MITNE-222

PL-MODT AND PL-MODMC TWO CODES FOR RELIABILITY AND AVAILABILITY ANALYSIS OF COMPLEX TECHNICAL SYSTEMS USING THE FAULT TREE MODULARIZATION TECHNIQUE

by

Mohammad Modarres
Lothar Wolf

November 1978

DEPARTMENT OF NUCLEAR ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts 02139

MITNE-222

PL-MODT AND PL-MODMC TWO CODES FOR RELIABILITY AND AVAILABILITY ANALYSIS OF COMPLEX TECHNICAL SYSTEMS USING THE FAULT TREE MODULARIZATION TECHNIQUE

by

Mohammad Modarres
Lothar Wolf

November 1978

Department of Nuclear Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

The methodology used in the PL-MOD code has been extended to include the time-dependent behavior of the fault tree components. Four classes of components are defined to model time-dependent fault tree leaves. Mathematical simplifications are applied to predict the time-dependent behavior of simple modules in the fault tree from its input components' failure data.

The extended code, PL-MODT, handles time-dependent problems based on the mathematical models that have been established. An automatic tree reduction feature is also incorporated into this code. This reduction is based on the Vesely-Fussell importance measure that the code calculates. A CUT-OFF value is defined and incorporated into the code. Any module or component in the fault tree whose V-F importance is less than this value will automatically be eliminated from the tree. In order to benchmark the PL-MODT code, a number of systems are analyzed. The results are in good agreement with other codes, such as FRANTIC and KITT. The computation times are comparable and in most of the cases are even lower for the PL-MODT code compared to the others.

In addition, a Monte-Carlo simulation code (PL-MODMC) is developed to propagate uncertainties in the failure rates of the components to the top event of a fault tree. An efficient sorting routine similar to the one used in the LIMITS code is employed in the PL-MODMC code. Upon modularization the code proceeds and propagates uncertainties in the failure rates through the tree. Large fault trees such as the LPRS fault tree as well as some smaller ones have been analyzed for simulation, and the results for the LPRS are in fair agreement with the WASH-1400 predictions for the number of simulations performed.

The codes PL-MODT and PL-MODMC are written in PL/1 language which offers the extensive use of the list processing tools. First experience indicates that these codes are very efficient and accurate, specifically for the analysis of very large and complex fault trees.

This work was performed under the financial sponsorship by U.S. NRC, whose support is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION TO THE MODULAR DECOMPOSITION APPROACH

1.1 Introduction

The method of modular decomposition of fault trees has recently become very attractive and has been proven to be a very efficient and reliable technique for the analysis of large fault trees [1] in the framework of the PL-MOD code written in PL/1 language.

The methodology employed by PL-MOD to modularize a fault tree consists of piecewise collapsing and modularizing portions of the tree. As a consequence, at the intermediate stages of this process some nodes are eliminated from the tree while others undergo changes in the type and number of inputs they have. In the next section some of the mathematical concepts used in the PL-MOD as well as the method utilized in the code are presented.

1.2 Modularization Technique: Its Mathematical Formulation and Its Application in the Code PL-MOD

Birnbaum and Esary [1] define a module as follows:

> "A module of a system is a subset of basic components of the system which are organized into some substructure of their own and which affect the system only through the performance of their substructure. Rephrasing, a module is an assembly of components which can itself be treated as a component of the system."

The coherent structure theory plays an important role in the analysis of systems using the modular concept. Coherent binary systems are systems whose performance improves as the performance

of their components improves. A coherent system is a system (C,ϕ) for which

$$\phi(\underset{\rightarrow}{x}) \leq \phi(\underset{\rightarrow}{y}) \quad \text{whenever } \underset{\rightarrow}{x} \leq \underset{\rightarrow}{y} \quad \text{i.e., } x_i \leq y_i \tag{1.2.1}$$

$$\phi(\underset{\rightarrow}{0}) = 0, \text{ where } \underset{\rightarrow}{0} = (0, 0,\ldots,0); \tag{1.2.2}$$

(i.e., a state where all components fail)

$$\phi(\underset{\rightarrow}{1}) = 1, \text{ where } \underset{\rightarrow}{1} = (1,1,\ldots,1) \tag{1.2.3}$$

(i.e., where all components perform).

Let $C = (C_1,C_2,\ldots,C_n)$ be a set of basic events, then $\underset{\rightarrow}{y} = (y_1,y_2,\ldots,y_n)$ is the vector of basic event outcomes, and the Boolean structure function $\phi(y)$ determines the state of the system (i.e., the top event in the fault tree).

$$\phi(y) = \begin{cases} 1 \text{ if the top event occurs} \\ 0 \text{ otherwise.} \end{cases}$$

For the AND and OR gates, the vector will be defined as follows:

$$\phi_{AND}(\underset{\rightarrow}{y}) = y_1 \cdot y_2 \cdot \ldots \cdot y_n = \prod_{i=1}^{n} y_i \tag{1.2.4}$$

$$\phi_{OR}(\underset{\rightarrow}{y}) = 1 - (1-y_1)(1-y_2)\ldots(1-y_n) = \coprod_{i=1}^{n} y_i \tag{1.2.5}$$

respectively.

A component C_i is inessential to a system (C,ϕ) if the performance of the component can have no effect on the performance of the structure, i.e., if

$$\phi(1_i,\underset{\rightarrow}{x}) = \phi(o_i,\underset{\rightarrow}{x}) \text{ for all } (\cdot i,\underset{\rightarrow}{x}) \tag{1.2.6}$$

where

$$(1_i,\underset{\rightarrow}{x}) = (x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n),$$

$$(0_i,\underset{\rightarrow}{x}) = (x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n),$$

$$(\cdot i,\underset{\rightarrow}{x}) = (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n).$$

Since inessential components can be detected from a system without any effect, we will generally assume that all components in the fault tree are essential and thus avoid some complications in the theory. Therefore, we always assume

$$\phi(1_i,\underset{\rightarrow}{x}) > (0_i,\underset{\rightarrow}{x}) \text{ for all } (\cdot i,\underset{\rightarrow}{x}) \qquad (1.2.7)$$

A cut set of a coherent binary system (C,ϕ) is a set of components $Q \subseteq C$ such that $\phi(\underset{\rightarrow}{0}^Q,\underset{\rightarrow}{1}^{Q'})=0$ where Q' is the complement of Q (i.e., $C=Q \cup Q'$), and $\underset{\rightarrow}{0}^Q,\underset{\rightarrow}{1}^Q$ are the component performance vectors for which $x_i=0$ when $C_i \varepsilon Q$ and $x_i=1$ when $C_i \varepsilon Q'$. A cut set Q is a <u>minimal cut set</u> of (C,ϕ) if there is no cut set P such that $P \subset Q$.

The path sets of the same system (C,ϕ) are in fact the cut sets of coherent system (C,ϕ^D), where $\phi^D(\underset{\rightarrow}{x}) = 1 - \phi(\underset{\rightarrow}{1} - \underset{\rightarrow}{x})$. (1.2.8) $\phi^D(\underset{\rightarrow}{x})$ is called the Dual Structure Function of the system (C,ϕ).

From the above definitions for a coherent function and by using the definition of a module, one can find [1] the structure function of the system using its modules.

Consider a subset A of components such that $A \subset C$. A is a modular set of (C,ϕ) if

$$\phi(\underset{\rightarrow}{x}) = \phi(\underset{\rightarrow}{x}^A,\underset{\rightarrow}{x}^{A'}) = \Psi[\chi_A(\underset{\rightarrow}{x}^A),\underset{\rightarrow}{x}^{A'}] \qquad (1.2.9)$$

where (A,χ_A) is a coherent binary system, and Ψ is the

structure function of a coherent binary system whose components are those $C_i \epsilon A'$. When A is a modular set we say that (A,χ_A) is a module of (C,ϕ). The application of this definition for a fault tree is demonstrated for the sample fault tree given in Figure 1.2.1. From what has been discussed so far in this section, the modules of the fault tree can be easily obtained. They are as follows:

$(A=\{a,b,c\},\ \chi_A = a \cup b \cup c)$

where $a \cup b$ denotes the union of a and b

$(A=\{d,e,f\}.\ \chi_A = d \cup e \cup f)$

$(A=\{a,b,c,d,e,f\},\ \chi_A = (a \cup b \cup c) \cap (d \cup e \cup f))$

where $(a') \cap (b')$ denotes the intersection of a' and b'

$(A=\{g,h,i\},\ \chi_A = (g \cup h \cup i)$

$(A=\{a,b,c,d\},\ \chi_A = ((a \cup b \cup c) \cap d))$

.

.

.

One of the useful representations of a Boolean function is its pivotal decomposition about one of its coordinates, i.e.,

$$\phi(x) = x_i \cdot \phi(1_i, \underset{\sim}{x}) + (1-x_i)\phi(0_i, \underset{\sim}{x}) \qquad (1.2.10)$$

If (A,χ_A) is a module of (C,ϕ), then from Eq. (1.2.9) we obtain

$$\phi(\underset{\sim}{x}) = \chi_A(\underset{\sim}{x}^A)\Psi(1,\underset{\sim}{x}^{A'}) + [1-\chi_A(\underset{\sim}{x}^A)]\ \Psi(0,\underset{\sim}{x}^{A'}) \qquad (1.2.11)$$

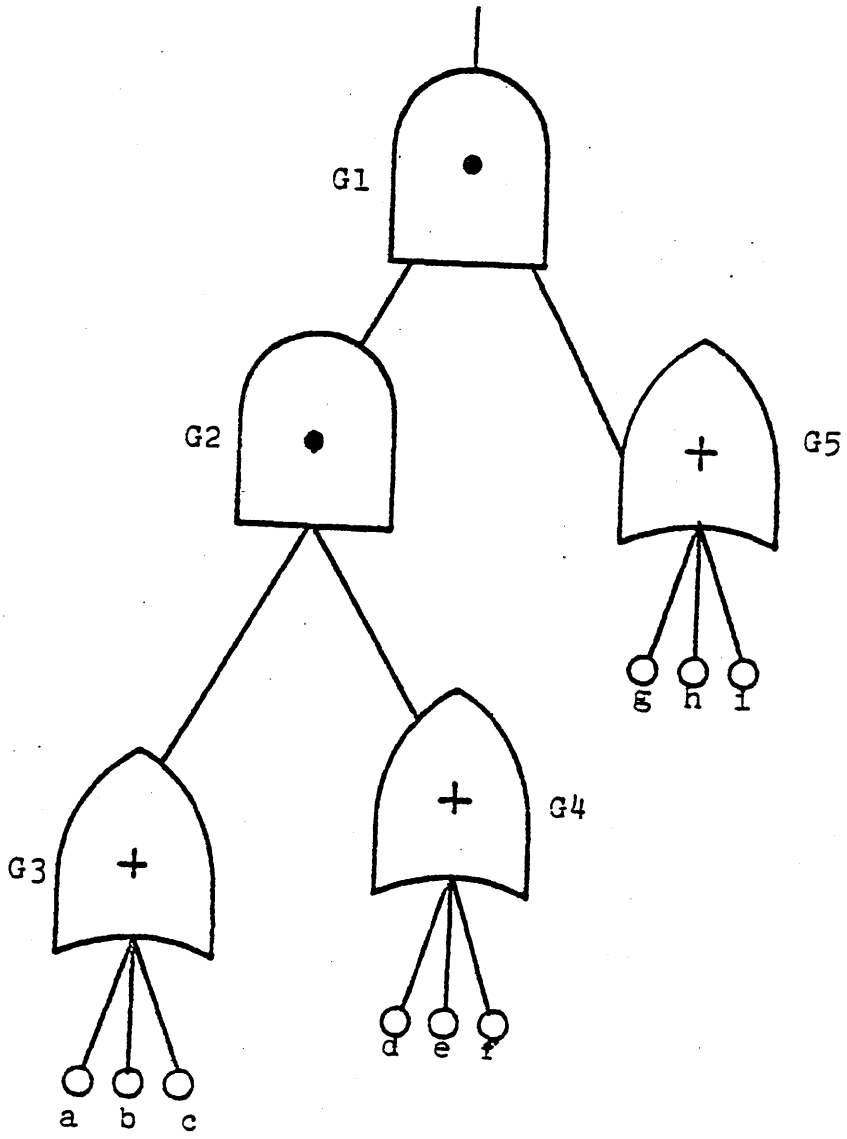


FIGURE 1.2.1

SAMPLE SUB-TREE I WITH NO REPLICATIONS

The method that is utilized in the PL-MOD code is based upon the definitions for the modularization given above. For example, the fault tree given in Figure 1.2.1 in which there are no replicated components is analyzed below by assuming that:

1. All the branches are independent, i.e., every intermediate gate event in the tree is modularizable.
2. The logic function $\chi_A(\underset{\sim}{x}^A)$ associated with each gate is either "prime" or "simple" having no inputs from other "simple" gates of the same type.

Thus, the fault tree of Figure 1.2.1 can be changed into another configuration as shown in Figure 1.2.2 which demonstrates the finest modular representation of Figure 1.2.1 and is obtained by coalescing gates G_1 and G_2. Its modular structure is given by the following set of recursive equations:

$$A_1=\{A_3,A_4,A_5\} \qquad \phi(\underset{\sim}{x})=\chi_{A_1}=\chi_{A_3}\cdot\chi_{A_4}\cdot\chi_{A_5}$$

$$A_3=\{a,b,c\}, \quad \chi_{A_3}=a \cup b \cup c$$

$$A_4=\{d,e,f\}, \quad \chi_{A_4}=d \cup e \cup f$$

$$A_5=\{g,h,i\}, \quad \chi_{A_5}=g \cup h \cup i$$

Suppose that the fault tree in Figure 1.2.1 has a replicated component as is shown in Figure 1.2.3. Then the following modules are obtained:

