf{z}^1$ overcome by $\mathbf{z}^2$ where $\mathbf{z}^2$ overcomes $\mathbf{z}^1$ if $\mathbf{z}^1 \leq \mathbf{z}^2$. If $\mathbf{z}^1 = \mathbf{z}^2$, then only eliminate one vector from the list.
6. The remaining $\mathbf{z}$ on the list are the upper boundary points to level k-1.
7. Clear the list, set $k = k - 1$, and return to step 2.

A potential upper boundary points to level k-1 cannot have two x_i decreased by 1 because of the following theorem.

THEOREM 3.21 If $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n) \in L_k$, then $\mathbf{x}' = (x_1-1, x_2-1, x_3, \ldots, x_n) \notin U_{k-1}$.

Proof: Suppose that $\mathbf{x}' \in U_{k-1}$. Then $\phi(x_1-1,x_2,\ldots,x_n) > k-1$. Since k is discrete, $\phi(x_1-1,x_2,\ldots,x_n) \geq k$. Now since $\mathbf{x} \in L_k$, $\phi(x_1-1,x_2,\ldots,x_n) < k$. Combining the last two inequalities results in a contradiction proving that the supposition is false.

Similar theorems could be proven when any number of x_i are decreased by 1 or more.

EXAMPLE 3.15 Consider a three-component general MCS with M=4, M_1=3, M_2=2, and M_3=1. Suppose the lower boundary points to level 2 are given by the customer as $L_2 = \{(3,0,0), (1,1,0), (2,0,1), (0,1,1), (0,2,0)\}$. Determine the upper boundary points to level 1.

At this point in the algorithm, k=2. The lower boundary points to level 2 are listed across the top of Table 3.10. The potential upper boundary points to level 1 for each $\mathbf{x} \in L_2$ are listed below each lower boundary point. The potential upper boundary points dominated by other $\mathbf{x} \in L_2$ are marked by an asterisk (*). The potential upper boundary points overcome by other potential upper boundary points are marked by a check mark (✓). Therefore, $U_1 = \{(2,0,0), (0,1,0), (1,0,1)\}$ which checks with the upper boundary points to level 1 given in the previous example.

Table 3.10 Potential Upper Boundary Points.

300	110	201	020	011
200	010	101 ✓	010 ✓	001 ✓
201 *	011 *	111 *	011 *	101 ✓
210 *	020 *	121 *	110 *	201 *
211 *	021 *	200 ✓	111 *	301 *
220 *	100 ✓	210 *	210 *	010 ✓
221 *	101	220 *	211 *	020 *
	200 ✓	300 *	310 *	110 *
	201 *	310 *	311 *	120 *
	300 *	320 *		210 *
	301 *			220 *
				310 *
				320 *

3.4 Summary

This chapter reviewed the structural, stochastic, and dynamic properties for the multistate model and generalized the same properties. New definitions were given for a k-out-of-n structure and for a general MCS in terms of lower and upper boundary points. Some of the common duality theorems were proven using the new definitions. Lower and upper bounds were established for the general multistate structure function. Counterexamples were found to disprove the general redundancy theorems for the general multistate case. The concepts of an alternate representation for $\phi(\mathbf{x})$, structural importance, and reliability importance were generalized. A computer program was written to find the exact probability distribution of the system. A separate computer program was created to calculate performance bounds for complex systems. To limit the amount of information needed from the customer, a third program was developed to determine the upper boundary points from the lower boundary points and vice versa.

4. THE CONTINUOUS MODEL

Chapter 2 discussed the binary mapping $\phi:\{0,1\}^n \rightarrow \{0,1\}$. Most of the previous work in reliability theory has adopted this binary model where the system and component states are restricted to one of two possible values. Chapter 3 explored two multistate mappings - $\phi:\{0,1,\ldots,M\}^n \rightarrow \{0,1,\ldots,M\}$ and $\phi:\{0,1,\ldots,M_i\}^n \rightarrow \{0,1,\ldots,M\}$. These models extended the framework to allow the system and component states to assume a finite number of values. The general multistate model allowed the number of system and component states to be different. Chapter 4 discusses the recent extension of the reliability model where the system and component states can degrade through a continuum of values. The continuous model was first developed using the mapping $\phi: \Delta \rightarrow [0,\infty)$ where Δ is some subset of $\mathbb{R}^n$, but most articles have focused on continuum structure functions that map from the unit hypercube to the unit interval, i.e. $\phi: [0,1]^n \rightarrow [0,1]$. In this chapter, as was done for the multistate model, the continuous model is generalized to $\phi: [0,M_i]^n \rightarrow [0,M]$.

4.1 Structural Properties

Structural properties characterize the deterministic relationship between the state of the system and the states of the components at a fixed moment in time.

4.1.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components comprising the system.
$\mathbb{R}^n$	a vector of n real numbers.
Δ	an arbitrary subset of $\mathbb{R}^n$.
$\mathbb{R}_+^n$	a specific subset of $\mathbb{R}^n$; the nonnegative orthant.
x_i	state of component i; $x_i \in [0,M_i]$.
M_i	best state of component i; $M_i < \infty$.
Ω_i	state space of component i; $\Omega_i = [0,M_i]$.
$\mathbf{x}$	component state vector; $\mathbf{x} = (x_1,x_2,\ldots,x_n)$.
S	component state space; $\{\mathbf{x} \mid x_i \in \Omega_i, \forall i\}$.
ϕ	state of the system; $\phi \in [0,M]$.
M	best state of the system; $M < \infty$.
Ω	state space of the system; $\Omega = [0,M]$.
$\phi(\mathbf{x})$	structure function; system state for $\mathbf{x}$.
$S(k)$	kth equivalence class; $\{\mathbf{x} \in S \mid \phi(\mathbf{x}) = k\}$, $k \in [0,M]$.
$\mathbf{x} < \mathbf{y}$	$x_i \le y_i \ \forall\ i$ and $x_i < y_i$ for at least one i.
$L(k)$	set of lower boundary points to level k, $k \in (0,M]$.
$U(k)$	set of upper boundary points to level k, $k \in [0,M)$.
P_j	jth minimal path set; $j=1,2,\ldots,s$.
$\boldsymbol{\alpha}$	$(\alpha,\alpha,\ldots,\alpha)$.
C	the set of component indices; $\{1,2,\ldots,n\}$.
$(j_i,\mathbf{x})$	$(x_1,\ldots,x_{i-1},j,x_{i+1},\ldots,x_n)$, $i=1,2,\ldots,n$ and $j \in \Omega_i$.
$\mathcal{L}(\mathbf{x},k)$	$\mathcal{L}(\mathbf{x},k) = \{(i,x_i)$ for all $x_i \ne 0\}$.
$\mathcal{U}(\mathbf{x},k)$	$\mathcal{U}(\mathbf{x},k) = \{(i,x_i)$ for all $x_i \ne M_i\}$.

4.1.2 Introduction

A natural extension of the multistate model allows the system and component states to be described by a continuous range of states. Although other authors had mentioned the possibility of continuous structure functions as early as 1978, Block and Savits [1984] were first to devise a model. Block and Savits defined continuum structure functions on the nonnegative orthant $\mathbb{R}_+^n$ and other subsets of $\mathbb{R}^n$. They derived results similar to those found for discrete structure functions. Subsequent authors (Baxter [1984,1986], Kaleva [1986], Baxter and Kim [1986], and Montero, Tejada, and Yáñez [1990]) have concentrated on continuum structure functions that map from $[0,1]^n$ to $[0,1]$. The continuous model is generalized to allow the continuous states of the system and each component to vary over a different range of values.

For a system with n components, the state of the ith component is given by the continuous variable $x_i \in [0,M_i]$ for $i=1,2,\ldots,n$ where M_i is the best state of component i. Let the state space for component i be designated by Ω_i. The component state vector $\mathbf{x} = (x_1,x_2,\ldots,x_n)$ is the vector of n component states and the component state space $S = \{\mathbf{x} \mid x_i \in \Omega_i, \forall i\}$ is the set of possible component state vectors.

The state of the system is given by the continuous variable $\phi \in [0,M]$. Let the state space for the system be designated by $\mathbf{\Omega}$. The model assumes that the state of the system is completely determined by the states of the n components. The relationship is described by the structure function $\phi(\mathbf{x})$, which is abbreviated by ϕ: $[0,M_i]^n \rightarrow [0,M]$.

For the multistate case, all component state vectors with the same system state were said to belong to the same equivalence class. A parallel definition can be derived for a general continuous structure function (general CSF).

DEFINITION. The <u>k^{th} equivalence class $S(k)$</u> of a general CSF is given by

$$S(k) = \{\mathbf{x} \in S \mid \phi(\mathbf{x}) = k\},\ k \in [0,M].$$

There are an infinite number of component state vectors and most equivalence classes contain an infinite number of vectors. In addition, there are an infinite number of equivalence classes. Still, each component state vector can belong to only one equivalence class.

EXAMPLE 4.1 Determine $S(k)$ for the 2-component general CSF:

$$\phi(\mathbf{x}) = x_1x_2 \quad \text{where } x_1 \in [0,2] \text{ and } x_2 \in [0,3].$$

$S(k) = \{\mathbf{x} \in S \mid x_1x_2 = k\}$ $k \in [0,6]$. As shown in Figure 4.1, $S(0) = \{\mathbf{x} \in S \mid \text{Min}\{x_1,x_2\} = 0\}$, $S(2) = \{\mathbf{x} \in S \mid x_1x_2 = 2\}$, and $S(6) = \{(2,3)\}$.

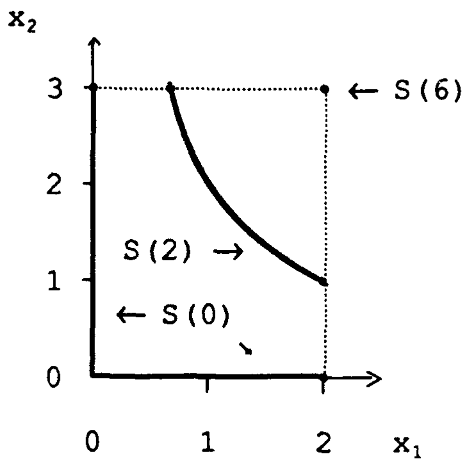


Figure 4.1 S(0), S(2), and S(6) for Example 4.1.

4.1.3 Special Structures

Montero et al. [1990] defined the concepts of minimal paths and minimal cuts at level k for a CSF. The definitions are extended to a general CSF, again using the terms lower boundary points and upper boundary points to level k. Note that the continuous model is complicated by problems with continuity. Specifically, lower boundary points to level k are only defined when a general CSF is right continuous and upper boundary points to level k are only defined when a general CSF is left continuous.

DEFINITION. The general CSF ϕ is <u>right continuous at $\mathbf{y}$</u> if for each $\mathbf{x} \in S$ and for each $\varepsilon > 0$, there is a $\delta > 0$ such that $|\phi(\mathbf{x}) - \phi(\mathbf{y})| < \varepsilon$ whenever $\mathbf{y} < \mathbf{x} < \mathbf{y} + \delta\,\mathbf{1}$.

DEFINITION. The general CSF ϕ is <u>left continuous at $\mathbf{y}$</u> if for each $\mathbf{x} \in S$ and for each $\varepsilon > 0$, there is a $\delta > 0$ such that $|\phi(\mathbf{x}) - \phi(\mathbf{y})| < \varepsilon$ whenever $\mathbf{y} - \delta\,\mathbf{1} < \mathbf{x} < \mathbf{y}$.

As with the general MCS, there are no lower boundary points to level 0 and no upper boundary points to level M.

DEFINITION. If the general CSF $\phi(\mathbf{x})$ is right continuous, then $\mathbf{x}$ is a <u>lower boundary point to level k</u> if $\phi(\mathbf{x}) \geq k$ and $\mathbf{y} < \mathbf{x}$ implies that $\phi(\mathbf{y}) < k$, $k \in (0,M]$.

Let L(k) designate the set of all lower boundary points to level k, $k \in (0,M]$.

DEFINITION. If the general CSF $\phi(\mathbf{x})$ is left continuous, then $\mathbf{x}$ is a upper boundary point to level k if $\phi(\mathbf{x}) \le k$ and $\mathbf{y} > \mathbf{x}$ implies that $\phi(\mathbf{y}) > k$, $k \in [0,M)$.

Let U(k) designate the set of all upper boundary points to level k, $k \in [0,M)$.

Like the general MCS, the general CSF definitions for series, parallel, and k-out-of-n structures are based on the number of lower and upper boundary points to level k.

DEFINITION. ϕ is a series general CSF iff ϕ has one lower boundary point to level j, $j \in (0,M]$ and n upper boundary points to level j, $j \in [0,M)$.

DEFINITION. ϕ is a parallel general CSF iff ϕ has n lower boundary point to level j, $j \in (0,M]$ and one upper boundary points to level j, $j \in [0,M)$.

DEFINITION. ϕ is a k-out-of-n general CSF iff ϕ has $\binom{n}{k}$ lower boundary points to level j, $j \in (0,M]$ and $\binom{n}{n-k+1}$ upper boundary points to level j, $j \in [0,M)$.

Series and parallel systems are special cases of the k-out-of-n structure. A series system is an n-out-of-n structure while a parallel system is a 1-out-of-n structure.

EXAMPLE 4.2 Suppose that $x_1 \in [0,1]$ and $x_2 \in [0,2]$. Determine S(k), L(k), and U(k), $k \in [0,1]$ for the 2-component general CSF:

$$\phi(\mathbf{x}) = \text{Max}\{x_1, \tfrac{1}{2}x_2\}.$$

$S(k) = \{\mathbf{x} \in S \mid \text{Max}\{x_1, \tfrac{1}{2}x_2\} = k\}$ for $k \in [0,1]$. The single upper boundary point to level k is $(k,2k)$, $k \in [0,1)$. The two lower boundary points to level k are $(0,2k)$ and $(k,0)$, $k \in (0,1]$. Therefore, $\phi(\mathbf{x})$ is a parallel general CSF. S(.5), L(.5), and U(.5) are shown in Figure 4.2.

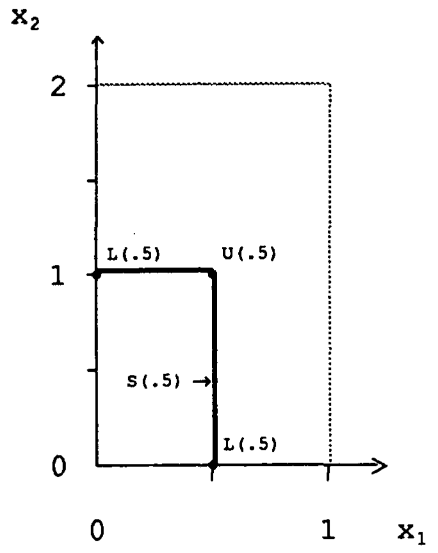


Figure 4.2 S(.5), L(.5), and U(.5) for Example 4.2.

EXAMPLE 4.3 Suppose that $x_1 \in [0,4]$ and $x_2 \in [0,8]$. Determine S(k), L(k), and U(k), $k \in [0,2]$ and the lower and upper boundary points to level k for the 2-component general CSF:

$$\phi(\mathbf{x}) = \text{Min}\{\tfrac{1}{2}x_1, \tfrac{1}{4}x_2\}.$$

$S(k) = \{\mathbf{x} \in S \mid \text{Min}\{\tfrac{1}{2}x_1, \tfrac{1}{4}x_2\} = k\}$ for $k \in [0,2]$. The single lower boundary point to level k is $(2k,4k)$, $k \in (0,2]$. The two upper boundary points to level k are $(4,4k)$ and $(2k,8)$, $k \in [0,2)$.

Therefore, $\phi(\mathbf{x})$ is a series general CSF. S(1), L(1), and U(1) are shown in Figure 4.3.

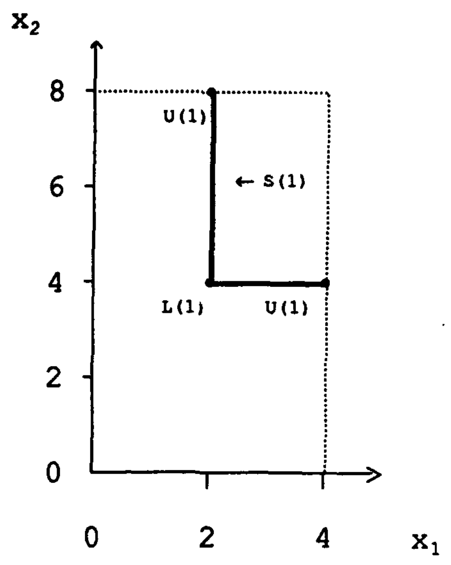


Figure 4.3 S(1), L(1), and U(1) for Example 4.3.

4.1.4 Coherent Structures

The authors who developed the CSF eliminated unrealistic structure functions by defining coherent systems. As with the MCS, there is no single definition for a coherent CSF.

Baxter [1984] defined a CSF to be coherent in much the same was as Barlow and Wu [1978] did for a multistate system.

DEFINITION. A CSF ϕ is <u>coherent</u> iff

i. $\Omega_i = \Omega = [0,1] \ \forall i$,

ii. $\phi(\mathbf{x})$ is increasing in each x_i, and

iii. $\phi(\mathbf{x}) = \underset{j=1,2,\ldots,s}{\text{Max}} \ \underset{i \in P_j}{\text{Min}} \ x_i$ where P_j is the jth minimal path set for the system's binary model.

This definition forces $\phi(\alpha) = \alpha$ for all $\alpha \in [0,1]$. It is desirable that $\phi(\mathbf{x}_0) = 0$ and $\phi(\mathbf{x}_M) = M$, but the model should not restrict the state of the system for specific $\mathbf{x} \in S$ since one component may be more important to the customer.

Paralleling the earlier definitions of multistate coherence by Griffith [1980], Baxter [1986] developed the following categories of coherence for a CSF: strictly coherent, coherent, and weakly coherent. Only the least restrictive definition for a weakly coherent system is repeated since it equates to the new component relevancy condition. Suppose that $C = \{1,2,\ldots,n\}$ is the set of component indices and S is the component state space.

DEFINITION. ϕ is <u>weakly coherent</u> iff

i. $\Omega_i = \Omega = [0,1]\ \forall i$,

ii. $\phi(\mathbf{x})$ is increasing in each x_i and

iii. $\sup_{\mathbf{x} \in S} [\phi(1_i,\mathbf{x}) - \phi(0_i,\mathbf{x})] > 0$ for each $i \in C$

where $(j_i,\mathbf{x})$ denotes $(x_1,\ldots,x_{i-1},j,x_{i+1},\ldots,x_n)$.

As a final note, Baxter [1986] also developed a CSF based on Natvig's definition of type-2 coherence [1982].

As with the general MCS, there are two aspects of the previous definitions for coherence that are too restrictive. First, it is not desirable to restrict the state spaces of the components and system to the same set. Also, the state of the system should not be restricted for specific $\mathbf{x} \in S$.

There are two aspects of the previous definitions for coherence that are desired. First, the system should not improve with the deterioration of a component. Second, the system should only contain relevant components. The next definition is used for a general coherent CSF:

DEFINITION. ϕ is a <u>general coherent CSF</u> iff

i. $\Omega_i = [0,M_i]$, $i=1,2,\ldots,n$,

ii. $\Omega = [0,M]$,

iii. $S(0)$ and $S(M)$ are not empty,

iv. $\phi(\mathbf{x})$ is increasing, and

v. $\forall$ component i, there exists an $\mathbf{x} \in L(k)$ such that $x_i \neq 0$ for some $k = (0,M]$ or there exists an $\mathbf{x} \in U(k)$ such that $x_i \neq M_i$ for some $k = [0,M)$.

4.1.5 Equivalent Coherent Structures

Block and Savits [1984] developed two alternative representations of the CSF using the binary decomposition techniques shown in Section 3.1.5. The representations are expanded to the general CSF. The type of continuity determines which representation is applicable.

Suppose the customer can specify the set of lower boundary points to level k, $k \in (0,M]$ for a right continuous general CSF. Define the following two indicator variables:

$$y_i(j) = \begin{cases} 0 & \text{if } x_i < j \\ 1 & \text{if } x_i \geq j \end{cases}$$

for $i=1,2,\ldots,n$ and $j \in (0,M_i]$.

$$\phi(\mathbf{y},k) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) < k \\ 1 & \text{if } \phi(\mathbf{x}) \geq k \end{cases} \qquad (4.1)$$

for $k \in (0,M]$.

Suppose the set of all lower boundary points to level k is designated by L(k). If $\mathbf{x} \in L(k)$, then let

$$\mathfrak{L}(\mathbf{x},k) = \{(i,x_i) \text{ for all } x_i \neq 0\}.$$

Block and Savits [1984] wrote the structure function as

$$\phi(\mathbf{y},k) = \underset{\mathbf{x}\in L(k)}{\text{Max}} \quad \underset{(i,j)\in \mathfrak{L}(\mathbf{x},k)}{\text{Min}} \; y_i(j).$$

From Equation 4.1, the value of the general CSF is given by

$$\phi(\mathbf{x}) = \int_0^M \phi(\mathbf{y},k)\,dk.$$

A similar decomposition can be used when the customer can specify the set of upper boundary points to level k, k $\in$ [0,M) for a left continuous general CSF. Define the following two indicator variables:

$$y_i(j) = \begin{cases} 0 & \text{if } x_i \leq j \\ 1 & \text{if } x_i > j \end{cases}$$

for i=1,2,...,n and $j \in [0,M_i)$.

$$\phi(\mathbf{y},k) = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) \leq k \\ 1 & \text{if } \phi(\mathbf{x}) > k \end{cases} \qquad (4.2)$$

for $k \in [0,M)$.

Suppose the set of all upper boundary points to level k is designated by U(k). If $\mathbf{x} \in U(k)$, then let

$$\mathfrak{U}(\mathbf{x},k) = \{(i,x_i) \text{ for all } x_i \neq M_i\}.$$

Block and Savits [1984] wrote the structure function as

$$\phi(\mathbf{y},k) = \underset{\mathbf{x}\in U(k)}{\mathrm{Min}} \ \underset{(i,j)\in \mathfrak{U}(\mathbf{x},k)}{\mathrm{Max}} \ y_i(j).$$

From Equation 4.2, the value of the general CSF is given by

$$\phi(\mathbf{x}) = \int_0^M \phi(\mathbf{y},k)\,dk.$$

The procedure is demonstrated for the case where there are a finite number of elements in each L(k) such as the parallel general CSF given in example 4.2.

EXAMPLE 4.4 Suppose a continuous general CSF with two mutually independent components is defined by the customer with the following the *sets of lower boundary points to level k*:

$L(k) = \{(0,2k),(k,0)\},\ k \in (0,1]$.

Write an equivalent expression for $\phi(\mathbf{x})$.

$\mathfrak{U}(\mathbf{x}^1,k) = \{(2,2k)\}$ and $\mathfrak{U}(\mathbf{x}^2,k) = \{(1,k)\}$.

$\phi(\mathbf{y},k) = \mathrm{Max}\{y_2(2k),y_1(k)\}$ and

$$\phi(\mathbf{x}) = \int_0^1 \phi(\mathbf{y},k)\ dk.$$

For the specific component state vector $\mathbf{x} = (.5,1.6)$,

$$y_1(j) = \begin{cases} 0 & \text{if } .5 < j \\ 1 & \text{if } .5 \geq j \end{cases} \text{ for } j \in (0,1]$$

and

$$y_2(j) = \begin{cases} 0 & \text{if } 1.6 < j \\ 1 & \text{if } 1.6 \geq j \end{cases} \quad \text{for } j \in (0,2]$$

so that $y_2(2k) \geq y_1(k)\ \forall\ k$. Thus, $\phi(\mathbf{y},k) = y_2(2k)$

and $\phi(\mathbf{x}) = \int_0^{.8} (1)\, dk + \int_{.8}^{1} (0)\, dk = .8$ which checks

with $\phi(\mathbf{x}) = \text{Max}\{.5,\ \tfrac{1}{2}(1.6)\} = .8$.

The procedure is also demonstrated for the case where there are an infinite number of elements in L(k) such as the general CSF given in example 4.1.

EXAMPLE 4.5 Suppose a continuous general CSF with two mutually independent components is defined by the customer so that $M=6$, $M_1=2$, $M_2=3$, and the lower boundary points to level k are given by:

$L(k) = \{(r, \frac{k}{r}) \text{ for } \frac{k}{3} \leq r \leq 2\},\ 0 < k \leq 6.$

Write an equivalent expression for $\phi(\mathbf{x})$.

The variable r in L(k) can be transformed to range between 0 and 1 so that for $0 < k \leq 6$:

$$L(k) = \{(\frac{s(6-k)+k}{3}, \frac{3k}{s(6-k)+k}) \text{ for } 0 \leq s \leq 1\}$$

$$\mathfrak{L}(\mathbf{x},k) = \{(1, \frac{s(6-k)+k}{3}), (2, \frac{3k}{s(6-k)+k}),\ 0 \leq s \leq 1\}$$

and $\phi(\mathbf{y},k) = \underset{0 \leq s \leq 1}{\text{Max}} \{y_1(\frac{s(6-k)+k}{3}) \cdot y_2(\frac{3k}{s(6-k)+k})\}$.

Therefore, $\phi(\mathbf{x}) = \int_0^6 \phi(\mathbf{y},k)\, dk$.

For the specific component state vector $\mathbf{x}$ =

$(1.2, .8)$, $y_1(\frac{s(6-k)+k}{3}) = 1$ if $1.2 \geq \frac{s(6-k)+k}{3}$

and $y_2(\frac{3k}{s(6-k)+k}) = 1$ if $.8 \geq \frac{3k}{s(6-k)+k}$. Both are

true only when $k \leq .96$ so the structure function

$\phi(\mathbf{x}) = \int_0^{.96} (1)\, dk + \int_{.96}^{6} (0)\, dk = .96$ which checks

with $\phi(\mathbf{x}) = (1.2)(.8)$ given in example 4.1.

As shown in example 4.5, even the simplest continuous structure functions do not lend themselves well to alternate representations. The infinite number of boundary points to level k makes a general equation for $\phi(\mathbf{y},k)$ unlikely and the integral representation of $\phi(\mathbf{x})$ ineffective.

4.1.6 Dual Structure Function

Block and Savits [1984] discussed the concept of the dual structure function for the continuum structure function $\phi(\mathbf{x}) = \max_{j=1,2,\ldots,s} \min_{i \in P_j} x_i$ where P_j is the jth minimal path set for the system's binary model. The notion of the dual is extended for the general CSF.

DEFINITION. Let ϕ be a structure function of a general CSF. The <u>dual structure function</u> ϕ^D is defined by

$$\begin{aligned} \phi^D(\mathbf{x}) &= M - \phi(M_1-x_1, M_2-x_2, \ldots, M_n-x_n) \\ &= M - \phi(\mathbf{x}_M - \mathbf{x}). \end{aligned}$$

The relationship between the primal and dual general CSF can be interpreted in the following manner. Let the primal components be in state x_i and the dual components be in state $x_i^D = M_i - x_i$. Then the primal system is in state k when the dual system is in state M - k.

The following theorems are identical to those given in section 3.1.6 and the proofs require only trivial changes.

THEOREM 4.1 If the primal is a general CSF, then the dual is also a general CSF.

THEOREM 4.2 $\mathbf{x}$ is a lower boundary point to level k for the general CSF ϕ iff $(\mathbf{x}_M - \mathbf{x})$ is an upper boundary point to level (M - k) for the dual general CSF ϕ^D.

THEOREM 4.3 The dual of a k-out-of-n general CSF is an (n-k+1)-out-of-n general CSF.

COROLLARY 4.1 The dual of a series general CSF of n components is a parallel general CSF of n components.

COROLLARY 4.2 The dual of a parallel general CSF of n components is a series general CSF of n components.

THEOREM 4.4 The dual of the dual is the primal.

EXAMPLE 4.6 Write the dual structure function $\phi^D(\mathbf{x})$ for the series structure function given in example 4.3.

$\phi(\mathbf{x}) = \text{Min}\{\frac{1}{2}x_1, \frac{1}{4}x_2\}$ with $x_1 \in [0,4]$, $x_2 \in [0,8]$ and $k \in [0,2]$. Therefore,

$$\phi^D(\mathbf{x}) = 2 - \phi(\mathbf{x}_M - \mathbf{x}) = 2 - \phi(4-x_1, 8-x_2)$$

$$= 2 - \text{Min}\left\{\frac{x_1}{2}, \frac{8-x_2}{4}\right\}$$

which is a parallel structure function with

$L(k) = \{(0,4k),(2k,0)\}$ for $k \in (0,2]$ and

$U(k) = \{(2k,4k)\}$ for $k \in [0,2)$.

4.1.7 Structural Importance

For the general CSF, component i is relevant if $\sup_{\mathbf{x} \in S} [\phi((M_i)_i,\mathbf{x}) - \phi(0_i,\mathbf{x})] > 0$. It is interesting to note that components can be irrelevant for the binary model but relevant for the continuous model.

EXAMPLE 4.7 In example 2.1, it was shown that component 2 is irrelevant for the binary structure

$$\phi(\mathbf{x}) = 1 - (1-x_1)(1-x_1x_2).$$

Show that component 2 is relevant for $\phi(\mathbf{x})$ defined as a general CSF with x_1, x_2, and $\phi \in [0,1]$.

Since $\phi(.5,1) = .75$ and $\phi(.5,0) = .5$, component 2 is relevant.

The system is coherent if, among other conditions, all components are relevant.

In Chapter 3, the structural importance of component i was defined as the proportion of the component state vectors where the relevance condition holds. Extending to the general CSF, structural importance for component i is given by

$$I_\phi(i) = \frac{1}{\prod_{j \neq i} M_j} \int_{\{\mathbf{x} | x_i = M_i\}} N(\mathbf{x})\, d\mathbf{x} \quad \text{where}$$

$$N(\mathbf{x}) = \begin{cases} 1 & \text{if } \sup_{\mathbf{x} \in S}[\phi((M_i)_i,\mathbf{x}) - \phi(0_i,\mathbf{x})] > 0 \\ 0 & \text{otherwise.} \end{cases}$$

4.1.8 Modules and Modular Decomposition

Definitions for modules and modular decomposition were given by Baxter and Kim [1986] for $\phi:[0,1]^n \rightarrow [0,1]$. The definitions can be extended for the general CSF.

DEFINITION - Suppose (C,ϕ) is a general CSF where C is the set of components. Suppose that $A \subset C$. Let A' denote the subset of C complementary to A. The general CSF (A,χ) is a <u>module</u> of (C,ϕ) if

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^A, \mathbf{x}^{A'}) = \psi[\chi(\mathbf{x}^A), \mathbf{x}^{A'}]$$

for all $\mathbf{x} \in S$. ψ is a CSF called the organizing structure.

DEFINITION - A <u>modular decomposition</u> of a general CSF (C,ϕ) is a set $\{(A_1,\chi_1),(A_2,\chi_2),\ldots,(A_k,\chi_k)\}$ of general CSFs along with the organizing structure ψ such that

i. $\{A_1,A_2,\ldots,A_k\}$ is a partition of C and

ii. $\phi(\mathbf{x}) = \psi[\chi_1(\mathbf{x}^{A_1}),\chi_2(\mathbf{x}^{A_2}),\ldots,\chi_k(\mathbf{x}^{A_k})]$ for all $\mathbf{x} \in S$.

As in the binary and multistate case, modular decomposition provides a method of breaking up a complex system into several more manageable problems. The smaller problems are solved and the organizing structure is used to combine the results.

4.2 Stochastic Properties

Up to this point, only the deterministic properties of the general CSF have been discussed. Next, the stochastic properties of the general CSF are explored.

4.2.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

n	number of components.
X_i	random variable for the state of component i.
x_i	fixed state of component i; $x_i \in \Omega_i$.
$\mathbf{X}$	random component state vector; $\mathbf{X} = (X_1, X_2, \ldots, X_n)$.
$\mathbf{x}$	fixed component state vector; $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.
$\phi(\mathbf{X})$	random variable for the state of the system.
ϕ	fixed state of the system; $\phi = \phi(\mathbf{x})$.
$Q(k)$	$\Pr[\phi(\mathbf{X}) \geq k]$, $k \in [0,M]$.
$Q_i(j)$	$\Pr[X_i \geq j]$, $i=1,2,\ldots,n$ and $j \in [0,M_i]$.
$Q(k-)$	$\Pr[\phi(\mathbf{X}) > k]$, $k \in [0,M]$.
$Q_i(j-)$	$\Pr[X_i > j]$, $i=1,2,\ldots,n$ and $j \in [0,M_i]$.

4.2.2 The Performance Function

For a system of n components, let X_i denote the random state of component i and x_i denote a specific state of component i. The random and specific states for all components are summarized in the random component state vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and the fixed component state vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Let $\phi(\mathbf{X})$ be the random system state and $\phi(\mathbf{x})$ or ϕ be a fixed system state.

For the general CSF, the problem is to find the performance distribution of the system, $Q(k)$, where

$$Q(k) = \Pr[\phi(\mathbf{X}) \geq k], \quad k \in [0,M]$$

given every component's performance distribution, $Q_i(j)$, where

$$Q_i(j) = \Pr[X_i \geq j],\ i=1,2,\ldots,n \text{ and } j \in [0,M_i].$$

If the random variables X_i, $i=1,\ldots,n$ are mutually independent, then $Q(k)$ may be expressed as a function of $Q_i(j)$. Because of problems with continuity, the performance distributions must often be written as strict inequalities

$$Q(k-) = \Pr[\phi(\mathbf{X}) > k],\ k \in [0,M]$$

and

$$Q_i(j-) = \Pr[X_i > j],\ i=1,2,\ldots,n \text{ and } j \in [0,M_i].$$

4.2.3 Exact Performance Distribution

The infinite number of boundary points to level k impedes the techniques used in Chapter 3 to find the exact performance distribution of the system (i.e. enumeration, inclusion-exclusion, and pivotal decomposition). For some cases Q(k) can be found directly by integration. The following example comes from Montero et al. [1990]:

EXAMPLE 4.8 Suppose a general CSF is defined by

$$\phi(\mathbf{x}) = x_1x_2$$

with mutually independent variables X_1 and X_2 ~ Uniform [0,1]. Find Q(k).

$$Q(k) = \Pr[\phi(\mathbf{x}) \geq k] = \int_k^1 \int_{\frac{k}{x_1}}^1 1\, dx_2\, dx_1 = \int_k^1 (1 - \frac{k}{x_1})\, dx_1$$

$$= 1 - k \ln |1| - k + k \ln |k|$$

$$= \begin{cases} 1 - k + k \ln k & k \in (0,1] \\ 1 & k = 0\ . \end{cases}$$

4.2.4 System Performance Bounds

The system performance bounds depend on the substitute characteristic for reliability used by the customer. Most authors have developed bounds for $\Pr[\phi(\mathbf{X}) \geq k]$.

Performance bounds can be constructed for independent and associated random variables. Of course, the bounds are more explicit if the random variables are independent. The next five sections generalize some bounds on $\Pr[\phi(\mathbf{X}) \geq k]$ derived by other authors.

4.2.4.1 Trivial Bounds

The trivial bounds for a right continuous general CSF are based on a single lower boundary point to level k.

THEOREM 4.5 Suppose $\phi(\mathbf{x})$ is a right continuous CSF. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in L(k)$, $k \in (0,M]$. Then

$$\prod_{i=1}^{n} Q_i(y_i) \leq Q(k) \leq 1 - \prod_{i=1}^{n} (1 - Q_i(y_i)).$$

Proof: The proof is similar to Theorem 3.14.

The trivial bounds for a left continuous general CSF are based on a single upper boundary point to level k.

THEOREM 4.6 Suppose $\phi(\mathbf{x})$ is a left continuous CSF. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in U(k)$, $k \in [0,M)$. Then

$$\prod_{i=1}^{n} Q_i(y_i) \leq Q(k) \leq 1 - \prod_{i=1}^{n} (1 - Q_i(y_i)).$$

Proof: The proof is similar to Theorem 3.14.

4.2.4.2 Path/Cut Bounds

Block and Savits [1984] developed the Path/Cut Bounds for the structure function $\phi: [0,1]^n \rightarrow [0,1]$. The bounds can be extended to the general CSF.

THEOREM 4.7 Let ϕ be a continuous general CSF with associated components. Then for all but a countable number of k,

$$\prod_{x \in U(k)} \Pr\left[\bigcup_{(i,j)\in \mathcal{U}(x,k)} \{X_i > j\}\right] \le Q(k) = Q(k-) \le \prod_{x \in L(k)} \Pr\left[\bigcap_{(i,j)\in \mathcal{L}(x,k)} \{X_i \ge j\}\right]$$

for k = (0,M].

The lower bound comes from the upper boundary points while the upper bound comes from the lower boundary points.

When the components are independent, the bounds can be explicitly derived from the performance distributions of the components.

THEOREM 4.8 Let ϕ be a general CSF with independent components. Then for all but a countable number of k,

$$\prod_{x \in U(k)} \prod_{(i,j)\in \mathcal{U}(x,k)} Q_i(j-) \le Q(k) = Q(k-) \le \prod_{x \in L(k)} \prod_{(i,j)\in \mathcal{L}(x,k)} Q_i(j)$$

for k = (0,M].

4.2.4.3 Min/Max Bounds

The Min/Max Bounds for $\phi:[0,1]^n \rightarrow [0,1]$ were developed by Block and Savits [1984] so that the lower bound comes from the lower boundary points while the upper bound comes from the upper boundary points. Generalizing,

THEOREM 4.9 Let ϕ be a continuous general CSF. Then the following bounds always hold for $k \in [0,M]$:

$$\sup_{\mathbf{x}\in L(k)} \Pr\left[\bigcap_{(i,j)\in \bar{L}(\mathbf{x},k)} \{X_i \geq j\}\right] \leq Q(k) \quad \text{and}$$

$$Q(k-) \leq \inf_{\mathbf{x}\in U(k)} \Pr\left[\bigcup_{(i,j)\in \bar{U}(\mathbf{x},k)} \{X_i > j\}\right].$$

If the components are associated, then

$$\sup_{\mathbf{x}\in L(k)} \left\{\prod_{(i,j)\in \bar{L}(\mathbf{x},k)} Q_i(j)\right\} \leq Q(k) \quad \text{and}$$

$$Q(k-) \leq \inf_{\mathbf{x}\in U(k)} \left\{\coprod_{(i,j)\in \bar{U}(\mathbf{x},k)} Q_i(j-)\right\}.$$

4.2.4.4 Combining Bounds

A combination of the bounds in Theorems 4.8 and 4.9 is appropriate for mutually independent components. Let ϕ be a general CSF with mutually independent components. Combined Bounds use the maximum lower bound and minimum upper bound found with Theorems 4.8 and 4.9.

4.2.4.5 Improved Path/Cut Bounds

Baxter and Kim [1986] showed that the Path/Cut Bounds can be improved with modular decomposition for the continuous model. To do this, Path/Cut Bounds are determined for each module. These bounds are then used to determine Path/Cut Bounds for the system. Baxter and Kim [1986] proved that the new bounds were always tighter than the Path/Cut Bounds found directly from the entire system.

4.2.4.6 General Bounds

Montero et al. [1990] developed general bounds without the need for $\phi(\mathbf{x})$ to be continuous. These bounds are more effective for the case when there are an infinite number of boundary points to level k.

THEOREM 4.9 Let $\phi(\mathbf{x})$ be a general CSF. Then

$$1 - \Pr[\mathbf{y} | \mathbf{y} \leq \mathbf{r}] \leq Q(k) \leq \Pr[\mathbf{y} | \mathbf{y} \geq \mathbf{q}]$$

where the vectors $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ are given by

$$r_i = \sup_{\mathbf{x} \in U(k)} x_i \quad \text{and} \quad q_i = \inf_{\mathbf{x} \in L(k)} x_i \quad \forall i.$$

4.3 Dynamic Properties

In the last two sections, the structural and stochastic properties of the general CSF were examined at a fixed moment in time. The next step is to consider dynamic models, where the component and system states vary with time.

4.3.1 Notation

The following notation is listed for the reader's convenience in the order of presentation:

t	fixed time; $t \geq 0$.
$X_i(t)$	state of component i at time t, $i=1,2,\ldots,n$.
$\mathbf{X}(t)$	vector of random component states at time t; $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$.
$\phi(\mathbf{X}(t))$	random system state at time t.
T^k	time for system state to reach or go below state k; $T^k = \inf\{t \mid \phi(X(t)) \leq k\}$.
T_i^j	time for state of component i to reach or go below state j; $T_i^j = \inf\{t \mid X_i(t) \leq j\}$.

$Q_k(t)$ performance distribution of the system; $Q_k(t) = \Pr[\phi(\mathbf{X}(t)) \geq k]$, $k=1,2,\ldots,M$.

$Q_{ij}(t)$ performance distribution for component i; $Q_{ij}(t) = Q_{ij}(t) = \Pr[X_i(t) \geq j]$, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$.

4.3.2 Distribution Representations

Let $\{X_i(t), t \geq 0\}$ for $i=1,2,\ldots,n$ be the decreasing and right continuous stochastic process representing the state of component i at time t, where t ranges over the nonnegative real numbers. The components and associated stochastic processes are assumed to be mutually independent. Let the vector of random component states at time t be designated by $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$. Let $\{\phi(\mathbf{X}(t)), t \geq 0\}$ denote the decreasing and right continuous stochastic process that represents the system state at time t.

4.3.3 Distribution Classes and Closure

Baxter [1984] discussed a dynamic stochastic model for the first-passage-time distribution which is defined as the first time a stochastic process enters the set of "bad" states from the set of "good" states. The first-passage-time distributions for the system and the components are given by

$$T^k = \inf \{t \mid \phi(X(t)) \leq k\} \text{ and}$$

$$T_i^j = \inf \{t \mid X_i(t) \leq j\}, \text{ respectively.}$$

Baxter [1984] used theorems given by Ross [1979] to prove IFRA and NBU closure with respect to the formation of coherent systems. That is, if $\{X_i(t), t \geq 0\}$, $i=1,2,\ldots,n$

are increasing independent IFRA/NBU processes, then $\{\phi(\mathbf{X}(t)), t \geq 0\}$ is also an IFRA/NBU process whenever ϕ is decreasing.

4.3.4 Exact System Performance

For the CSF, the problem is to find the performance distribution of the system given by

$$Q_k(t) = \Pr[\phi(\mathbf{X}(t)) \geq k] = \Pr[T^k \geq k] \text{ for } k= [0,M]$$

from each component's performance distribution given by

$$Q_{ij}(t) = \Pr[X_i(t) \geq j] = \Pr[T_i^j > t],\ i=1,2,\ldots,n \text{ and } j=[0,M_i].$$

Although this formation works well for the multistate case, an infinite number of distributions must be specified for the continuous case. Attempts to bound system performance result in similar difficulties. Thus, the integration technique demonstrated in Example 4.8 provides a more desirable method for calculating the exact system performance.

4.4 Summary

This chapter reviewed the structural, stochastic, and dynamic properties for the continuous model and extended the same properties to the general CSF. New definitions were given for k-out-of-n structures. Component relevance was defined in terms of lower and upper boundary points. An alternate representation for $\phi(\mathbf{x})$ was generalized along with many other structural and stochastic properties. Although the theory was sound, the continuous model resulted in an infinite number of boundary points. This severely limited the practical applications of the continuous model.

5. CUSTOMER-DRIVEN MULTISTATE RELIABILITY MODEL

Binary reliability models were developed for their mathematical simplicity. Unfortunately, they are a gross oversimplification of reality for most systems. Continuous reliability models represent reality better, but the amount of computations renders the model ineffective except for the most simple structures. As a compromise, most systems should be analyzed with discrete multistate models.

To support this choice, an example is presented where a customer could make a better decision by using the discrete multistate model. Suppose the customer must choose between two systems with the probability distributions given in Table 5.1. For System A, $E[\phi(\mathbf{X})] = 2.06$, $Var[\phi(\mathbf{X})] = 0.7164$, and the system's performance distribution is $\mathbf{q} = (.91,.9,.2,.05)$. For System B, $E[\phi(\mathbf{X})] = 3.24$, $Var[\phi(\mathbf{X})] = 1.5424$, and the performance distribution is $\mathbf{q} = (.99,.8,.75,.7)$. The choice between the two systems is not clear and will depend on how the customer weighs the different performance measures.

Next, suppose that the same two system state probability distributions are condensed into a binary model where system states 0 through 1 are considered "bad" and states 2 through 4 are considered "good" (Table 5.2). For System A, $E[\phi(\mathbf{X})] = Pr[\phi(\mathbf{X}) \geq 1] = .9$ and $Var[\phi(\mathbf{X})] = .09$. For System B, $E[\phi(\mathbf{X})] = .8$ and $Var[\phi(\mathbf{X})] = .16$. The binary model deludes the customer by removing the customer's ambivalence for the two systems.

Table 5.1 System State Probability Distributions for Two Systems.

System State (k)	$\Pr[\phi(\mathbf{X}) = k]$	
	System A	System B
0	.09	.01
1	.01	.19
2	.7	.05
3	.15	.05
4	.05	.7

Table 5.2 Condensed System State Probability Distributions for Two Systems.

System State (k)	$\Pr[\phi(\mathbf{X}) = k]$	
	System A	System B
0	.1	.2
1	.9	.8

Of course, the same argument can be used to promote the continuous model over the multistate model. For example, suppose a general CSF is defined by $\phi(\mathbf{x}) = x_1 x_2$ with mutually independent components whose states are defined by the random variables X_1 and X_2 ~ Uniform[0,3]. Then

$$Q(k) = \Pr[\phi(\mathbf{x}) \geq k] = \int_{\frac{k}{3}}^{3} \int_{\frac{k}{x_1}}^{3} \frac{1}{9}\, dx_2\, dx_1 = \frac{1}{9} \int_{\frac{k}{3}}^{3} (3 - \frac{k}{x_1})\, dx_1$$

$$= \begin{cases} 1 - \frac{k}{9} \ln 3 - \frac{k}{9} + \frac{k}{9} \ln \frac{k}{3} & k \in (0,9] \\ 1 & k = 0. \end{cases}$$

$$E[\phi(\mathbf{X})] = \int_{0}^{9} Q(k)\, dk = 2.25 \text{ and } \mathrm{Var}[\phi(\mathbf{X})] = 3.9375.$$

A logical multistate model to condense the continuous model is shown in Table 5.3 with $P_{ij} = \Pr[X_i = j] = .25$ for $i=1,2$ and $j=0,1,2,3$. When summarized in this way, the multistate model results in $E[\phi(\mathbf{X})] = 2.25$, $\mathrm{Var}[\phi(\mathbf{X})] = 7.1875$, and $\mathbf{q} = (.5625, .5, .375, .25, .1875, .1875, .0625, .0625, .0625)$. Thus, some of the characteristics of the continuous system are lost when the multistate model is used to represent the continuous system.

Table 5.3 $\phi(\mathbf{x})$ for the Multistate Model.

		x_2			
		0	1	2	3
x_1	0	0	0	0	0
	1	0	1	2	3
	2	0	2	4	6
	3	0	3	6	9

In theory, the continuous model is the best choice for representing a continuous system; however, the continuous model is not a viable option. When the structure function is known explicitly (which is usually not the case), the integration becomes unmanageable for all but the most trivial structures. When the structure function is known implicitly, an infinite number of boundary points are needed to describe the system, to find the exact performance distribution, or to derive system performance bounds. For these reasons, the remainder of this chapter is dedicated to the development and evaluation of the multistate model.

A useful model serves some practical end. The purpose of the customer-driven multistate reliability model is to evaluate substitute characteristics for reliability that allow the customer to distinguish between systems. Since the customer uses the output of the model to make a decision, the model must be generated from the viewpoint of the customer. Otherwise, the model will not supply the information that the customer needs to make a choice.

Generating the model from the viewpoint of the customer means involving the customer at every step in the development and evaluation of the model. The following sections of this chapter provide a detailed discussion of how the customer participates in:

1. Defining the number of system and component states,
2. Estimating the component state probabilities,
3. Defining the system,
4. Estimating the system state probabilities, and
5. Determining the substitute characteristics for reliability.

5.1 Defining the Number of System and Component States

Defining the number of system and component states is referred to as state classification. The customer must determine which performance measure will be used for state classification. Once this is done, the procedure for state classification depends on whether the selected performance measure is discrete or continuous.

5.1.1 Discrete State Classification

State classification for a discrete performance measure is straightforward. The discrete realizations of the chosen performance measure are enumerated for the customer. Next, the customer assigns a state to every realization according to the level of detail needed for him to make a decision. The number of distinct states specified by the customer provides the state classification used in the multistate model. The following example demonstrates this technique for two different customers.

EXAMPLE 5.1 A fighter squadron consists of twenty F-16 aircraft. Each aircraft can perform in three roles: air-to-air, air-to-ground, and intercept. Give two unique state classifications for a single aircraft using the same performance measure.

Suppose the commander of the squadron chooses the number of roles each aircraft currently supports as his performance measure. The realizations of the performance measure are 0, 1, 2, and 3 roles. One commander could ask for the maximum amount of detail, requiring four unique states for the number of roles supported. A second commander might only be interested in determining whether or not an aircraft is capable of all three roles. In this case, only two states are required to make the distinction.

5.1.2 Continuous State Classification

State classification of a continuous performance measure is slightly more complex. The range of possible values for the chosen performance measure is shown to the customer. The customer divides the continuous range into several distinct regions as illustrated in Figure 5.1.

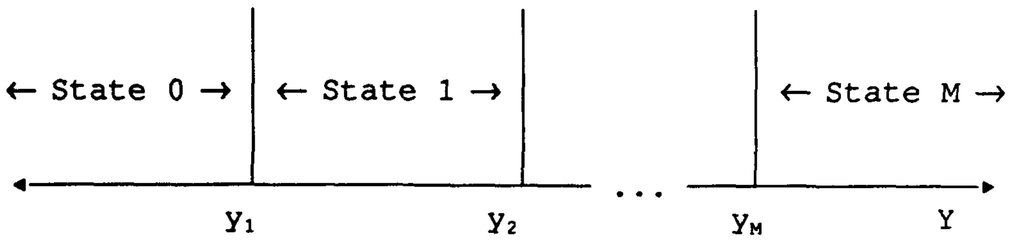


Figure 5.1 State Classification.

Let X represent the discrete state and Y represent the continuous performance measure. State 0 is assigned to X at the lowest value of Y. Y is increased until the customer reaches a threshold where a state change is desired (y_1). State 1 is assigned to X above y_1. Y is increased further until another threshold is found (y_2), no new discrete states are desired, or the continuous performance measure reaches its target value. The procedure is summarized with the following transformation:

$$X = \begin{cases} 0 & \text{if } -\infty < Y \le y_1 \\ 1 & \text{if } \; y_1 < Y \le y_2 \\ \vdots & \qquad \vdots \\ M & \text{if } \; y_M < Y < \infty. \end{cases}$$

The procedure can be discussed more technically if the customer determines the thresholds between discrete states $(y_1, y_2, \ldots, y_M)$ through an economic analysis. This can be done by comparing the cost of various countermeasures to the cost associated with any deviation from the target value of the continuous performance measure.

As shown in Figure 5.2, suppose that the continuous performance measure, Y, has a target value designated by y^* and any value less than y^* results in some loss to the customer given by

$$L(y) = k\,(y^* - y)^2.$$

Suppose the customer has two countermeasures, costing L_1 and L_2, that return the performance measure to y^*. The customer will not execute the first countermeasure until Y results in a loss of at least L_1, that is until $y_1 \le y^* - \sqrt{\frac{L_1}{k}}$. The customer will not take advantage of the second countermeasure until $y_2 \le y^* - \sqrt{\frac{L_2}{k}}$. Assuming that $L_2 < L_1$, then $y_1 < y_2$.

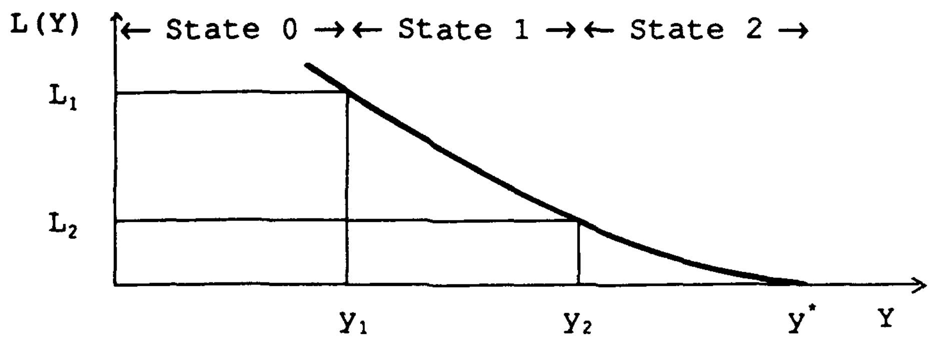


Figure 5.2 State Classification With Loss Function.

Two countermeasures divide the continuous performance measure into three distinct states and provide an economic means to model how the customer determines the thresholds between the discrete states (y_1, y_2). More countermeasures result in more discrete states. The following example demonstrates this technique for a car battery that can be recharged or replaced.

EXAMPLE 5.2 Suppose the performance measure for a car battery is the voltage and the target value is 12 volts. Let $L(y) = 10\,(12 - y)^2$. The two countermeasures are to recharge the battery for \$10 or to replace the battery for \$40. Divide the continuous performance measure into three states.

$$L_1 = 40 \text{ and } y_1 \le 12 - \sqrt{\frac{40}{10}} = 10.$$

$$L_2 = 10 \text{ and } y_2 \le 12 - \sqrt{\frac{10}{10}} = 11.$$

This gives the following three discrete states:

$$X = \begin{cases} 0 & \text{if } \quad 0 \le Y \le 10 \\ 1 & \text{if } \; 10 < Y \le 11 \\ 2 & \text{if } \; 11 < Y \le 12. \end{cases}$$

At times, the customer may use more than one performance measure to determine the discrete state. In this case, an effort must be made to convert the performance measures to some common measure. Again, this is often accomplished through economic analysis.

5.2 Estimating Component State Probabilities

Component state probabilities can be estimated by selecting a distribution for the time spent in each state and estimating the parameter(s) for the distribution. The desired system lifetime, t^*, must be given by the customer.

Let T_{ij} be a random variable representing the time component i stays in state j, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$. Let t^* be the desired system lifetime provided by the customer. Let $X_i(t)$ be the random state of component i at time t. Define the probability that component i is in state j or higher at time t as

$$Q_{ij}(t) = \Pr[X_i(t) \geq j] = \Pr[T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ij} > t].$$

Note that $Q_{i0}(t) = 1.0$ and $Q_{i,M_i+1}(t) = 0.0$.

Suppose that the distribution of T_{ij} is exponential with parameter λ_i and that $T_{i1}, T_{i2}, \ldots, T_{i,M_i}$ are mutually independent $\forall$ i. Then the distribution of $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ij}$ is the convolution of (M_i-j+1) exponential distributions each with parameter λ_i which is an Erlang distribution with shape parameter (M_i-j+1) and scale parameter λ_i. For Erlang (η,λ), $R(t) = \Pr[T > t] = 1 - F(t) = \sum_{k=0}^{\eta-1} \frac{(\lambda t)^k \exp(-\lambda t)}{k!}$. Therefore, $Q_{ij}(t^*) = \sum_{k=0}^{M_i-j} \frac{(\lambda_i t^*)^k \exp(-\lambda_i t^*)}{k!}$. The probability of component i being in state j at time t^* can be found from

$$P_{ij}(t^*) = \Pr[X_i(t^*) = j] = Q_{ij}(t^*) - Q_{i,j+1}(t^*)$$

for i=1,2,...,n and j=0,1,...,M_i. Note that the parameter λ_i must be estimated for every component.

The same procedure can be used whenever the distribution of the convolution is known. For example, if $T_{ij} \sim$ Erlang(η_{ij},λ_i), then $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ik} \sim$ Erlang$(\sum_{j=k}^{M_i} \eta_{ij},\lambda_i)$ and if $T_{ij} \sim$ Normal(μ_{ij},σ_{ij}^2), then $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ik} \sim$ Normal$(\sum_{j=k}^{M_i} \mu_{ij},\sum_{j=k}^{M_i} \sigma_{ij}^2)$. Additional steps must be taken for the normal case since the left-hand tail probabilities allow the random variable for time to be negative. When the specific distribution of the convolution is unknown, the form and parameter(s) can be estimated by simulation.

Returning to the exponential case, the parameter λ_i can be estimated by adapting a technique discussed by Kapur and Lamberson [1977] for counting failures over a time interval. Their technique was based on the situation with n test stands where items are replaced as they fail and the test is stopped at a predetermined time. Again, assume the distribution of T_{ij} is exponential with parameter λ_i for every j=1,2,...,M_i. When a component changes state, it can be replaced with a new component at its best state or the test can be continued with the degraded component.