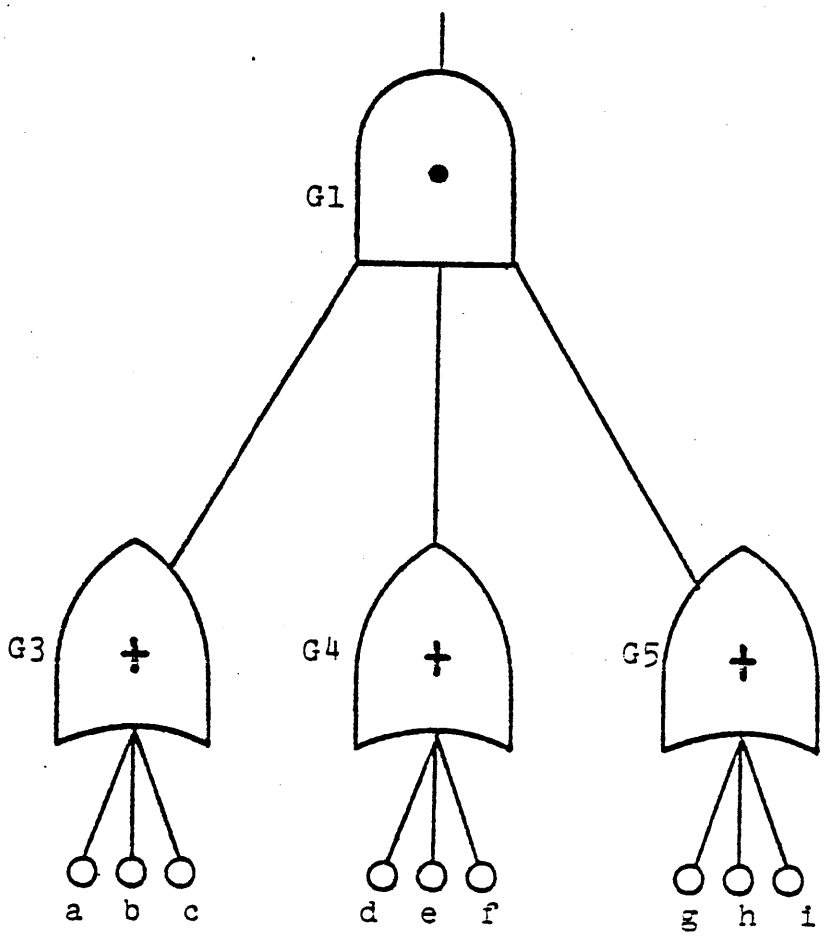


FIGURE 1.2.2

FINEST MODULAR REPRESENTATION OF SAMPLE SUB-TREE I

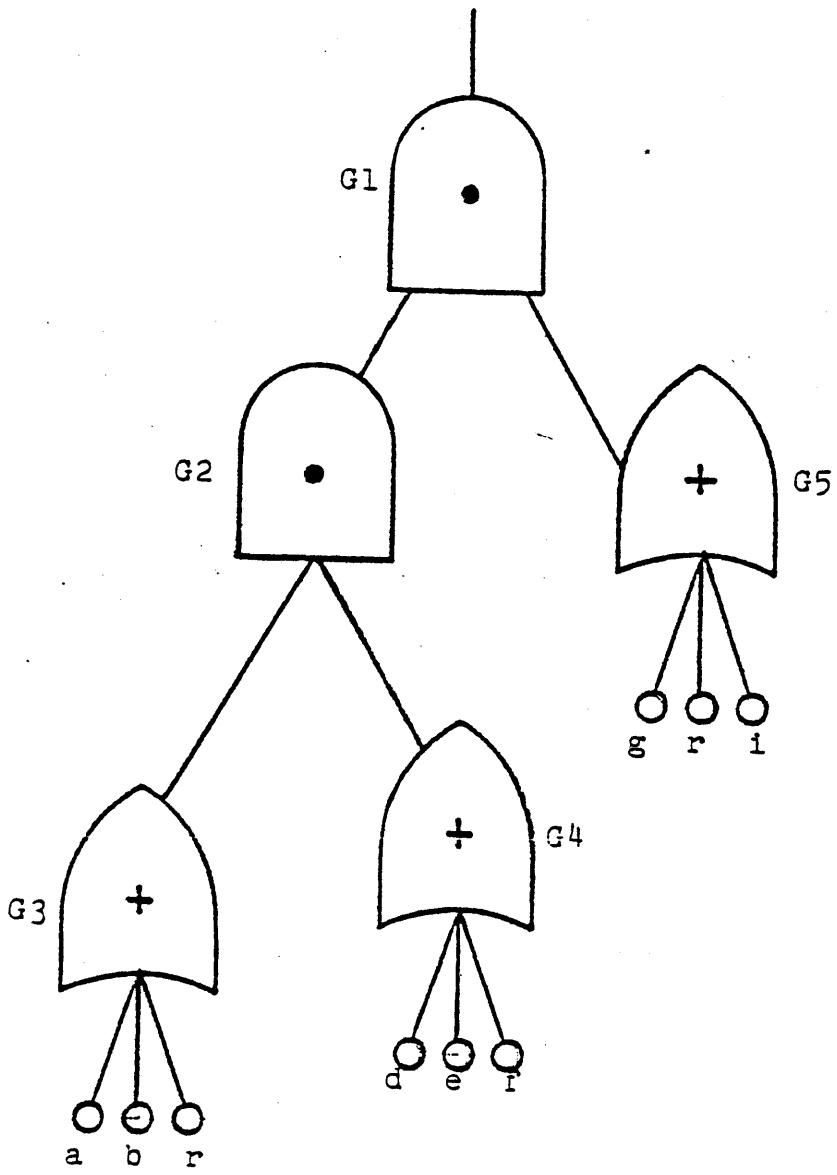


FIGURE 1.2.3

SAMPLE SUB-TREE II WITH REPLICATIONS

$A_3=\{a,b\}$, $\chi_{A_3}=a \cup b$

$A_4=\{d,e,f\}$, $\chi_{A_4}=d \cup e \cup f$

$A_5=\{g,i\}$, $\chi_{A_5}=g \cup i$

Furthermore, these modules together with the replicated event, r, will become the input to a higher order prime gate as shown in Figure 1.2.4.

$$\phi(\underset{\sim}{x}) = (x_r, \underset{\sim}{x}^{A_3}, \underset{\sim}{x}^{A_4}, \underset{\sim}{x}^{A_5})$$

with the following modular minimal cut sets

$S_1 = (1,0,1,0)$

$S_2 = (0,1,1,1)$

It must be stressed here that the algorithm given by Chatterjee [2] was devised for deriving the modular composition of a fault tree given the minimal cut set structure description of the tree. In complete contrast with this, the modularization algorithm used in PL-MOD (Figure 1.2.5) derives the modular composition of a fault tree directly from its diagram description. It is important to realize that the modular minimal cut sets which are derived from the above algorithm are compact representations of the usual basic event minimal cut sets.

Once the modular structure of a fault tree has been obtained, the quantitative evaluation of reliability and importance parameters of the tree are efficiently performed. In particular, the

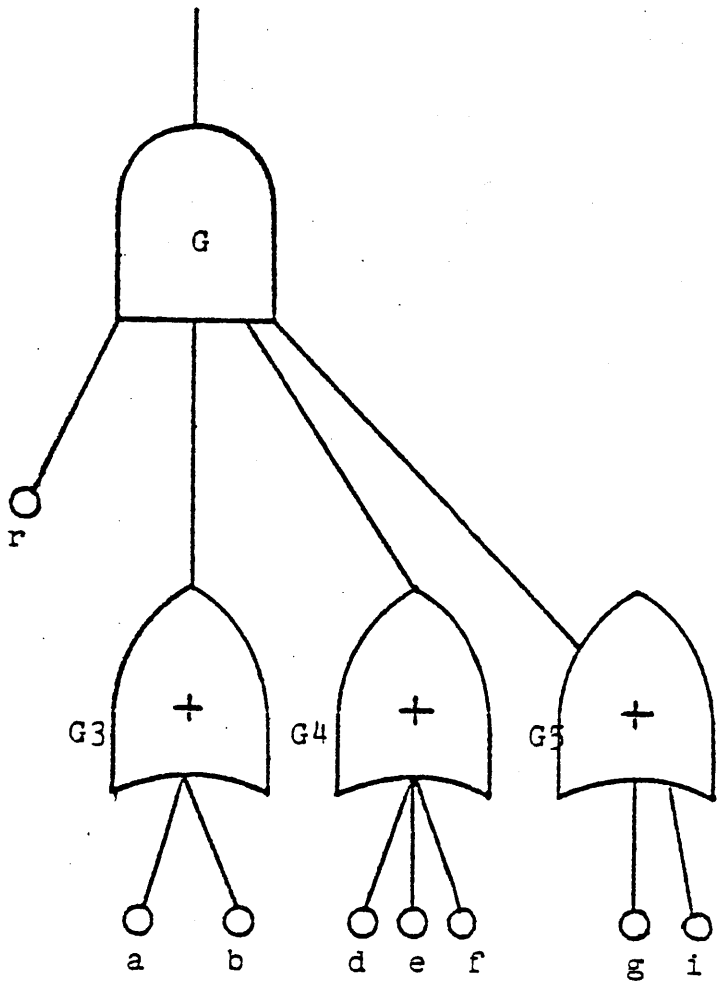


FIGURE 1.2.4

FINEST MODULAR REPRESENTATION OF SAMPLE SUB-TREE II

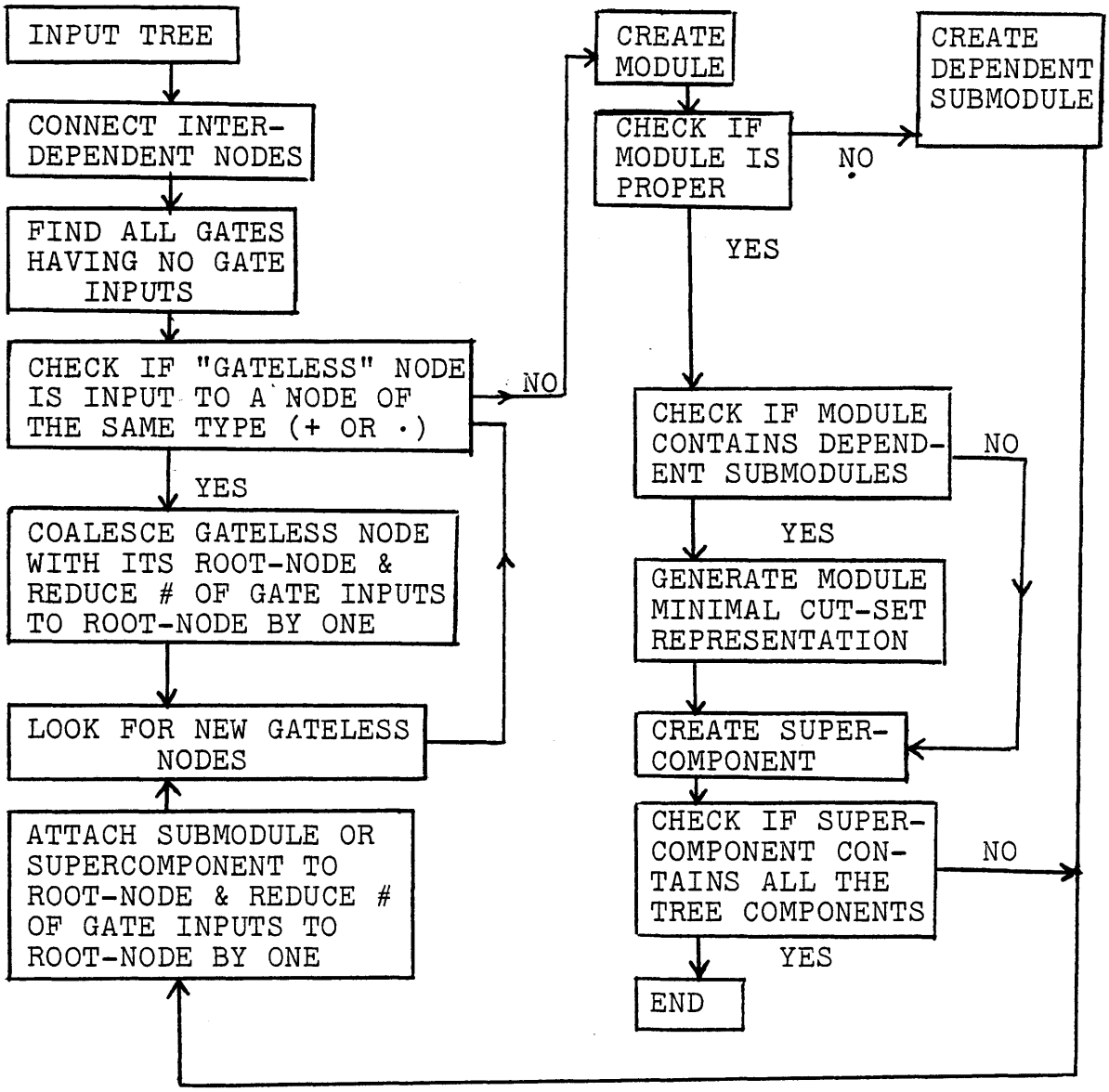


FIGURE 1.2.5

FAULT TREE MODULARIZATION ALGORITHM

probability of the occurrence of the top event is obtained by means of a series of recursive calculations requiring the evaluation of the probability expectation value of each of the modules combined in the tree.

With respect to the importance measure, Olmos [3] has shown how the Vesely-Fussell importance can be applied for modules and how the importance of basic events can be easily computed from a knowledge of the modular structure of the fault tree by successively using recursive modular equations (see Chapter 3) and the modular importance chain rule.

PL-MOD is written in PL/1 language, because it provides several features normally found only in assembler or list processing languages. The essence of list processing is the ability to dynamically allocate blocks of core storage to link these blocks together into a structure to store and retrieve data from the blocks. It should be noticed that list processing for complicated data structures such as those required by PL-MOD are very difficult if not impossible to achieve through manipulations using FORTRAN.

From the foregoing evidence the computational advantages of PL-MOD to analyze and to evaluate fault trees in a modular manner are:

a) The probabilities of the occurrence for the top and intermediate gate events are efficiently computed by evaluating these modular events in the same order in which they are generated;

b) The modular and component importance measures are easily computed by starting at the top tree event and successively using a modular importance chain rule.

c) For complex fault trees necessitating the use of minimal cut-set upper bounds for their quantification, sharper bounds will result by using minimal cut-set upper bounds at the level of modular gates.

The efficiency and the accuracy of the PL-MOD method has been demonstrated in reference [3] where PL-MOD has been tested against MOCUS.

REFERENCES

[1] Z.W. Birnbaum and J.D. Esary, "Modules of Coherent Binary Systems", J. Soc. Indust. Appl. Math., Vol. 13, No. 2, (June 1965).

[2] P. Chatterjee, "Modularization of Fault Trees: A Method to Reduce the Cost of Analysis", Reliability and Fault Tree Analysis Conference, SIAM (1975).

[3] J. Olmos, "A Modular Approach to Fault Tree and Reliability Analysis", Ph.D. Thesis, MIT Dept. of Nucl. Engng. (August 1977).

[4] J. Olmos and L. Wolf, "A Modular Approach to Fault Tree and Reliability Analysis", Transactions A.N.S. 26 (1977).

[5] R.E. Barlow and F. Proschan, "Statistical Theory of Reliability and Life Testing", Holt, Rinehart and Winston (1975).

[6] H.E. Lambert, "Fault Trees for Decision Making in Systems Analysis", UCRL-51829 (October 1975).

[7] J.B. Fussell et al., "MOCUS - A Computer Program to Obtain Minimal Sets from Fault Trees", Aeroject Nuclear Co., ANCR-1156C (August 1974).

[8] "Reactor Safety Study, Appendix II (Volume 2) PWR Fault Trees", WASH-1400 Draft (August 1974).

2. TIME DEPENDENT FAULT TREE ANALYSIS

2.1 Introduction

The use of modularization techniques in fault tree analysis provides an efficient and fast method to determine the unavailabilities of the modules and the top tree event. The modularization technique is specifically advantageous for the analysis of fault trees comprised of components with time-dependent behaviors. This is because the time-dependent evaluation involves many calculations of the same kind at different time steps.

In this study, four different classes of components are considered to be used in the PL-MOD code for evaluating the fault tree. The four classes of components are:

1. Time-independent or constant unavailability components (i.e., $\bar{A}$ = constant).
2. Nonrepairable components in which the failure rates are time independent.
3. Repairable components, failure of which is immediately detected (revealed faults). The failure and repair rates are time independent for this class of components.
4. Repairable components, failure of which is detected upon inspection (periodically tested components).

A detailed discussion of these four classes of components is provided in Sections 2.2 through 2.5. Subroutine NUMERO in the original PL-MOD code is altered such that it can handle these four classes of components mentioned above. A detailed discussion of these changes is presented in Section 2.9 in order to clarify

method utilized in the code. The new code is named PL-MODT, and the abbreviation, T, stands for the transient features of the code.

In the PL-MODT code, all features of the PL-MOD code are essentially kept the same. The new code is able to handle very large fault trees with time-dependent components as well as small trees in an efficient, fast, and economical way. During this development, an attempt has been made to incorporate most of the important features of the present time-dependent codes such as KITT [6] and FRANTIC [4] into PL-MODT. These features will be discussed in more detail later in this chapter.

PL-MODT has been benchmarked against the KITT and the FRANTIC codes, and the results obtained indicate that the code PL-MODT is adequate for time-dependent calculations. Examples of this benchmark analysis are presented in Section 2.10.

2.2 Class 1 Components: Time Independent Components

A time-independent leaf (component) in a fault tree takes into account the presence of failures whose probability of occurrence does not change during the component's operation. These failures can occur essentially because of either a natural phenomenon such as earthquake, flood, tornado, etc., or a physical phenomenon such as operator faults, airplane crash, etc. The occurrence of these faults has a specific likelihood which is not time dependent. For example, if a fault tree leaf could

be in n-different failed states given that there exists only one non-failed state, then a Markovian model can be formulated such as the one shown in Figure 2.2.1.

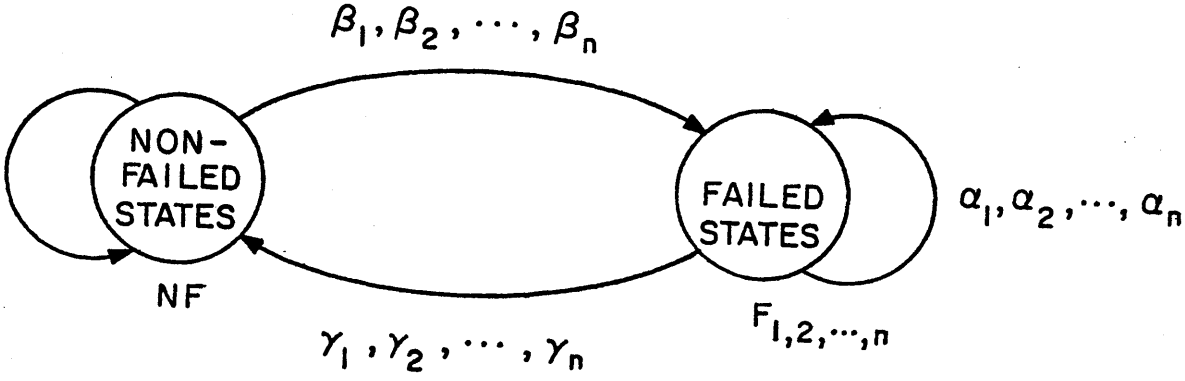


Figure 2.2.1 A Markovian model for time independent components

Each of these n failed states could be one of the failure modes of the time-independent component. For example, faults caused by an operator fault, airplane crash, or by a missile produced from a turbine failure are three different mutually exclusive faults occupying three failed states of the components.

In Figure 2.2.1, different transition probabilities are given by α, β, and δ. Therefore, the probability that the component is in its non-failed state can be calculated as follows. The transition matrix $\underline{A}$ is given by Eq. (2.2.1) below:

$$\underline{A} = \begin{array}{c} \\ NF \\ F_1 \\ F_2 \\ \vdots \\ F_n \end{array} \begin{array}{c} \begin{array}{cccc} NF & F_1 & F_2 & \cdots & F_n \end{array} \\ \begin{pmatrix} \alpha & \beta_1 & \beta_2 & \cdots & \beta_n \\ \delta_1 & \alpha_1 & 0 & \cdots & 0 \\ \delta_2 & 0 & \alpha_2 & & \\ \vdots & \vdots & \vdots & \text{All zeros} & \\ \delta_n & \text{All zeros} & & & \alpha_n \end{pmatrix} \end{array} \qquad (2.2.1)$$

If $\underline{P} = [P_{NF} \quad P_{F_1} \ldots P_{F_n}]$, then the solution of the equation $\underline{P} \cdot \underline{A} = \underline{P}$ would provide a discrete value for P_{NF} which is also a time-independent value for the probability that the leaf is in its non-failed state. The following example is given below makes the above discussion more clear.

Example: Figure 2.2.2 shows different transition probabilities for a component with one non-failed state and two failed states. The problem is to calculate the likelihood that the component is in its non-failed state.

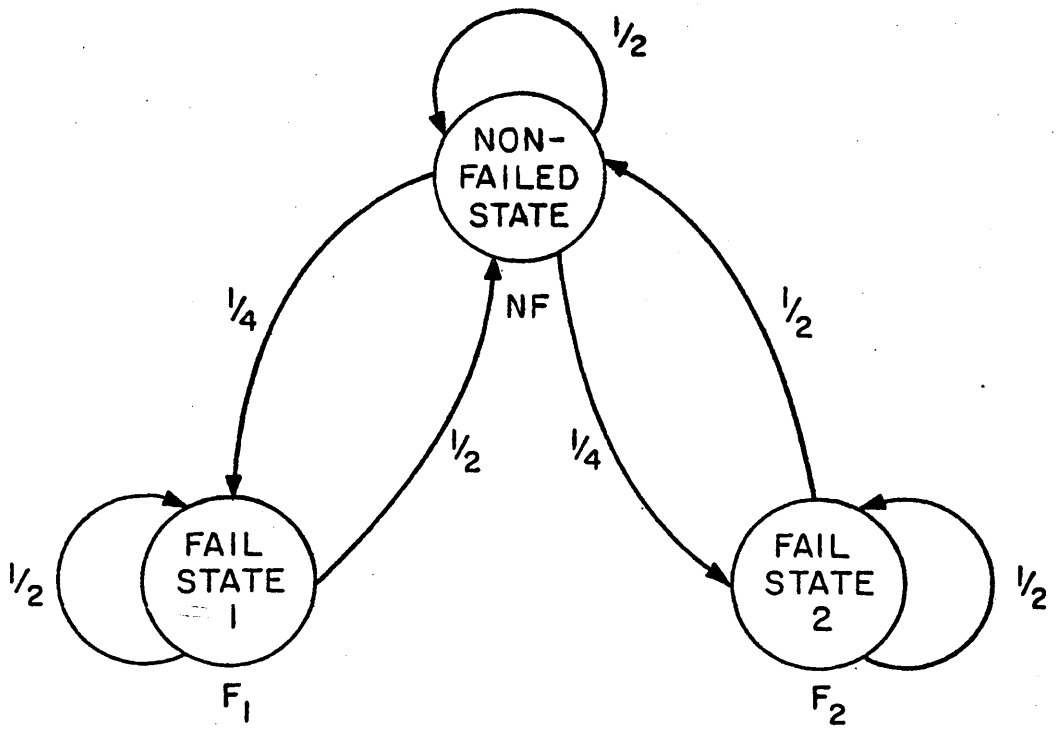


Figure 2.2.2 A Markovian model.

The transition matrix $\underline{A}$ for this example is given as:

$$\underline{A} = \begin{array}{c} \\ NF \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{ccc} NF & F_1 & F_2 \end{array} \\ \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \end{array}$$

Therefore, the matrix equation has the following form:

$$[P_{NF} \quad P_{F_1} \quad P_{F_2}]\ \underline{A} = [P_{NF} \quad P_{F_1} \quad P_{F_2}]$$

Solving the above equation one finds the matrix $\underline{P}$

$$\underline{P} = [\tfrac{1}{2} \quad \tfrac{1}{4} \quad \tfrac{1}{4}]$$

and thus $P_{NF} = \frac{1}{2}$

2.3 Class 2 Components: Nonrepairable Components

For the class 2, 3, and 4 components, the total unavailability can be divided into two separate parts: first, the probability that at an initial time (t = 0) the component K is down (unavailable); second, the probability that during the operation, component K becomes unavailable. Therefore, one can write

$$\bar{A}(t) = (1-V_i)\bar{A}_w(t) + V_i\bar{A}_d(t)$$

where

V_i = probability that K is down at t=0

$\bar{A}_v(t)$ = probability that K is down at t=t given that K was up at t=0

$\bar{A}_d(t)$ = probability that K is down at t=t given that k was down at t=0

For class 2, 3, and 4 components, we assume that $V_i=0$ which is a valid assumption most of the time for most of the components. Therefore, it is assumed that $\bar{A}(t)=\bar{A}_v(t)$.

In the code PL-MODT the unavailability equations are formulated in terms of $\bar{A}_v(t)$ since this reduces the amount of calculations.

The value of $\bar{A}_v(t)$ for any class 2 component for a mission time t may be calculated exactly by the expression

$$\bar{A}_v = 1-\exp\left[-\int_{t_1}^{t_2} h(t)\,dt\right] \qquad (2.3.1)$$

where $h(t)$ = hazard rate (instantaneous rate)

t = stated time duration of the mission which begins at time t and ends at t_2.

Eq. (2.3.1) is the general form of the unavailability and does not contain any approximation. However, since the hazard rate h(t) is time variant, the unavailability should be calculated through times of break-in as well as wear-out. Generally, data are not available to give a good description of the hazard rate through a component's lifetime. Also, it has been demonstrated

that for most components, there is a long period of useful life wherein the hazard rate is relatively constant. Under these assumptions, Eq. (2.3.1) can be approximated as

$$\bar{A}_v(t) = 1-\exp[-\lambda t] \qquad (2.3.2)$$

where λ is the constant component failure rate which is a characteristic of the exponential distribution. We will see later in Section 2.4 that for a special case, class 3 components can be reduced to class 2 and therefore Eq. (2.3.2) should be obtainable from class 3 unavailability equations. Equation (2.3.2) has been adopted in PL-MODT to calculate the unavailability of non-repairable components.

2.4 Class 3 Components: Repairable Components

For this class of components, it is assumed again that $\bar{A}(t)=\bar{A}_v(t)$. For calculating $\bar{A}_v(t)$ it would be more convenient to use a Markovian approach by using a constant failure rate $\lambda(hr^{-1})$ and a constant repair rate $\mu(hr^{-1})$. Figure 2.4.1 presents a Markovian model for this class of components

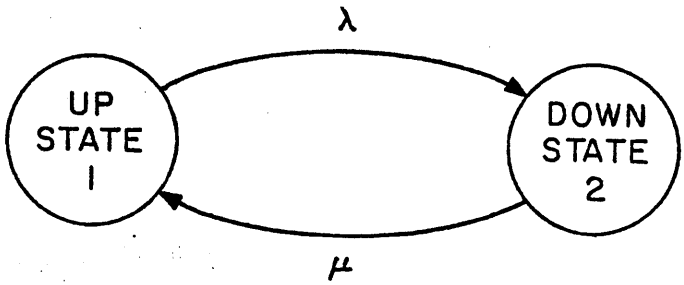


Figure 2.4.1 A Markovian model for repairable components (revealed faults)

It is assumed that the transition rates λ and μ have a probability density function f(t) and g(t), respectively. Assuming that state 1 is the UP state, and state 2 is the DOWN state, then

$P_{12}(T)$ = probability that the component K goes from 1 to 2 in the time interval Δt(t to t + Δt).

Therefore

$$P_{12} = \lambda\Delta t + O(\Delta t^2 + \ldots\ldots) \qquad (2.4.1)$$

higher order terms

Similarly, $P_{21} = \mu\Delta t$ (2.4.2)

Using Eqs. (2.4.1) and (2.4.2) we can determine the transition matrix $\overline{A}$ as follows:

$$P' = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & \quad 2 \end{array} \\ \begin{pmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ \mu\Delta t & 1-\mu\Delta t \end{pmatrix} \end{array} \qquad (2.4.3)$$

By subtracting 1 from the diagonal elements of the matrix $\underline{P}$ one is able to find the identity matrix.

$$\underline{P} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & \quad 2 \end{array} \\ \begin{pmatrix} -\lambda\Delta t & \lambda\Delta t \\ \mu\Delta t & -\mu\Delta t \end{pmatrix} \end{array} \qquad (2.4.4)$$

From the identity matrix $\underline{P}$ we get the transition matrix which reads

$$\underline{A} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix} \qquad (2.4.5)$$

To evaluate $P_2(t) = \bar{A}_v(t)$ the Laplace transform is applied.

$$(S\underline{I} - \underline{A}) = \begin{pmatrix} S+\lambda & -\lambda \\ -\mu & S+\mu \end{pmatrix} \tag{2.4.6}$$

The inverse of the matrix, Eq. (2.4.6), is calculated as follows

$$(S\underline{I}-\underline{A})^{-1} = \begin{pmatrix} \frac{S+\mu}{S(S+\lambda+\mu)} & \frac{\lambda}{S(S+\lambda+\mu)} \\ \frac{\mu}{S(S+\lambda+\mu)} & \frac{S+\lambda}{S(S+\lambda+\mu)} \end{pmatrix}$$

$$\equiv \frac{1}{S}\begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \\ \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix} + \frac{1}{S+\lambda+\mu}\begin{pmatrix} \frac{\lambda}{\lambda+\mu} & -\frac{\lambda}{\lambda+\mu} \\ -\frac{\mu}{\lambda+\mu} & \frac{\mu}{\lambda+\mu} \end{pmatrix} \tag{2.4.7}$$

Taking the inverse of Eq. (2.4.7) we can evaluate $\bar{A}_v(t)$

$$P(t) = \begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \\ \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix} + e^{-(\lambda+\mu)t}\begin{pmatrix} \frac{\lambda}{\lambda+\mu} & -\frac{\lambda}{\lambda+\mu} \\ \frac{-\mu}{\lambda+\mu} & \frac{\mu}{\lambda+\mu} \end{pmatrix} \tag{2.4.8}$$

From Eq. (2.4.8) it follows that

$$\bar{A}_v(t) = P_2(t) = \frac{\lambda}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)t}\right) \tag{2.4.9}$$

Equation (2.4.9) approaches an asymptotic value of

$$\bar{A}_v(t) = \frac{\lambda}{\lambda+\mu} \tag{2.4.10}$$

as the time t gets large.

Equation (2.4.10) could be reformulated as follows:

$$\bar{A}_v = \frac{\tau_r}{\theta+\tau_r} \tag{2.4.11}$$

where $\theta = \frac{1}{\lambda}$ = mean time between failures (MTBF)

and $\tau_r = \frac{1}{\mu}$ = mean time to repair (MTTR)

Sometimes Eq. (2.4.10) is given in the following form:

$$\bar{A}_v = \frac{\lambda\tau_r}{\lambda\tau_r+1} \tag{2.4.12}$$

Equation (2.4.9) can be used to determine unavailability of non-repairable components. It is known that for non-repairable components the repair rate is zero because no repair is conducted which corresponds to $\tau_r = \infty$. Therefore, by setting $\mu=0$ in Eq. (2.4.9), it reduces to the form given by Eq. (2.3.2), i.e., $\bar{A}_v(t)=1-e^{-\lambda t}$.

Equation (2.4.9) is used in the code PL-MODT to determine the unavailability of components and some simple modules created during the fault tree modularization.

2.5 Class 4 Components: Periodically Tested Components

Suppose that the component k is inspected at the times t_1, $t_2,\ldots,t_n$. If k is found to be failed, it will therefore be repaired.

θ_n = time needed to inspect an intact component at the n-th inspection.

τ_n = time needed to repair a failed component at the n-th inspection; and

$\theta_n + \tau_n$ = time needed to inspect a failed component at the n-th inspection.

In that case,

$$\theta_n + \tau_n < t_{n+1} - t_n$$

Assuming that the times t_n are known (inspection times), then the unavailability for any time t given $t_n < t < t_{n+1}$ could be calculated.

If

$$x_n = t_n - t_{n-1} - \theta_{n-1} \quad \text{for } n>1$$

$$y_n = x_n - \tau_n - 1 \quad \text{and}$$

$$x_1 = t_1$$

Then, the unavailability at different times $t_1, t_2, \ldots, t_n$ would be

$$\bar{A}_v(t_1) = F(x_1)$$

$$\bar{A}_v(t_2) + F(x_1)F(y_2)+[1-F(x_1)]F(x_2) = F(x_2)-[F(x_2)-F(y_2)]F(x_1)$$

$$\bar{A}_v(t_3) = \{F(x_2)-[F(x_2)-F(y_2)]F(x_1)\}\ F(y_3) + \{1-F(x_2)+[F(x_2)-F(y_2)]F(x_1)\}\ F(x_3) =$$

$$= F(x_3)-[F(x_3)-F(y_3)]F(x_2)+[F(x_3)-F(y_3)][F(x_2)-F(y_2)]F(x_1)$$

$$\vdots$$

$$\bar{A}_v(t_n) = \sum_{j=1}^{n} (-1)^{n-j} F(x_j) \prod_{k=j+1}^{n} [F(x_k)-F(y_k)] \qquad (2.5.1)$$

Equation (2.5.1) can be considerably simplified, and in the particular case of equal inspection time and inspection period,

that is:

$$\text{Inspection interval} = t_2 - t_1 = t_3 - t_2 = \ldots = t_n - t_{n-1} = \eta$$

$$\text{Inspection period (Duration)} = \theta_1 = \theta_2 = \ldots = \theta_n = \theta$$

$$\text{Repair period (Duration)} = \tau_1 = \tau_2 = \ldots = \tau_n = \tau$$

Equation (2.5.1) becomes

$$\bar{A}_v(t_n) = F(\eta-\theta) \sum_{j=1}^{n} [F(\eta-\theta-t)-F(\eta-\theta)]^{n-j} + [F(t_1)-F(\eta-\theta)]$$

$$[F(\eta-\theta-t)-F(\eta-\theta)]^{n-1} \qquad (2.5.2)$$

Equation (2.5.2) can be written as follows:

$$\bar{A}_v(t_n) = F(\eta-\theta)\frac{1-[F(\eta-\theta)-F(\eta-\theta-\tau)]^n}{1+F(\eta-\theta)-F(\eta-\theta-\tau)} +$$

$$[F(t_1)-F(\eta-\theta)][F(\eta-\theta-\tau)-F(\eta-\theta)]^{n-1} \qquad (2.5.3)$$

Since $F(\eta-\theta)-F(\eta-\theta-\tau)$ in Eq. (2.5.3) is usually very small, then Eq. (2.5.3) can be approximated by

$$\lim_{n\to\infty} \bar{A}_v(t_n) = \frac{F(\eta-\theta)}{1+F(\eta-\theta)-F(\eta-\theta-\tau)} \qquad (2.5.4)$$

To calculate the unavailability as a function of time, we can apply the theorems of the sum of the probabilities and of the conditional probability. Therefore one can write

$$\bar{A}_v(t) = \bar{A}_v(t_n)\,\beta(t_n,t) + [1-\bar{A}_v(t_n)]\,\alpha(t_n,t) \qquad (2.5.5)$$

where

$\alpha(t_n,t)$ = probability that the component k is down at time t given k was up at t_n and,

$\beta(t_n,t)$ = probability that component k is down at time t given k was down at t_n.