Kapur and Lamberson [1977] derived the maximum likelihood estimator for λ. Their methodology is modified to derive the maximum likelihood estimator for λ_i. Suppose that n identical multistate components in state M_i are placed on test. The test is stopped at a prespecified time t^*. When a component degrades from state M_i to state M_i-1, it could be replaced with another component at its best state. Since it was assumed that T_{ij} are independent and identically distributed for every $j=1,2,...,M_i$, the degraded component is left in place to await a state change from M_i-1 to M_i-2. The multistate component is replaced by a new component only when the worst state is reached. Suppose that there are n test stands designated by $i=1,2,...,n$. Let X_i be the number of state changes on the ith test stand and Y be the total number of state changes observed during the test time t^*. Then

$$Y = \sum_{i=1}^{n} X_i .$$

If X_i is distributed as a Poisson with parameter $\lambda_i t^*$, then Y is also Poisson with parameter $n\lambda_i t^*$. Assuming a constant number of state changes $Y = r$, the likelihood function for the distribution of Y can be written as

$$L(\lambda_i) = \frac{(n\lambda_i t^*)^r \exp(-n\lambda_i t^*)}{r!}$$

so that

$$\ln L = r \ln (n\lambda_i t^*) - n\lambda_i t^* - \ln (r!) .$$

To find the value of λ_i where ln L is a maximum,

$$\frac{\partial \ln L}{\partial \lambda_i} = \frac{r}{\lambda_i} - nt^{*} = 0$$

so that $\hat{\lambda}_i = \frac{r}{nt^{*}}$. As expected, the estimate of λ_i is the number of state changes divided by the product of the number of multistate components placed on test and the prespecified test time.

A similar derivation can be done for the case where n multistate components are placed on test that is stopped after a prespecified number of state changes. In general, the estimate of the parameter λ_i is given by the number of state changes divided by the total time on test.

5.3 Defining the System

The equivalence class S_k was defined as the set of $\mathbf{x} \in S$ such that $\phi(\mathbf{x}) = k$, $k=0,1,\ldots,M$. The boundary between S_k and S_{k+1} was defined by either the lower boundary points to level $k + 1$ or the upper boundary points to level k. Recall that the boundary points completely specify the system and that they are defined by the customer.

For simple systems, the customer might be able to provide all the required boundary points through a few uncomplicated statements. For example, suppose the customer wishes to model the four tires of an automobile and he provides the following statements:

1. The system has four tires (components) with either high tread wear (component state 0), moderate tread

wear (component state 1), or low tread wear (component state 2).

2. The system state is worst (system state 0) if any of the tires has high tread wear (component state 0).

3. Otherwise, the system state is the number of tires (system states 0, 1, 2, 3 or 4) with low tread wear (component state 2).

From this limited amount of information, the lower and upper boundary points can be generated as shown in Table 5.4:

Table 5.4 Lower and Upper Boundary Points.

Equivalence Class	Lower Boundary Points	Upper Boundary Points
S_0	-	1111 0222 2022 2202 2220
S_1	1112 1121 1211 2111	0222 2022 2202 2220 1112 1121 1211 2111
S_2	1122 1212 1221 2112 2121 2211	0222 2022 2202 2220 1122 1212 1221 2112 2121 2211
S_3	2221 2212 2122 1222	2221 2212 2122 1222
S_4	2222	-

The lower (upper) boundary points are found by determining when a decrease (increase) in the state of any one of the components forces a decrease (increase) in the state of the system. In practice, finding all boundary points can be a very time consuming process.

The following algorithm was developed to help find the lower boundary points. The idea is to find a single lower boundary point to level M and to use the fact that all other lower boundary points to level M must be efficient (not less than or greater than) with respect to that point. Of course, generating all efficient points (Algorithm Steps 3 and 6) is difficult for any large problem. However, the customer's interpretation of the system may allow the efficient points to be found quickly. The method continues by finding a single lower boundary point to level M-1 and stops after the lower boundary points to level 1 have been found.

Algorithm for Finding Lower Boundary Points

1. Begin with the perfect component state vector, $\mathbf{x}_M = (M_1, M_2, \ldots, M_n)$. By Theorem 3.1, $\mathbf{x}_M \in S_M$. So there must be at least one lower boundary point to level M.

2. Find a lower boundary point to level M:

    a. Check if $\mathbf{x}_M$ is a lower boundary point.

    b. If not, decrease the states of the components in $\mathbf{x}_M$ until a lower boundary point is found.

    c. Label the lower boundary point to level M, L_{M1}.

3. All other lower boundary points to level M must be efficient with respect to L_{M1}. Decrease the states of some of the components in L_{M1} and determine the <u>minimal</u> increases in the states of the other components to return the system to state M. Label these lower boundary points consecutively, $L_{M2}, L_{M3}, \ldots, L_{M,s_M}$ where s_M is the number of lower boundary points to level M.

4. Set k = M - 1. The existence of at least one lower boundary point to level k is guaranteed by the definition of a general MCS.

5. Find a lower boundary point to level k:

   a. Check if a lower boundary point from a higher level is also a lower boundary point to level k.

   b. If not, decrease the states of the components in one of the lower boundary points to level k+1 until a lower boundary point to level k is found.

   c. Label the lower boundary point to level k, L_{k1}.

6. All other lower boundary points to level k must be efficient with respect to L_{k1}. Decrease the states of some of the components in L_{k1} and determine the <u>minimal</u> increases in the states of the other components to return the system to state k. Label these lower boundary points consecutively, $L_{k2}, L_{k3}, \ldots, L_{k,s_k}$ where s_k is the number of lower boundary points to level k.

7. Stop if k = 1. Otherwise, set k = k-1 and return to step 5.

A similar algorithm was developed to help find the upper boundary points. Generating all efficient points (Algorithm Steps 3 and 6) requires most of the calculations.

Algorithm for Finding Upper Boundary Points

1. Begin with the worst component state vector, $\mathbf{x}_0 = (0,0,\ldots,0)$. By Theorem 3.1, $\mathbf{x}_0 \in S_0$. So there must be at least one upper boundary point to level 0.

2. Find an upper boundary point to level 0:

   a. Check if $\mathbf{x}_0$ is an upper boundary point.

   b. If not, increase the states of the components in $\mathbf{x}_0$ until an upper boundary point is found.

   c. Label the upper boundary point to level 0, U_{01}.

3. All other upper boundary points to level 0 must be efficient with respect to U_{01}. Increase the states of some of the components in U_{01} and determine the <u>minimal</u> decreases in the states of the other components to return the system to state 0. Label these upper boundary points consecutively, U_{02}, $L_{03}, \ldots, L_{0,s_0}$ where s_0 is the number of upper boundary points to level 0.

4. Set k = 1. The existence of at least one upper boundary point to level k is guaranteed by the definition of a general MCS.

5. Find an upper boundary point to level k:

   a. Check if an upper boundary point from a lower level is also an upper boundary point to level k.

   b. If not, increase the states of the components in one of the upper boundary points to level k-1 until an upper boundary point to level k is found.

   c. Label the upper boundary point to level k, U_{k1}.

6. All other upper boundary points to level k must be efficient with respect to U_{k1}. Increase the states of some of the components in U_{k1} and determine the <u>minimal</u> decreases in the states of the other components to

return the system to state k. Label these upper boundary points consecutively, U_{k2}, U_{k3}, ..., U_{k,t_k} where t_k is the number of upper boundary points to level k.

7. Stop if k = M. Otherwise, set k = k+1 and return to step 5.

Once generated, the lower or upper boundary points can be used to form a structure function that is equivalent to the structure function implicitly used by the customer to define the system. Block and Savits [1982] generated the structure function by decomposing the multistate system into several binary structures, one for each level of the system. The binary structures for level k are based on the lower or upper boundary points to level k. Their technique was expanded for the general MCS in section 3.1.5 and was implemented in the computer program found in Appendix A.

The customer should not have to provide both the lower and upper boundary points. With this in mind, the FORTRAN computer program in Appendix C was written to convert the lower boundary point to level k to upper boundary points to level k-1, k=1,2,...,M and vice versa. If both sets of boundary points are provided, the same program can be used to make sure that no boundary points were missed. The amount of calculations required for the conversion limits the usefulness of the program for large problems.

5.4 Estimating System State Probabilities

Enumeration, inclusion-exclusion, pivoting, and modular decomposition are four techniques commonly used to estimate the system state probabilities. These techniques are studied in section 3.2.4. The computer program given in Appendix A implements the first three techniques directly and the fourth technique indirectly. It has been tested for moderately large problems of about 10 components, each with 4 states.

For larger problems, two options are available. If the customer can decompose the problem into smaller problems with an organizing structure, then one of the three techniques can be used on each subproblem and on the organizing structure. However, for multistate systems, modular decomposition requires a system with some physical interpretation. This is because functional block diagrams and fault trees cannot be used as an aid for decomposition.

The second option for larger problems is to use the performance bounds given in section 3.2.5. The bounds were implemented by the FORTRAN program listed in Appendix B.

5.5 Determining Substitute Characteristics for Reliability

For binary models, reliability was defined as the probability that the system functions. For multistate models, there are different degrees of functioning so a new measure of system performance is required. El-Neweihi et al. [1978] suggested $E[\phi(X)]$ or the expected system state. Butler [1979] promoted $Pr[\phi(X) \geq k]$, especially when the customer was willing to divide system states into two categories ($\geq k$ or $< k$). Griffith [1980] used $E[u(\phi(X))]$ or the expected utility of the system state.

Each of these definitions provides a measure of the performance for multistate systems. However, it is the customer that evaluates the system performance, so it must be the customer that indicates the most appropriate definition. If the customer wants to measure the center and spread of the distribution, then $E[\phi(X)]$ and $Var[\phi(X)]$ seem appropriate. If the customer can separate the system's probability distribution into "good" and "bad" states, then $Pr[\phi(X) \geq k]$ works well. If the customer wants to evaluate efficient performance distributions, then $E[u(\phi(X))]$ allows the customer to weigh the different possibilities.

The second objective of this research is to develop a new substitute characteristic for multistate reliability based on the expected loss to the customer. The new measure should be sensitive to the pattern of degradation about a desired system lifetime t^*. In other words, the measure should be a function of the number state reductions as well as the time of each state change relative to t^*.

To lend some credibility to this approach, the binary model is discussed from this perspective. Suppose that T is a random variable for the time to failure of the system. Let the binary loss function be given by

$$L(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq t^* \\ 0 & \text{if } t > t^* . \end{cases}$$

Then the expected loss is

$$\mathcal{L} = E[L(T)] = \int_{\text{all } t} L(t)f(t)dt = \int_0^{t^*}(1)f(t)dt = 1 - R(t^*).$$

Therefore, for the binary model and the given loss function, minimizing the expected loss is equivalent to maximizing the reliability at the desired system lifetime.

For the multistate model, suppose that T_k are mutually independent random variables for the time the system spends in state k, k=1,2,...,M. Let the loss function with a fixed rate of increase be given by the following:

$$L(t_1,t_2,\ldots t_M) = \begin{cases} 0 & \text{if } t_M > t^* \\ 1 & \text{if } t_M+t_{M-1} > t^*, t_M \le t^* \\ 2 & \text{if } t_M+t_{M-1}+t_{M-2} > t^*, t_M+t_{M-1} \le t^* \\ \vdots & \\ M & \text{if } t_M+t_{M-1}+\ldots+t_1 \le t^* . \end{cases}$$

Using Theorem 3.10 twice, the expected loss is

$\mathcal{L} = E[L(T_1,T_2,\ldots,T_M)]$

$= \Pr[L(T_1,T_2,\ldots,T_M) \ge 1] + \Pr[L(T_1,T_2,\ldots,T_M) \ge 2] + \ldots$

$\quad + \Pr[L(T_1,T_2,\ldots,T_M) \ge M]$

$= \{1 - \Pr[L(T_1,T_2,\ldots,T_M) \le 0]\} + \{1 - \Pr[L(T_1,T_2,\ldots,T_M) \le 1]\}$

$\quad + \ldots + \{1 - \Pr[L(T_1,T_2,\ldots,T_M) \le M-1]\}$

$= \{1 - \Pr[\phi(\mathbf{X}(t^*)) \ge M]\} + \{1 - \Pr[\phi(\mathbf{X}(t^*)) \ge M-1]\} + \ldots$

$\quad + \{1 - \Pr[\phi(\mathbf{X}(t^*)) \ge 1]\}$

$= M - Q_M(t^*) - Q_{M-1}(t^*) - \ldots - Q_1(t^*)$

$= M - E[\phi(\mathbf{X}(t^*))].$

Note that for M = 1, $\mathfrak{L} = 1 - Q_1(t^*) = 1 - R(t^*)$ as was shown for the binary model.

Next, consider the following loss function with a faster rate of increase:

$$L(t_1, t_2, \ldots t_M) = \begin{cases} 0 & \text{if } t_M > t^* \\ 1^2 & \text{if } t_M + t_{M-1} > t^*, t_M \le t^* \\ 2^2 & \text{if } t_M + t_{M-1} + t_{M-2} > t^*, t_M + t_{M-1} \le t^* \\ \vdots & \\ M^2 & \text{if } t_M + t_{M-1} + \ldots + t_1 \le t^* . \end{cases}$$

By conditional expectation, the expected loss is

$$\mathfrak{L} = E[L(T_1, T_2, \ldots, T_M)]$$

$$= \sum_{k=0}^{M} E[L(T_1, T_2, \ldots, T_M) \mid \phi(\mathbf{X}(t^*)) = k] \cdot Pr[\phi(\mathbf{X}(t^*)) = k]$$

$$= \sum_{k=0}^{M} (M-k)^2 \cdot Pr[\phi(\mathbf{X}(t^*)) = k]$$

$$= M^2 \sum_{k=0}^{M} Pr[\phi(\mathbf{X}(t^*)) = k] - 2M \sum_{k=0}^{M} k \; Pr[\phi(\mathbf{X}(t^*)) = k]$$

$$+ \sum_{k=0}^{M} k^2 \; Pr[\phi(\mathbf{X}(t^*)) = k]$$

$$= M^2(1) - 2M \; E[\phi(\mathbf{X}(t^*))] + \{E[\phi(\mathbf{X}(t^*))]\}^2 + Var[\phi(\mathbf{X}(t^*))].$$

Note that for M = 1, $\mathfrak{L} = 1 - 2R(t^*) + \{R(t^*)\}^2 + R(t^*)\{1 - R(t^*)\}$ $= 1 - R(t^*)$ as was shown for the binary model.

If the system states are assigned arbitrarily, then all substitute characteristics should be calculated with respect to the chosen performance measure. Suppose the loss function is given by $L(y) = c(y^* - y)^2$ where y^* is a target value and y are discrete thresholds separating a continuous performance measure. By conditional expectation, the expected loss is

$$\mathfrak{L} = E[L(T_1, T_2, \ldots, T_M)]$$

$$= \sum_{\text{all } y} E[L(T_1, T_2, \ldots, T_M) \mid Y = y] \cdot \Pr[Y = y]$$

$$= \sum_{\text{all } y} c(y^* - y)^2 \cdot \Pr[Y = y]$$

$$= c\ \{(y^*)^2 \sum_{\text{all } y} \Pr[Y = y] - 2y^* \sum_{\text{all } y} y \Pr[Y = y] + \sum_{\text{all } y} y^2 \Pr[Y = y]\}$$

$$= c\ \{(y^*)^2(1) - 2y^* E[Y] + \{E[Y]\}^2 + \mathrm{Var}[Y]\}.$$

For this case, the expected loss includes both the mean and variance within a single performance measure.

The final loss function provides a new substitute characteristic for reliability sensitive to the pattern of degradation about a lifetime specified by the customer. It assigns different weights to each state decrease. Suppose that T_k are mutually independent random variables for the time spent in system state k, k=1,2,...,M. Let the loss function be given by

$$L(t_1, t_2, \ldots t_M) = \begin{cases} 0 & \text{if } t_M > t^* \\ c_M(t^* - t_M)^2 & \text{if } t_M + t_{M-1} > t^*,\ t_M \le t^* \\ \sum_{j=M-1}^{M} c_j(t^* - \sum_{i=j}^{M} t_i)^2 & \text{if } t_M + t_{M-1} + t_{M-2} > t^*,\ t_M + t_{M-1} \le t^* \\ & \vdots \\ \sum_{j=1}^{M} c_j(t^* - \sum_{i=j}^{M} t_i)^2 & \text{if } t_M + t_{M-1} + \ldots + t_1 \le t^* \end{cases}$$

where c_k is the cost for leaving system state k per squared unit of time . Let $P_k(t^*) = \Pr[\phi(\mathfrak{X}(t^*)) = k]$. Assuming that $T_k \sim \exp(\lambda)$ k=1,2,...,M, then from the convolution formulas developed in section 5.2,

$$P_k(t^*) = \begin{cases} \dfrac{(\lambda t^*)^{M-k} \exp(-\lambda t^*)}{(M-k)!} & k = 1,2,\ldots,M \\ 1 - \sum_{k=1}^{M} P_k(t^*) & k = 0. \end{cases}$$

By conditional expectation, the expected loss is

$$\mathscr{L} = E[L(T_1,T_2,\ldots,T_M)]$$

$$= \sum_{k=0}^{M} E[L(T_1,T_2,\ldots,T_M) \mid \phi(\mathbf{X}(t^*)) = k] \cdot P_k(t^*)$$

$$= \sum_{k=0}^{M} E\left[\sum_{j=k+1}^{M} c_j(t^*-\sum_{i=j}^{M} T_i)^2 \middle| \phi(\mathbf{X}(t^*)) = k\right] P_k(t^*)$$

$$= \sum_{k=0}^{M} E\left[\sum_{j=k+1}^{M} c_j(t^*-\sum_{i=j}^{M} T_i)^2 \middle| \sum_{i=k}^{M} T_i > t^*, \sum_{i=k+1}^{M} T_i \le t^*\right] P_k(t^*)$$

$$= \sum_{k=0}^{M} \left\{(t^*)^2 \sum_{j=k+1}^{M} c_j - 2t^* \sum_{j=k+1}^{M} c_j E\left[\sum_{i=j}^{M} T_i \middle| \sum_{i=j-1}^{M} T_i > t^*, \sum_{i=j}^{M} T_i \le t^*\right]\right.$$

$$\left. + \sum_{j=k+1}^{M} c_j E\left[\left(\sum_{i=j}^{M} T_i\right)^2 \middle| \sum_{i=j-1}^{M} T_i > t^*, \sum_{i=j}^{M} T_i \le t^*\right]\right\} P_k(t^*).$$

Using the fact that $\sum_{i=j}^{M} T_i \sim$ Erlang $(M-j+1,\lambda)$, the expected loss can be determined using the computer program listed in Appendix D. The user inputs t^*, M, λ, and c_k for k=1,...,M. The program determines the theoretical expected loss and checks the result by simulation.

If the customer chooses to use several conflicting substitute characteristics for reliability, then the problem becomes a discrete multiobjective optimization problem. Many techniques already exist for finding the best compromise solution through interaction with the customer.

5.6 Summary

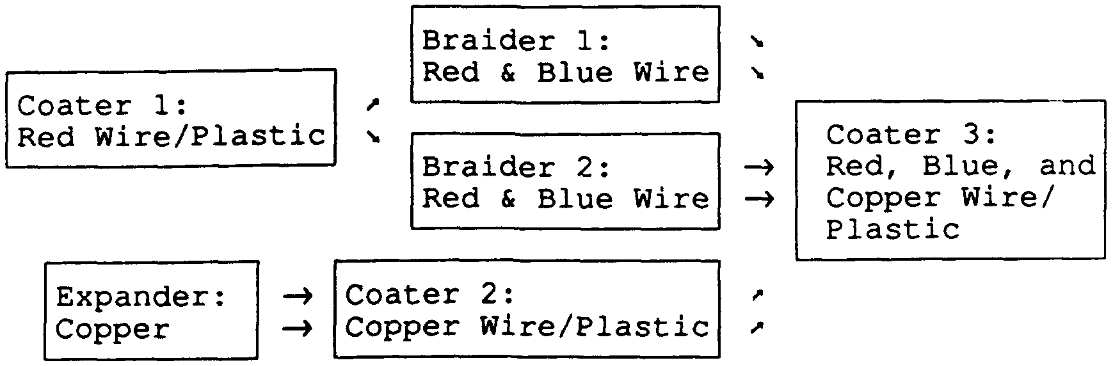
This chapter promoted the multistate model over both the binary and continuous model. It emphasized the customer's role in the development and evaluate of the multistate model. The credibility and usefulness of the model are increased by involving the customer at every step.

6. APPLICATIONS

This chapter demonstrates the development and evaluation of the general multistate model for several problems. The examples were chosen from such diverse areas as production and assembly, military battle planning, and wearout analysis to demonstrate the versatility and flexibility of the general multistate model.

6.1 Production and Assembly Process

Suppose that the multistate model is used for the wire flowshop shown in Figure 6.1 where a cable is created from the following raw materials: red wire, blue wire, copper, and plastic. The red wire is purchased bare and is covered with plastic on coating machine 1. Next, the red and blue wires are intertwined on one of two braiding machines. The process has redundancy at this point since a single braider causes production to backup. At the same time, copper bars are worked to form a copper grounding wire on the expanding machine. Next, the copper wire is covered with plastic on



Braider 1: Red & Blue Wire
Coater 1: Red Wire/Plastic
Braider 2: Red & Blue Wire
Coater 3: Red, Blue, and Copper Wire/ Plastic
Expander: Copper
Coater 2: Copper Wire/Plastic

Figure 6.1. Process Flow Diagram.

coating machine 2. During the final assembly, the braided wires and the coated copper wire are covered with plastic on coating machine 3 to form the desired cable.

The wire flowshop operates under the Just-In-Time philosophy where the proper amount of raw materials are delivered by each supplier every hour. The wire flowshop has been designed so that if 100% of the raw materials needed for an hour of production are delivered and if all machines operate at an ideal production rate, then the system will produce 100% of the desired amount of cable. Any deviation in the hourly delivery of raw materials or in the production efficiency of the expander or coaters will decrease the flowshop's hourly output of cable. The only machines with excess production capacity are the two braiders.

There are several measures commonly used to describe the production rate of a system. Throughput is defined as the number of complete assemblies that can be generated in a fixed period of time. Average flow time is the average length of time from the arrival of raw materials to the completion of the finished product. Bottleneck flow rate is the production rate at the system's bottleneck. Another measure is the *instantaneous production potential* which is a function of the percentage of raw materials delivered and the production efficiency of the machines. The multistate model will be used to evaluate the instantaneous production potential for the system.

Step 1: Define the Number of System and Component States.

Suppose that the wire flowshop operates in a cycle where the four suppliers deliver raw materials to the plant every hour. However, the suppliers do not consistently deliver the entire amount of raw materials required for an hour of work. For accounting purposes, the suppliers have agreed to adhere to one of the following three options:

1. Deliver 100% of the materials for 1 hour of work,
2. Deliver 50% of the materials for 1 hour of work, or
3. Make no delivery.

This is an example that requires the state classification of a discrete performance measure. As defined in Table 6.1, let the variables x_i, i=1,2,3,4 represent the percentage of red wire, blue wire, copper, and plastic delivered for 1 hour of work. The maximum number of states possible is 3 (M_i=2). If the customer chooses to use the greatest amount of detail, then the component states for the raw materials should be described as shown in Table 6.2.

In addition, the machines do not always operate at peak efficiency. The production rate of each machine continuously degrades. Suppose that the customer does not want to model the production rate as a continuous random variable. This is an example that requires the state classification of a continuous performance measure. Let the customer's quadratic loss function be given by $L(y) = \$160(1-y)^2$ where y is a percentage of the ideal production rate.

Table 6.1 Production Component Definitions.

Variable	Definition
x_1	Percentage of Red Wire Delivered
x_2	Percentage of Blue Wire Delivered
x_3	Percentage of Copper Delivered
x_4	Percentage of Plastic Delivered
x_5	Coater 1 Percentage of Ideal Prod. Rate
x_6	Coater 2 Percentage of Ideal Prod. Rate
x_7	Coater 3 Percentage of Ideal Prod. Rate
x_8	Braider 1 Percentage of Ideal Prod. Rate
x_9	Braider 2 Percentage of Ideal Prod. Rate
x_{10}	Expander Percentage of Ideal Prod. Rate

Table 6.2 Component States and Descriptions.

Variable	State	Description
x_1, x_2, x_3, x_4	0	None of Order Delivered
	1	50% of Order Delivered
	2	100% of Order Delivered
$x_5, x_6, x_7, x_8, x_9, x_{10}$	0	0% of Ideal Production Rate
	1	50% of Ideal Production Rate
	2	75% of Ideal Production Rate
	3	100% of Ideal Production Rate

Assume that the three countermeasures for restoring a machine to the ideal production rate cost \$10, \$40, and \$160. The customer will never choose the \$10 countermeasure until $y = .75$ because $L(y) < 10$ for $.75 < y \leq 1.0$. Therefore, the customer makes no distinction for production rates in the range $(.75, 1.00]$. Likewise, the customer will not choose the

\$40 countermeasure until $y = .5$ because $L(y) < 40$ for $.5 < y \leq 1.0$. The \$160 countermeasure is not implemented until $y = 0.0$. In summary, 4 states (M_1=3) are needed to describe production rates in the following ranges: [0.0,0.0], (0.0,.50], (.50,.75], and (.75,1.0]. As defined in Table 6.1, the variables x_5, x_6, x_7, x_8, x_9, and x_{10} denote the states of the three coaters, the two braiders, and the expander, respectively. Table 6.2 gives the state classification for each machine's percentage of the ideal production rate.

The purpose of the model is to measure the distribution for the instantaneous production potential of the system. Now that the number of states are known for each of the components, the largest number of system states can be found. The system production rate will be affected by both the percentage of raw materials delivered and the percentage of the ideal production rate for each machine. The system can only assume the states shown in Table 6.3 given the component states listed in Table 6.2.

Table 6.3 System States and Descriptions.

State	Description
0	0% of Ideal Production Potential
1	50% of Ideal Production Potential
2	75% of Ideal Production Potential
3	100% of Ideal Production Potential

Step 2: Estimate Component State Probabilities.

It is fairly simple to see how the component state probabilities for x_i, i=1,2,3,4 can be estimated from previous experience with the suppliers. Data should be collected on the percentage of the order received each hour from each supplier. Suppose that this technique results in the component state probabilities which are summarized in Table 6.4.

Table 6.4 Component State Probabilities.

Variable	State			
	0	1	2	3
x_1	.0100	.0400	.9500	-
x_2	.0100	.0300	.9600	-
x_3	.0100	.0100	.9800	-
x_4	.0100	.0200	.9700	-
x_5	.0002	.0045	.0905	.9048
x_6	.0002	.0045	.0905	.9048
x_7	.0002	.0045	.0905	.9048
x_8	.0474	.1438	.3595	.4493
x_9	.0474	.1438	.3595	.4493
x_{10}	.0011	.0164	.1638	.8187

The component state probabilities for x_i, i=5,6,7,8,9,10 can be estimated with the convolution technique introduced in section 5.2. Let T_{ij} be a random variable representing the time component i stays in state j, i=5,6,7,8,9,10 and j=1,2,3. Suppose that T_{ij}, i=5,6,7 and j=1,2,3 are mutually independent exponential variables with λ_i = .1. The distribution of $T_{i,M_i} + T_{i,M_{i-1}} + \ldots + T_{ij}$ is the convolution of (M_i-j+1) exponential distributions each with parameter λ_i which is an Erlang distribution with shape parameter (M_i-j+1) and scale parameter λ_i. In general, for Erlang (η,λ),

$$R(t) = Pr[T > t] = 1 - F(t) = \sum_{k=0}^{\eta-1} \frac{(\lambda t)^k \exp(-\lambda t)}{k!}.$$

Therefore, $Q_{ij}(t) = \sum_{k=0}^{M_i-j} \frac{(\lambda_i t)^k \exp(-\lambda_i t)}{k!}$. For i=5,6,7

$Q_{i3}(1) = e^{-.1} = .9048$,

$Q_{i2}(1) = e^{-.1} + .1\ e^{-.1} = .9953$,

$Q_{i1}(1) = e^{-.1} + .1\ e^{-.1} + \frac{(.1)^2\ e^{-.1}}{2!} = .9998$, and

$Q_{i0}(1) = 1.0$. Thus, the component state probabilities are $P_{i0}(1) = Q_{i0}(1) - Q_{i1}(1) = .0002$, $P_{i1}(1) = Q_{i1}(1) - Q_{i2}(1) = .0045$, $P_{i2}(1) = Q_{i2}(1) - Q_{i3}(1) = .0905$, and $P_{i3}(1) = Q_{i3}(1) - Q_{i4}(1) = .9048$.

Suppose that T_{ij}, i=8,9 and j=1,2,3 are mutually independent exponential variables with $\lambda_i = .8$. Using the same procedure, the component state probabilities are $P_{i0}(1) = .0474$, $P_{i1}(1) = .1438$, $P_{i2}(1) = .3595$, and $P_{i3}(1) = .4493$.

Suppose that T_{ij}, i=10 and j=1,2,3 are mutually independent exponential variables with $\lambda_i = .2$. Using the same procedure, the component state probabilities are $P_{i0}(1) = .0011$, $P_{i1}(1) = .0164$, $P_{i2}(1) = .1638$, and $P_{i3}(1) = .8187$. Each of the component state probability distributions are listed in Table 6.4.

Step 3: Define the System.

The customer can define the system by specifying which component state vectors belong to each equivalence class. This system has $3^4 \cdot 4^6 = 331,776$ different component state vectors. Fortunately, it is not necessary to determine the equivalence class for every $\mathbf{x} \in S$. The customer only needs to specify when a decrease (increase) in the state of any one of the n components forces a decrease (increase) in the state of the system. For this specific problem, the customer needs to specify when a decrease in the percentage of raw materials delivered or in the percentage of the ideal production rates of the machines forces a decrease in the production potential of the system.

Suppose the customer states that any decrease in the percentage of raw materials delivered causes a corresponding decrease in the system's production potential. Also, assume that the customer states that any decrease in the percentage of the ideal production rate of the coaters or the enlarger causes a corresponding decrease in the system's production potential. Finally, suppose that the customer considers the parallel operation of the two braiders to be additive. For example, a 50% production rate on both braiders satisfies a 100% production potential for the system as will a 100% production rate on a single braider.

Beginning with system state 3 and the perfect component state vector, $\mathbf{x}_M = (2,2,2,2,3,3,3,3,3,3)$. To be consistent, the customer will say that $\mathbf{x}_M$ is not a lower boundary point to level 3 since x_8 (Braider 1) and x_9 (Braider 2) can be decreased without leaving system state 3. The customer has implied that only x_8 and x_9 can be changed without leaving system state 3. Since the customer considers the percentage of the production rates of the braiders to be additive, the lower boundary points to level 3 are (2,2,2,2,3,3,3,3,0,3), (2,2,2,2,3,3,3,0,3,3), and (2,2,2,2,3,3,3,1,1,3).

The procedure is similar for the lower boundary points to level 2. Start by looking for the lowest possible states for the variables $x_1,x_2,x_3,x_4,x_5,x_6,x_7$, and x_{10} so that the system is in state 2. From the customer's interpretation of the system, each of the variables must be in state 2. Any further decrease in these variables results in a decrease in the production potential of the system. Again, using the customer's additive sense of the variables x_8 and x_9, the lower boundary points to level 2 are (2,2,2,2,2,2,2,2,0,2), (2,2,2,2,2,2,2,0,2,2), and (2,2,2,2,2,2,2,1,1,2). Similar logic gives the following lower boundary points to level 1: (1,1,1,1,1.1,1,1,0,1), and (1,1,1,1,1,1,1,0,1,1). The upper boundary points to level k can also be determined in this manner. All the lower boundary points and the upper boundary points to level k are given in Table 6.5. Note that the entire system of 331,776 component state vectors is summarized with just 8 lower boundary points and 29 upper boundary points!

Table 6.5 Lower and Upper Boundary Points.

Level	Lower Boundary Points	Upper Boundary Points
0	None	0222333333 2022333333 2202333333 2220333333 2222033333 2222303333 2222330333 2222333003 2222333330
1	1111111101 1111111011	1222333333 2122333333 2212333333 2221333333 2222133333 2222313333 2222331333 2222333103 2222333013 2222333331
2	2222222202 2222222022 2222222112	1222333333 2122333333 2212333333 2221333333 2222233333 2222323333 2222332333 2222333203 2222333023 2222333332
3	2222333303 2222333033 2222333113	None

Step 4: Estimate System State Probabilities.

The FORTRAN program called MAIN in Appendix A implements several techniques for calculating the exact probability distribution of a general MCS. It includes routines for enumeration, inclusion-exclusion, and pivotal decomposition using either the lower or the upper boundary points. Each technique requires the following information from the customer to completely describe the system:

1. The customer's determination of the number of distinctive system and component states,
2. The probability distribution of each component, and
3. The customer's definition of the system in terms of either the lower or the upper boundary points.

This information was collected in the previous three steps. The program outputs the exact probability distribution of the system. For the production and assembly process, the program produced the results given in Table 6.6.

Table 6.6 System State Probabilities.

k	$\Pr[\phi(\mathbf{X})) = k]$
0	.043191
1	.130323
2	.327007
3	.499479

Step 5: Determine Substitute Characteristics for Reliability.

The final step is to evaluate substitute characteristics for reliability that summarize the system state probabilities for the customer. Evaluating the expected system state results in $E[\phi(\mathbf{X})] = 2.282774$. Does this mean that we can interpolate to find an expected production potential for the system of $.282774(1.0-.75)+.75 = .8207$? The answer is no! Making the calculation directly, the expected production potential is (.130323)(.5) + (.327007)(.75) + (.499479)(1) = .8099. Thus, it makes no sense to evaluate $E[\phi(\mathbf{X})]$ and $\mathrm{Var}[\phi(\mathbf{X})]$ because of the arbitrary way the system states were defined (i.e. no system state was used for a 25% production potential because it was not possible).

One solution is to redefine the system states from 0 to 4 so that $E[\phi(\mathbf{X})] = 3.239597$ and interpolation gives the proper expected production potential of .239597(1.0-.75)+.75 = .8099. A better approach is to evaluate the substitute characteristics directly in terms of the customer's real variable of interest. In the case of the production and assembly process, the substitute characteristics are computed in terms of Y, the random variable for the instantaneous production potential of the system. The customer may choose to use one or several of the performance measures listed in Table 6.7 for evaluating the production process.

Table 6.7 Substitute Characteristics.

Substitute Characteristic	Calculated Value
$E[Y]$	0.809896
$\mathrm{Var}[Y]$	0.060070
$\Pr[Y \geq .5]$	0.956809
$\Pr[Y \geq .75]$	0.826486
$\Pr[Y \geq 1.0]$	0.499479
$E[L(Y)]$	\$153.94

The calculations for each substitute characteristic are obvious with the exception of the expected loss, which requires the customer's loss function due to variation from the system's ideal production rate. Let $L(y)=\$1600(1-y)^2$ where y is a percentage of the ideal production rate. Then the expected loss $\mathcal{L} = E[L(Y)] = \int_{\text{all } y} L(y)\, f(y)\, dy$. A discrete approximation of f(y) is given in Table 6.6. Therefore, $\mathcal{L} \approx 0(.499479) + 100(.327007) + 400(.130323) + 1600(.043191) = \$153.94$ every hour. Since the loss function is quadratic, $E[L(Y)] = \$16000[1 - 2\mu_y + \mu_y^2 + \sigma_y^2] \approx \$16000[1 -2(.809896) + (.809896)^2 + .060070] = \$153.94$ which agrees with the previous approximation for expected loss.

6.2 Military Battle Planning

Suppose the Army wishes to evaluate the "reliability" of a battle plan. The plan includes two attack groups, each supported by a separate artillery unit. Each attack group has the potential for accomplishing up to 2 out of 4 mission objectives without effective artillery support. With effective artillery support, the potential of each attack group is doubled so that a single attack group can now accomplish all 4 mission objectives. Suppose that the 4 mission objectives can be assigned to any attack group. The goal of the multistate model is to evaluate how effectively the battle groups accomplish the mission objectives.

Step 1: Define the Number of System and Component States.

Let the variables x_1 and x_2 represent the effectiveness of the artillery support for attack groups 1 and 2, respectively. From the description, it is only necessary to distinguish between two levels of artillery support: effective and not effective. Let the variables x_3 and x_4 represent the number of objectives accomplished by attack groups 1 and 2, respectively. From the description, each attack group can accomplish 0, 1, or 2 mission objectives without artillery support. Using the largest number of states, x_1, x_2, x_3, and x_4 are defined in Tables 6.8 and 6.9.

Table 6.8 Battle Plan Component Definitions.

Variable	Definition
x_1	Artillery Effectiveness for Attack Group 1
x_2	Artillery Effectiveness for Attack Group 2
x_3	Objectives Accomplished by Attack Group 1
x_4	Objectives Accomplished by Attack Group 2

Table 6.9 Component States and Descriptions.

Variable	State	Description
x_1, x_2	0	Artillery Not Effective
	1	Artillery Effective
x_3, x_4	0	No Objectives Accomplished
	1	One Objective Accomplished
	2	Two Objectives Accomplished

The model will estimate the probability distribution for the number of mission objectives accomplished by the two attack groups. It is possible to accomplish 0, 1, 2 , 3, or 4 mission objectives which are the system states important to the Army (Table 6.10).

Table 6.10 System States and Descriptions.

State	Description
0	No Objectives Accomplished
1	One Objective Accomplished
2	Two Objectives Accomplished
3	Three Objectives Accomplished
4	Four Objectives Accomplished

Step 2: Estimate Component State Probabilities.

After the distinct states are defined by the Army, the probability distributions of the artillery units and the attack groups must be estimated. Suppose that Table 6.11 contains estimates derived from training exercises performed by each artillery unit and each attack group.

Table 6.11 Component State Probabilities.

Variable	State		
	0	1	2
x_1	.3	.7	-
x_2	.2	.8	-
x_3	.01	.89	.1
x_4	.05	.8	.15

Step 3: Define the System.

The battle plan example has $2^2 \cdot 3^2 = 36$ different component state vectors. The problem size was intentionally restricted so that the entire component state space could be separated into equivalence classes for the reader as shown in Table 6.12.

Table 6.12 Equivalence Classes.

k	No. of Vectors	$\mathbf{x} \in S_k$
0	4	0000 0100 1000 1100
1	4	0001 0010 0110 1001
2	9	0002 0011 0020 0101 0120 1002 1010 1101 1110
3	4	0012 0021 0111 1011

4	15	0022 0102 0112 0121 0122
		1012 1020 1021 1022 1102
		1111 1112 1120 1121 1122

It is not necessary to determine the equivalence class for every component state vector. As shown before, the customer must only specify when a decrease in the state of any one of the n components forces a decrease in the state of the system. For this problem, the Army must specify when a decrease in the combat effectiveness of the artillery units or in the number of objectives accomplished by the attack groups forces a decrease in the total number of objectives accomplished.

Using the algorithm found in section 5.3, start with M = 2 and the perfect component state vector, $\mathbf{x}_M = (1,1,2,2)$. From the Army's description, $\mathbf{x}_M$ is not a lower boundary point to level 4 since x_2 and x_4 (artillery unit 2 and attack group 2) can be removed from the battle without leaving system state 4. However, this change results in $\mathbf{x}^1 = (1,0,2,0)$ which is a lower boundary point to levels 3 and 4 since $\phi(0,0,2,0) = 2$ and $\phi(1,0,1,0) = 2$. The symmetric nature of this problem implies that $\mathbf{x}^2 = (0,1,0,2)$ is also a lower boundary point to levels 3 and 4. All other lower boundary points to level 4 must be efficient with respect to $\mathbf{x}^1$ and $\mathbf{x}^2$. The analysis continues by arbitrarily choosing $\mathbf{x}^1$.

The next step is to systematically decrease the states of some of the components in $\mathbf{x}^1$ and to determine the minimal increases in the states of the other components to return the system to state 4. For example, if x_1 is decreased to state 0 in $\mathbf{x}^1$, then two different options will return the system to state 4 in a minimal fashion: $\mathbf{x}^3 = (0,0,2,2)$ and $\mathbf{x}^4 = (0,1,2,1)$. Both vectors are lower boundary points to level 4. The symmetric nature of this problem implies that $\mathbf{x}^5 = (1,0,1,2)$ is also a lower boundary point.

Continuing with $\mathbf{x}^1$, if x_3 is decreased to state 1, then two different options return the system to state 4: $\mathbf{x}^6 = (1,1,1,1)$ and $\mathbf{x}^* = (1,0,1,2)$. $\mathbf{x}^6$ is new a lower boundary points to level 4 and $\mathbf{x}^*$ was found earlier. At this point, the decrease of more than one of the components in $\mathbf{x}^1$ needs to be explored as well as decreasing the components by more than one state. The remaining details of this procedure are left to the reader.

Similar logic gives the lower boundary points to levels 3, 2, and 1 as well as the upper boundary points to levels 3, 2, 1, and 0. All the lower and upper boundary points to level k are listed in Table 6.13. Either set of boundary points are enough to completely describe the system.

Table 6.13 Lower and Upper Boundary Points.

Level	Lower Boundary Points	Upper Boundary Points
0	-	1100
1	0001 0010	1100 1001 0110
2	0002 0011 0020 0101 1010	1110 1101 1002 0120 0011
3	0012 0021 0102 0111 1011 1020	1110 1101 1011 1002 0120 0111 0021 0012
4	0022 0102 0121 1012 1020 1111	-

Once the lower boundary points have been found, an alternate representation of the customer's structure function can be written. Using the formulas found in section 3.1.5:

$\phi^1(\mathbf{y}) = \text{Max}\{y_{41}, y_{31}\}$,

$\phi^2(\mathbf{y}) = \text{Max}\{y_{42}, y_{31}y_{41}, y_{32}, y_{21}y_{41}, y_{11}y_{31}\}$,

$\phi^3(\mathbf{y}) = \text{Max}\{y_{31}y_{42}, y_{32}y_{41}, y_{21}y_{42}, y_{21}y_{31}y_{41}, y_{11}y_{31}y_{41}, y_{11}y_{32}\}$,

$\phi^4(\mathbf{y}) = \text{Max}\{y_{32}y_{42}, y_{21}y_{42}, y_{21}y_{32}y_{41}, y_{11}y_{31}y_{42}, y_{11}y_{32}, y_{11}y_{21}y_{31}y_{41}\}$,

and $\phi(\mathbf{x}) = \phi^1(\mathbf{y}) + \phi^2(\mathbf{y}) + \phi^3(\mathbf{y}) + \phi^4(\mathbf{y})$.

Suppose $\mathbf{x} = (0,1,2,0)$. Then $\mathbf{y} = (y_{11},y_{21},y_{31},y_{32},y_{41},y_{42}) = (0,1,1,1,0,0)$. Therefore, $\phi^1(\mathbf{y})=1$, $\phi^2(\mathbf{y})=1$, $\phi^3(\mathbf{y})=0$, $\phi^4(\mathbf{y})=0$, and $\phi(\mathbf{x}) = 2$. This agrees with the value given in Table 6.12. It can also be quickly verified that the derived structure function is equivalent to the supposedly unknown structure function implicitly being used by the Army:

$$\phi(\mathbf{x}) = \text{Min}\{\, 4,\ x_3 2^{x_1} + x_4 2^{x_2} \,\} .$$

As a final check, the boundary point conversion program found in Appendix C was run to verify that all lower and upper boundary points were found. The lower boundary points to level k generated the upper boundary points to level k-1 for k=1,2,3 and vice versa. Therefore, none of the boundary points were missed.

Step 4: Estimate System State Probabilities.

For the battle plan example, the information from Tables 6.9, 6.10, 6.11, and 6.13 was supplied to the FORTRAN program in Appendix A. The enumeration, inclusion-exclusion, and pivotal decomposition techniques were exercised with both the upper and lower boundary points. The program always produced the probability distribution for the system which is listed in Table 6.14. Note that the program results would have been inconsistent if some of the boundary points had been missed.

Table 6.14 System State Probabilities.

k	$Pr[\phi(\mathbf{X}) = k]$
0	0.00050
1	0.01495
2	0.08207
3	0.28337
4	0.61911

Step 5: Determine Substitute Characteristics for Reliability.

Each of the substitute characteristics for "reliability" provides a measure of the effectiveness of the attack plan. Since the systems states correspond directly to the number of objectives accomplished, the system performance can be measured in terms of either the system state, $\phi(\mathbf{X})$, or the number of mission objectives accomplished, Y.

The Army can use any or all of the performance measures listed in Table 6.15 for evaluating the effectiveness of the battle plan. The center and spread of the distribution is indicated by $E[\phi(\mathbf{X})] = 3.50564$ and $Var[\phi(\mathbf{X})] = 0.50981$. The Army should attempt to increase $E[\phi(\mathbf{X})]$ while decreasing $Var[\phi(\mathbf{X})]$. If the Army can label the accomplishment of 2 or more objectives as "good", then $Pr[\phi(\mathbf{X}) \geq 2] = 0.98455$ is an appropriate measure of system performance. The Army should attempt to increase this probability. Finally, the Army will strive to minimize $E[L(\phi(\mathbf{X}))]$.

Table 6.15 Substitute Characteristics.

Substitute Characteristic	Calculated Value
$E[\phi(\mathbf{X})] = E[Y]$	3.50564
$Var[\phi(\mathbf{X})] = Var\ [Y]$	0.50981
$Pr[\phi(\mathbf{X}) \geq 1] = Pr[Y \geq 1]$	0.99950
$Pr[\phi(\mathbf{X}) \geq 2] = Pr[Y \geq 2]$	0.98455
$Pr[\phi(\mathbf{X}) \geq 3] = Pr[Y \geq 3]$	0.90248
$Pr[\phi(\mathbf{X}) \geq 4] = Pr[Y \geq 4]$	0.61911
$E[L(\phi(\mathbf{X}))] = E[L(Y)]$	\$5322.60

The calculations for each substitute characteristic are obvious with the exception of the expected loss. Suppose that the Army's discrete loss function due to any deviation from accomplishing all 4 objectives is

$$L(y) = \begin{cases} \$200{,}000 & \text{if } y=0 \\ \$50{,}000 & \text{if } y=1 \\ \$20{,}000 & \text{if } y=2 \\ \$10{,}000 & \text{if } y=3 \\ \$0 & \text{if } y=4 \end{cases}$$

where y is the number of objectives accomplished. Then the expected loss $\mathcal{L} = E[L(Y)] = \sum_{y=0}^{4} L(y)\,p(y)$ where p(y) is the probability mass function given in Table 6.14. Therefore, $\mathcal{L} = 0(.61911) + 10000(.28337) + 20000(.08207) + 50000(.01495) + 200000(.0005) = \$5322.60$.

6.3 Tire Tread Wear

Suppose the manufacturer of a car tire wishes to evaluate the wear of two newly developed tires: brand X and brand Y. Each brand of tire is mounted on the right-front, left-front, right-rear, and left-rear positions of identical test vehicles. All tires start with 8/32" of tread. The test vehicles are driven on a track and the amount of tread left on each tire is recorded to the nearest 1/32" at the end of a 60,000 mile test. The multistate model will be used to compare the wear of the two brands of tire.

Step 1: Define the Number of System and Component States.

Let the variables x_1, x_2, x_3, and x_4 represent the tread left on the right-front, left-front, right-rear, and left-rear tires, respectively. Since the data was collected to the nearest 1/32", it is possible to use 9 states for each variable. The manufacturer does not recommend driving on tires unless more than 2/32" of tread remains. They also decide that a breakdown of tread wear into 2 additional equal categories will be sufficient to compare the two brands. These choices produce the definitions and descriptions for x_1, x_2, x_3, and x_4 given in Tables 6.16 and 6.17.

Table 6.16 Tire Tread Wear Component Definitions.

Variable	Definition
x_1	Tread Remaining on Right-Front Tire
x_2	Tread Remaining on Left-Front Tire
x_3	Tread Remaining on Right-Rear Tire
x_4	Tread Remaining on Left-Rear Tire

Table 6.17 Component States and Descriptions.

Variable	State	Description
x_1, x_2, x_3, x_4	0	High Tread Wear 0/32" - 2/32" Tread Remains
	1	Moderate Tread Wear 3/32" - 5/32" Tread Remains
	2	Low Tread Wear 6/32" - 8/32" Tread Remains

The manufacturer insists that the car is in the worst state (system state 0) if any of the tires has high tread wear (component state 0). Otherwise, the manufacturer wants to distinguish between various system states according to the number of tires with low tread wear (component state 2). The last condition will require an additional 5 system states, producing the six system states shown in Table 6.18.

Table 6.18 System States and Descriptions.

State	Description
0	Any Tire With High Tread Wear
1	No Tires With Low Tread Wear and No Tires With High Tread Wear (All Tires With Moderate Tread Wear)
2	One Tire With Low Tread Wear and No Tires With High Tread Wear
3	Two Tires With Low Tread Wear and No Tires With High Tread Wear
4	Three Tires With Low Tread Wear and No Tires With High Tread Wear
5	Four Tires With Low Tread Wear and No Tires With High Tread Wear

<u>Step 2: Estimate Component State Probabilities</u>.

The probability distribution of every component must be estimated for each brand. As in the first example, this can be done with the convolution technique. This procedure will not be demonstrated again. Suppose that the component state probabilities for brand X and brand Y were estimated and are given in Tables 6.19 and 6.20.

Table 6.19 Brand X Component State Probabilities.

Variable	State		
	0	1	2
x_1	.03	.87	.1
x_2	.06	.89	.05
x_3	.01	.94	.05
x_4	.03	.88	.09

Table 6.20 Brand Y Component State Probabilities.

Variable	State		
	0	1	2
x_1	.05	.83	.12
x_2	.06	.86	.08
x_3	.04	.89	.07
x_4	.08	.82	.1

Step 3: Define the System.

The tire tread wear example has $3^4 = 81$ component state vectors. It is not necessary to determine the equivalence class for every $\mathbf{x} \in S$. The manufacturer need only say when a decrease in the state of a tire forces a decrease in the number of tires with low tread wear. Since this is how the manufacturer defined the system states, each $\mathbf{x}$ in the kth equivalence class will be both a lower boundary point and an upper boundary point to level k.

Using the algorithm for finding all lower boundary points, the highest system state is M = 5 and the perfect component state vector is $\mathbf{x}_M = (2,2,2,2)$. $\mathbf{x}_M$ is a lower boundary point to level 5. So $L_{51} = (2,2,2,2)$. No other lower boundary points to level 5 can exist since no other component state vectors are efficient with respect to $\mathbf{x}_M$.

Start searching for the lower boundary points to level 4 by decreasing the states of the components of L_{51} until a lower boundary point is found. Reducing x_1 by one in L_{51} results in the first lower boundary point to level 4, L_{41} =

(1,2,2,2). Using the notion that the other lower boundary points to level 4 must be efficient with respect to L_{41} gives L_{42} = (2,1,2,2), L_{43} = (2,2,1,2), and L_{44} = (2,2,2,1).

Next, search for lower boundary points to level 3 by decreasing the states of the components of L_{41} until a lower boundary point is found. Reducing x_2 by one in L_{41} results in the first lower boundary point to level 3, L_{31} = (1,1,2,2). Using the notion that the other lower boundary points to level 3 must be efficient with respect to L_{31} results in L_{32} = (1,2,1,2), L_{33} = (1,2,2,1), L_{34} = (2,1,1,2), L_{35} = (2,1,2,1), and L_{36} = (2,2,1,1).

The first lower boundary point to level 2 is found by reducing x_3 by one in L_{31} so that L_{21} = (1,1,1,2). The remaining efficient points are L_{22} = (1,1,2,1), L_{23} = (1,2,1,1), and L_{24} = (2,1,1,1). Reducing x_4 by one in L_{21} results in the only lower boundary point to level 1, L_{11} = (1,1,1,1). The algorithm ends since lower boundary points are not defined for level 0. Similar logic gives the upper boundary points to levels 0, 1, 2, 3, and 4. All lower and upper boundary points are listed in Table 6.21.

The boundary point conversion program in Appendix C was used to make sure that no boundary points were missed. The lower boundary points to level k generated the upper boundary points to level k-1, k=1,2,...,M and the upper boundary points to level k generated the lower boundary points to level k+1, k=0,1,...,M-1.

Table 6.21 Lower and Upper Boundary Points.

Level	Lower Boundary Points	Upper Boundary Points
0	-	0222 2022 2202 2220
1	1111	0222 2022 2202 2220 1111
2	1112 1121 1211 2111	0222 2022 2202 2220 1112 1121 1211 2111
3	1122 1212 1221 2112 2121 2211	0222 2022 2202 2220 1122 1212 1221 2112 2121 2211
4	2221 2212 2122 1222	2221 2212 2122 1222
5	2222	-

<u>Step 4: Estimate System State Probabilities</u>.

For the tire wear example, the information collected for brand X and brand Y was supplied to the FORTRAN program in Appendix A. Each technique in the program gave the system state probabilities listed in Table 6.22.

Table 6.22 System State Probabilities By Brand.

k	$\Pr[\phi(\mathbf{X}) = k]$	
	Brand X	Brand Y
0	.124398	.211302
1	.640501	.520931
2	.209179	.228274
3	.024660	.036832
4	.001239	.002593
5	.000023	.000067

Step 5: Determine Substitute Characteristics for Reliability.

The manufacturer can use any or all of the performance measures listed in Table 6.23 for evaluating the wear of the two brands of tires. Neither brand is superior with respect to every performance measure: Brand X has a higher expected system state and lower variability; Brand Y has a better performance distribution for all but $\Pr[\phi(\mathbf{X}) \geq 1]$. The calculation of expected loss requires the specification of the manufacturer's loss function. Suppose that the manufacturer gives the following loss function:

Table 6.23 Substitute Characteristics By Brand.

Substitute Characteristic	Calculated Value	
	Brand X	Brand Y
$E[\phi(\mathbf{X})]$	1.137910	1.098683
$\mathrm{Var}[\phi(\mathbf{X})]$	0.424717	0.601575
$\Pr[\phi(\mathbf{X}) \geq 1]$	0.875602	0.788697
$\Pr[\phi(\mathbf{X}) \geq 2]$	0.235101	0.267766
$\Pr[\phi(\mathbf{X}) \geq 3]$	0.025922	0.039492
$\Pr[\phi(\mathbf{X}) \geq 4]$	0.001262	0.002660
$\Pr[\phi(\mathbf{X}) \geq 5]$	0.000023	0.000067
$E[L(\phi(\mathbf{X}))]$	\$153.36	\$148.85

$$L(\phi(\mathbf{X}(t^*))) = \begin{cases} \$200 & \text{if } \phi(\mathbf{X}(t^*)) = 0 \\ \$190 & \text{if } \phi(\mathbf{X}(t^*)) = 1 \\ \$30 & \text{if } \phi(\mathbf{X}(t^*)) = 2 \\ \$20 & \text{if } \phi(\mathbf{X}(t^*)) = 3 \\ \$10 & \text{if } \phi(\mathbf{X}(t^*)) = 4 \\ \$0 & \text{if } \phi(\mathbf{X}(t^*)) = 5 \end{cases}$$

which reflects that most customers want to purchase new tires once system state 1 is reached. From this loss function, the expected losses for Brand X and Brand Y are \$153.36 and \$148.85, respectively.

6.4 Summary

The customer was involved in the generation of the multistate model. The production example described the state classification of performance measures and the convolution technique for estimating component state probabilities. The battle plan example demonstrated the algorithm for finding boundary points and the alternate representation of $\phi(\mathbf{x})$. The tire tread wear example explained how to use the model for comparing two systems and how different conclusions could be reached by considering different performance measures.

7. FURTHER RESEARCH, SUMMARY, AND CONCLUSIONS

This chapter provides several promising directions for further research, gives a summary of the main contributions shown in this dissertation, and presents concluding remarks regarding the importance of these new results.

7.1 Directions for Further Research

There were several topics related to the multistate reliability model that were not completely finalized. The most promising directions for further research are discussed in the next four sections.

7.1.1 Fuzzy Sets

In traditional set theory, sets are a well-defined collection of elements. In other words, an element either is or is not a member of the set. In fuzzy set theory, sets are an ill-defined collection of elements. A membership function is used to indicate the degree of membership for each element in the fuzzy set.

Park [1987] defined a fuzzy set $\tilde{A}$ in $\mathbf{X}$ with the set of ordered pairs, $\{(\mathbf{x}, g_{\tilde{A}}(\mathbf{x}))\}$. The real value in the interval $[0,1]$ given by the membership function, $g_{\tilde{A}}(\mathbf{x})$, represents the degree of membership for each point in $\mathbf{X}$. $\mathbf{X}$ is an explicit support set usually taken to be $\mathbb{R}^n$. For example, suppose $\tilde{A}$ is the fuzzy set of "good" students. Let $\mathbf{X} = (x_L, x_A,, x_V, x_Q)$ represent a student's percentile rank on language, auditory, visual, and quantitative tests. Each student's degree of membership in $\tilde{A}$ depends on the specific membership function used to determine what is meant by a "good" student. Suppose that the school board uses the following membership function to determine a student's potential:

$$g_A(\mathbf{x}) = .2x_L + .3x_A + .1x_V + .4x_Q.$$

Larger values of $g_A(\mathbf{x})$ indicate a higher degree of membership for each student in the fuzzy set of "good" students.

A fuzzy number is a fuzzy set defined on the real axis. Kaleva [1986] described the performance of the components in a binary model with fuzzy numbers and used some existing properties of fuzzy numbers to determine the performance of the system in terms of another fuzzy number.

Two ideas for introducing fuzzy sets to the multistate reliability model are given next. Up to this point, each component state vector was a member of a single equivalence class based on the system state. Therefore, equivalence classes relied on traditional set theory. Fuzzy sets could be applied by using a membership function to determine the degree of membership for $\mathbf{x}$ in each equivalence class. This would allow the customer to be subjective about the state of the system for each $\mathbf{x}$. Second, the term "reliable" could be thought of as a fuzzy set. A membership function derived from the customer would be used to determine a component's degree of membership in the set of "reliable" components by combining substitute characteristics for reliability.