Therefore,

$$\alpha(t_n,t) = 1(t-t_n) - 1(t-t_n-\theta_n) + F(t-t_n-\theta_n) \qquad (2.5.6)$$

and

$$\beta(t_n,t) = 1(t-t_n)-1(t-t_n-\theta_n-\tau_n)+F(t-t_n-\theta_n-\tau_n) \qquad (2.5.7)$$

where

$1(t-t_n)$ is a unit step function given as follows:

$$1(x) = \begin{cases} 1 & \text{for } x \geq 0_+ \\ 0 & \text{for } x \leq 0_- \end{cases}$$

Now if $F(t) = 1-\exp[-\lambda t]$ where λ is the constant failure rate of the component, after some approximations [2] we obtain

$$\bar{A}_v(t) = 1-e^{-(t-m\eta)\lambda eff}\,[1-e^{-(\frac{t-m\eta}{\theta})^q}] \qquad (2.5.8)$$

where

$$\lambda eff = \frac{\eta-\theta}{\eta}(\frac{\eta-\theta}{\eta}+\frac{2\tau}{\eta})+2[1-\frac{\Gamma(\frac{1}{q})}{q}]\frac{\theta}{\eta^2} \qquad (2.5.9)$$

$\Gamma(\frac{1}{q})$ = gamma function of $\frac{1}{q}$ and,

$q = L_n(3-L_n\theta\lambda)$ and, (2.5.10)

$m = 1, 2, \ldots, n$

Figure 2.5.1 shows the unavailability calculated by Eq. (2.5.8) for one inspection interval and the following data for a specific periodically tested component

$$\theta = 1.5 \text{ hrs}, \quad \tau = 19 \text{ hrs}, \quad \eta = 720 \text{ hrs}, \quad \lambda = 3\text{x}10^{-6}\text{hr}^{-1}$$

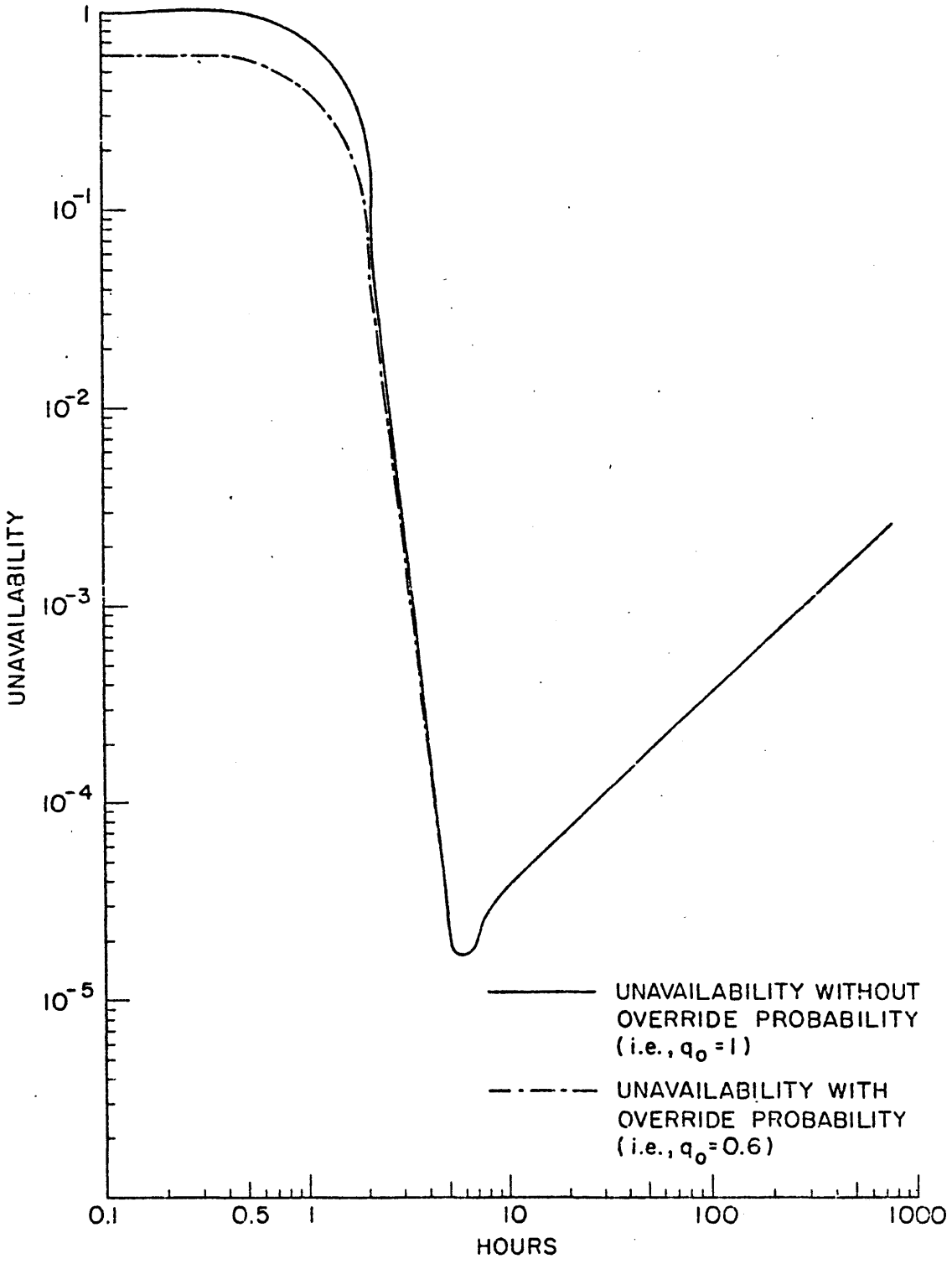


FIGURE 2.5.1

UNAVAILABILITY OF A COMPONENT AS A FUNCTION OF TIME USING EQ. (2.5.8)

It should be noted here that Eq. (2.5.8) can be put into a simpler form. For instance, take

$$[1 - e^{-(\frac{t-m\eta}{\theta})^q}] \quad \text{part of Equation (2.5.8)}$$

For typical values of q, η, and θ, this part of the equation approaches one about the end of the inspection period (i.e., for $t>m\eta+\theta$). This results, however,in an unavailability equation as follows

$$\bar{A}_v(t) = 1-e^{-(t-m\eta)\lambda eff}$$

Therefore, if this approximation is used for the operating times $t>m\eta+\theta$, some computing time can be saved. Thus, for only a small fraction of time θ, the complete Eq. (2.5.8) will be used whereas for the rest of the test interval the simplified form can be applied. (Note θ is always in the order of a few hours and η is in the order of several days or even a few months.) The simplified form of Eq. (2.5.8) is not incorporated into PL-MODT yet, but it is recommended to use this equation instead of Eq. (2.5.8) for $t>m\eta+\theta$ in the future.

Equation (2.5.8) provides the unavailability of a periodically tested component with equal test intervals. It is known, however, that most of the time the first test interval is longer than the subsequent ones and, therefore, Eq. (2.5.8) is not quite adequate for this interval. Eq. (2.5.8) must be modified such that the longer test interval can be handled. Specifically at t=0 we have $\bar{A}_v(0)=0$ and $\eta= \eta_1$ for the first test interval, with $\eta_2=\eta_3=\ldots\eta_n=\eta'$

for subsequent ones. A new form of Eq. (2.5.8) is actually utilized in the PL-MODT code for the first test interval by changing $t \rightarrow t+\theta$ and setting $m=1$, and $\eta=\eta_1$. Therefore, Eq. (2.5.8) becomes

$$\bar{A}_{v_1}(t) = 1-e^{-(t+\theta-\eta_1)\lambda eff} \cdot [1-e^{-(\frac{t-\eta_1}{\theta}+1)^q}] \qquad (2.5.11)$$

Also, it should be noticed that in the PL-MODT the second term in Eq. (2.5.9) is approximated as follows

$$2[1-\frac{\Gamma(1/q)}{q}]\theta/\eta^2 = 0.2 \times \theta/\eta^2 \qquad (2.5.12)$$

This approximation will save substantial amounts of computation time and will not change the computed unavailabilities since θ/η^2 is small compared to the first term of Eq. (2.5.9).

Override probability* P can be accounted for in Eq. (2.5.8) by simple multiplying $\bar{A}_v(t)$ during the inspection period (θ) by the override probability P. Therefore,

$$\bar{A}_{v_o}(t) = \bar{A}_v(t) \cdot P \qquad (2.5.13)$$

where P is given by

$$P = q_o + (1-q_o)(1-e^{-\lambda eff \cdot \eta}) \qquad (2.5.14)$$

and q_o = override unavailability.

Eq. (2.5.14) can be derived by using Eq. (2.5.8) to calculate the unavailability at the end of the test interval. The effect of

*Override unavailability is the probability that a component cannot function properly during its inspection period if it is demanded. Therefore, with an override unavailability equal to one, the component is totally unavailable during the inspection interval.

incorporating an override probability is also shown in Figure 2.5.1 for q_o=0.6. The overall effect is a reduction in the average unavailability of the component and thus an increase in the system availability.

2.6 General Time Dependent Relations for the Evaluation of Fault Trees by Using the Modular Concept

Take a module with a set $\{m_1, m_2, \ldots, m_n\}$. The structure function for this module will be

$$\sigma_M(t) = \beta[\sigma_1(t), \sigma_2(t),\ldots,\sigma_m(t)] \qquad (2.6.1)$$

where

$$\sigma_i(t) = \sigma_{mi}(t) \quad (1 = 1, 2,\ldots,n) \quad \text{also,}$$

$$\sigma_i = \begin{cases} 1 \text{ when module i has occurred at time t} \\ 0 \text{ otherwise} \end{cases}$$

The expectation value of Eq. (2.6.1) is as follows

$$h_\sigma(t) = E\{\sigma_M(t)\} \qquad (2.6.2)$$

Similar to the steady state analysis, for a simple AND gate module in which $M = \{M_1, M_2, \ldots, M_n, \Omega\}$ we have

$$h_\sigma(t) = E\{\sigma_M(t)\} = h_{\sigma_1}(t)\cdot h_{\sigma_2}(t)\ldots h_{\sigma_m}(t) = \prod_{i=1}^{n} h_{\sigma_i}(t) \qquad (2.6.3)$$

Similar expressions can be derived for an OR gate module, namely for a set M

$$M = \{M_1, M_2, \ldots, M_n, U\}$$

$$h_\beta(t) = 1-[1-h_{\sigma_1}(t)][1-h_{\sigma_2}(t)]\ldots(1-h_{\sigma_n}(t)] = \coprod_{i=1}^{n} h_i(t) \quad (2.6.4)$$

For higher order modules, we have the following relationship

$$\sigma_M(t) = \coprod_{j=1}^{N_k} \prod_{i\varepsilon k_j} \sigma_i(t) \quad (2.6.5)$$

where N_k=number of modules and components connected to a higher order module or the top event.

Using the minimal cut-set upper bound formula, one obtains

$$h_\beta(t) = \coprod_{j=1}^{N_k} \prod_{i\varepsilon k_j} h_{\sigma_i}(t) \approx \sum_{j=1}^{N_k} \prod_{i\varepsilon k_j} h_{\sigma_i}(t) \quad (2.6.7)$$

which is simply the union of modules and components which are attached to a higher order module or the top event.

2.7 General Relations for Time Dependent Simple Modules Consisting of Only Repairable Components (Class 3 Components)

The unavailability of a repairable component as given by Eqs. (2.4.9) and (2.4.10) can be used to derive an approximate failure rate Λ and a mean dead time τ for simple AND and OR gates.

A simple AND gate or module is a module consisting of only simple component inputs. From Eq. (2.4.10) by employing the common assumption that $\lambda_i\tau_i << 1$, one obtains

$$\bar{h}_i(t) = \bar{F}_i \lesssim \lambda_i\tau_i \quad (2.7.1)$$

where

$\bar{h}_i(t)$ = unavailability of the component i input to a module.

Once these basic relationships have been established, the next step is to find the failure and repair rates of the module.

The primary event in an AND module can occur in the time interval t to t+dt, with the remaining events having already occured at time t, or the second primary event can occur in the time interval t to t+dt, with the remaining events having already occured at time t, or.... Keeping these observations in mind, the following equation is obtained

$$f(t) = pr\{F \cap R\} = \sum_{j=1}^{n} F_j(t)\lambda_i.dt \prod_{\substack{i=1 \\ i\neq j}}^{n} \overline{F}_i(t) \qquad (2.7.2)$$

where n = number of components input to the AND gate

$\overline{F}(t) = 1-F(t)$

f(t) = probability density function (p.d.f.) of an AND gate.

The p.d.f. can also be obtained as follows: If a cumulative probability density function (c.p.d.f.) for an AND gate is designated as $\overline{H}(t)$, then

$$\overline{H}(t) = \prod_{i=1}^{n} \overline{F}_i(t) \qquad (2.7.3)$$

By substituting Eq. (2.7.3) into Eq.(2.7.2), the following expression results

$$f(t) = \sum_{j=1}^{n} F_j(t)\lambda_j \times \frac{H(t)}{\overline{F}_j(t)} \qquad (2.7.4)$$

Since $F_j(t)$ is close to unity and the remaining terms are very small, Eq. (2.7.4) can be approximated as

$$f(t) \lesssim \bar{H}(t) \sum_{j=1}^{n} \frac{\lambda_j}{\bar{F}_j(t)} \tag{2.7.5}$$

By substituting Eqs. (2.7.1) and (2.7.3) into Eq. (2.7.5), the following result is obtained:

$$f(t) = f = \prod_{i=1}^{n} \lambda_i \tau_i \sum_{j=1}^{n} \frac{1}{\tau_j} \tag{2.7.6}$$

By using the general hazard rate formula, it is possible to calculate Λ for the AND gate as

$$\Lambda(t) = \frac{f(t)}{R(t)} \tag{2.7.7}$$

where

$R(t)$ = reliability of the AND gate.

$R(t)$ may be approximated by $R(t) \simeq 1$ since the numerator is small. Therefore, Eq. (2.7.7) reduces to the following form $\Lambda_{AND}(t) = f(t)$ or, from Eq. (2.7.6),

$$\Lambda_{AND} \simeq \prod_{i=1}^{n} \lambda_i \tau_i \sum_{i=1}^{n} \frac{1}{\tau_i} \tag{2.7.8}$$

Similarly, by using the definition of the mean dead time for a simple gate, it follows that

$$\tau(t) = \frac{\bar{H}(t)}{H(t)\ \Lambda(t)} \tag{2.7.9}$$

Therefore, the mean dead time for a simple AND gate can be obtained from the definition in Eq. (2.7.9)

$$\tau_{AND} = \frac{\prod_{i=1}^{n} \lambda_i \tau_i}{(1 - \prod_{i=1}^{n} \lambda_i \tau_i)(\prod_{i=1}^{n} \lambda_i \tau_i \sum_{j=1}^{n} \frac{1}{\tau_j})} \qquad (2.7.10)$$

Since $\lambda_i \tau_i << 1$, it can be assumed that $(1 - \prod_{i=1}^{n} \lambda_i \tau_i) \simeq 1$ and thus Eq. (2.7.10) can be written as follows

$$\tau_{AND} = \frac{1}{\sum_{j=1}^{n} \frac{1}{\tau_j}} \qquad (2.7.11)$$

In a similar way, one can obtain the value of $\Lambda(t)$ and $\tau(t)$ for OR gates. The first primary event in an OR gate can occur in the time interval t to t+dt with the remaining events not having occurred at time t, or the second primary event can occur in the time interval t to t+dt with the remaining events not having occurred at time t or,....