7.1.2 Reliability Polynomial

The definition of the reliability function for the binary model was given in section 2.2.2. If the components are mutually independent, $r = r(\mathbf{p})$ where $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $p_i = \Pr[X_i = 1]$ for $i=1,2,\ldots,n$. For the special case when $p_1 = p_2 = \ldots = p_n = p$, the reliability function is called the reliability polynomial and is denoted by $r(p)$. Barlow and Proschan [1981] wrote the reliability polynomial for a k-out-of-n system as

$$r(p) = \sum_{i=k}^{n} \binom{n}{i} p^{i} (1-p)^{n-i} .$$

Thus, the reliability polynomial for a series (n-out-of-n) system is $r(p) = p^n$ and for a parallel (1-out-of-n) system is $r(p) = 1 - (1 - p)^n$.

A more general form of the reliability polynomial for the binary model is shown next. Again, the independent components have the same reliability, p. Let $C_{n,k}$ designate the set of component state vectors with k components working and (n - k) components failed. The cardinality of $C_{n,k}$, denoted by $|C_{n,k}|$, is $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$. Let A_k denote the set of component state vectors in $C_{n,k}$ that cause the system to work. Let $|A_k|$ denote the cardinality of A_k. Barlow and Iyer [1988] wrote the general reliability polynomial as

$$r(p) = \sum_{k=0}^{n} |A_k| \, p^{k} (1-p)^{n-k} . \qquad (7.1)$$

Yao [1991] transformed Equation (7.1) to express the reliability polynomial directly in polynomial form:

$$r(p) = \sum_{k=0}^{n} D_k \, p^k \qquad (7.2)$$

where $D_k = \sum_{i=0}^{k} |A_i| \, |C_{n-i,k-i}| \, (-1)^{k-i}$.

EXAMPLE 7.1 For a binary system of 3 independent components, each with reliability p, determine the reliability polynomial for the structure given in Figure 7.1 directly from the structure function and using the Equations (7.1) and (7.2).

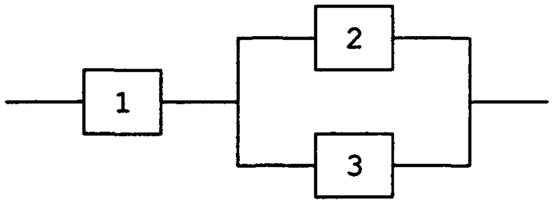


Figure 7.1 Structure for Example 7.1.

$\phi(\mathbf{x}) = x_1 \, (1 - (1 - x_2)(1 - x_3))$

$= x_1 \, (1 - (1 - x_2 - x_3 + x_2x_3))$

$= x_1 \, (x_2 + x_3 - x_2x_3) = x_1x_2 + x_1x_3 - x_1x_2x_3.$

$r(p) = E[\phi(\mathbf{x})] = 2p^2 - p^3.$

$C_{3,0} = \{(0,0,0)\}$	$A_0 = \emptyset$
$C_{3,1} = \{(1,0,0),(0,1,0),(0,0,1)\}$	$A_1 = \emptyset$
$C_{3,2} = \{(1,1,0),(1,0,1),(0,1,1)\}$	$A_2 = \{(1,1,0),(1,0,1)\}$
$C_{3,3} = \{(1,1,1)\}$	$A_3 = \{(1,1,1)\}$

Using Equation (7.1),

$r(p) = 0p^0(1-p)^3 + 0p(1-p)^2 + 2p^2(1-p) + p^3(1-p)^0$

$= 2p^2 - 2p^3 + p^3$

$= 2p^2 - p^3.$

Using Equation (7.2),

$D_0 = 0,\ D_1 = 0 + 0 = 0,$

$D_2 = 0 + 0 + 2 \binom{1}{0} (-1)^0 = 2,$ and

$D_3 = 0 + 0 + 2 \binom{1}{1} (-1)^1 + 1 \binom{0}{0} (-1)^0 = -1.$

$r(p) = 0 + 0p + 2p^2 - p^3.$

Essentially, this method groups together the component state vectors which have the same number of working components.

Some preliminary work has been done on an equivalent derivation for the multistate reliability model based on the multinomial distribution. Paralleling the binary model, the components must be mutually independent with an identical probability distribution, $\mathbf{p} = (p_0, p_1, \ldots, p_M)$. Thus, all components have the same number of states. Let the random variables X_j, $j=0,1,\ldots,M$, represent the number of components in state j. The probability of x_0 components in state 0, x_1 components in state 1, ..., and x_M components in state M is given by the following multinomial distribution:

$$\binom{n}{x_0, x_1, \ldots, x_M} p_0^{x_0}\, p_1^{x_1} \cdots p_M^{x_M}$$

where $x_0+x_1+\ldots+x_M = n$ and $p_0+p_1+\ldots+p_M = 1$. This method groups together the component state vectors which have the same number of components in each of the states.

Let $C_{n, x_1, x_2, \ldots, x_M}$ designate the set of component state vectors with x_i components in state i, $i=1,2,\ldots,M$, and x_0 =

$n-(x_1+x_2+\ldots+x_M)$ components in state 0. The cardinality of $C_{n,x_1,x_2,\ldots,x_M}$, denoted by $|C_{n,x_1,x_2,\ldots,x_M}|$, is determined from $\binom{n}{x_0,x_1,\ldots,x_M} = \frac{n!}{x_0!\,x_1!\ldots x_n!}$. Let $A^k_{x_1,x_2,\ldots,x_M}$ denote the set of component state vectors in $C_{n,x_1,x_2,\ldots,x_M}$ that cause the system to be in state k. Let $|A^k_{x_1,x_2,\ldots,x_M}|$ denote the cardinality of $A^k_{x_1,x_2,\ldots,x_M}$. Then the multinomial which gives the system's probability distribution is

$$r(k;\mathbf{p}) = \sum_{x_1=0}^{n} \sum_{x_2=0}^{n-x_1} \cdots \sum_{x_M=0}^{n-\sum_{i=1}^{M-1} x_i} |A^k_{x_1,x_2,\ldots,x_M}|\; p_0^{\,n-\sum_{i=1}^{M} x_i}\, p_1^{x_1} \cdots p_M^{x_M} \quad (7.3)$$

for system states k=0,1,2,...,N. When M = N = 1, the multinomial in Equation (7.3) reduces to Equation (7.1).

EXAMPLE 7.2 For a multistate system of 3 mutually independent components, each with probability distribution $\mathbf{p} = (p_0,p_1,p_2)$, determine the multinomial for $\phi(\mathbf{x}) = \text{Min}\{x_1,x_2,x_3\}$ using lower boundary points and Equation (7.3).

The lower boundary point to level 2 is (2,2,2) and the lower boundary point to level 1 is (1,1,1). Therefore, $Q_2 = Q_{12}Q_{22}Q_{32}$ and $Q_1 = Q_{11}Q_{21}Q_{31}$.

By subtracting,

$P_2 = Q_2 - 0 = (p_2)^3$,

$P_1 = Q_1 - Q_2 = (p_1 + p_2)^3 - (p_2)^3$

$\quad = 3(p_1)(p_2)^2 + 3(p_1)^2(p_2) + (p_1)^3$

$P_0 = 1 - (p_2)^3 - (p_1 + p_2)^3$

$C_{3,0,0} = \{(0,0,0)\}$

$C_{3,0,1} = \{(0,0,2),(0,2,0),(2,0,0)\}$

$C_{3,0,2} = \{(0,2,2),(2,0,2),(2,2,0)\}$

$C_{3,0,3} = \{(2,2,2)\}$

$C_{3,1,0} = \{(0,0,1),(0,1,0),(1,0,0)\}$

$C_{3,1,1} = \{(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)\}$

$C_{3,1,2} = \{(1,2,2),(2,1,2),(2,2,1)\}$

$C_{3,2,0} = \{(0,1,1),(1,0,1),(1,1,0)\}$

$C_{3,2,1} = \{(1,1,2),(1,2,1),(2,1,1)\}$ and

$C_{3,3,0} = \{(1,1,1)\}$.

$|A^2_{0,3}| = 1$, $|A^1_{1,2}| = 3$, $|A^1_{2,1}| = 3$, and $|A^1_{3,0}| = 1$.

Using Equation (7.3),

$r(2,\mathbf{p}) = (p_2)^3$,

$r(1,\mathbf{p}) = 3(p_1)(p_2)^2 + 3(p_1)^2(p_2) + (p_1)^3$, and

$r(0,\mathbf{p}) = 1 - r(2;\mathbf{p}) - r(1:\mathbf{p})$ which agrees with the results found with the lower boundary points.

Additional research is required to write the multinomial in terms of the performance distribution and to use the boundary points to determine $A^k_{x_1, x_2, \ldots, x_n}$.

7.1.3 Expected Loss

In section 5.5, expected loss was used to introduce an innovative new substitute characteristic for reliability. A loss function was developed which was sensitive to the pattern of degradation about the desired system lifetime, t^*. Although the expected loss was discussed with respect to the system states, the expected loss can also be found for each individual component. Additional research is required to determine the expected loss of the system directly from the expected loss of the components.

Some preliminary results have been derived for the multistate model when the loss function has a fixed rate of increase. Suppose that the loss function for the system is given by $L = M - \phi(\mathbf{x})$ for $\phi(\mathbf{x}) = 0,1,\ldots,M$ and that the loss function for component i is given by $L_i = M_i - x_i$ for $x_i = 0,1,\ldots,M_i$. Let $\mathfrak{L}$ and $\mathfrak{L}_i$ represent the expected loss for the system and component i, respectively. It can be easily shown that $\mathfrak{L} = M - E[\phi(\mathbf{X})]$ and $\mathfrak{L}_i = M_i - E[X_i]$.

Suppose a series structure is defined as $\phi_s(\mathbf{x}) = \prod_{i=1}^{n} x_i$.

Then $E[\phi_s] = \prod_{i=1}^{n} E[X_i] = \prod_{i=1}^{n} (M_i - (M_i - E[X_i])) = \prod_{i=1}^{n} (M_i - \mathfrak{L}_i)$.

Therefore, $M - E[\phi_s] = \mathfrak{L}_s = M - \prod_{i=1}^{n} (M_i - \mathfrak{L}_i)$. Suppose that a parallel structure is defined as $\phi_p(\mathbf{x}) = M - \prod_{i=1}^{n} (M_i - x_i)$. Then $E[\phi_p] = M - \prod_{i=1}^{n} (M_i - E[X_i]) = M - \prod_{i=1}^{n} \mathfrak{L}_i$. Therefore, $M - E[\phi_p] = \mathfrak{L}_p = \prod_{i=1}^{n} \mathfrak{L}_i$. For the special case of the binary model, when $M_1 = M_2 = \ldots = M_n = M = 1$, $\mathfrak{L}_s = 1 - \prod_{i=1}^{n} (1 - \mathfrak{L}_i)$ and $\mathfrak{L}_p = \prod_{i=1}^{n} \mathfrak{L}_i$.

EXAMPLE 7.3 Suppose that n=2, $M_1=3$, $M_2=2$, M=6, and

$\phi_s(\mathbf{x}) = x_1x_2$ as shown in Table 7.1.

Table 7.1 $\phi_s(\mathbf{x})$ for Example 7.3.

		x_2		
		0	1	2
x_1	0	0	0	0
	1	0	1	2
	2	0	2	4
	3	0	3	6

Suppose that the probability distributions for the mutually independent components are:

$P_{10} = .1 \quad P_{11} = .2 \quad P_{12} = .3 \quad P_{13} = .4$

$P_{20} = .1 \quad P_{21} = .7 \quad P_{22} = .2$

Show that if $L_i = M_i - x_i$ for $i=1,2$ and $L_s = M - \phi(\mathbf{x})$, then $\mathcal{L}_s = M - (M_1 - \mathcal{L}_1)(M_2 - \mathcal{L}_2)$.

$E[X_1] = 0(.1)+1(.2)+2(.3)+3(.4) = 2$ and $\mathcal{L}_1 = M_1 - E[X_1] = 3 - 2 = 1$. $E[X_2] = 0(.1)+1(.7)+2(.2) = 1.1$ and $\mathcal{L}_2 = M_2 - E[X_2] = 2 - 1.1 = .9$. Thus $\mathcal{L}_s = 6 - (3 - 1)(2 - .9) = 3.8$.

Checking this result directly from the system,

$\mathcal{L}_s = E[L] = 6(.19)+5(.14)+4(.25)+3(.28)+2(.06) = 3.8$.

EXAMPLE 7.4 Suppose that $n=2$, $M_1=3$, $M_2=2$, $M=6$, and $\phi_p(\mathbf{x}) = 6 - (3 - x_1)(2 - x_2)$ as shown in Table 7.2.

Table 7.2 $\phi_p(\mathbf{x})$ for Example 7.4.

		x_2		
		0	1	2
x_1	0	0	3	6
	1	2	4	6
	2	4	5	6
	3	6	6	6

Suppose that the probability distributions for the mutually independent components are the same as in the previous example. Show that if $L_i = M_i - x_i$ for i=1,2 and $L_p = M - \phi(\mathbf{x})$, then $\mathcal{L}_p = \mathcal{L}_1\mathcal{L}_2$.

As before, $E[X_1] = 2$, $\mathcal{L}_1 = 1$, $E[X_2] = 1.1$, and $\mathcal{L}_2 = .9$.

Therefore, $\mathcal{L}_p = (1)(.9) = 0.9$.

Checking this result directly from the system,

$\mathcal{L}_p = E[L] = 6(.01)+4(.02)+3(.07)+2(.17)+1(.21) = 0.9$.

Other results have been derived for the multistate model when the loss function has a faster rate of increase. Let the loss function for the system be given by $L = (M - \phi(\mathbf{x}))^2$ for $\phi(\mathbf{x}) = 0,1,\ldots,M$ and the loss function for component i be given by $L_i = (M_i - x_i)^2$ for $x_i = 0,1,\ldots,M_i$. It can be easily shown that $\mathcal{L} = (M - E[\phi(\mathbf{X})])^2 + Var[\phi(\mathbf{X})]$ and that $\mathcal{L}_i = (M_i - E[X_i])^2 + Var[X_i]$.

No relationship between the expected loss of the components and the expected loss of the system has been discovered for a series structure defined by $\phi_s(\mathbf{x}) = \prod_{i=1}^{n} x_i$ using the quadratic loss functions. However, a relationship has been found for a parallel structure defined by $\phi_p(\mathbf{x}) = M - \prod_{i=1}^{n}(M_i - x_i)$. Rearranging terms and squaring both sides gives $(M - \phi_p)^2 = \prod_{i=1}^{n}(M_i - x_i)^2$. Taking the expected value this equation results in $\mathcal{L}_p = \prod_{i=1}^{n}\mathcal{L}_i$.

EXAMPLE 7.5 For the problem given in Example 7.4 and the probability distributions given in Example 7.3, show that if $L_i = (M_i - x_i)^2$ for i=1,2 and $L_p = (M - \phi(\mathbf{x}))^2$, then $\mathcal{L}_i = (M_i - E[X_i])^2 + Var[X_i]$ for i=1,2 and $\mathcal{L}_p = \mathcal{L}_1\mathcal{L}_2$.

$E[X_1] = 1$ and $Var[X_1] = 1$. So $\mathcal{L}_1 = (3 - 2)^2 + 1 = 2$ which checks with $\mathcal{L}_1 = 9(.1)+4(.2)+1(.3) = 2$.

$E[X_2] = 1.1$ and $Var[X_2] = .29$. So $\mathcal{L}_2 = (2 - 1.1)^2 + .29 = 1.1$ which checks with $\mathcal{L}_2 = 4(.1)+1(.7) = 1.1$.

$\mathcal{L}_p = 36(.01)+16(.02)+9(.07)+4(.17)+1(.21) = 2.2$. Thus, $\mathcal{L}_p = 2.2 = (2)(1.1) = \mathcal{L}_1\mathcal{L}_2$.

Further research is needed to find the relationship between the expected loss of the components and the expected loss of the system for the loss function proposed in section 5.5 that is sensitive to the pattern of degradation about the desired system lifetime, t^*.

7.1.4 Reliability Estimation

Life testing is the process of finding point estimates or confidence intervals for the parameter(s) of a chosen failure distribution. Kapur and Lamberson [1977] discussed the choice of the exponential distribution,

$$f(t;\lambda) = \lambda \exp(-\lambda t) \qquad t \geq 0,\ \lambda > 0$$

as an appropriate failure model. The maximum likelihood estimator for λ is given by $\hat{\lambda} = \frac{r}{T}$ where r is the total number of failures and T is the total time on test. If n items are tested for a prespecified time (t^*) and failed items are replaced, then $T = nt^*$. For this time truncated test, the $100(1-\alpha)\%$ two-sided confidence interval for λ is

$$\frac{\chi^2_{1-\alpha/2,2r}}{2T} \leq \lambda \leq \frac{\chi^2_{\alpha/2,2(r+1)}}{2T} .$$

Note that the $100(1-\alpha)\%$ one-sided upper confidence interval for λ is still valid when no failures occur.

Kapur and Lamberson [1977] used the invariance property of maximum likelihood estimators to derive the point estimate

$$\hat{R}(t) = \exp(-\hat{\lambda}t)$$

for the reliability function and the $100(1-\alpha)\%$ two-sided confidence interval

$$\exp\left\{\frac{(-t)\,\chi^2_{\alpha/2,2(r+1)}}{2T}\right\} \leq R(t) \leq \exp\left\{\frac{(-t)\,\chi^2_{1-\alpha/2,2r}}{2T}\right\}$$

for the reliability function when $t \geq 0$.

Section 5.2 contained some preliminary work for testing multistate components by counting the number of state changes in a given time interval (t^*) and replacing a component upon entering state 0. Suppose that T_{ij} is the random variable for the time component i spends in state j, $i=1,2,\ldots,n$ and $j=1,2,\ldots,M_i$. Suppose that $T_{ij} \sim$ exponential with parameter λ_i for every $j=1,2,\ldots,M_i$. For a time truncated test, the maximum likelihood estimator for λ_i was shown to be $\hat{\lambda}_i = \frac{r}{nt^*}$ where r is the number of state changes and n is the number of multistate components on test. The method is valid since the memoryless property of the exponential distribution renews each component after every state change.

Confidence intervals for λ_i must still be developed. However, the invariance property of maximum likelihood estimators can be used to derive point estimates for the probability distribution. Let $P_{ij}(t^*) = \Pr[X_i(t^*) = j]$. Paralleling the work in section 5.5, when $T_{ij} \sim \exp(\lambda_i)$ for $j=1,2,\ldots,M_i$, then

$$P_{ij}(t^*) = \begin{cases} \dfrac{(\lambda_i t^*)^{M_i-j} \exp(-\lambda_i t^*)}{(M_i - j)!} & j = 1,2,\ldots,M_i \\ 1 - \displaystyle\sum_{j=1}^{M_i} P_{ij}(t^*) & j = 0. \end{cases} \tag{7.4}$$

Substituting $\hat{\lambda}_i$ in Equation (7.4) produces maximum likelihood estimators (MLEs) for the probability distribution.

EXAMPLE 7.6 Suppose that 6 identical multistate components are placed on test for 10 hours and 15 state changes are observed. Assuming $T_{ij} \sim \exp(\lambda_i)$ for $j=1,2,3$, determine the MLEs for the probability distribution of component i.

$\hat{\lambda}_i = 15/60 = .25$, $M_i = 3$, and $t^* = 10$.

Using Equation (7.4),

$$\hat{P}_{i3}(10) = \frac{(2.5)^0 \exp(-2.5)}{0!} = 0.0821,$$

$$\hat{P}_{i2}(10) = \frac{(2.5)^1 \exp(-2.5)}{1!} = 0.2052,$$

$\hat{P}_{11}(10) = \frac{(2.5)^2 \exp(-2.5)}{2!} = 0.2565$, and

$\hat{P}_{10}(10) = 1 - \hat{P}_{13}(10) - \hat{P}_{12}(10) - \hat{P}_{11}(10) = 0.4562$.

The MLEs for the probability distribution produce MLEs for the substitute characteristics for reliability.

EXAMPLE 7.7 Use the MLEs found in Example 7.6 to produce MLEs for $E[X_1]$, $Var[X_1]$, and $E[L]$.

$\hat{E}[X_1] = 1(.2565)+2(.2052)+3(.0821) = .9132,$

$\hat{E}[X_1^2] = 1(.2565)+4(.2052)+9(.0821) = 1.8162,$

$\hat{\sigma}^2_{X_1} = 1.8162 - (.9132)^2 = .9883.$

Using $c_1 = 50$, $c_2 = 5$, $c_3 = 1$, and the last lost function given in section 5.5, $\hat{E}[L] = \$573.68$.

Further research is necessary to relax the assumptions made for the distribution of T_{ij}.

7.2 Summary

This section provides a summary of the contributions given in the dissertation. A literature review was conducted to study the existing structural, stochastic, and dynamic properties of the binary, multistate, and continuous models.

Structural, stochastic, and dynamic properties were developed for the general multistate reliability model after the model was modified with a new definition for component relevance. New structural properties included a definition for a k-out-of-n structure, a definition for a general MCS, bounds on the structure function, alternate representations for $\phi(\mathbf{x})$, proofs for many duality theorems, and a measure of structural importance. New stochastic properties included a generalization for reliability importance, a program that determines the exact probability distribution for the system, a program that calculates performance bounds for more complex systems, and a program that converts boundary points.

The continuous model was expanded to allow a different range of states for the components and the system. New structural properties included a definition for a k-out-of-n structure, a definition for a general CSF, an alternate representation for $\phi(\mathbf{x})$, proofs for many duality theorems, and a measure of structural importance. New stochastic and dynamic properties were not developed because the continuous model resulted in an infinite number of boundary points.

Next, the voice of the customer was incorporated into the general multistate reliability model. A method for state classification was developed to allow the customer to define the number of system and component states. A technique using the convolution of random variables was devised to estimate the component state probabilities. A procedure for obtaining boundary points from the customer was designed to determine the customer's definition of the system. A computer program was written to estimate the system state probabilities. The expected loss was introduced as a substitute characteristic for reliability and a computer program was written to find the expected loss.

Several applications were given to demonstrate the customer-driven reliability model. Finally, some preliminary results were presented for fuzzy sets, the reliability polynomial, expected loss, and reliability estimation.

7.3 Conclusions

The binary model is the most commonly used reliability model. However, most components and systems do not progress directly from a working state to a failed state. Instead, the state degrades through a continuum of values which is better represented by a continuous model. Unfortunately, the continuous model results in an overwhelming number of calculations, making the model impractical for all but the simplest structures.

The general multistate model is a sensible compromise between the continuous and binary models. The model provides more information than the binary model and requires less calculations than the continuous model. Allowing a different number of states for each component and the system makes the model more appealing to the customer. Involving the customer at every step in the development and evaluation of the multistate model increases the creaibility and usefulness of the model.

LITERATURE CITED

General References:

Clarke, A. and Disney, R. (1970), Probability and Random Processes for Engineers and Scientists, John Wiley and Sons: New York, NY.

Gitlow, H., Gitlow, S., Oppenheim, A., and Oppenheim, R. (1989), Tools and Methods for the Improvement of Quality, Irwin: Boston, MA.

Binary Model:

Barlow, R. E. and Iyer, S. (1988), "Computational Complexity of Coherent Systems and the Reliability Polynomial," Probability in the Engineering and Informational Sciences, 2, 461-469.

Barlow, R. E. and Proschan, F. (1981), Statistical Theory of Reliability and Life Testing Probability Models, TO BEGIN WITH: Silver Spring, MD.

Birnbaum, Z. W., Esary, J. D. and Saunders, S. C. (1961), "Multi-Component Systems and Structures and Their Reliability," Technometrics, 3, 1, 55-77.

Birnbaum, Z. W. and Esary, J. D. (1965), "Modules of Coherent Binary Systems," SIAM Journal on Applied Mathematics, 13, 2, 444-462.

Birnbaum, Z. W. (1969), "On the Importance of Different Components in a Multi-Component System," In Multivariate Analysis II, P. R. Krishnaiah, editor, Academic Press: New York, NY, 581-592.

Bodin, L. D. (1970), "Approximations to System Reliability Using a Modular Decomposition," Technometrics, 12, 2, 335-344.

Esary, J. D. and Proschan, F. (1963a), "Coherent Structures of Non-Identical Components," Technometrics, 5, 2, 191-209.

Esary, J. D. and Proschan, F. (1963b), "Relationship Between the System Failure Rate and Component Failure Rate," Technometrics, 5, 2, 183-189.

Esary, J. D. and Proschan, F. (1970), "A Reliability Bound for Systems of Maintained, Interdependent Components," Journal of the American Statistical Society, 65, 329 - 338.

Esary, J. D., Marshall, A. W., and Proschan, F. (1970), "Some Reliability Applications of the Hazard Transform," SIAM Journal on Applied Mathematics, 18, 4, 849-860.

Feller, W. (1968), An Introduction to Probability Theory and Its Applications, 3rd edition, John Wiley and Sons: New York, NY.

Kaleva, O. (1986), "Fuzzy Performance of a Coherent System," Journal of Mathematical Analysis and Applications," 117, 1, 234-246.

Kapur, K. C. and Lamberson, L. R. (1977), Reliability in Engineering, John Wiley and Sons: New York, NY.

Park, K. S. (1987), "Fuzzy Set Apportionment of System Reliability," IEEE Transactions on Reliability, R-36, 1, 129-132.

Ross, S. M. (1989), Introduction To Probability Models, 4th edition, Academic Press: San Diego, CA.

Yao, Z. (1991), "Introduction to the Theory of General-System Reliability Functions," Submitted for publication in IEEE Transactions on Reliability.

Multistate Model:

Abouammoh, A. M. and Al-Kadi, M. A. (1991), "Component Relevancy in Multistate Reliability Models," IEEE Transactions on Reliability, 40, 3, 370-374.

Barlow, R. E. and Wu, A. S. (1978), "Coherent Systems With Multi-State Components," Mathematics of Operations Research, 3, 4, 275-281.

Block, H. W. and Savits, T. H. (1982), "A Decomposition For Multistate Monotone Systems," Journal of Applied Probability, 19, 2, 391-402.

Borges, W. D. S. and Rodrigues, F. W. (1983), "An Axiomatic Characterization of Multistate Coherent Structures," Mathematics of Operations Research, 8, 3, 435-438.

Butler, D. A. (1979), "A Complete Importance Ranking For Components of Binary Coherent Systems, With Extensions To Multi-State Systems," Naval Research Logistics Quarterly, 26, 4, 565-578.

Butler, D. (1982), "Bounding the Reliability of Multistate Systems," Operations Research, 30, 3, 530-544.

El-Neweihi, E., Proschan, F. and Sethuraman, J. (1978), "Multistate Coherent Systems," Journal of Applied Probability, 15, 4, 675-688.

Fardis, M. N. and Cornell, C. A. (1981), "Analysis of Coherent Multistate Systems," IEEE Transactions on Reliability, R-30, 2, 117-122.

Griffith, W. S. (1980), "Multistate Reliability Models," Journal of Applied Probability, 17, 3, 735-744.

Hudson, J. C. (1981), The Structure and Reliability of Multistate Systems with Multistate Components, Ph.D. Dissertation, Department of Industrial Engineering, Wayne State University, Detroit, Michigan.

Hudson, J. C. and Kapur, K. C. (1983a), "Modules in Coherent Multistate Systems," IEEE Transactions on Reliability, R-32, 2, 183-185.

Hudson, J. C. and Kapur, K. C. (1983b), "Reliability Analysis for Multistate Systems with Multistate Components," IIE Transactions, 15, 2, 127-135.

Hudson, J. C. and Kapur, K. C. (1985), "Reliability Bounds for Systems with Multistate Components," Operations Research, 33, 1, 153-160.

Iyer, S. (1989), "Exact Reliability Computation for Multistate Coherent Systems," For the SQC Unit, Indian Statistical Institute, Bombay, India 400 020.

Janan, X. (1985), "On Multistate Systems Analysis," IEEE Transactions on Reliability, R-34, 4, 329-337.

Natvig, B. (1982), "Two Suggestions of How To Define a Multistate Coherent System," Advances in Applied Probability, 14, 2, 434-455.

Ohi, F. and Nishida T. (1984), "On Multistate Coherent Systems," IEEE Transactions on Reliability, R-33, 4, 284-287.

Ross, S. M. (1979), "Multivalued State Component Systems," The Annals of Probability, 7, 2, 379-383.

Wood, A. P. (1985), "Multistate Block Diagrams and Fault Trees," IEEE Transactions on Reliability, R-34, 3, 236-240.

Continuous Model:

Baxter, L. A. (1984), "Continuum Structures I," Journal of Applied Probability, 21, 4, 802-815.

Baxter, L. A. (1986), "Continuum Structures II," Mathematical Proceedings of the Cambridge Philosophical Society," 99, 2, 331-338.

Baxter, L. A. and Kim, C. (1986), "Bounding The Stochastic Performance Of Continuum Structure Functions. I," Journal of Applied Probability, 23, 3, 660-669.

Block, H. W. and Savits, T. H. (1984), "Continuous Multistate Structure Functions," Operations Research, 32, 3, 703-714.

Montero, J., Tejada, J. and Yáñez, J. (1990), "Structural Properties of Continuum Systems," European Journal of Operational Research, 45, 2-3, 231-240.

BIBLIOGRAPHY

Abraham, J. A. (1979), "An Improved Algorithm for Network Reliability," IEEE Transactions on Reliability, R-28, 1, 58-61.

Almassy, G. (1979), "Limits of Models in Reliability Engineering," Proceedings of the 1979 Annual Reliability and Maintainability Symposium, 364-367.

Aven, T. (1985), "Reliability Evaluation of Multistate Systems with Multistate Components," IEEE Transactions On Reliability, R-34, 5, 473-479.

Bossche, A. (1987), "Calculation of Critical Importance for Multi-State Components," IEEE Transactions On Reliability, R-36, 2, 247-249.

Burdick, G. R., Fussell, J. B., Rasmuson, F. M. and Wilson, J. R. (1977), "Phased Mission Analysis: A Review of New Developments and An Application," IEEE Transactions on Reliability, R-26, 1, 43-49.

Cafaro, G., Corsi F. and Vacca F. (1986), "Multistate Markov Models and Structural Properties of the Transition-Rate Matrix," IEEE Transactions On Reliability, R-35, 2, 192-200.

Doulliez, P. and Jamoulle (1972), "Transportation Networks With Random Arc Capacities," revue francaise d' Automatique Informatique Recherche Operationnelle, 6 anneé, v-3, Novembre, 45-59.

Ebrahimi, N. (1984), "Multistate Reliability Models," Naval Research Logistics Quarterly, 31, 4, 671-680.

El-Neweihi, E. (1980), "Multistate Reliability Models: A Survey," Air Force Office of Scientific Research (AFOSR) Technical Report No. 76-30501, University of Illinois, Department of Mathematics, Champaign-Urbana, IL.

Elsayed, E. A. and Zebib, A. (1979), "A Reparable Multistate Device," IEEE Transactions on Reliability, R-28, 1, 81-82.

Esary, J. D. and Marshall, A. W. (1970), "Coherent Life Functions," SIAM Journal on Applied Mathematics, 18, 4, 810-814.

Funnemark, E. and Natvig, B. (1985), "Bounds for the Availabilities In A Fixed Interval For Multistate Monotone Systems," Advances in Applied Probability, 17, 3, 638-665.

Garg, R. C. and Kumar, A. (1977), "A Complex System with Two Types of Failure & Repair," IEEE Transactions on Reliability, R-26, 4, 299-300.

Golomb, S. W. (1971), "Mathematical Models: Uses and Limitations," IEEE Transactions on Reliability, R-20, 3, 130-131.

Gopol, K., Aggarwal, K. K. and Gupta, J. S. (1978), "Reliability Analysis of Multistate Device Networks," IEEE Transactions on Reliability, R-27, 3, 233-235.

Hatoyama, Y. (1979), "Reliability Analysis of 3-State Systems," IEEE Transactions on Reliability, R-28, 5, 386-393.

Hjort, N. L., Natvig, B. and Funnemark, E. (1985), "The Association In Time Of A Markov Process With Application To Multistate Reliability Theory," Journal of Applied Probability, 22, 2, 473-479.

Hwang, F. K. and Yao, Y. C. (1989), "Multistate Consecutively-Connected Systems," IEEE Transactions on Reliability, R-38, 4, 472-474.

Iyer, R. K. and Downs, T. (1978), "A Moment Approach to Evaluation and Optimization of Complex System Reliability," IEEE Transactions on Reliability, R-27, 3, 226-229.

Kapur, K. C. (1975), "Optimization in Design by Reliability," AIIE Transactions, 7, 2, 185-192.

Kapur, K. C. (1988), "Product and Process Design Optimization by Design of Experiments Using Taguchi Methods," SAE Technical Paper Series No. 880821, Earthmoving Industry Conference, Peoria, Illinois.

Kapur, K. C. (1991), "Quality Improvement Through Robust Design," To be presented at the 1991 International Industrial Engineering Conference, May 20-22, Detroit, Michigan.

Kapur, P. K. and Kapoor, K. R. (1978), "Stochastic Behaviour of Some 2-Unit Redundant Systems," IEEE Transactions on Reliability, R-27, 5, 382-385.

Karpinski J. (1986), "A Multistate System Under an Inspection and Review Policy," IEEE Transactions on Reliability, R-35, 1, 76-77.

Mohamed, A. (1990), Multicriteria Optimization Applied to Multistate Repairable Components, Ph.D. Dissertation, School Of Industrial Engineering, The University of Oklahoma, Norman, Oklahoma.

Montero, J. (1991), "General Reliability Bounds: Some Comments," Submitted to the Journal of the Operational Research Society.

Moore, E. F. and Shannon, C. E. (1956), "Reliable Circuits Using Less Reliable Relays," Journal of the Franklin Institute, 262, 3 & 4, 191-208 & 281-298.

Natvig, B. and Streller, A. (1984), "The Steady-State Behaviour Of Multistate Monotone Systems," Journal of Applied Probability, 21, 4, 826-835.

Pedar, A and Sarma, V. V. S. (1981), "Phased-Mission Analysis for Evaluating the Effectiveness of Aerospace Computing-Systems," IEEE Transactions on Reliability, R-30, 5, 429-437.

Proctor, C. L. II and Proctor, C. L. (1977), "Multistate-Time Dependent System Modeling," Proceedings of the 1977 Annual Reliability and Maintainability Symposium, 401-403.

Sakawa, M. (1978), "Multiobjective Optimization by the Surrogate Worth Trade-Off Method," IEEE Transactions on Reliability, R-27, 5, 311-314.

Satyanarayana, A. and Chang, M. K. (1983), "Network Reliability and the Factoring Theorem," Networks, 13, 107-120.

Shao, J. and Kapur, K. C. (1989), "Multilevel Modular Decomposition for Multistate Systems", Proceedings of the 1989 Annual Reliability and Maintainability Symposium, 102-107.

Singh, B. and Proctor, C. L. (1976), "Reliability Analysis of Multistate Device Networks," Proceeding of the 1976 Annual Reliability and Maintainability Symposium, 31-35.

Appendix A. Exact System Performance Program

```
**********************************************************
*  WRITTEN BY:  Ralph Boedigheimer                       *
*  LAST UPDATE:  7 Oct 91                                *
**********************************************************

**********************************************************
*  This is the main program that runs all other programs *
*  and calculates reliability for a multistate system.   *
**********************************************************
      program main
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    answer - variable for interactive feedback          *
**********************************************************
      real answer
**********************************************************
*  The main menu is presented to the user.  One of the   *
*  given options must be selected.                       *
**********************************************************
5     answer=0.0
      do while ((answer.lt.1.0).or.(answer.gt.10.0).or.
     +(amod(answer,1.0).ne.0.0))
        print *,'ENTER SELECTION FROM THE FOLLOWING MENU:'
        print *,'  1.  INPUT A NEW SYSTEM DESCRIPTION.'
        print *,'  2.  DISPLAY THE CURRENT SYSTEM.'
        print *,'  3.  USE ENUMERATION.'
        print *,'  4.  USE LOWER BOUNDARY POINTS.'
        print *,'  5.  USE UPPER BOUNDARY POINTS.'
        print *,'  6.  USE DECOMPOSITION (LBPs) - AVEN.'
        print *,'  7.  USE DECOMPOSITION (UBPs) - AVEN.'
        print *,'  8.  USE DECOMPOSITION (LBPs) - IYER.'
        print *,'  9.  USE DECOMPOSITION (UBPs) - IYER.'
        print *,' 10.  EXIT THE PROGRAM.'
        read *,answer
        print *
      enddo
**********************************************************
*  The program routes to the appropriate subroutine and  *
*  then returns to the main menu.                        *
**********************************************************
      go to (10,20,30,40,50,60,70,80,90,100),answer
10    call system
      go to 5
20    call display
      go to 5
30    call enum
      go to 5
40    call lower
      go to 5
```

```
50     call upper
       go to 5
60     call decomplow
       go to 5
70     call decomphi
       go to 5
80     call declower
       go to 5
90     call decupper
       go to 5
100    stop
       end

**********************************************************
*  This program is used to enter a description of the    *
*  multistate system being studied.                      *
**********************************************************
       subroutine system
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*.                boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
**********************************************************
       common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
       integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
       real prob (100,0:100)
**********************************************************
*  The required information is interactively entered.    *
**********************************************************
       print *,'ENTER THE MAXIMUM SYSTEM STATE (1-100).'
       read *,msys
       print *
       print *,'ENTER THE NUMBER OF COMPONENTS (1-100).'
       read *,ncomp
       print *
       do 10 i=1,ncomp
         print *,'ENTER THE MAXIMUM STATE OF COMPONENT',i,
     +           '(1-100).'
         read *,m(i)
10     continue
       do 20 i=(ncomp+1),100
```

```
          m(i)=0
 20     continue
        m(0)=0
        do 40 i=1,ncomp
          print *
          do 30 j=0,m(i)
            print *,'FOR COMPONENT',i,'ENTER THE ',
      +             'PROBABILITY OF BEING IN STATE',j
            read *,prob(i,j)
 30       continue
 40     continue
        print *
        do 60 k=1,msys
          print *,'ENTER THE NUMBER OF LOWER BOUNDARY',
      +           ' POINTS TO LEVEL',k,'(1-100).'
          read *,s(k)
          print *
          do 50 j=1,s(k)
            print *,'FOR LEVEL',k,'ENTER LOWER BOUNDARY',
      +             ' POINT #',j
            read *,(lbp(i,j,k),i=1,ncomp)
 50       continue
          print *
 60     continue
        do 80 k=0,msys-1
          print *,'ENTER THE NUMBER OF UPPER BOUNDARY',
      +           ' POINTS TO LEVEL',k,'(1-100).'
          read *,t(k)
          print *
          do 70 j=1,t(k)
            print *,'FOR LEVEL',k,'ENTER UPPER BOUNDARY',
      +             ' POINT #',j
            read *,(ubp(i,j,k),i=1,ncomp)
 70       continue
          print *
 80     continue
        return
        end

**********************************************************
*  This program displays the description of the system.  *
**********************************************************
        subroutine display
**********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    msys - the maximum state of the system               *
*    ncomp - the number of components in the system       *
*    m(i) - the maximum state of component i              *
*    s(k) - number of lower boundary points to level k    *
*    lbp(i,j,k) - the ith element of the jth lower        *
*                 boundary point to level k               *
*    t(k) - number of upper boundary points to level k    *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                 boundary point to level k               *
*    prob(i,j) - probability of component i in state j    *
***********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real prob (100,0:100)
***********************************************************
*  The current system is displayed for the user.         *
***********************************************************
      print *,'Maximum System State:',msys
      print *
      print *,'Number of Components:',ncomp
      print *
      print *,'Component Max State Vector: (',
     +        (m(i),i=1,ncomp),')'
      print *
      do 10 i=1,ncomp
        print 99,'Component',i,'Probabilities:',
     +           (prob(i,j),j=0,m(i))
10    continue
      print *
      do 30 k=1,msys
        print *,'System Level',k
        do 20 j=1,s(k)
          print *,'Lower Boundary Point #',j,': (',
     +    (lbp(i,j,k),i=1,ncomp),')'
20      continue
        print *
30    continue
      print *
      do 50 k=0,msys-1
        print *,'System Level',k
        do 40 j=1,t(k)
          print *,'Upper Boundary Point #',j,': (',
     +    (ubp(i,j,k),i=1,ncomp),')'
40      continue
        print *
50    continue
      print *
99    format(a9,1x,i2,1x,a14,2x,100(f5.3,1x))
      return
      end

***********************************************************
*  This program enumerates all possible component state   *
*  vectors and determines the probability of the vector.  *
*  Then it determines the system state for the vector and*
*  tallies the overall probability for each system state.*
**********************************************************
      subroutine enum
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    philower - the system state with lower boundary pts.*
*    phiupper - the system state with upper boundary pts.*
*    nvec - total number of component state vectors      *
*    divider - variable used to change base              *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    x(i) - the ith element of a component state vector  *
*    answer - variable for interactive feedback          *
*    prob(i,j) - probability of component i in state j   *
*    pvec - probability of a component state vector      *
*    plev(k) - probability of system being in state k    *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,philower,phiupper,nvec,divider,
     +m(0:100),s(100),lbp(100,100,100),t(0:100),
     +ubp(100,100,0:100),x(100)
      real answer,prob(100,0:100)
      double precision pvec,plev(0:100)
      external philower,phiupper
**********************************************************
*  The probability of each system state is set to zero.  *
**********************************************************
      do 10 k=0,msys
        plev(k)=0.0
10    continue
**********************************************************
*  The total number of component state vectors is found. *
**********************************************************
      nvec=1
      do 20 i=1,ncomp
        nvec=nvec*(m(i)+1)
20    continue
**********************************************************
*  Determine whether lower or upper boundary points will *
*  be used to determine the state of the system.         *
**********************************************************
      answer=0.0
      do while ((answer.lt.1.0).or.(answer.gt.2.0).or.
     +(amod(answer,1.0).ne.0.0))
        print *,'ENTER SELECTION FROM THE FOLLOWING MENU:'
        print *,'  1.  USE LOWER BOUNDARY POINTS.'
        print *,'  2.  USE UPPER BOUNDARY POINTS.'
        read *, answer
        print *
      enddo
**********************************************************
*  The probability of each component state vector is     *
*  found.  The system state is determined and the        *
*  probability is tallied under that state.              *
**********************************************************
      do 40 k=0,(nvec-1)
        divider=1
        pvec=1.0
        do 30 i=1,ncomp
          divider=divider*(m(i-1)+1)
          x(i)=mod((int(k/divider)),(m(i)+1))
          pvec=pvec*prob(i,x(i))
30      continue
        if (answer.eq.1.0) then
          n=philower(x)
        elseif (answer.eq.2.0) then
          n=phiupper(x)
        endif
        plev(n)=plev(n)+pvec
40    continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
      do 50 k=0,msys
        print 99,'Probability of state',k,'is',plev(k)
50    continue
      print *
      print *
99    format(a20,1x,i2,1x,a2,1x,f8.6)
      return
      end

**********************************************************
*  The program can evaluate the structure function for a *
*  component state vector given the following:           *
*      1)  The maximum state of the system,              *
*      2)  The number of components in the system,       *
*      3)  The maximum state of each component, and      *
*      4)  The lower boundary points to level k.         *
**********************************************************
      function philower(x)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system       *
*    prod - system state of a binary structure            *
*    sum - contains the subscript of the binary vector    *
*    philower - the system state with lower boundary pts.*
*    temp - used to find the max of the binary function  *
*    m(i) - the maximum state of component i              *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower        *
*                 boundary point to level k               *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                 boundary point to level k               *
*    x(i) - the ith element of the component state vector*
*    y(i) - the ith element of the binary state vector   *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,prod,sum,philower,temp,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +x(100),y(10000)
**********************************************************
*  Convert the multistate vector to a binary vector.     *
**********************************************************
      k=0
      do 20 i=1,ncomp
        do 10 j=1,m(i)
          k=k+1
          if (x(i).ge.j) then
            y(k)=1
          else
            y(k)=0
          endif
10      continue
20    continue
**********************************************************
*  Evaluate the binary structure functions with the      *
*  binary vector for every level and sum to give the     *
*  desired result.                                       *
**********************************************************
      philower=0
      do 60 k=1,msys
        temp=0
        do 50 j=1,s(k)
          prod=1
          do 40 i=1,ncomp
            if (lbp(i,j,k).ne.0) then
              sum=0
              do 30 n=1,(i-1)
                sum=sum+m(n)
30            continue
              sum=sum+lbp(i,j,k)
```

```
                prod=prod*y(sum)
              endif
40          continue
            temp=max(temp,prod)
50        continue
          philower=philower+temp
60      continue
        return
        end

***********************************************************
*  The program can evaluate the structure function for a *
*  component state vector given the following:           *
*       1)  The maximum state of the system,             *
*       2)  The number of components in the system,      *
*       3)  The maximum state of each component, and     *
*       4)  The upper boundary points to level k.        *
***********************************************************
      function phiupper(x)
***********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    msys - the maximum state of the system               *
*    ncomp - the number of components in the system       *
*    prod - used to find the max of the binary function   *
*    sum - contains the subscript of the binary vector    *
*    phiupper - the system state with upper boundary pts.*
*    temp - system state of a binary structure            *
*    m(i) - the maximum state of component i              *
*    s(k) - number of lower boundary points to level k    *
*    lbp(i,j,k) - the ith element of the jth lower        *
*                 boundary point to level k               *
*    t(k) - number of upper boundary points to level k    *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                 boundary point to level k               *
*    x(i) - the ith element of the component state vector*
*    y(i) - the ith element of the binary state vector    *
***********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,prod,sum,phiupper,temp,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +x(100),y(10000)
***********************************************************
*  Convert the multistate vector to a binary vector.      *
***********************************************************
      k=0
      do 20 i=1,ncomp
        do 10 j=0,m(i)-1
          k=k+1
          if (x(i).gt.j) then
            y(k)=1
          else
            y(k)=0
          endif
10      continue
20    continue
**********************************************************
*  Evaluate the binary structure functions with the      *
*  binary vector for every level and sum to give the     *
*  desired result.                                       *
**********************************************************
      phiupper=0
      do 60 k=0,msys-1
        prod=1
        do 50 j=1,t(k)
          temp=0
          do 40 i=1,ncomp
            if (ubp(i,j,k).ne.m(i)) then
              sum=1
              do 30 n=1,(i-1)
                sum=sum+m(n)
30            continue
              sum=sum+ubp(i,j,k)
              temp=max(temp,y(sum))
            endif
40        continue
          prod=prod*temp
50      continue
        phiupper=phiupper+prod
60    continue
      return
      end

**********************************************************
*  This program determines the probability of each system*
*  state directly from the lower boundary points.        *
**********************************************************
      subroutine lower
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    cmblower - real function that find all combinations *
*    prob(i,j) - probability of component i in state j   *
*    sum - the sum of all combinatorial summations       *
*    plev(k) - probability of a system being in state k  *
*    cplev(k) - probability of system in state k or more *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real cmblower,prob(100,0:100)
      double precision sum,plev(0:100),cplev(0:100)
      external cmblower
**********************************************************
*  The known cumulative probabilities are entered.       *
**********************************************************
      cplev(0)=1.0
      cplev(msys+1)=0.0
**********************************************************
*  The cumulative probability of the system being in     *
*  state k or higher is found for every system state.    *
**********************************************************
      do 20 k=1,msys
        sum=0.0
        do 10 j=1,s(k)
          sum=sum+((-1)**(j+1))*cmblower(s(k),j,k)
10      continue
        cplev(k)=sum
20    continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
      do 30 k=0,msys
        plev(k)=cplev(k)-cplev(k+1)
        print 99,'Probability of state',k,'is',plev(k)
30    continue
      print *
      print *
99    format(a20,1x,i2,1x,a2,1x,f8.6)
      return
      end

**********************************************************
*  This program determines all possible combinations of  *
*  vectors to consider.  It is required for calculation  *
*  of the intersection of events in the inclusion-       *
*  exclusion formula.                                    *
**********************************************************
      function cmblower(n,r,k)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    ichange - the element that is changed               *
*    r - the number of vectors to choose                       *
*    n - the total number of vectors                           *
*    itop - maximum state of the intersection of vectors *
*    m(i) - the maximum state of component i                   *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower             *
*                 boundary point to level k                    *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper             *
*                 boundary point to level k                    *
*    vec(i) - the intersection vector                          *
*    lrg(i) - the largest vector in position i                 *
*    store - a temporary storage location                      *
*    cmbupper -variable used to return probability             *
*    prob(i,j) - probability of component i in state j   *
*    prod - probability of a component state vector            *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                                  *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,ichange,r,n,numb,itop,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +vec(0:100),lrg(0:100)
      real store,cmblower,prob(100,0:100)
      double precision prod,cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The sum of the probabilities of all combinations of   *
*  lower boundary points to level k taken r at a time is *
*  found.                                                *
**********************************************************
      do 30 i=0,r
        vec(i)=i
        lrg(i)=n-r+i
30    continue
      ichange=r
      cmblower=0.0
      do while (ichange.gt.0)
        ichange=r
        do 60 numb=(vec(ichange-1)+1),n
```

```
          vec(ichange)=numb
          prod=1.0
          do 50 i=1,ncomp
            itop=0
            do 40 j=1,r
              itop=max(lbp(i,vec(j),k),itop)
40          continue
            prod=prod*cprob(i,itop)
50        continue
          cmblower=cmblower+prod
60      continue
        do while ((vec(ichange-1)).eq.(lrg(ichange-1))
     +  .and.(ichange.gt.1))
          ichange=ichange-1
        enddo
        ichange=ichange-1
        vec(ichange)=vec(ichange)+1
        do 70 i=(ichange+1),r
          vec(i)=vec(i-1)+1
70      continue
      enddo
      return
      end

**********************************************************
*  This program determines the probability of each system*
*  state directly from the upper boundary points.        *
**********************************************************
      subroutine upper
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points at level 1   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                  boundary point to level k             *
*    cmbupper - real function that find all combinations *
*    prob(i,j) - probability of component i in state j   *
*    sum - the sum of all combinatorial summations       *
*    plev(k) - probability of a system being in state k  *
*    cplev(k) - probability of system in state k or less *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real cmbupper,prob(100,0:100)
      double precision sum,plev(0:100),cplev(-1:100)
      external cmbupper
**********************************************************
*  The known cumulative probabilities are entered.       *
**********************************************************
      cplev(msys)=1.0
      cplev(-1)=0.0
**********************************************************
*  The cumulative probability of the system being in     *
*  state k or higher is found for every system state.    *
**********************************************************
      do 20 k=0,msys-1
        sum=0.0
        do 10 j=1,t(k)
          sum=sum+((-1)**(j+1))*cmbupper(t(k),j,k)
10      continue
        cplev(k)=sum
20    continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
      do 30 k=0,msys
        plev(k)=cplev(k)-cplev(k-1)
        print 99,'Probability of state',k,'is',plev(k)
30    continue
      print *
      print *
99    format(a20,1x,i2,1x,a2,1x,f8.6)
      return
      end

**********************************************************
*  This program determines all possible combinations of  *
*  vectors to consider.  It is required for calculation  *
*  of the intersection of events in the inclusion-       *
*  exclusion formula.                                    *
**********************************************************
      function cmbupper(n,r,k)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    ichange - the element that is changed               *
*    r - the number of vectors to choose                 *
*    n - the total number of vectors                     *
*    ibot - the min state of the intersection of vectors *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of lower boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                 boundary point to level k               *
*    vec(i) - the intersection vector                     *
*    lrg(i) - the largest vector in position i            *
*    store - a temporary storage location                 *
*    cmbupper - variable used to return probability       *
*    prob(i,j) - probability of component i in state j    *
*    prod - probability of a component state vector       *
*    cprob(i,j) - probability of component i in state     *
*                 j or lower                              *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,ichange,r,n,numb,itop,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +vec(0:100),lrg(0:100)
      real store,cmbupper,prob(100,0:100)
      double precision prod,cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=0,m(i)
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The sum of the probabilities of all combinations of   *
*  upper boundary points to level k taken r at a time is *
*  found.                                                *
**********************************************************
      do 30 i=0,r
        vec(i)=i
        lrg(i)=n-r+i
30    continue
      ichange=r
      cmbupper=0.0
      do while (ichange.gt.0)
        ichange=r
        do 60 numb=(vec(ichange-1)+1),n
          vec(ichange)=numb
          prod=1.0
          do 50 i=1,ncomp
            ibot=m(i)
            do 40 j=1,r
              ibot=min(ubp(i,vec(j),k),ibot)
40          continue
            prod=prod*cprob(i,ibot)
```

```
50          continue
            cmbupper=cmbupper+prod
60        continue
          do while ((vec(ichange-1)).eq.(lrg(ichange-1))
     +    .and.(ichange.gt.1))
            ichange=ichange-1
          enddo
          ichange=ichange-1
          vec(ichange)=vec(ichange)+1
          do 70 i=(ichange+1),r
            vec(i)=vec(i-1)+1
70        continue
        enddo
        return
        end

**********************************************************
*  This program determines the probability of each system*
*  state by decomposition using lower boundary points.   *
**********************************************************
        subroutine decomplow
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                            *
*    prod - probability of a component state vector      *
*    plev(k) - probability of a system being in state k  *
*    cplev(k) - probability of system in state k or more *
**********************************************************
        common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
       +t(0:100),ubp(100,100,0:100),prob(100,0:100)
        integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
       +t(0:100),ubp(100,100,0:100)
        integer bo(100),b(100),bbo(100,1000),bb(100,1000),
       +vo(100),v(100),ymin(100),lo,ss,h1,h2,iter
        real prob(100,0:100),cprob(100,0:100)
        double precision prod,plev(0:100),cplev(0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
```

```
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The system is decomposed with Aven's algorithm.       *
**********************************************************
****** Step 1
      do 110 k=1,msys
        iter=1
        cplcv(k)=0.0
        do 30 i=1,ncomp
          bo(i)=m(i)
          ymin(i)=bo(i)
          b(i)=0
30      continue
****** Step 2
        do while (iter.ne.0)
          h1=-1
          lo=1
          do 60 j=1,s(k)
            iflag=0
            do 40 i=1,ncomp
              if (lbp(i,j,k).gt.bo(i)) then
                iflag=1
              endif
40          continue
            if (iflag.eq.0) then
              h2=0
              do 50 i=1,ncomp
                if(lbp(i,j,k).lt.ymin(i)) then
                  ymin(i)=lbp(i,j,k)
                endif
                h2=h2+bo(i)-max(lbp(i,j,k),b(i))
50            continue
              if (h2.gt.h1) then
                h1=h2
                lo=j
              endif
            endif
60        continue
****** Step 3 & 4
          prod=1.0
          do 70 i=1,ncomp
            v(i)=max(ymin(i),b(i))
            vo(i)=max(lbp(i,lo,k),b(i))
            prod=prod*(cprob(i,vo(i))-cprob(i,bo(i)+1))
70        continue
```

```
            cplev(k)=cplev(k)+prod
****** Step 5
            ss=0
            do 90 i=1,ncomp
              if (v(i).lt.vo(i)) then
                ss=ss+1
                do 80 j=1,ncomp
                  if (j.ne.i) then
                    bbo(j,ss+iter-1)=bo(j)
                  else
                    bbo(j,ss+iter-1)=vo(j)-1
                  endif
                  if (j.lt.i) then
                    bb(j,ss+iter-1)=vo(j)
                  else
                    bb(j,ss+iter-1)=v(j)
                  endif
80              continue
              endif
90          continue
            iter=ss+iter-1
****** Step 6
            do 100 i=1,ncomp
              bo(i)=bbo(i,iter)
              b(i)=bb(i,iter)
              ymin(i)=bo(i)
100         continue
          enddo
110     continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
        cplev(0)=1.0
        cplev(msys+1)=0.0
        do 120 k=0,msys
          plev(k)=cplev(k)-cplev(k+1)
          print 99,'Probability of state',k,'is',plev(k)
120     continue
        print *
        print *
99      format(a20,1x,i2,1x,a2,1x,f8.6)
        return
        end

**********************************************************
*  This program determines the probability of each system*
*  state by decomposition using upper boundary points.   *
**********************************************************
        subroutine decomphi
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system                  *
*    ncomp - the number of components in the system          *
*    m(i) - the maximum state of component i                 *
*    s(k) - number of lower boundary points to level k       *
*    lbp(i,j,k) - the ith element of the jth lower           *
*                  boundary point to level k                 *
*    t(k) - number of upper boundary points to level k       *
*    ubp(i,j,k) - the ith element of the jth upper           *
*                 boundary point to level k                  *
*    prob(i,j) - probability of component i in state j       *
*    cprob(i,j) - probability of component i in state        *
*                 j or higher                                *
*    prod - probability of a component state vector          *
*    plev(k) - probability of a system being in state k      *
*    cplev(k) - probability of system in state k or less     *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      integer bo(100),b(100),bbo(100,1000),bb(100,1000),
     +vo(100),v(100),ymax(100),lo,ss,h1,h2,iter
      real prob(100,0:100),cprob(100,0:100)
      double precision prod,plev(0:100),cplev(0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The system is decomposed with Aven's algorithm.       *
**********************************************************
****** Step 1
      do 110 k=0,msys-1
        iter=1
        cplev(k)=0.0
        do 30 i=1,ncomp
          bo(i)=0
          ymax(i)=bo(i)
          b(i)=m(i)
30      continue
****** Step 2
        do while (iter.ne.0)
          h1=-1
          lo=1
          do 60 j=1,t(k)
            iflag=0
            do 40 i=1,ncomp
              if (ubp(i,j,k).lt.bo(i)) then
                iflag=1
              endif
40          continue
            if (iflag.eq.0) then
              h2=0
              do 50 i=1,ncomp
                if(ubp(i,j,k).gt.ymax(i)) then
                  ymax(i)=ubp(i,j,k)
                endif
                h2=h2-bo(i)+min(ubp(i,j,k),b(i))
50            continue
              if (h2.gt.h1) then
                h1=h2
                lo=j
              endif
            endif
60        continue
****** Step 3 & 4
          prod=1.0
          do 70 i=1,ncomp
            v(i)=min(ymax(i),b(i))
            vo(i)=min(ubp(i,lo,k),b(i))
            prod=prod*(cprob(i,bo(i))-cprob(i,vo(i)+1))
70        continue
          cplev(k)=cplev(k)+prod
****** Step 5
          ss=0
          do 90 i=1,ncomp
            if (v(i).gt.vo(i)) then
              ss=ss+1
              do 80 j=1,ncomp
                if (j.ne.i) then
                  bbo(j,ss+iter-1)=bo(j)
                else
                  bbo(j,ss+iter-1)=vo(j)+1
                endif
                if (j.lt.i) then
                  bb(j,ss+iter-1)=vo(j)
                else
                  bb(j,ss+iter-1)=v(j)
                endif
80            continue
            endif
90        continue
          iter=ss+iter-1
****** Step 6
          do 100 i=1,ncomp
```

```
             bo(i)=bbo(i,iter)
             b(i)=bb(i,iter)
             ymax(i)=bo(i)
100        continue
         enddo
110    continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
       cplev(-1)=0.0
       cplev(msys)=1.0
       do 120 k=0,msys
         plev(k)=cplev(k)-cplev(k-1)
         print 99,'Probability of state',k,'is',plev(k)
120    continue
       print *
       print *
99     format(a20,1x,i2,1x,a2,1x,f8.6)
       return
       end

**********************************************************
*  This subroutine uses decomposition and lower boundary *
*  points to find the probability of each system state.  *
**********************************************************
       subroutine declower
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    wk1-4(i,j) - temporary working matrices             *
*    icnt(i) - used to store subproblem sizes            *
*    prob(i,j) - probability of component i in state j   *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                            *
*    cplev(k) - probability of system in state k or more *
**********************************************************
       common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
      +t(0:100),ubp(100,100,0:100),prob(100.0:100)
       integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
      +t(0:100),ubp(100,100,0:100),wk1(100,1000),
      +wk2(100,1000),wk2(100,1000),wk3(100,1000),
      +wk4(100,1000),icnt(0:100)
       real prob(100,0:100),cprob(100,0:100),cplev(0:100)
      common/ralph/cprob(100,0:100),rel
      external lexinc,calclow,divide
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description   *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  For each level k, the lower boundary points are used  *
*  to decompose the system into disjoint problems.       *
**********************************************************
      do 120 k=1,msys
        icomp=1
        rel=0.0
        do 40 i=1,ncomp
          do 30 j=1,s(k)
            wk1(i,j)=lbp(i,j,k)
30        continue
40      continue
        call lexinc(wk1,ncomp,s(k))
        call calclow(icomp,wk1,ncomp,s(k),wk2,jcnt)
        do while (jcnt.ge.1)
          call divide (wk2,icomp,jcnt,icnt,iter)
          jcnt=0
          icomp=icomp+1
          do 90 isub=1,iter+1
            do 60 i=1,ncomp
              do 50 j=icnt(isub-1),icnt(isub)-1
                wk3(i,j-icnt(isub-1)+1)=wk2(i,j)
50            continue
60          continue
            call calclow(icomp,wk3,ncomp,icnt(isub)-
     +                icnt(isub-1),wk4,newj)
            do 80 i=1,ncomp
              do 70 j=1,newj
                wk1(i,jcnt+j)=wk4(i,j)
70            continue
80          continue
            jcnt=jcnt+newj
90        continue
          do 110 i=1,ncomp
            do 100 j=1,jcnt
              wk2(i,j)=wk1(i,j)
100         continue
110       continue
```

```
         enddo
         cplev(k)=rel
120    continue
***********************************************************
*  The probability of each system state is printed.       *
***********************************************************
       cplev(0)=1.0
       cplev(msys+1)=0.0
       do 130 k=0,msys
         rlev=cplev(k)-cplev(k+1)
         print 99,'Probability of state',k,'is',rlev
130    continue
       print *
       print *
99     format(a20,1x,i2,1x,a2,1x,f8.6)
       return
       end

***********************************************************
*  This program sorts vectors lexicographically.          *
***********************************************************
       subroutine lexinc(wk,ihigh,jhigh)
***********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    wk - array of vectors to be sorted                   *
*    ihigh - the ith dimension of wk                      *
*    jhigh - the jth dimension of wk                      *
***********************************************************
       integer wk(100,1000)
***********************************************************
*  The vectors are sorted from the last element to the    *
*  first in increasing order with a bubble sort routine.  *
***********************************************************
       do 40 i1=ihigh,1,-1
         do 30 j1=1,jhigh-1
           num=jhigh-j1
           do 20 j2=1,num
             if (wk(i1,j2).gt.wk(i1,j2+1)) then
               do 10 i2=1,ihigh
                 itemp=wk(i2,j2)
                 wk(i2,j2)=wk(i2,j2+1)
                 wk(i2,j2+1)=itemp
10             continue
             endif
20         continue
30       continue
40     continue
       return
       end

***********************************************************
```

```
*  This program is used to branch the lower boundary     *
*  points into disjoint subproblems.                     *
**********************************************************
      subroutine calclow(icomp,wk1,ihigh,jhigh,wk2,jcnt)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    wk1-2(i,j) - temporary working matrices             *
*    icnt(i) - array used to store subproblem sized      *
*    prob(i,j) - probability of component i in state j   *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                            *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100.0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),wk1(100,1000),
     +wk2(100,1000),icnt(0:100)
      real prob(100,0:100),cprob(100,0:100)
      common/ralph/cprob(100,0:100),rel
      external lexinc,divide,elim
**********************************************************
*  Temporary lower boundary points are generated.        *
**********************************************************
      jwk1=1
      jwk2=1
      do 40 ifix=0,m(icomp)
        do while ((wk1(icomp,jwk1).eq.ifix).and.
     +           (jwk1.le.jhigh))
          do 10 i=1,ihigh
            wk2(i,jwk2)=wk1(i,jwk1)
10        continue
          jwk1=jwk1+1
          jwk2=jwk2+1
        enddo
        if (ifix.lt.m(icomp)) then
          do 30 j=1,jwk1-1
            do 20 i=1,ihigh
              if (i.eq.icomp) then
                wk2(i,jwk2)=ifix+1
              else
                wk2(i,jwk2)=wk1(i,j)
              endif
```

```
20          continue
            jwk2=jwk2+1
30        continue
        endif
40    continue
      jcnt=jwk2-1
**********************************************************
*  The problem is logically separated into subproblems.  *
**********************************************************
      call divide(wk2,icomp,jcnt,icnt,iter)
**********************************************************
*  Dominated lower boundary points are marked.           *
**********************************************************
      do 80 isub=0,iter
        do 70 j1=icnt(isub),icnt(isub+1)-2
          do 60 j2=j1+1,icnt(isub+1)-1
            iflag=0
            do 50 i=1,ihigh
              if (wk2(i,j1).ge.wk2(i,j2)) then
                iflag=iflag+1
              endif
50          continue
            if (iflag.eq.ihigh) then
              wk2(1,j1)=-1
            endif
60        continue
70      continue
80    continue
**********************************************************
*  Dominated lower boundary points are eliminated.       *
**********************************************************
      call elim(wk2,ncomp,jcnt)
**********************************************************
*  The problem is logically separated into subproblems.  *
**********************************************************
      call divide(wk2,icomp,jcnt,icnt,iter)
**********************************************************
*  Single lower boundary points are marked and the       *
*  probability of the lower boundary point is tallied.   *
**********************************************************
      do 110 isub=0,iter
        if ((icnt(isub+1)-icnt(isub)).eq.1) then
          prod=1.0
          do 90 i=1,icomp
            prod=prod*prob(i,wk2(i,icnt(isub)))
90        continue
          do 100 i=icomp+1,ihigh
            prod=prod*cprob(i,wk2(i,icnt(isub)))
100       continue
          rel=rel+prod
          wk2(1,icnt(isub))=-1
        endif
110   continue
**********************************************************
*  Single lower boundary points are eliminated.          *
**********************************************************
      call elim(wk2,ncomp,jcnt)
**********************************************************
*  The remaining problem is sorted lexicographically.    *
**********************************************************
      call lexinc(wk2,ihigh,jcnt)
      return
      end

      subroutine divide(wk,icomp,jcnt,icnt,iter)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    wk(i) - a temporary working matrix                  *
*    icomp - the component being pivoted on              *
*    jcnt - the jth dimension of wk                      *
*    icnt - an array used to store subproblem sizes      *
*    iter - the number of subproblems                    *
**********************************************************
      integer wk(100,1000),icnt(0:100)
      icnt(0)=1
      iter=0
      do 20 j=2,jcnt
        iflag=0
        do 10 i=1,icomp
          if (wk(i,j).eq.wk(i,j-1)) then
            iflag=iflag+1
          endif
10      continue
        if (iflag.ne.icomp) then
          iter=iter+1
          icnt(iter)=j
        endif
20    continue
      icnt(iter+1)=jcnt+1
      return
      end

      subroutine elim(wk,ihigh,jcnt)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    wk - a temporary working matrix                     *
*    ihigh - the ith dimension of wk                     *
*    jcnt - the jth dimension of wk                      *
**********************************************************
      integer wk(100,1000)
      do 30 j2=jcnt,1,-1
        if (wk(1,j2).eq.-1) then
```

```
            do 20 j1=j2,jcnt-1
              do 10 i=1,ihigh
                wk(i,j1)=wk(i,j1+1)
10            continue
20          continue
            jcnt=jcnt-1
          endif
30      continue
        return
        end

*********************************************************
*  This subroutine uses decomposition and upper boundary *
*  points to find the probability of each system state.  *
*********************************************************
        subroutine decupper
*********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    wk1-4(i,j) - temporary working matrices             *
*    icnt(i) - used to store subproblem sizes            *
*    prob(i,j) - probability of component i in state j   *
*    cprob(i,j) - probability of component i in state    *
*                 j or lower                             *
*    cplev(k) - probability of system in state k or less *
*********************************************************
        common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
       +t(0:100),ubp(100,100,0:100),prob(100.0:100)
        integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
       +t(0:100),ubp(100,100,0:100),wk1(100,1000),
       +wk2(100,1000),wk2(100,1000),wk3(100,1000),
       +wk4(100,1000),icnt(0:100)
        real prob(100,0:100),cprob(100,0:100),cplev(0:100)
        common/ralph/cprob(100,0:100),rel
        external lexdec,calchi,divide
*********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description   *
*********************************************************
        do 20 i=1,ncomp
          store=0.0
          do 10 j=0,m(i)
            cprob(i,j)=store+prob(i,j)
            store=cprob(i,j)
10        continue
20      continue
**********************************************************
*  For each level k, the upper boundary points are used  *
*  to decompose the system into disjoint problems.       *
**********************************************************
        do 120 k=0,msys-1
          icomp=1
          rel=0.0
          do 40 i=1,ncomp
            do 30 j=1,t(k)
              wk1(i,j)=ubp(i,j,k)
30          continue
40        continue
          call lexdec(wk1,ncomp,t(k))
          call calchi(icomp,wk1,ncomp,t(k),wk2,jcnt)
          do while (jcnt.ge.1)
            call divide (wk2,icomp,jcnt,icnt,iter)
            jcnt=0
            icomp=icomp+1
            do 90 isub=1,iter+1
              do 60 i=1,ncomp
                do 50 j=icnt(isub-1),icnt(isub)-1
                  wk3(i,j-icnt(isub-1)+1)=wk2(i,j)
50              continue
60            continue
              call calchi(icomp,wk3,ncomp,icnt(isub)-
     +                  icnt(isub-1),wk4,newj)
              do 80 i=1,ncomp
                do 70 j=1,newj
                  wk1(i,jcnt+j)=wk4(i,j)
70              continue
80            continue
              jcnt=jcnt+newj
90          continue
            do 110 i=1,ncomp
              do 100 j=1,jcnt
                wk2(i,j)=wk1(i,j)
100           continue
110         continue
          enddo
          cplev(k)=rel
120     continue
**********************************************************
*  The probability of each system state is printed.      *
**********************************************************
        cplev(-1)=0.0
        cplev(msys)=1.0
        do 130 k=0,msys
          rlev=cplev(k)-cplev(k-1)
          print 99,'Probability of state',k,'is',rlev
130     continue
        print *
        print *
99      format(a20,1x,i2,1x,a2,1x,f8.6)
        return
        end

**********************************************************
*  This program sorts vectors lexicographically.         *
**********************************************************
        subroutine lexdec(wk,ihigh,jhigh)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    wk - array of vectors to be sorted                  *
*    ihigh - the ith dimension of wk                     *
*    jhigh - the jth dimension of wk                     *
**********************************************************
        integer wk(100,1000)
**********************************************************
*  The vectors are sorted from the last element to the   *
*  first in decreasing order with a bubble sort routine. *
**********************************************************
        do 40 i1=ihigh,1,-1
          do 30 j1=1,jhigh-1
            num=jhigh-j1
            do 20 j2=1,num
              if (wk(i1,j2).lt.wk(i1,j2+1)) then
                do 10 i2=1,ihigh
                  itemp=wk(i2,j2)
                  wk(i2,j2)=wk(i2,j2+1)
                  wk(i2,j2+1)=itemp
10              continue
              endif
20          continue
30        continue
40      continue
        return
        end

**********************************************************
*  This program is used to branch the lower boundary     *
*  points into disjoint subproblems.                     *
**********************************************************
        subroutine calchi(icomp,wk1,ihigh,jhigh,wk2,jcnt)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower         *
*                 boundary point to level k                *
*    t(k) - number of upper boundary points to level k     *
*    ubp(i,j,k) - the ith element of the jth upper         *
*                 boundary point to level k                *
*    wk1-2(i,j) - temporary working matrices               *
*    icnt(i) - array used to store subproblem sized        *
*    prob(i,j) - probability of component i in state j     *
*    cprob(i,j) - probability of component i in state      *
*                 j or lower                               *
***********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100.0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),wk1(100,1000),
     +wk2(100,1000),icnt(0:100)
      real prob(100,0:100),cprob(100,0:100)
      common/ralph/cprob(100,0:100),rel
      external lexdec,divide,elim
***********************************************************
*  Temporary upper boundary points are generated.         *
***********************************************************
      jwk1=1
      jwk2=1
      do 40 ifix=m(icomp),0,-1
        do while ((wk1(icomp,jwk1).eq.ifix).and.
     +            (jwk1.le.jhigh))
          do 10 i=1,ihigh
            wk2(i,jwk2)=wk1(i,jwk1)
10        continue
          jwk1=jwk1+1
          jwk2=jwk2+1
        enddo
        if (ifix.gt.0) then
          do 30 j=1,jwk1-1
            do 20 i=1,ihigh
              if (i.eq.icomp) then
                wk2(i,jwk2)=ifix-1
              else
                wk2(i,jwk2)=wk1(i,j)
              endif
20          continue
            jwk2=jwk2+1
30        continue
        endif
40    continue
      jcnt=jwk2-1
***********************************************************
*  The problem is logically separated into subproblems.  *
***********************************************************
      call divide(wk2,icomp,jcnt,icnt,iter)
```

```
***********************************************************
*  Dominated lower boundary points are marked.            *
***********************************************************
      do 80 isub=0,iter
        do 70 j1=icnt(isub),icnt(isub+1)-2
          do 60 j2=j1+1,icnt(isub+1)-1
            iflag=0
            do 50 i=1,ihigh
              if (wk2(i,j1).le.wk2(i,j2)) then
                iflag=iflag+1
              endif
50          continue
            if (iflag.eq.ihigh) then
              wk2(1,j1)=-1
            endif
60        continue
70      continue
80    continue
***********************************************************
*  Dominated lower boundary points are eliminated.        *
***********************************************************
      call elim(wk2,ncomp,jcnt)
***********************************************************
*  The problem is logically separated into subproblems.   *
***********************************************************
      call divide(wk2,icomp,jcnt,icnt,iter)
***********************************************************
*  Single lower boundary points are marked and the        *
*  probability of the lower boundary point is tallied.    *
***********************************************************
      do 110 isub=0,iter
        if ((icnt(isub+1)-icnt(isub)).eq.1) then
          prod=1.0
          do 90 i=1,icomp
            prod=prod*prob(i,wk2(i,icnt(isub)))
90        continue
          do 100 i=icomp+1,ihigh
            prod=prod*cprob(i,wk2(i,icnt(isub)))
100       continue
          rel=rel+prod
          wk2(1,icnt(isub))=-1
        endif
110   continue
***********************************************************
*  Single lower boundary points are eliminated.           *
***********************************************************
      call elim(wk2,ncomp,jcnt)
***********************************************************
*  The remaining problem is sorted lexicographically.     *
***********************************************************
      call lexdec(wk2,ihigh,jcnt)
      return
      end
```