Similar to Eq. (2.7.2), the following equation results.

$$g(t).dt = \sum_{j=1}^{n} G_j(t) \lambda_j dt \prod_{\substack{i=1 \\ i \neq j}}^{n} G_i(t) \qquad (2.7.12)$$

Applying the same approximations made for Eq. (2.7.2), Eq. (2.7.12) can be reduced to the following form

$$g(t) = L(t) \sum_{j=1}^{n} \lambda_j \qquad (2.7.13)$$

where

$$L(t) = \prod_{i=1}^{n} G_i(t) \simeq 1 - \sum_{i=1}^{n} \bar{G}_i(t)$$

From Equation (2.7.13) it follows that

$$\Lambda_{OR} = g = (1 - \sum_{i=1}^{n} \lambda_i \tau_i) \sum_{i=1}^{n} \lambda_i \qquad (2.7.14)$$

Using a definition similar to that in Eq. (2.7.9) for the dead time,

$$\tau(t) = \frac{\bar{L}(t)}{L(t)\ \Lambda(t)}$$

By approximating $\prod_{i=1}^{n} \lambda_i \tau_i$ as $\sum_{i=1}^{n} \lambda_i \tau_i$, then τ_{OR} follows as

$$\tau_{OR} = \frac{\sum_{i=1}^{n} \lambda_i \tau_i}{(1 - \sum_{j=1}^{n} \lambda_i \tau_i)^2 \sum_{i=1}^{n} \lambda_i} \qquad (2.7.15)$$

A special case in which $(1 - \sum_{i=1}^{n} \lambda_i \tau_i)^2$ is close to unity would result in very simple forms for Λ_{OR} and τ_{OR}, namely,

$$\Lambda_{OR} = \sum_{i=1}^{n} \lambda_i \qquad (2.7.16)$$

and

$$\tau_{OR} = \sum_{i=1}^{n} \lambda_i \tau_i \ / \sum_{i=1}^{n} \lambda_i \qquad (2.7.17)$$

The values obtained for Λ and τ for simple <u>AND</u> and <u>OR</u> modules can be further investigated. For example, take Eqs. (2.7.8) and (2.7.11) for an AND module. If we were to approximate this module behaving as a simple component, with the same approximation stated in Eq. (2.7.1), it follows that the unavailability of the module is given by

$$\bar{A}_{AND} = \Lambda_{AND} \cdot \tau_{AND} = \prod_{i=1}^{n} \lambda_i \tau_i \qquad (2.7.18)$$

Equation (2.7.18) can also be obtained by using the asymptotic unavailabilities of individual input components to this AND gate. Similarly, from Eqs. (2.7.16) and (2.7.17) it follows that

$$\bar{A}_{OR} = \Lambda_{OR} \cdot \tau_{OR} = \sum_{i=1}^{n} \lambda_i \tau_i \qquad (2.7.19)$$

which is again the exact asymptotic unavailability of this OR module.

Equations (2.7.18) and (2.7.19) show that the approximation of assuming that a simple module behaves as a simple component would have no effect whatsoever on the simple module's asymptotic unavailability, if and only if $\Lambda \cdot \tau << 1$.

The above discussion indicates that within a certain range of time, Eq. (2.4.9) could be used to determine the unavailability of the module. By examining many typical simple modules, it has been found that the approximation of assuming that modules would behave as components will not provide an adequate value for the unavailability of the module for times $t < 2\tau$. Figure 2.7.1 shows a comparison between the exact unavailability of a typical simple AND module with two input components; Table 2.7.1 summarizes the parametric characteristics of these input components.

Table 2.7.1

Component Input to a Simple AND Module

Input Component	Failure Rate (hr^{-1})	Mean Dead Time (hr)
1	3×10^{-4}	100
2	2×10^{-5}	150

Therefore,

$$\Lambda_{AND} = 9 \times 10^{-5} \times \frac{1}{60} = 1.5 \times 10^{-6} \text{ hr}^{-1}$$

$$\tau_{AND} = \frac{1}{\frac{1}{100} + \frac{1}{150}} = 60 \text{ hrs}$$

2.8 General Relationships for a Time-Dependent Simple Module Consisting of Only Non-Repairable Components (Class 2 Components)

For a non-repairable component, the unavailability is given by Eq. (2.3.2). For a simple component i, the unavailability would be

$$\overline{F}_i(t) = 1 - e^{-\lambda t} \qquad (2.8.1)$$

For small values of $\lambda_i t$ we can approximate Eq. (2.8.1) as $\overline{F}_i(t) \simeq \lambda_i t$. Therefore, by using Eq. (2.6.3) one obtains under these circumstances

$$\Lambda_{AND}(t) = \left(\prod_{i=1}^{n} \lambda_i\right) t^{n-1} \qquad (2.8.2)$$

where n = number of inputs to the AND gate. For the case of an

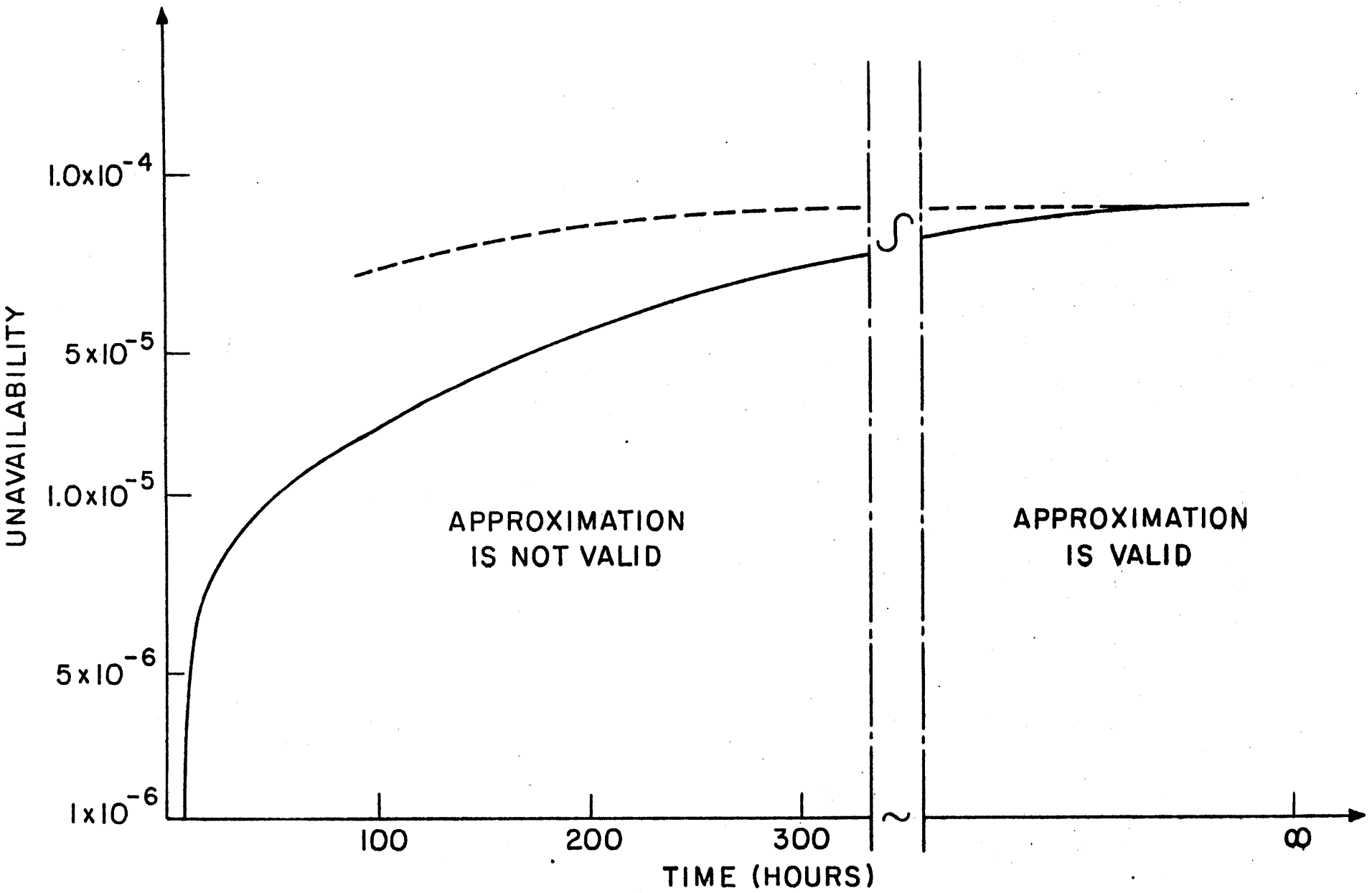


FIGURE 2.7.1

UNAVAILABILITY OF AN AND MODULE AS A FUNCTION OF TIME

OR gate, Λ_{OR} is derived by using Eq. (2.6.4)

$$\Lambda_{OR} = \sum_{i=1}^{n} \lambda_i \qquad (2.8.3)$$

Treating an OR module as a simple component will result in exactly the same unavailabilities and no approximation is involved by using Eq. (2.8.3). Treating an AND gate as a simple component by using Eq. (2.8.2) is possible only if $\lambda_i t$ is small, namely, $\lambda_i t<0.1$. In that case, Eq. (2.8.2) provides the failure rate of a simple AND module. Therefore, for $t>0.1/\lambda_i$ the approximation is no longer valid and the general equation

$$\overline{A}(t) = \prod_{i=1}^{n} \overline{F}_i(t) \quad \text{must be used.}$$

Table 2.8.1 summarizes the formulas discussed in this section.

2.9 Description of the PL-MODT Code

The original PL-MOD code consists of only Class 1 components, and therefore it is only able to evaluate time-independent components. The other three classes of components have been incorporated into the PL-MOD code, and thus the extended version is called PL-MODT, which also comprises all the other features of the PL-MOD code. The incorporation of Class 2,3, and 4 components is performed in five separate steps.

Step 1 includes the development of subroutine SHOHREH, which evaluates time-dependent unavailability of fault trees consisting of only non-repairable components (Class 2 components).

Table 2.8.1

Failure Characteristics of Simple Modules
Having Repairable or Non-repairable Component Inputs

Type of Module	Repairable Components		Non-repairable Components
	Λ	τ	Λ
AND	$\left(\prod_{i=1}^{n} \lambda_i \tau_i\right) \sum_{i=1}^{n} \frac{1}{\tau_i}$	$\frac{1}{\sum_{i=1}^{n} \frac{1}{\tau_i}}$	$\left(\sum_{i=1}^{n} \lambda_i\right) t^{n-1}$
OR	$\sum_{i=1}^{n} \lambda_i$	$\frac{\sum_{i=1}^{n} \lambda_i \tau_i}{\sum_{i=1}^{n} \lambda_i}$	$\sum_{i=1}^{n} \lambda_i$
K-O-N	$\sum_{k=1}^{m^*}\left(\prod_{i=1}^{k} \lambda_i \tau_i\right) \sum_{i=1}^{k} \frac{1}{\tau_i}$	$\frac{\sum_{k=1}^{m} \prod_{i=1}^{k} \lambda_i \tau_i}{\sum_{k=1}^{m}\left(\prod_{i=1}^{k} \lambda_i \tau_i\right) \sum_{i=1}^{k} \frac{1}{\tau_i}}$	$\sum_{i=1}^{m}\left(\prod_{i=1}^{k} \lambda_i\right) t^{k-1}$

* $m = {}_kC_N = \frac{N!}{k!(N-k)!}$

Step 2 includes the development of subroutine CECA which evaluates time-dependent unavailability of fault trees consisting of only repairable components (Class 3 components).

Step 3 includes the development of subroutine SHARAREH which evaluates time-dependent unavailability of fault trees consisting of only periodically tested components (Class 4 components).

Step 4 consists of a further development of the old NUMERO subroutine to enable it to calculate the average unavailability over the period of system operation. In addition, the old subroutines PLUS and EXPECT were modified to allow the code the treatment of unavailabilities which are close to one. This seems necessary for Class 4 components during the component inspection period, where unavailabilities are usually close to 1.

Step 5 comprises the development of the SHARAREH subroutine such that the PL-MODT code contains the ability to handle large fault trees consisting of a combination of different classes of components with different time-dependent behaviors.

These five steps are discussed in full detail in the following sections.

2.9.1 Developments Included in Step 1

In the SHOHREH procedure, the value of the unavailability for simple AND modules is calculated at different time steps (DEL9T(1,j)) by using Eqs. (2.8.2) and (2.3.2) up to a time t where $t=0.1/\lambda$ and λ is the largest failure rate of the components on the module. After the time t, the components are considered individually and therefore only Eq. (2.3.2) is used.

For the simple OR modules, the unavailability is calculated by using Eqs. (2.8.3) and (2.3.2). No limitation exists for the OR module and thus Eqs. (2.8.2) and (2.3.2) are used for the entire operational period.

The total number of time steps in each time interval mesh is calculated by

$$TIE(1,I) = \frac{AUN(1,I)}{DEL9T(1,I)} .;$$

where AUN(1,I) is the duration in which the time step DELqT(1,I) is applied and,

DEL9T(1,I) is the time step for the I-th mesh interval.

At each TIE(1,I) time step, first the unavailability values are calculated for the various modules and components of the fault tree using Eq. (2.3.2). Next, these unavailabilities are assigned to the arrays STATE(1,I) and STATD(1,I). Finally, the subroutines EXPECT, DOT, PLUS, MINUP, and IMPORTANCE are called respectively to calculate the top event and higher order module's unavailability as well as the Vesely-Fussell importance measures. The same procedure is applied for all of the other time interval meshes (AUN(1,1), AUN(1,2),...,AUN(1,N)). At the end, the mean unavailability is calculated when the time exceeds $T= \sum_{I=1}^{N} AUN(1,I)$ and the program stops.

As explained before, for each time interval mesh there exists a corresponding time step DEL9T(1,I) and, therefore, a corresponding number of time steps TIE(1,I). Thus, the total number of unavailability values calculated for the top event would be

$$TIE = \sum_{I=1}^{N} TIE(1,I)$$

where N = total number of time interval meshes used.

2.9.2 Developments Included in Step 2

In the second step, the CECA subroutine was developed. Similar to the SHOHREH subroutine, different time interval meshes along with their corresponding time steps are used. These values are stored in the allocated arrays AUN(1,I) and DEL9T(1,I). Four other arrays--STSTS, STATT, STATTE, and STATED--are allocated to store the failure and repair rates of free and replicated components, respectively.

At the beginning of the operation, the approximations discussed in Section 2.8 of this chapter are not applicable due to the small unavailability values and, therefore, the unavailabilities of replicated and free components must be calculated directly by using Eq. (2.4.9). As time progresses the components approach their asymptotic unavailability values and, therefore, when the unavailability errors are small enough (usually when $t \simeq 3\tau$), the code automatically uses the approximations summarized in Table 2.8.1. As discussed before, these formulas asymptotically approach the exact unavailability for simple modules..

Unavailability values calculated at each time step will be again assigned to the STATE (1,I) and STATD(1,I) arrays. After calling subroutines EXPECT, DOT, PLUS, MINUP and IMPORTANCE, the top event and higher order module's unavailability will be calculated. When

no time interval meshes remain, the average unavailability will be determined by using the procedure incorporated into the NUMERO subroutine. This will be discussed in Section 2.9.4.

2.9.3 Developments Included in Step 3

In this step, the SHARAREH subroutine is developed for periodically tested components. In this subroutine the use of different mesh intervals plays an important role in the accurate determination of the top event unavailability. For example, during the inspection period, very small time step meshes must be used in order to calculate the detailed behavior of the system within this interval, which usually results in large system unavailabilities during this short time interval. (See examples presented in Section 2.10.)

The arrays STATT and STATD are allocated to store the failure rates of free and replicated components respectively. The arrays ETTA and ETTAD are used to store the test interval, TTETA and TTETAD to store the inspection period, whereas TAVV and TAVVD are employed to store the repair duration.

The values of λ_{eff} and q from Eqs. (2.5.9) and (2.5.10) are calculated for each free and replicated component and are stored in the arrays STATTE and STATTED, QUUE and QUUED, respectively for free and replicated components. At each time step, a special procedure is employed to determine the value of m in Eq. (2.5.8). This value provides the specific test interval that should be used. For m=1, the code will use Eq. (2.5.11), since it indicates

that the component is in its first test interval. For m>1, Eq. (2.5.8) will be used under the assumption that the subsequent test intervals are identical (i.e., only the first test interval differs from the others).

The unavailabilities calculated at each time step are assigned to the arrays STATE(1,I) and STATD(1,I) in order to calculate the top and higher order module unavailabilities. For periodically tested components, no approximations such as those discussed in Section 2.7 can be applied, since it seems that no correlation exists between a simple module inspection and repair duration and its input components' parameters (i.e., η, θ, τ, and λ).

Finally, the average unavailability during each test interval is calculated in the NUMERO procedure. If the override unavailability has a value less than unity, and if the time step is within the inspection period of the component, then the value of the predicted unavailability will be multiplied by a value called POORD, where POORD is the override probability.

$$\text{POORD} = q_o + (1 - q_o)(1 - e^{-\lambda_{eff} \cdot \eta})$$

q_o = override unavailability

Therefore, the arrays STATE(1,I) and STATD(1,I) will change to the following form.

STATE(1,I) = POORD * STATE(1,I) and

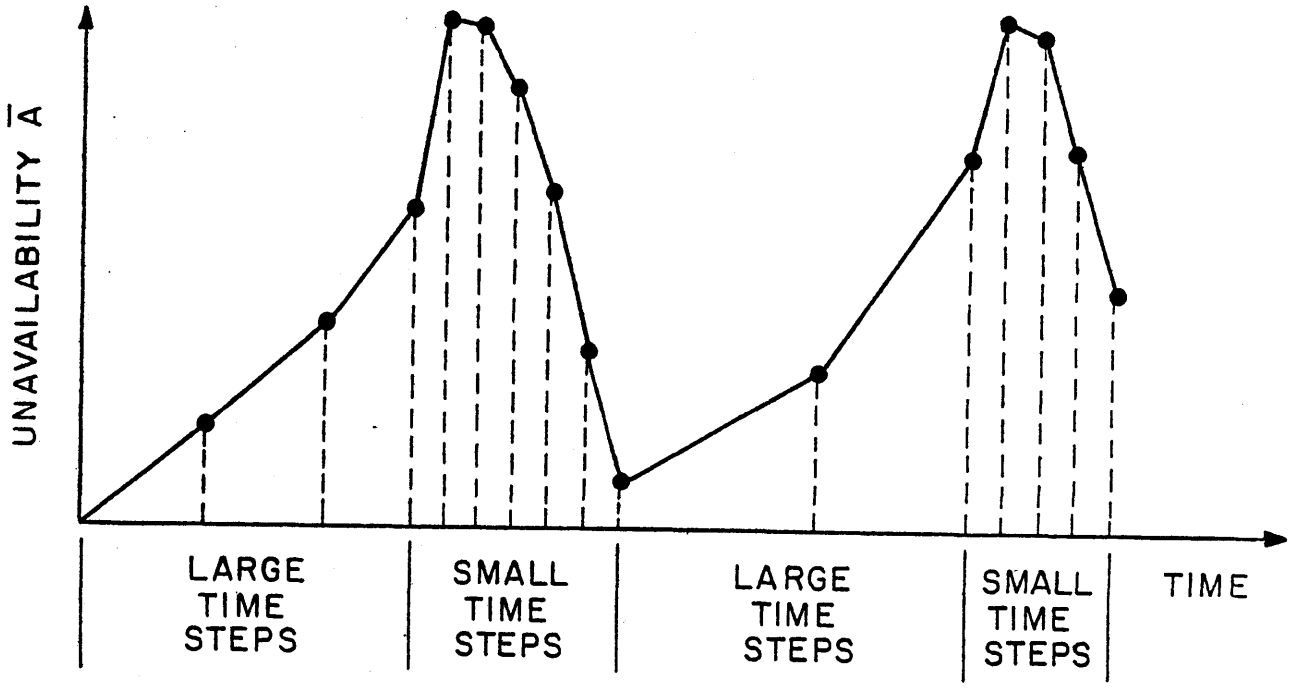
STATD(1,I) = POORD * STATD(1,I)

where STATE(1,I) and STATD(1,I) are calculated by Eq. (2.5.8).

2.9.4 Developments Included in Step 4

In the fourth step, the NUMERO subroutine is further extended to calculate the average unavailability of the top event. For this purpose, the following assumption is made. After the unavailabilities of two successive time steps are found for the top event, it is assumed that the unavailability changes linearly between these two points. Therefore, the unavailability as a function of time for the top event of a specific fault tree can be calculated for different time steps by the PL-MODT. For the special case of a fault tree of only periodically tested components, the top event unavailability behaves as shown in Figure 2.9.1.

Figure 2.9.1: Unavailability as a Function of Time



The total area under the curve in Figure 2.9.1 is calculated by adding the area occupied by each time step interval, and using the linear unavailability approximation between the two time steps. The total area is then divided by the operating time t, where $T = \sum_{i=1}^{N} AUN(1,I)$ to get the average unavailability of the system.

In the PL-MOD version of the code, the approximation states in Eq. (2.6.7) is used to calculate the union of all components and modules attached to a higher order module or the top event. In the time-dependent version of the code, this approximation is no longer appropriate, since sometimes we are dealing with unavailabilities ranging from 0.2 to 1, which precludes the use of the minimal cut-set upper bound formula, Eq. (2.6.7). Thus, the general form

$$h_\beta(t) = \coprod_{j=1}^{N_k} \prod_{i \varepsilon k_j} h_{r_i}(t)$$

is incorporated into the subroutines PLUS and MINUP. However, if the unavailabilities input to a higher order module or the top event are individually smaller than 0.2, then the program is constructed such that the old versions of subroutines PLUS and MINUP are used. As an example, part of the changes stated above are presented here.

```
      IF REX < 0.2 THEN GO TO ZACH;
```

(NOTE: ZACH will follow the old PLUS subroutine, and REX is unavailability calculated for the module.)

```
IF (PROP.LIM=1  ε  PROP.TIL(1)=0) THEN GO TO SLUA;
DO J=1 TO PROP. LIM;
REX=REX*(1-STATE(L,PROP.TIL(J)));
END;
SLUA:  IF (PROP.MIM=1 ε PROP.PIM(1)=NULL) THEN GO TO SLUB;
DO J=1 TO PROP.MIM;
IF (PROP.PIM(J)→PROP.HOST¬=NULL) THEN DO;
PR=PROP.PIM(J)→ PROP.HOST;
REX=REX*(1-PER.REL(1));
END;
ELSE REX=REX*(1-PROP.PIM(J)→PROP.REL(1));
END;
SLUB:  REX=1-REX;
ZACH: .....
```

The dummy variable REX and other similar variables are declared on FLOAT DECIMAL (16) so that the result from subtraction of small unavailabilities from 1 will not be truncated.

2.9.5 Developments Included in Step 5

In the last step, the SHARAREH subroutine is further developed in order to combine different classes of time-dependent components. This enables the code to treat any combination of repairable, non-repairable and periodically tested components in a fault tree. This specific problem is solved by adding a zero test interval condition option for those components which are not periodically tested (i.e., ETTA(1,I)=ETTAD(1,I)=0). If the zero test interval

condition arises, then the computer code will automatically apply Eq. (2.4.9) and thus will not evaluate Eqs. (2.5.8) through (2.5.11).

It should be noted here that in the case of non-repairable components, Eq. (2.4.9) reduces to Eq. (2.3.2) with the repair rate equal to zero (i.e., $\mu = 0$). Therefore, for a non-repairable component we have ETTA(1,I) = ETTAD(1,I)=TTETA(1,I)=TTETAD(1,I)=0. This results in an equation of the form given by Eq. (2.3.2).

One final note discusses the usefulness of the subroutines SHOHREH and CECA. Since the SHARAREH subroutine can handle not only periodically tested components but also repairable (revealed fault) and non-repairable compoennts, then there is seemingly no need for the subroutines SHOHREH and CECA. However, it should be pointed out here that the use of SHOHREH and CECA subroutines saves computation time due to approximation incorporated into them. Additionally, no procedure exists in these subroutines which would recognize the class of each component at each individual time step. Furthermore, if the fault tree components are only of one class, then the corresponding subroutine should be used to save computation time, although the unavailabilities will naturally be the same by using the SHARAREH subroutine. A variable ESF is used in the PL-MODT code to determine which subroutine is going to be used. For ESF=1, all components are considered to have steady-state point value unavailability (i.e., Class 1 components). For ESF=2,all components are considered to be repairable (revealed fault) components (i.e., Class 3 components). Finally, for ESF=4

all components are periodically tested components (i.e., Class 4 components) or a combination of the above-mentioned classes of components.

2.10 Examples

In order to more easily comprehend the meaning of the input description, four examples are given in this section.

2.10.1 Low Pressure Recirculation System (LPRS) for PWRs (Class 1)

This system is part of the Emergency Core Cooling Recirculation System and consists of the containment pump and two pumps in parallel redundancy. Figure 2.10.1 shows a simplified flow diagram and Figure 2.10.2 shows its associated reduced fault tree. It should be referred to in connection with the following discussion. All values with the exception of V_{24} and V_{25} are aligned for injection into the cold legs. V_{24} and/or V_{25} should be open in order to start the LPRS and close V_2. These valves, whether locally or remotely operated, are all manually operated. Details of the system as well as its fault tree can be taken from WASH-1400 [3].

The LPRS unavailability estimates are given in WASH-1400 as follows:

$$Q_{med}^{WASH} = 1.3 \times 10^{-2}$$

$$Q_{lower}^{WASH} = 4.4 \times 10^{-3}$$

$$Q_{upper}^{WASH} = 2.7 \times 10^{-2}$$

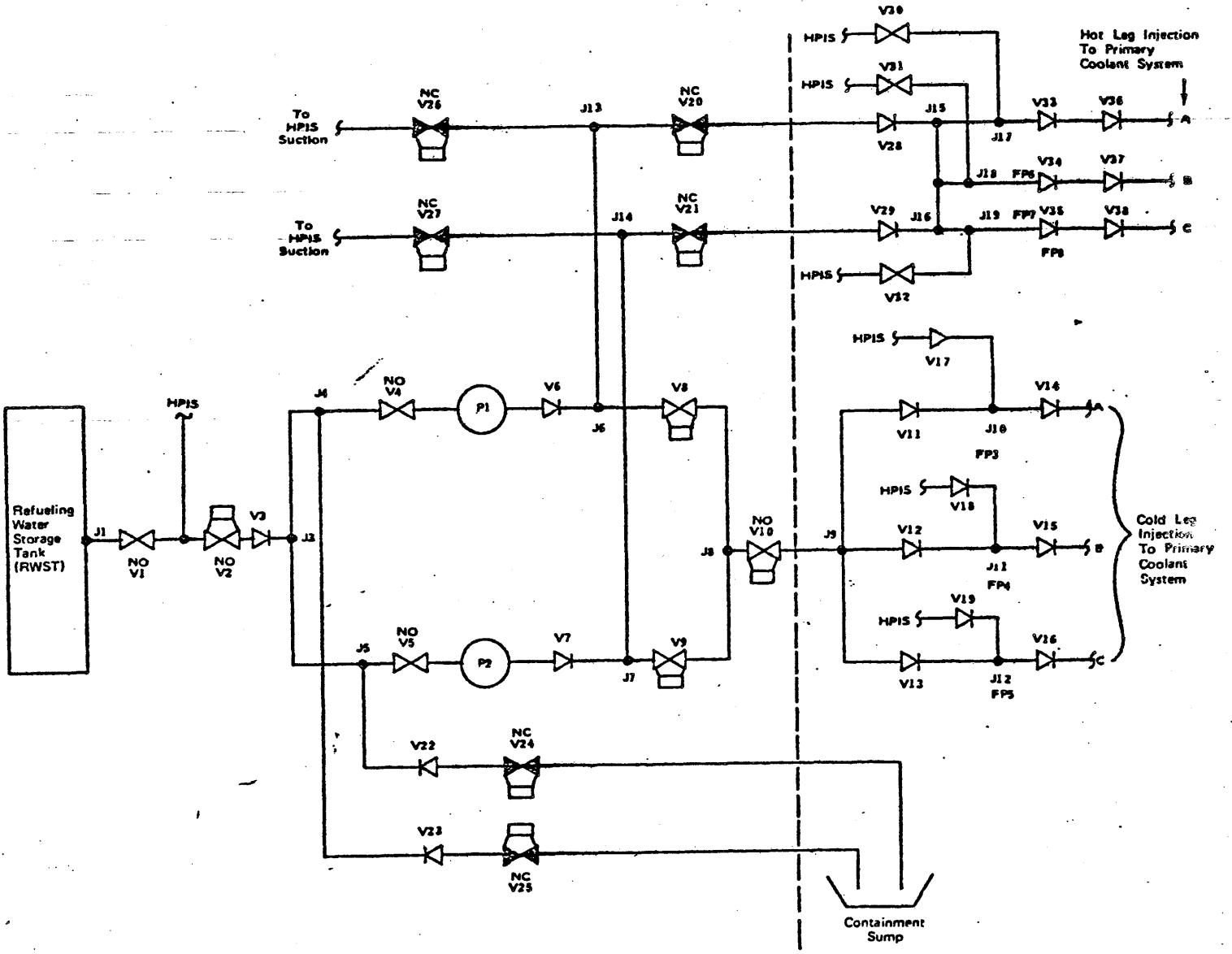


Figure 2.10.1: Simplified Flow Diagram LPRS

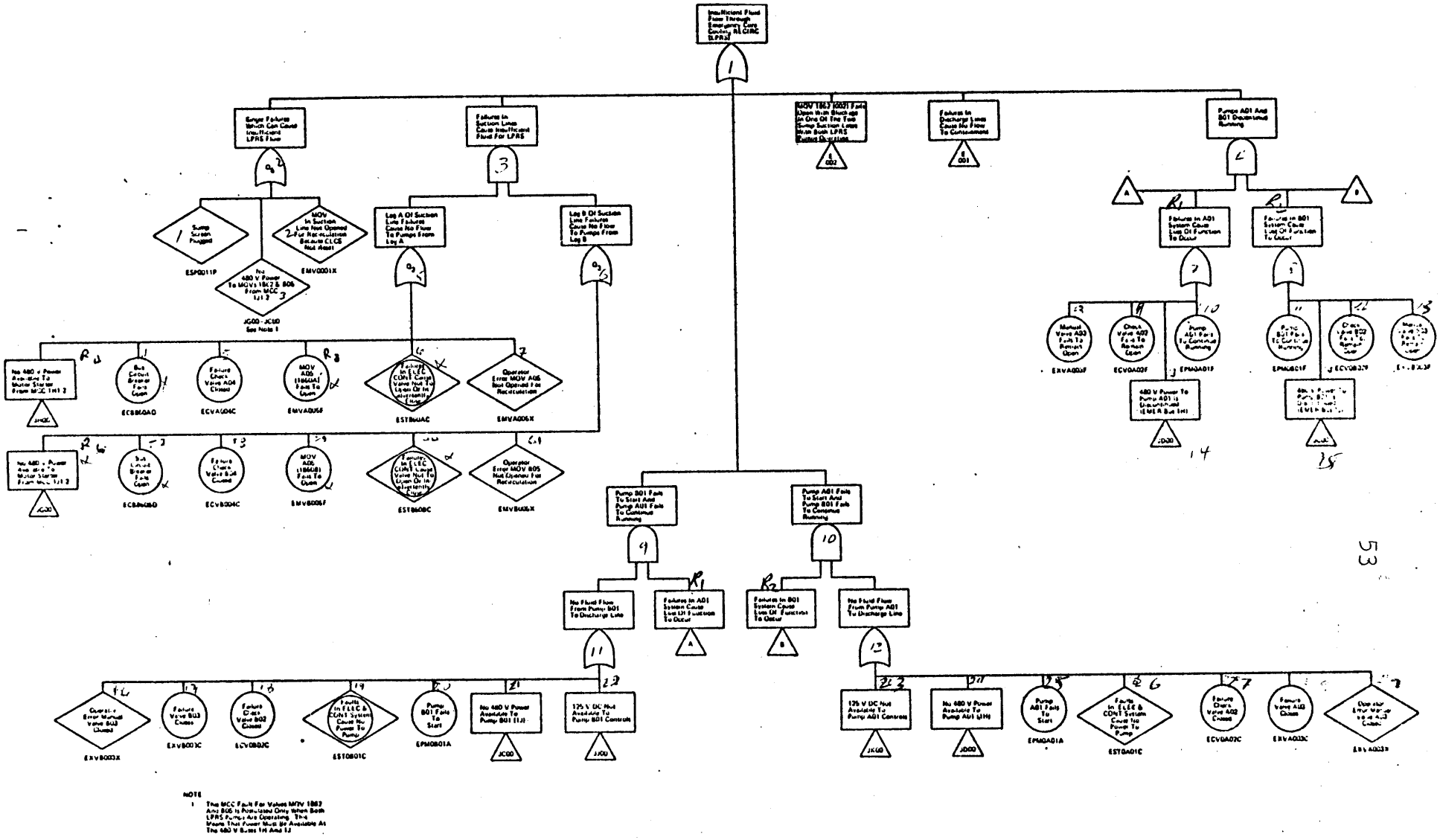
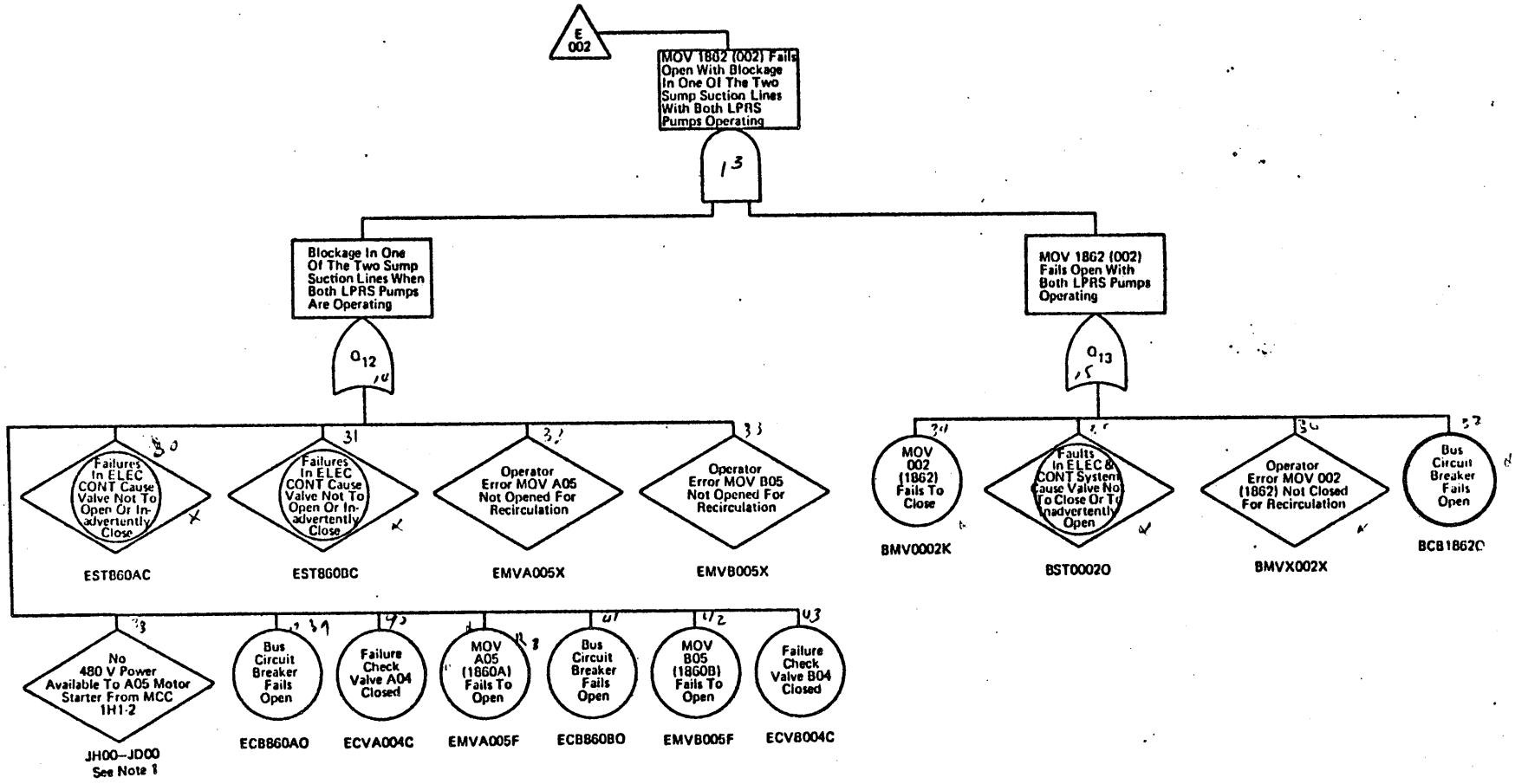