Appendix B. Bounding System Performance Program

```
**********************************************************
*  WRITTEN BY:  Ralph Boedigheimer                       *
*  LAST UPDATE:  16 Dec 91                               *
**********************************************************

**********************************************************
*  This is the main program that runs all other programs *
*  and estimates reliability for a multistate system.    *
**********************************************************
      program bounds
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    answer - variable for interactive feedback          *
**********************************************************
      real answer
**********************************************************
*  The main menu is presented to the user.  One of the   *
*  given options must be selected.                       *
**********************************************************
5     answer=0.0
      do while ((answer.lt.1.0).or.(answer.gt.10.0).or.
     +(amod(answer,1.0).ne.0.0))
        print *,'ENTER SELECTION FROM THE FOLLOWING MENU:'
        print *,'  1.  INPUT A NEW SYSTEM DESCRIPTION.'
        print *,'  2.  DISPLAY THE CURRENT SYSTEM.'
        print *,'  3.  TRIVIAL BOUNDS (LBPs).'
        print *,'  4.  TRIVIAL BOUNDS (UBPs).'
        print *,'  5.  PATH/CUT BOUNDS.'
        print *,'  6.  MIN/MAX BOUNDS.'
        print *,'  7.  COMBINED BOUNDS.'
        print *,'  8.  INCLUSION-EXCLUSION BOUNDS (LBPs).'
        print *,'  9.  INCLUSION-EXCLUSION BOUNDS (UBPs).'
        print *,' 10.  EXIT THE PROGRAM.'
        read *,answer
        print *
      enddo
**********************************************************
*  The program routes to the appropriate subroutine and  *
*  then returns to the main menu.                        *
**********************************************************
      go to (10,20,30,40,50,60,70,80,90,100),answer
10    call system
      go to 5
20    call display
      go to 5
30    call ltrivial
      go to 5
40    call utrivial
      go to 5
```

```
50     call pathcut
       go to 5
60     call minmax
       go to 5
70     call combined
       go to 5
80     call lower
       go to 5
90     call upper
       go to 5
100    stop
       end

**********************************************************
*  This program is used to enter a description of the    *
*  multistate system being studied.                      *
**********************************************************
       subroutine system
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
**********************************************************
       common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
      +t(0:100),ubp(100,100,0:100),prob(100,0:100)
       integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
      +t(0:100),ubp(100,100,0:100)
       real prob (100,0:100)
**********************************************************
*  The required information is interactively entered.    *
**********************************************************
       print *,'ENTER THE MAXIMUM SYSTEM STATE (1-100).'
       read *,msys
       print *
       print *,'ENTER THE NUMBER OF COMPONENTS (1-100).'
       read *,ncomp
       print *
       do 10 i=1,ncomp
         print *,'ENTER THE MAXIMUM STATE OF COMPONENT',i,
      +        '(1-100).'
         read *,m(i)
         print *
10     continue
```

```
      do 20 i=(ncomp+1),100
        m(i)=0
20    continue
      m(0)=0
      do 40 i=1,ncomp
        print *
        do 30 j=0,m(i)
          print *,'FOR COMPONENT',i,'ENTER THE ',
     +            'PROBABILITY OF BEING IN STATE',j
          read *,prob(i,j)
30      continue
40    continue
      do 60 k=1,msys
        print *,'ENTER THE NUMBER OF LOWER BOUNDARY',
     +          ' POINTS TO LEVEL',k,'(1-100).'
        read *,s(k)
        print *
        do 50 j=1,s(k)
          print *,'FOR LEVEL',k,'ENTER LOWER BOUNDARY',
     +            ' POINT #',j
          read *,(lbp(i,j,k),i=1,ncomp)
50      continue
        print *
60    continue
      do 80 k=0,msys-1
        print *,'ENTER THE NUMBER OF UPPER BOUNDARY',
     +          ' POINTS TO LEVEL',k,'(1-100).'
        read *,t(k)
        print *
        do 70 j=1,t(k)
          print *,'FOR LEVEL',k,'ENTER UPPER BOUNDARY',
     +            ' POINT #',j
          read *,(ubp(i,j,k),i=1,ncomp)
70      continue
        print *
80    continue
      return
      end

**********************************************************
*  This program displays the description of the system.  *
**********************************************************
      subroutine display
**********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    msys - the maximum state of the system               *
*    ncomp - the number of components in the system       *
*    m(i) - the maximum state of component i              *
*    s(k) - number of lower boundary points to level k    *
*    lbp(i,j,k) - the ith element of the jth lower        *
*                  boundary point to level k              *
*    t(k) - number of upper boundary points to level k    *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                 boundary point to level k               *
*    prob(i,j) - probability of component i in state j    *
***********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real prob (100,0:100)
***********************************************************
*  The current system is displayed for the user.          *
***********************************************************
      print *,'Maximum System State:',msys
      print *
      print *,'Number of Components:',ncomp
      print *
      print *,'Component Max State Vector:(',
     +        (m(i),i=1,ncomp),')'
      print *
      do 10 i=1,ncomp
        print 99,'Component',i,'Probabilities:',
     +           (prob(i,j),j=0,m(i))
10    continue
      print *
      do 30 k=1,msys
        print *,'System Level',k
        do 20 j=1,s(k)
          print *,'Lower Boundary Point #',j,': (',
     +            (lbp(i,j,k),i=1,ncomp),')'
20      continue
        print *
30    continue
      print *
      do 50 k=0,msys-1
        print *,'System Level',k
        do 40 j=1,t(k)
          print *,'Upper Boundary Point #',j,': (',
     +            (ubp(i,j,k),i=1,ncomp),')'
40      continue
        print *
50    continue
      print *
99    format(a9,1x,i2,1x,a14,2x,100(f5.3,1x))
      return
      end

***********************************************************
*  This program determines the trivial bounds using a     *
*  single lower boundary point.                           *
***********************************************************
```

```
      subroutine ltrivial
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real prob(100,0:100),cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from        *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
      do 50 k=1,msys
        print *,'System Level',k
        do 30 j=1,s(k)
          print *,'Lower Boundary Point #',j,': (',
     +            (lbp(i,j,k),i=1,ncomp),')'
30      continue
        print *
        answer=0.0
        do while ((answer.lt.1.0).or.(answer.gt.s(k)).or.
     +  (amod(answer,1.0).ne.0.0))
          print *,'ENTER THE LOWER BOUNDARY POINT #.'
          read *,answer
          print *
        enddo
        j=int(answer)
        qlow=1.0
        qhigh=1.0
        do 40 i=1,ncomp
          qlow=qlow*cprob(i,lbp(i,j,k))
          qhigh=qhigh*(1.0-cprob(i,lbp(i,j,k)))
40      continue
        print *,qlow,'<= Q(',k,') <=',1.0-qhigh
        print *
50    continue
      print *
      print *
      return
      end

**********************************************************
*  This program determines the trivial bounds using a    *
*  single upper boundary point.                          *
**********************************************************
      subroutine utrivial
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real prob(100,0:100),cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
      do 50 k=0,msys-1
        print *,'System Level',k
        do 30 j=1,t(k)
          print *,'Upper Boundary Point #',j,': (',
     +            (ubp(i,j,k),i=1,ncomp),')'
30      continue
        print *
        answer=0.0
        do while ((answer.lt.1.0).or.(answer.gt.t(k)).or.
     +  (amod(answer,1.0).ne.0.0))
          print *,'ENTER THE UPPER BOUNDARY POINT #.'
          read *,answer
          print *
        enddo
        j=int(answer)
        qlow=1.0
        qhigh=1.0
        do 40 i=1,ncomp
          qlow=qlow*cprob(i,ubp(i,j,k)+1)
          qhigh=qhigh*(1.0-cprob(i,ubp(i,j,k)+1))
40      continue
        print *,qlow,'<= Q(',k+1,') <=',1.0-qhigh
        print *
50    continue
      print *
      print *
      return
      end

**********************************************************
*  This program determines the path/cut bounds for one of*
*  the measures of reliability.                          *
**********************************************************
      subroutine pathcut
**********************************************************
```

```
*  VARIABLE DESCRIPTIONS:                                    *
*    msys - the maximum state of the system                  *
*    ncomp - the number of components in the system          *
*    m(i) - the maximum state of component i                 *
*    s(k) - number of lower boundary points to level k       *
*    lbp(i,j,k) - the ith element of the jth lower           *
*                 boundary point to level k                  *
*    t(k) - number of upper boundary points to level k       *
*    ubp(i,j,k) - the ith element of the jth upper           *
*                 boundary point to level k                  *
*    prob(i,j) - probability of component i in state j       *
*    cprob(i,j) - probability of component i in state        *
*                 j or higher                                *
**************************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,philower,phiupper,nvec,divider,
     +m(0:100),s(100),lbp(100,100,100),t(0:100),
     +ubp(100,100,0:100),x(100)
      real prob(100,0:100),cprob(100,0:100)
**************************************************************
*  The cumulative probability array is found from the        *
*  probability array entered in the system description.      *
**************************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**************************************************************
*  The path/cut bounds are determined assuming the           *
*  components are mutually independent.                      *
**************************************************************
      do 70 k=1,msys
        qlow=1.0
        do 40 j=1,t(k-1)
          prod=1.0
          do 30 i=1,ncomp
            if (ubp(i,j,k-1).ne.m(i)) then
              temp=1.0-cprob(i,ubp(i,j,k-1)+1)
              prod=prod*temp
            endif
30        continue
          temp=1.0-prod
          qlow=qlow*temp
40      continue
        qhigh=1.0
        do 60 j=1,s(k)
          prod=1.0
          do 50 i=1,ncomp
            if (lbp(i,j,k).ne.0.0) then
              temp=cprob(i,lbp(i,j,k))
              prod=prod*temp
            endif
50        continue
          temp=1.0-prod
          qhigh=qhigh*temp
60      continue
        print *,qlow,'<= Q(',k,') <=',1.0-qhigh
70    continue
      print *
      print *
      return
      end

**********************************************************
*  This program determines the min/max bounds for one of *
*  the measures of reliability.                          *
**********************************************************
      subroutine minmax
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                            *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,philower,phiupper,nvec,divider,
     +m(0:100),s(100),lbp(100,100,100),t(0:100),
     +ubp(100,100,0:100),x(100)
      real prob(100,0:100),cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
```

```
10      continue
20    continue
**********************************************************
*  The min/max bounds are determined assuming the        *
*  components are mutually independent.                  *
**********************************************************
      do 70 k=1,msys
        qlow=0.0
        do 40 j=1,s(k)
          prod=1.0
          do 30 i=1,ncomp
            if (lbp(i,j,k).ne.0.0) then
              temp=cprob(i,lbp(i,j,k))
              prod=prod*temp
            endif
30        continue
          qlow=max(qlow,prod)
40      continue
        qhigh=1.0
        do 60 j=1,t(k-1)
          prod=1.0
          do 50 i=1.ncomp
            if (ubp(i,j,k-1).ne.m(i)) then
              temp=1.0-cprob(i,ubp(i,j,k-1)+1)
              prod=prod*temp
            endif
50        continue
          temp=1.0-prod
          qhigh=min(qhigh,temp)
60      continue
        print *,qlow,'<= Q(',k,') <=',qhigh
70    continue
      print *
      print *
      return
      end

**********************************************************
*  This program determines the combined bounds for one of*
*  the measures of reliability.                          *
**********************************************************
      subroutine combined
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper          *
*                 boundary point to level k                 *
*    prob(i,j) - probability of component i in state j      *
*    cprob(i,j) - probability of component i in state       *
*                 j or higher                               *
*************************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,philower,phiupper,nvec,divider,
     +m(0:100),s(100),lbp(100,100,100),t(0:100),
     +ubp(100,100,0:100),x(100)
      real prob(100,0:100),cprob(100,0:100)
*************************************************************
*  The cumulative probability array is found from the       *
*  probability array entered in the system description.     *
*************************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
*************************************************************
*  The combined bounds are determined assuming the          *
*  components are mutually independent.                     *
*************************************************************
      do 110 k=1,msys
        qlow1=0.0
        do 40 j=1,s(k)
          prod=1.0
          do 30 i=1,ncomp
            if (lbp(i,j,k).ne.0.0) then
              temp=cprob(i,lbp(i,j,k))
              prod=prod*temp
            endif
30        continue
          qlow1=max(qlow1,prod)
40      continue
        qlow2=1.0
        do 60 j=1,t(k-1)
          prod=1.0
          do 50 i=1,ncomp
            if (ubp(i,j,k-1).ne.m(i)) then
              temp=1.0-cprob(i,ubp(i,j,k-1)+1)
              prod=prod*temp
            endif
50        continue
          temp=1.0-prod
          qlow2=qlow2*temp
60      continue
```

```
         qhigh1=1.0
         do 80 j=1,t(k-1)
           prod=1.0
           do 70 i=1,ncomp
             if (ubp(i,j,k-1).ne.m(i)) then
               temp=1.0-cprob(i,ubp(i,j,k-1)+1)
               prod=prod*temp
             endif
70         continue
           temp=1.0-prod
           qhigh1=min(qhigh1,temp)
80       continue
         qhigh2=1.0
         do 100 j=1,s(k)
           prod=1.0
           do 90 i=1,ncomp
             if (lbp(i,j,k).ne.0.0) then
               temp=cprob(i,lbp(i,j,k))
               prod=prod*temp
             endif
90         continue
           temp=1.0-prod
           qhigh2=qhigh2*temp
100      continue
         qhigh2=1.0-qhigh2
         qlow=max(qlow1,qlow2)
         qhigh=min(qhigh1,qhigh2)
         print *,qlow,'<= Q(',k,') <=',qhigh
110    continue
       print *
       print *
       return
       end

**********************************************************
*  This program bounds the probability distribution     *
*  using the inclusion-exclusion principle and lower    *
*  boundary points.                                     *
**********************************************************
       subroutine lower
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    cmblower - real function that finds all combinations*
*    prob(i,j) - probability of component i in state j   *
*    sum - the sum of all combinatorial summations       *
*    plev(k) - probability of a system in state k        *
*    cplev(k) - probability of system in state k or more *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real cmblower,prob(100,0:100)
      double precision sum
      external cmblower
**********************************************************
*  The cumulative probability of the system being in     *
*  state k or higher is found for every system state.    *
**********************************************************
      answer=0.0
      do while ((answer.le.0.0).or.(answer.ge.1.0))
        print *,'ENTER THE ACCEPTABLE TOLERANCE',
     +          ' (0 < TOL < 1).'
        read *,answer
        print *
      enddo
      tol=answer
      do 10 k=1,msys
        sum=0.0
        j=1
        diff=1.0
        qlow=0.0
        qhigh=1.0
        diff=qhigh-qlow
        do while ((j.le.s(k)).and.(tol.lt.diff))
          sum=sum+((-1)**(j+1))*cmblower(s(k),j,k)
          if ((j.ne.s(k)).and.(mod(j,2).ne.0.0)) then
            qhigh=sum
          elseif ((j.ne.s(k)).and.(mod(j,2).eq.0.0)) then
            qlow=sum
          elseif (j.eq.s(k)) then
            qlow=sum
            qhigh=sum
          endif
          diff=qhigh-qlow
          j=j+1
        enddo
        print *,qlow,'<= Q(',k,') <=',qhigh
10    continue
      print *
      print *
      return
      end
```

```
**********************************************************
*  This program determines all possible combinations of  *
*  vectors to consider.  It is required for calculation  *
*  of the intersection of events in the inclusion-       *
*  exclusion formula.                                    *
**********************************************************
      function cmblower(n,r,k)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    ichange - the element that is changed               *
*    r - the number of vectors to choose                 *
*    n - the total number of vectors                     *
*    itop - the max state of the intersection of vectors *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    vec(i) - the intersection vector                    *
*    lrg(i) - the largest vector in position i           *
*    store - a temporary storage location                *
*    cmbupper -variable used to return probability       *
*    prob(i,j) - probability of component i in state j   *
*    prod - probability of a component state vector      *
*    cprob(i,j) - probability of component i in state    *
*                 j or higher                            *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,ichange,r,n,numb,itop,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +vec(0:100),lrg(0:100)
      real store,cmblower,prob(100,0:100)
      double precision prod,cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 2  i=1,ncomp
        store=0.0
        do 10 j=m(i),0,-1
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The sum of the probabilities of all combinations of   *
*  lower boundary points to level k taken r at a time    *
*  is found.                                              *
**********************************************************
      do 30 i=0,r
        vec(i)=i
        lrg(i)=n-r+i
30    continue
      ichange=r
      cmblower=0.0
      do while (ichange.gt.0)
        ichange=r
        do 60 numb=(vec(ichange-1)+1),n
          vec(ichange)=numb
          prod=1.0
          do 50 i=1,ncomp
            itop=0
            do 40 j=1,r
              itop=max(lbp(i,vec(j),k),itop)
40          continue
            prod=prod*cprob(i,itop)
50        continue
          cmblower=cmblower+prod
60      continue
        do while ((vec(ichange-1)).eq.(lrg(ichange-1))
     +  .and.(ichange.gt.1))
          ichange=ichange-1
        enddo
        ichange=ichange-1
        vec(ichange)=vec(ichange)+1
        do 70 i=(ichange+1),r
          vec(i)=vec(i-1)+1
70      continue
      enddo
      return
      end

**********************************************************
*  This program bounds the probability distribution      *
*  using the inclusion-exclusion principle and upper
*  boundary points.                                       *
**********************************************************
      subroutine upper
**********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    msys - the maximum state of the system               *
*    ncomp - the number of components in the system       *
*    m(i) - the maximum state of component i              *
*    s(k) - number of lower boundary points to level k    *
*    lbp(i,j,k) - the ith element of the jth lower        *
*                 boundary point to level k               *
*    t(k) - number of upper boundary points at level l    *
*    ubp(i,j,k) - the ith element of the jth upper        *
*                  boundary point to level k              *
*    cmbupper - real function that find all combinations *
*    prob(i,j) - probability of component i in state j    *
*    sum - the sum of all combinatorial summations        *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real cmbupper,prob(100,0:100)
      double precision sum
      external cmbupper
**********************************************************
*  The cumulative probability of the system in state     *
*  k or lower is found for every system state.           *
**********************************************************
      answer=0.0
      do while ((answer.le.0.0).or.(answer.ge.1.0))
        print *,'ENTER THE ACCEPTABLE TOLERANCE ',
     +         '(0 < TOL < 1).'
        read *,answer
        print *
      enddo
      tol=answer
      do 10 k=0,msys-1
        sum=0.0
        j=1
        diff=1.0
        qlow=0.0
        qhigh=1.0
        diff=qhigh-qlow
        do while ((j.le.t(k)).and.(tol.lt.diff))
          sum=sum+((-1)**(j+1))*cmbupper(t(k),j,k)
          if ((j.ne.t(k)).and.(mod(j,2).ne.0.0)) then
            qlow=1.0-sum
          elseif ((j.ne.t(k)).and.(mod(j,2).eq.0.0)) then
            qhigh=1.0-sum
          elseif (j.eq.t(k)) then
            qlow=1.0-sum
            qhigh=1.0-sum
          endif
          diff=qhigh-qlow
          j=j+1
        enddo
        print *,qlow,'<= Q(',k+1,') <=',qhigh
10    continue
      print *
      print *
      return
      end
```

```
**********************************************************
*  This program determines all possible combinations of  *
*  vectors to consider.  It is required for calculation  *
*  of the intersection of events in the inclusion-       *
*  exclusion formula.                                    *
**********************************************************
      function cmbupper(n,r,k)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    ichange - the element that is changed               *
*    r - the number of vectors to choose                 *
*    n - the total number of vectors                     *
*    ibot - the min state of the intersection of vectors *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of lower boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    vec(i) - the intersection vector                    *
*    lrg(i) - the largest vector in position i           *
*    store - a temporary storage location                *
*    cmbupper - variable used to return probability      *
*    prob(i,j) - probability of component i in state j   *
*    prod - probability of a component state vector      *
*    cprob(i,j) - probability of component i in state    *
*                 j or lower                             *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,ichange,r,n,numb,itop,m(0:100),
     +s(100),lbp(100,100,100),t(0:100),ubp(100,100,0:100),
     +vec(0:100),lrg(0:100)
      real store,cmbupper,prob(100,0:100)
      double precision prod,cprob(100,0:100)
**********************************************************
*  The cumulative probability array is found from the    *
*  probability array entered in the system description.  *
**********************************************************
      do 20 i=1,ncomp
        store=0.0
        do 10 j=0,m(i)
          cprob(i,j)=store+prob(i,j)
          store=cprob(i,j)
10      continue
20    continue
**********************************************************
*  The sum of the probabilities of all combinations of   *
*  lower boundary points to level k taken r at a time     *
*  is found.                                              *
***********************************************************
      do 30 i=0,r
        vec(i)=i
        lrg(i)=n-r+i
30    continue
      ichange=r
      cmbupper=0.0
      do while (ichange.gt.0)
        ichange=r
        do 60 numb=(vec(ichange-1)+1),n
          vec(ichange)=numb
          prod=1.0
          do 50 i=1,ncomp
            ibot=m(i)
            do 40 j=1,r
              ibot=min(ubp(i,vec(j),k),ibot)
40          continue
            prod=prod*cprob(i,ibot)
50        continue
          cmbupper=cmbupper+prod
60      continue
        do while ((vec(ichange-1)).eq.(lrg(ichange-1))
     +  .and.(ichange.gt.1))
          ichange=ichange-1
        enddo
        ichange=ichange-1
        vec(ichange)=vec(ichange)+1
        do 70 i=(ichange+1),r
          vec(i)=vec(i-1)+1
70      continue
      enddo
      return
      end
```