Figure 2.10.2: **LPRS Reduced Fault Tree (Sh. 1)**



NOTE:

1. These MCC Faults Postulated Only When Both LPRS Pumps Are Operating. This Means That Power Must Be Available At The 480 V Buses 1H And 1J.

Figure 2.10.2: (cont.) **LPRS Reduced Fault Tree (Sh. 2)**

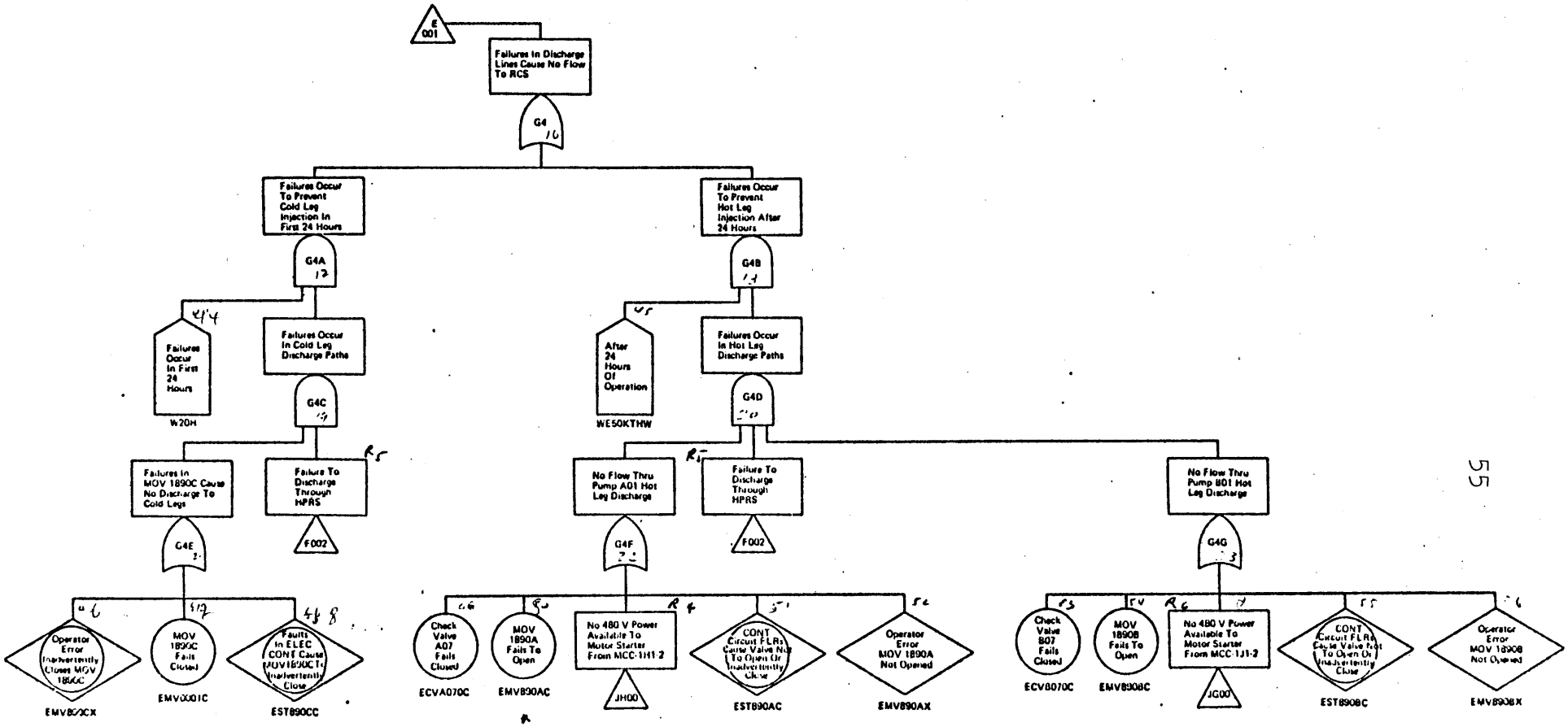


Figure 2.10.2: LPRS Reduced Fault Tree (Sh. 3) (cont.)

where the lower and upper bounds were evaluated by a Monte-Carlo simulation by using the minimal cut-set approach. The point estimates for single and double failures, test and maintenance as well as common mode failures are:

$$Q_{single} = 1.1 \times 10^{-5}$$

$$Q_{double} = 2.7 \times 10^{-3}$$

$$Q_{test\ \&\ main} = 1.0 \times 10^{-4}$$

$$Q_{common} = 6.0 \times 10^{-3}$$

respectively.

This system has been analyzed by PL-MOD to determine the point estimate probability for the occurrence of the top event and the Vesely-Fussell importance of the components. The reduced fault tree contains a total of 61 non-replicated basic events, 4 replicated events and 2 replicated modular gates.

The point unavailability computed by PL-MOD using the modular approach is

$$Q = 4.83 \times 10^{-3}$$

The total computation time for this example was 0.46 sec. and included the modularization, the evaluation of the top event probability and the determination of the importance measure for all components and modules in the fault tree.

2.10.2 Auxiliary Feedwater System - A Comparison Between PL-MODT and FRANTIC (Class 4)

The Aux-Feed System is shown in Figure 2.10.3 and described in detail in [4]. As can be seen from the figure, the system consists of two diesels in parallel with a pump and two valves. The pump and valves are in series. The block diagram used in FRANTIC [4] is a simplified version of the one shown in WASH-1400. It is assumed that the components of the system are periodically tested. The data are summarized in Table 2.10.1.

Figure 2.10.4 compares the results for the point unavailability computed by the two codes FRANTIC and PL-MODT. It should be noticed that PL-MODT gives slightly lower values for the unavailability for the system during the operation time. During the inspection time of valve 1, pump and valve 2, and diesel 4 ($720 < t < 721.5$) both codes give essentially the same answer. However, for the repair time interval, a larger difference appears as can be seen from Figure 2.10.5. This is mainly due to the fact that the analytical equation in PL-MODT gives lower values for the unavailability. This difference vanishes as the end of the inspection period is approached. Thereafter, both codes predict about the same value for the unavailability as depicted in Figure 2.10.5. To evaluate unavailabilities shown in Figure 2.10.5, a finer time step mesh was used.

PL-MODT treats the group: valve 1, pump and valve 2, as a module and therefore the unavailability of this module will be given automatically in the output. Therefore, the unavailability

Figure 2.10.3

Block Diagram of the Aux-Feed System

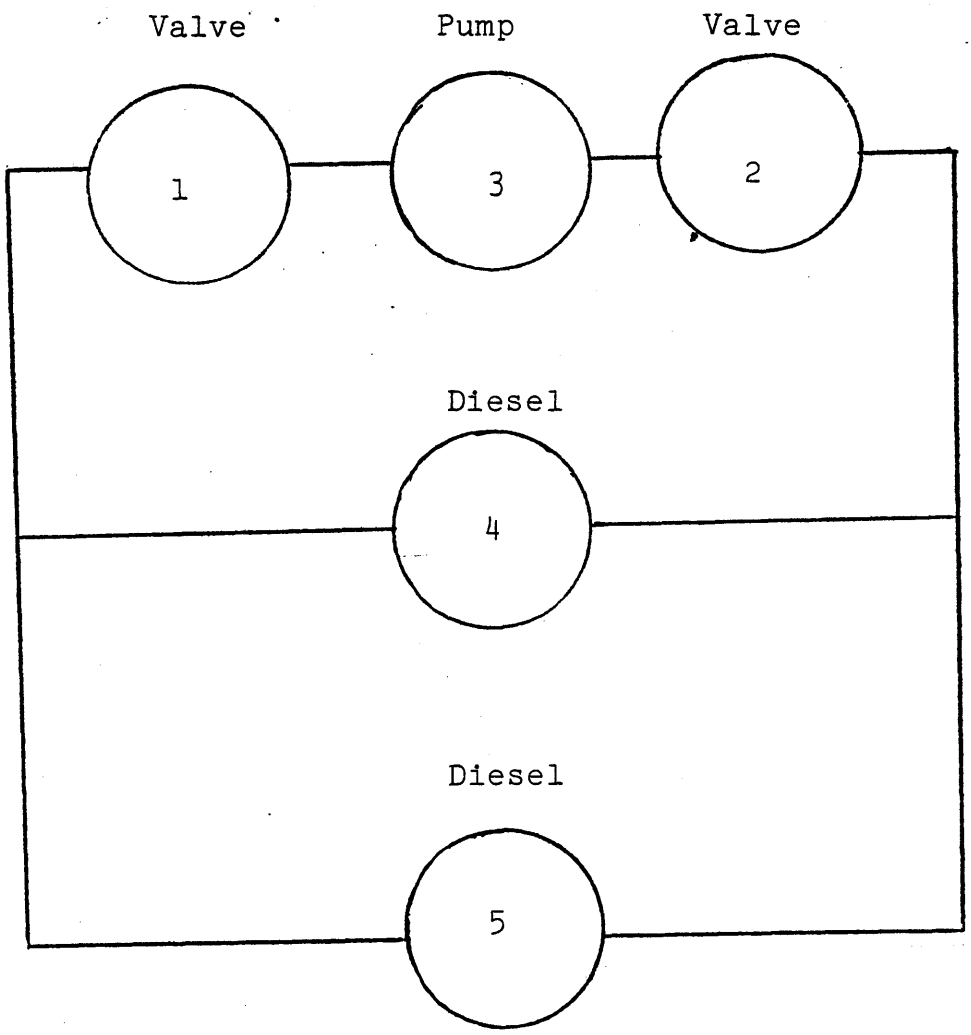


TABLE 2.10.1

THE COMPONENT INPUT DATA FOR THE CODES FRANTIC AND PL-MODT

Component Number	Component name	Failure Rate $(\times 10^{-i})(\lambda)$	Test Interval (η) (days)	First Test Interval q.T.	Inspection Duration(θ) (hours)	Repair Duration(η) (hours)	(q_o) Override Unavailability
1	Valve	0.3	30	30	1.5	7	1.0
2	Valve	0.3	30	30	1.5	7	1.0
3	Pump	3	30	30	1.5	19	1.0
4	Diesel	42	60	30	1.5	21	1.0
5	Diesel	42	60	60	1.5	21	1.0

Figure 2.10.4

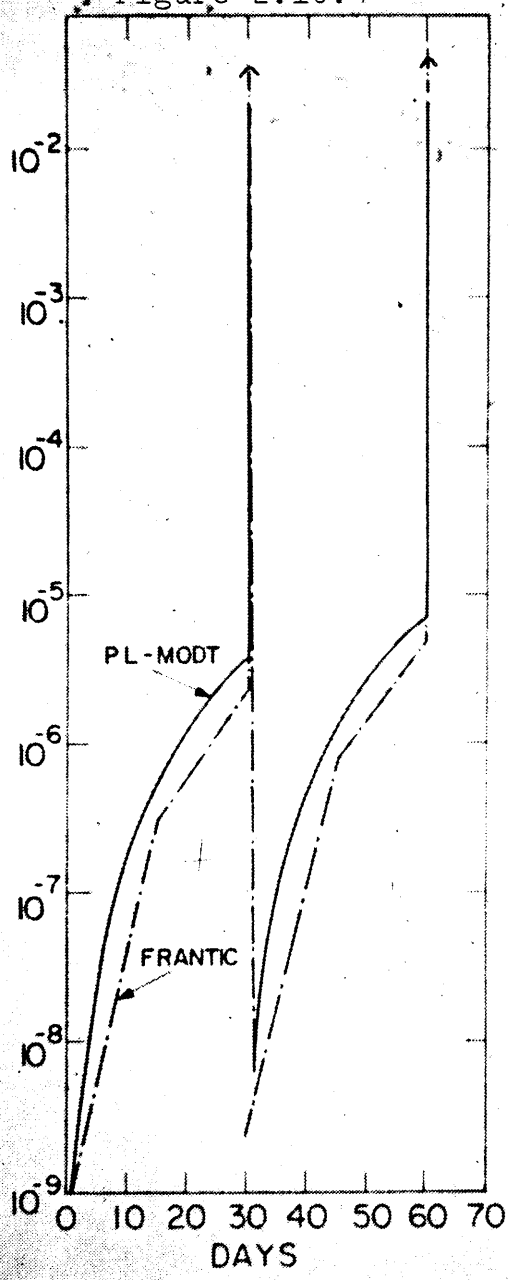
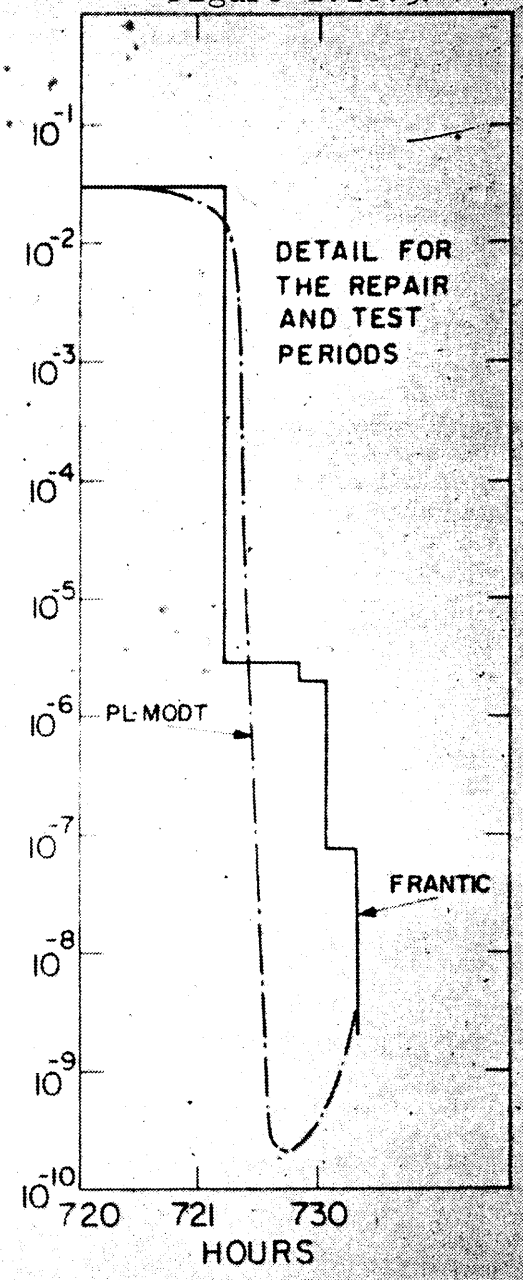


Figure 2.10.5



COMPARISON BETWEEN THE TIME-DEPENDENT UNAVAILABILITIES FOR THE AUX-FEED SYSTEM AS CALCULATED BY FRANTIC AND PL-MODT

for each branch could automatically be obtained from the same computer run for unavailability calculations of the top event.