Appendix C. Boundary Point Conversion Program

```
**********************************************************
*  WRITTEN BY:  Ralph Boedigheimer                        *
*  LAST UPDATE:  20 Feb 92                                *
**********************************************************

**********************************************************
*  This is the main program that runs all other programs.*
**********************************************************
      program bpconv
**********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    answer - variable for interactive feedback           *
**********************************************************
      real answer
**********************************************************
*  The main menu is presented to the user.  One of the    *
*  given options must be selected.                        *
**********************************************************
5     answer=0.0
      do while ((answer.lt.1.0).or.(answer.gt.5.0).or.
     +(amod(answer,1.0).ne.0.0))
        print *,'ENTER SELECTION FROM THE FOLLOWING MENU:'
        print *,'  1.  INPUT A NEW SYSTEM DESCRIPTION.'
        print *,'  2.  DISPLAY THE CURRENT SYSTEM.'
        print *,'  3.  CONVERT LBPs to UBPs.'
        print *,'  4.  CONVERT UBPs to LBPs.'
        print *,'  5.  EXIT THE PROGRAM.'
        read *,answer
        print *
      enddo
**********************************************************
*  The program routes to the appropriate subroutine and   *
*  then returns to the main menu.                         *
**********************************************************
      go to (10,20,30,40,50),answer
10    call system
      go to 5
20    call display
      go to 5
30    call lbptoubp
      go to 5
40    call ubptolbp
      go to 5
50    stop
      end

**********************************************************
*  This program is used to enter a description of the     *
*  multistate system being studied.                       *
```

```
**********************************************************
      subroutine system
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      real prob (100,0:100)
**********************************************************
*  The required information is interactively entered.    *
**********************************************************
      print *,'ENTER THE MAXIMUM SYSTEM STATE (1-100).'
      read *,msys
      print *
      print *,'ENTER THE NUMBER OF COMPONENTS (1-100).'
      read *,ncomp
      print *
      do 10 i=1,ncomp
        print *,'ENTER THE MAXIMUM STATE OF COMPONENT',i,
     +          '(1-100).'
        read *,m(i)
        print *
10    continue
      do 20 i=(ncomp+1),100
        m(i)=0
20    continue
      m(0)=0
      do 40 i=1,ncomp
        print *
        do 30 j=0,m(i)
          print *,'FOR COMPONENT',i,'ENTER THE',
     +          ' PROBABILITY OF BEING IN STATE',j
          read *,prob(i,j)
30      continue
40    continue
      print *
      do 60 k=1,msys
        print *,'ENTER THE NUMBER OF LOWER BOUNDARY',
     +          ' POINTS TO LEVEL',k,'(1-100).'
```

```
           read *,s(k)
           print *
           do 50 j=1,s(k)
             print *,'FOR LEVEL',k,'ENTER LOWER BOUNDARY',
     +             ' POINT #',j
             read *,(lbp(i,j,k),i=1,ncomp)
50         continue
           print *
60       continue
         do 80 k=0,msys-1
           print *,'ENTER THE NUMBER OF UPPER BOUNDARY',
     +             ' POINTS TO LEVEL',k,'(1-100).'
           read *,t(k)
           print *
           do 70 j=1,t(k)
             print *,'FOR LEVEL',k,'ENTER UPPER BOUNDARY',
     +             ' POINT #',j
             read *,(ubp(i,j,k),i=1,ncomp)
70         continue
           print *
80       continue
         return
         end

**********************************************************
*  This program displays the description of the system.  *
**********************************************************
         subroutine display
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    prob(i,j) - probability of component i in state j   *
**********************************************************
         common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
         integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
         real prob (100,0:100)
**********************************************************
*  The current system is displayed for the user.         *
**********************************************************
         print *,'Maximum System State:',msys
         print *
```

```
      print *,'Number of Components:',ncomp
      print *
      print *,'Component Max State Vector: (',
     +        (m(i),i=1,ncomp),')'
      print *
      do 10 i=1,ncomp
        print 99,'Component',i,'Probabilities:',
     +           (prob(i,j),j=0,m(i))
10    continue
      print *
      do 30 k=1,msys
        print *,'System Level',k
        do 20 j=1,s(k)
          print *,'Lower Boundary Point #',j,': (',
     +            (lbp(i,j,k),i=1,ncomp),')'
20      continue
        print *
30    continue
      print *
      do 50 k=0,msys-1
        print *,'System Level',k
        do 40 j=1,t(k)
          print *,'Upper Boundary Point #',j,': (',
     +            (ubp(i,j,k),i=1,ncomp),')'
40      continue
        print *
50    continue
      print *
99    format(a9,1x,i2,1x,a14,2x,100(f5.3,1x))
      return
      end

**********************************************************
*  This program converts lower boundary points to level  *
*  k to upper boundary points to level k-1.              *
**********************************************************
      subroutine lbptoubp
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    vec(i) - a temporary work vector                    *
*    perm(i) - a permanent work vector                   *
*    list(i,j) - a storage vector for each level         *
*    jcnt - the jth dimension of list                       *
*    prob(i,j) - probability of component i in state j     *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      integer vec(0:100),perm(100),list(100,10000)
      real prob(100,0:100)
      external elim,lexdec
**********************************************************
*   A list of potential upper boundary points is generated*
*   from all the lower boundary points.                   *
**********************************************************
      do 150 k=1,msys
        jcnt=0
        do 70 j=1,s(k)
          do 60 i=1,ncomp
            do 10 i1=1,ncomp
              vec(i1)=lbp(i1,j,k)
10          continue
            vec(i)=lbp(i,j,k)-1
            do 20 i2=1,ncomp
              perm(i2)=vec(i2)
20          continue
            if (vec(i).ge.0) then
              ichange=ncomp
              do while (ichange.gt.0)
                ichange=ncomp
                do 40 i3=vec(ncomp),m(ncomp)
                  vec(ncomp)=i3
                  if (perm(i).eq.vec(i)) then
                    jcnt=jcnt+1
                    do 30 i4=1,ncomp
                      list(i4,jcnt)=vec(i4)
30                  continue
                  endif
40              continue
                ichange=ichange-1
                do while ((vec(ichange).eq.m(ichange))
     +          .and.(ichange.gt.0))
                  ichange=ichange-1
                enddo
                vec(ichange)=vec(ichange)+1
                do 50 i5=ichange+1,ncomp
                  vec(i5)=perm(i5)
50              continue
              enddo
            endif
60        continue
70      continue
```

```
**********************************************************
*  All potential upper boundary points that are dominated*
*  by other lower boundary points are marked and removed.*
**********************************************************
        do 100 j1=1,s(k)
          do 90 j2=1,jcnt
            iflag=0
            do 80 i=1,ncomp
              if (lbp(i,j1,k).le.list(i,j2)) then
                iflag=iflag+1
              endif
80          continue
            if (iflag.eq.ncomp) then
              list(1,j2)=-1
            endif
90        continue
100     continue
        call elim(list,ncomp,jcnt)
**********************************************************
*  The list matrix is lexicographically sorted in       *
*  decreasing order.                                     *
**********************************************************
        call lexdec(list,ncomp,jcnt)
**********************************************************
*  All potential upper boundary points that are overcome *
*  by other potential upper boundary points are marked   *
*  and removed from the list.                            *
**********************************************************
        do 130 j2=jcnt,2,-1
          do 120 j1=j2-1,1,-1
            iflag=0
            do 110 i=1,ncomp
              if (list(i,j1).ge.list(i,j2)) then
                iflag=iflag+1
              endif
110         continue
            if (iflag.eq.ncomp) then
              list(1,j2)=-1
            endif
120       continue
130     continue
        call elim(list,ncomp,jcnt)
**********************************************************
*  The remaining vectors on the list are upper boundary  *
*  points to level k-1.                                  *
**********************************************************
        print *'For level',k-1
        do 140 j=1,jcnt
          print *,'Upper Boundary Point #',j,': (',
     +    (list(i,j),i=1,ncomp),')'
140     continue
```

```
        print *
150   continue
      return
      end

      subroutine elim(wk,ihigh,jcnt)
***********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    wk - a temporary working matrix                      *
*    ihigh - the ith dimension of wk                      *
*    jcnt - the jth dimension of wk                       *
***********************************************************
      integer wk(100,10000)
      do 30 j2=jcnt,1,-1
        if (wk(1,j2).eq.-1) then
          do 20 j1=j2,jcnt-1
            do 10 i=1,ihigh
              wk(i,j1)=wk(i,j1+1)
10          continue
20        continue
          jcnt=jcnt-1
        endif
30    continue
      return
      end

      subroutine lexdec(wk,ihigh,jhigh)
***********************************************************
*  VARIABLE DESCRIPTIONS:                                 *
*    wk - a temporary working matrix                      *
*    ihigh - the ith dimension of wk                      *
*    jhigh - the jth dimension of wk                      *
***********************************************************
      integer wk(100,10000)
      do 40 i1=ihigh,1,-1
        do 30 j1=1,jhigh-1
          num=jhigh-j1
          do 20 j2=1,num
            if (wk(i1,j2).lt.wk(i1,j2+1)) then
              do 10 i2=1,ihigh
                itemp=wk(i2,j2)
                wk(i2,j2)=wk(i2,j2+1)
                wk(i2,j2+1)=itemp
10            continue
            endif
20        continue
30      continue
40    continue
      return
      end
```

```
**********************************************************
*  This program converts upper boundary points to level  *
*  k to lower boundary points to level k+1.              *
**********************************************************
      subroutine ubptolbp
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    msys - the maximum state of the system              *
*    ncomp - the number of components in the system      *
*    m(i) - the maximum state of component i             *
*    s(k) - number of lower boundary points to level k   *
*    lbp(i,j,k) - the ith element of the jth lower       *
*                 boundary point to level k              *
*    t(k) - number of upper boundary points to level k   *
*    ubp(i,j,k) - the ith element of the jth upper       *
*                 boundary point to level k              *
*    vec(i) - a temporary work vector                    *
*    perm(i) - a permanent work vector                   *
*    list(i,j) - a storage vector for each level         *
*    jcnt - the jth dimension of list                    *
*    prob(i,j) - probability of component i in state j   *
**********************************************************
      common msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100),prob(100,0:100)
      integer msys,ncomp,m(0:100),s(100),lbp(100,100,100),
     +t(0:100),ubp(100,100,0:100)
      integer vec(0:100),perm(100),list(100,10000)
      real prob(100,0:100)
      external elim,lexinc
**********************************************************
*  A list of potential lower boundary points is generated*
*  from all the upper boundary points.                   *
**********************************************************
      do 150 k=0,msys-1
        jcnt=0
        do 70 j=1,t(k)
          do 60 i=1,ncomp
            do 10 i1=1,ncomp
              vec(i1)=ubp(i1,j,k)
10          continue
            vec(i)=ubp(i,j,k)+1
            do 20 i2=1,ncomp
              perm(i2)=vec(i2)
20          continue
            if (vec(i).le.m(i)) then
              ichange=ncomp
              do while (ichange.gt.0)
                ichange=ncomp
                do 40 i3=vec(ncomp),0,-1
                  vec(ncomp)=i3
                  if (perm(i).eq.vec(i)) then
```

```
                    jcnt=jcnt+1
                    do 30 i4=1,ncomp
                      list(i4,jcnt)=vec(i4)
30                  continue
                  endif
40              continue
                ichange=ichange-1
                do while ((vec(ichange).eq.0).and.
     +          (ichange.gt.0))
                  ichange=ichange-1
                enddo
                vec(ichange)=vec(ichange)-1
                do 50 i5=ichange+1,ncomp
                  vec(i5)=perm(i5)
50              continue
              enddo
            endif
60        continue
70      continue
**********************************************************
*  All potential lower boundary points that are dominated*
*  by other upper boundary points are marked and removed.*
**********************************************************
        do 100 j1=1,t(k)
          do 90 j2=1,jcnt
            iflag=0
            do 80 i=1,ncomp
              if (ubp(i,j1,k).ge.list(i,j2)) then
                iflag=iflag+1
              endif
80          continue
            if (iflag.eq.ncomp) then
              list(1,j2)=-1
            endif
90        continue
100     continue
        call elim(list,ncomp,jcnt)
**********************************************************
*  The list matrix is lexicographically sorted in        *
*  increasing order.                                     *
**********************************************************
        call lexinc(list,ncomp,jcnt)
**********************************************************
*  All potential lower boundary points that are overcome *
*  by other potential lower boundary points are marked   *
*  and removed from the list.                            *
**********************************************************
        do 130 j2=jcnt,2,-1
          do 120 j1=j2-1,1,-1
            iflag=0
            do 110 i=1,ncomp
```

```
              if (list(i,j1).le.list(i,j2)) then
                iflag=iflag+1
              endif
110         continue
            if (iflag.eq.ncomp) then
              list(1,j2)=-1
            endif
120       continue
130     continue
        call elim(list,ncomp,jcnt)
**********************************************************
*  The remaining vectors on the list are lower boundary  *
*  points to level k+1.                                  *
**********************************************************
        print *'For level',k+1
        do 140 j=1,jcnt
          print *,'Lower Boundary Point #',j,': (',
    +     (list(i,j),i=1,ncomp),')'
140     continue
        print *
150   continue
      return
      end

      subroutine lexinc(wk,ihigh,jhigh)
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    wk - a temporary working matrix                     *
*.   ihigh - the ith dimension of wk                     *
*    jhigh - the jth dimension of wk                     *
**********************************************************
      integer wk(100,10000)
      do 40 i1=ihigh,1,-1
        do 30 j1=1,jhigh-1
          num=jhigh-j1
          do 20 j2=1,num
            if (wk(i1,j2).gt.wk(i1,j2+1)) then
              do 10 i2=1,ihigh
                itemp=wk(i2,j2)
                wk(i2,j2)=wk(i2,j2+1)
                wk(i2,j2+1)=itemp
10            continue
            endif
20        continue
30      continue
40    continue
      return
      end
```

Appendix D. Expected Loss Program

```
**********************************************************
*  WRITTEN BY:  Ralph Boedigheimer                       *
*  LAST UPDATE:  20 Apr 92                               *
**********************************************************

**********************************************************
*  This program determines the expected loss for a       *
*  quadratic loss function that is sensitive to the      *
*  pattern of degradation about a specified lifetime.  It*
*  assumes that the distributions for the time spent in  *
*  state k are mutually independent and exponential with *
*  common parameter lambda.  It allows different costs   *
*  for leaving the various states.                       *
**********************************************************
      program loss
**********************************************************
*  VARIABLE DESCRIPTIONS:                                *
*    c(k) - cost for leaving state k                     *
*    lam - common parameter of exponential distributions *
*    p(k) - probability of being in state k at tstar     *
*    tstar - desired lifetime for the customer           *
*    m - maximum state of system or component            *
*    iter - number of iterations for the simulation      *
*    t(j) - exponential variate for time spent in state k*
*    loss - loss from one iteration of the simulation    *
*    avgloss - mean of the simulation losses             *
**********************************************************
      real c(100),lam,p(0:100),tstar
      integer m,iter
      real t(0:100),loss,avgloss
      external gamdf
**********************************************************
*  The system or component description is entered.       *
**********************************************************
      print *,'SYSTEM OR COMPONENT DESCRIPTION -'
      print *
      print *,'  ENTER DESIRED LIFETIME (T*).'
      read *,tstar
      print *,'  ENTER THE MAXIMUM STATE (M).'
      read *,m
      do 10 k=m,1,-1
        print *,'  ENTER THE COST OF LEAVING STATE',k,'.'
        read *,c(k)
10    continue
      print *,'  ENTER THE COMMON LAMBDA.'
      read *,lam
**********************************************************
*  The state probabilities are calculated.               *
**********************************************************
```

```
      totp=0.0
      do 30 k=m,1,-1
        ifac=1
        do 20 i=m-k,1,-1
          ifac=ifac*i
20      continue
        p(k)=(lam*tstar)**(m-k)*exp(-1.0*lam*tstar)/ifac
        totp=totp+p(k)
30    continue
      p(0)=1.0-totp
      print *
      print *,'State Probabilities:'
      do 40 k=0,m
        print *,'P(',k,') =',p(k)
40    continue
**********************************************************
*  The theoretical expected loss is calculated.          *
**********************************************************
      tsum=0.0
      do 60 k=0,m-1
        csum=0.0
        dsum=0.0
        esum=0.0
        do 50 j=k+1,m
          csum=csum+c(j)
          dsum=dsum+c(j)*(m-j+1)/lam
     +          *gamdf(lam*tstar,real(m-j+2))
     +          /gamdf(lam*tstar,real(m-j+1))
          esum=esum+c(j)*(m-j+1)*(m-j+2)/(lam**2.0)
     +          *gamdf(lam*tstar,real(m-j+3))
     +          /gamdf(lam*tstar,real(m-j+1))
50      continue
        csum=csum*tstar**2.0
        dsum=dsum*(-2.0)*tstar
        tsum=tsum+(csum+dsum+esum)*p(k)
60    continue
      print *
      print *,'Theoretical Expected Loss =',tsum
**********************************************************
*  The expected loss is approximated with a simulation.  *
**********************************************************
      print *
      print *
      print *,'*** SIMULATION ***'
      print *
      print *,'ENTER THE NUMBER OF ITERATIONS.'
      read *,iter
      cost=0.0
      do 90 ii=1,iter
        icnt=0
        sum=0.0
```

```
          do while ((sum.le.tstar).and.(icnt.le.m))
            t(m-icnt)=(-1.0/lam)*alog(1.0-rand(0.0))
            sum=sum+t(m-icnt)
            icnt=icnt+1
          enddo
          do 80 j=m-icnt+2,m
            tott=0.0
            do 70 i=j,m
              tott=tott+t(i)
70          continue
            loss=loss+c(j)*(tstar-tott)**2
80        continue
90      continue
        avgloss=loss/iter
        print *
        print *,'Estimated Expected Loss =',avgloss

        end
```

VITA

Ralph Boedigheimer is a Major in the United States Air Force with over 12 years of service. He entered the United States Air Force Academy (USAFA) and graduated in 1980 with a Bachelor of Science Degree in Mathematics. After receiving his commission, he served as a scientific analyst at Nellis Air Force Base. Major Boedigheimer began graduate work at the Air Force Institute of Technology and received a Master of Science Degree in Operations Research with distinction in 1983. He worked for the Air Force Operational Test and Evaluation Center until December 1987 and instructed at USAFA until his entry into the School of Industrial Engineering at the University of Oklahoma in August 1989. He is a published author and a member of several professional organizations and honor societies.