The CPU time for PL-MODT was 0.57 seconds for modularizing the fault tree and calculating the unavailabilities for 40 time steps as well as the importances for components and modules.

As the number of components increases and more inspected components become involved, the differences during the repair time should vanish. This will become more apparent in the next example.

2.10.3 Example of a Simple Electric Circuit Using FRANTIC and PL-MODT

Figure 2.10.6 shows a simple electric system which has been discussed in [5]. The purpose of the system is to provide light by the bulb when the switch is closed, the relay 1 contacts closed and the contacts of relay 2 (a normally closed relay) are opened. Should the contacts of relay 1 open, the light will go out and the operator will immediately open the switch which in turn causes the contacts of the relay 2 to close which restores the light. In what follows, operator failures, wiring failures as well as secondary failures will be neglected. The fault tree for this system is shown in Figure 2.10.7.

Failure rates, repair times and test periods for the various components are summarized in Table 2.10.2. No replicated component or module exists in this system.

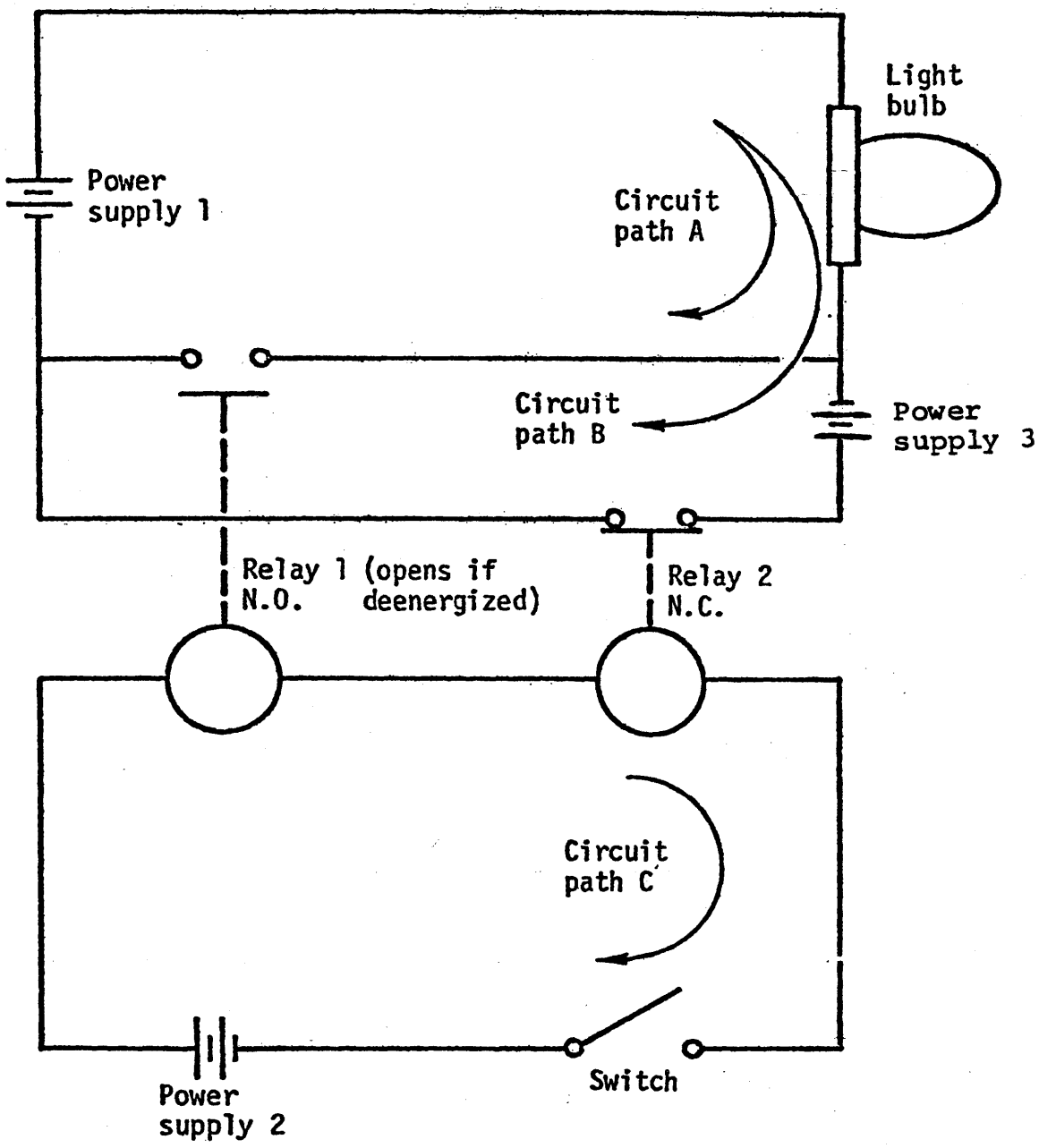


Figure 2.10.6: Sample System for Mutually Exclusive Events

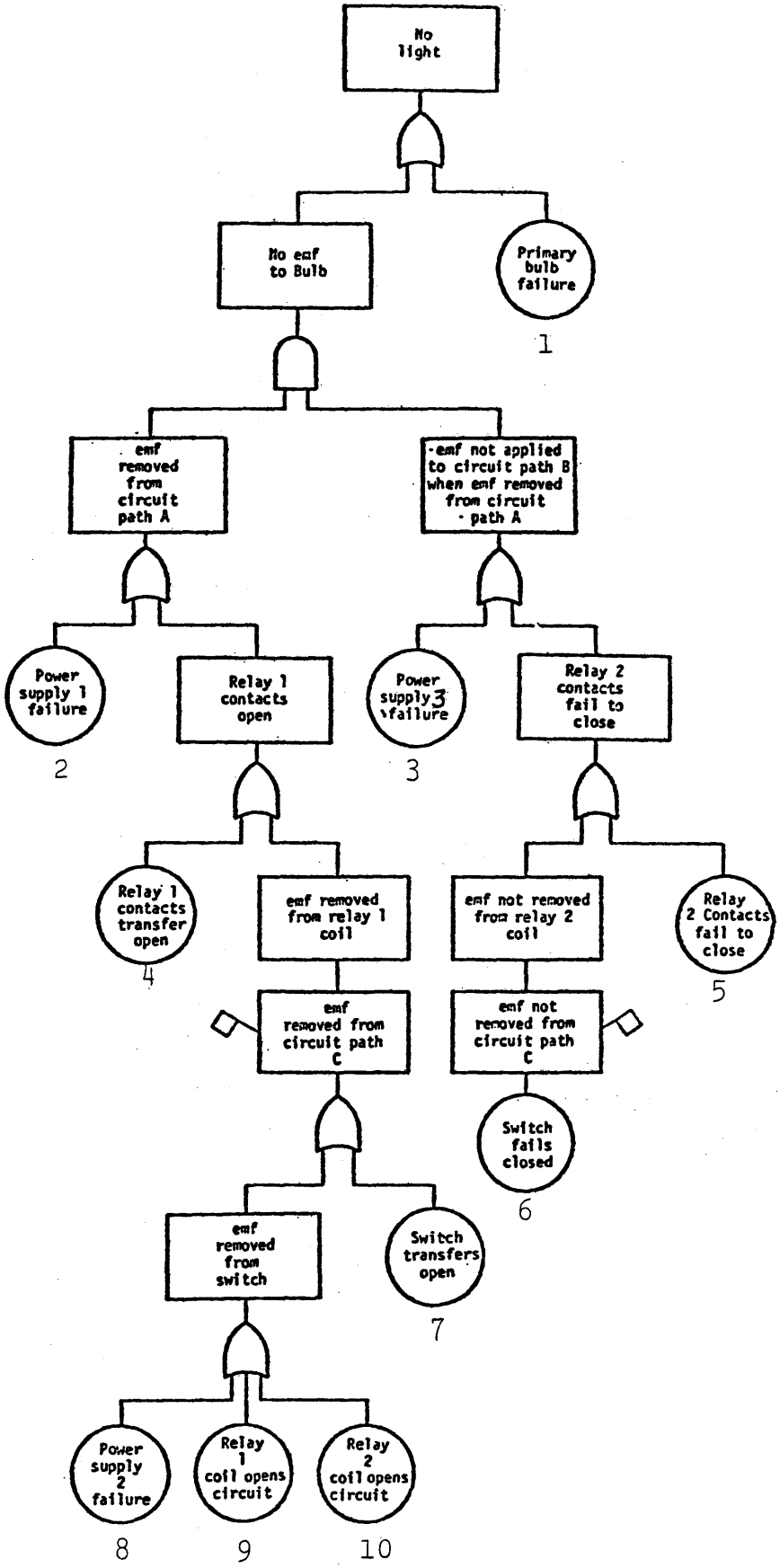


Figure 2.10.7 : **Fault Tree for Sample System in Figure 2.10.6**

TABLE 2.10.2

INSPECTION PARAMETERS FOR THE CIRCUIT EXAMPLE

Component #	Failure rate hr^{-1}	Inspection Time (hrs)	First (Days) Time Interval	Repair Time (hrs)	Override Unavailability	Test 1(days) Interval
1	2.0×10^{-4}	1.0	7	2.0	1.0	7
2	2.8×10^{-5}	0.5	7	1.0	1.0	14
3	2.8×10^{-5}	1.0	7	1.5	1.0	14
4	3.2×10^{-3}	0.5	7	2.0	1.0	7
5	4.1×10^{-4}	0.5	14	1.5	1.0	21
6	3.2×10^{-4}	1.0	28	1.5	1.0	28
7	2.8×10^{-3}	0.5	14	2.0	1.0	21
8	2.8×10^{-3}	1.0	14	1.5	1.0	14
9	4.5×10^{-3}	0.5	7	2.5	1.0	7
10	4.5×10^{-3}	1.5	7	3.0	1.0	7

In order to enable FRANTIC to analyze this system, the system's unavailability function must be provided as input. This function was found to be

$$QS = \{ 1.0-(1-Q(1))\{ 1-[(1-Q(2))(1-Q(4))(1-Q(7))(1-Q(8))*$$
$$*(1-Q(9))(1-Q(10))][1-(1-Q(3))(1-Q(6))(1-Q(5))\}\}$$

Naturally, for PL-MODT the fault tree was directly inputted because there is no need for a system function. As output, PL-MODT gives the four following modules.

Module #4: components 5, 6, and 3
Module #3: components 4, 7, 8, 9, 10, and 2
Module #2: modules 4 and 3
Module #1: component 1 and module 2

Figure 2.10.8 compares the output of both codes for one complete period of 28 days. As can be seen, the results are overall in very close agreement during the operational period. Differences are attributed to the fact that for this example the failure rates are comparatively high, and the FRANTIC prediction, by using a linear approximation for the unavailability, at large times would not be valid.

Again, during the inspection period both codes give essentially the same results. However, for the repair period (see Figure 2.10.9) differences show up again which were observed already in Example 2.10.2. It should be noticed that these differences are

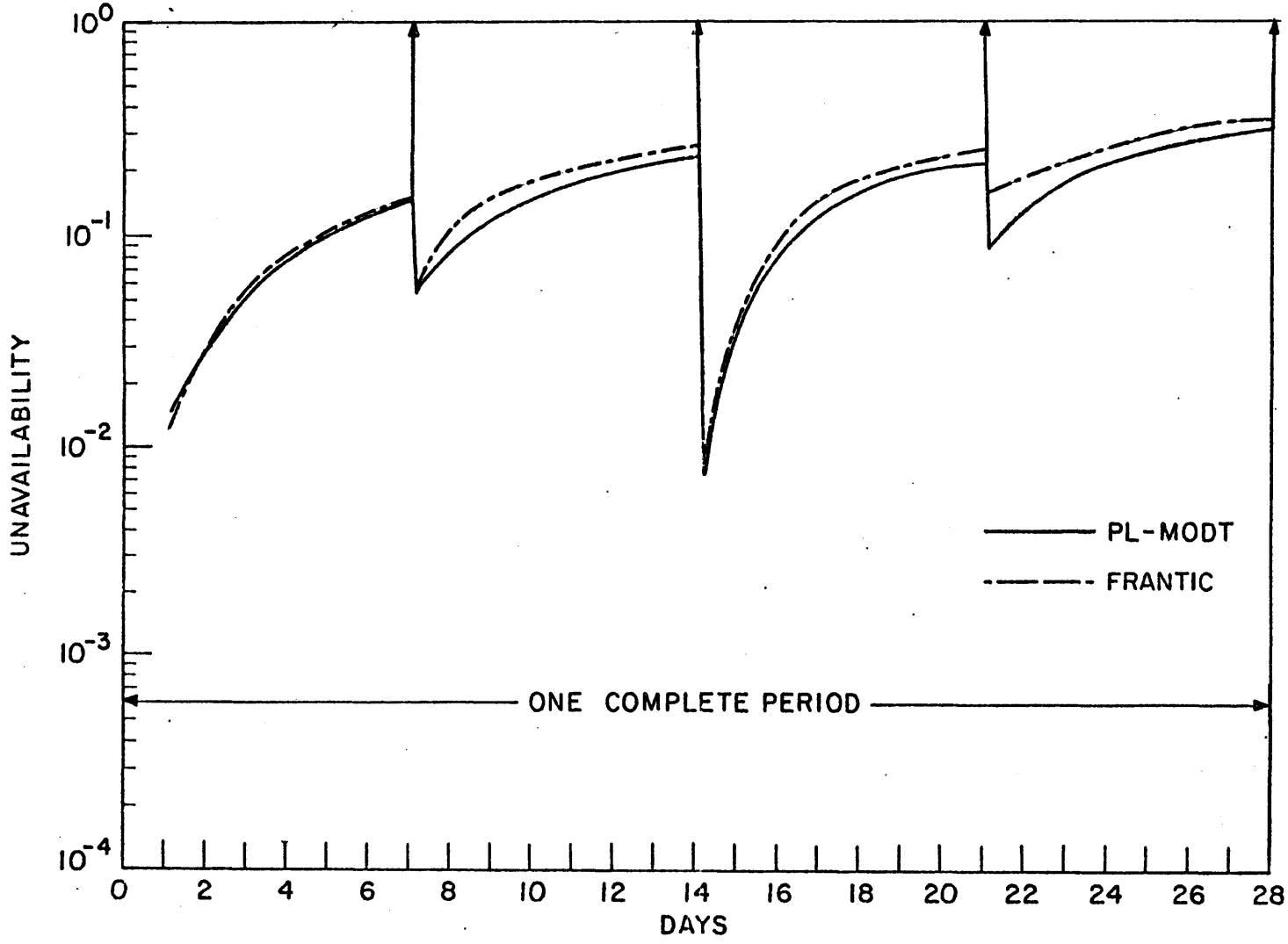


Figure 2.10.8: Comparison Between the Unavailabilities for the Electrical System During Its Operation as Calculated by FRANTIC and PL-MODT

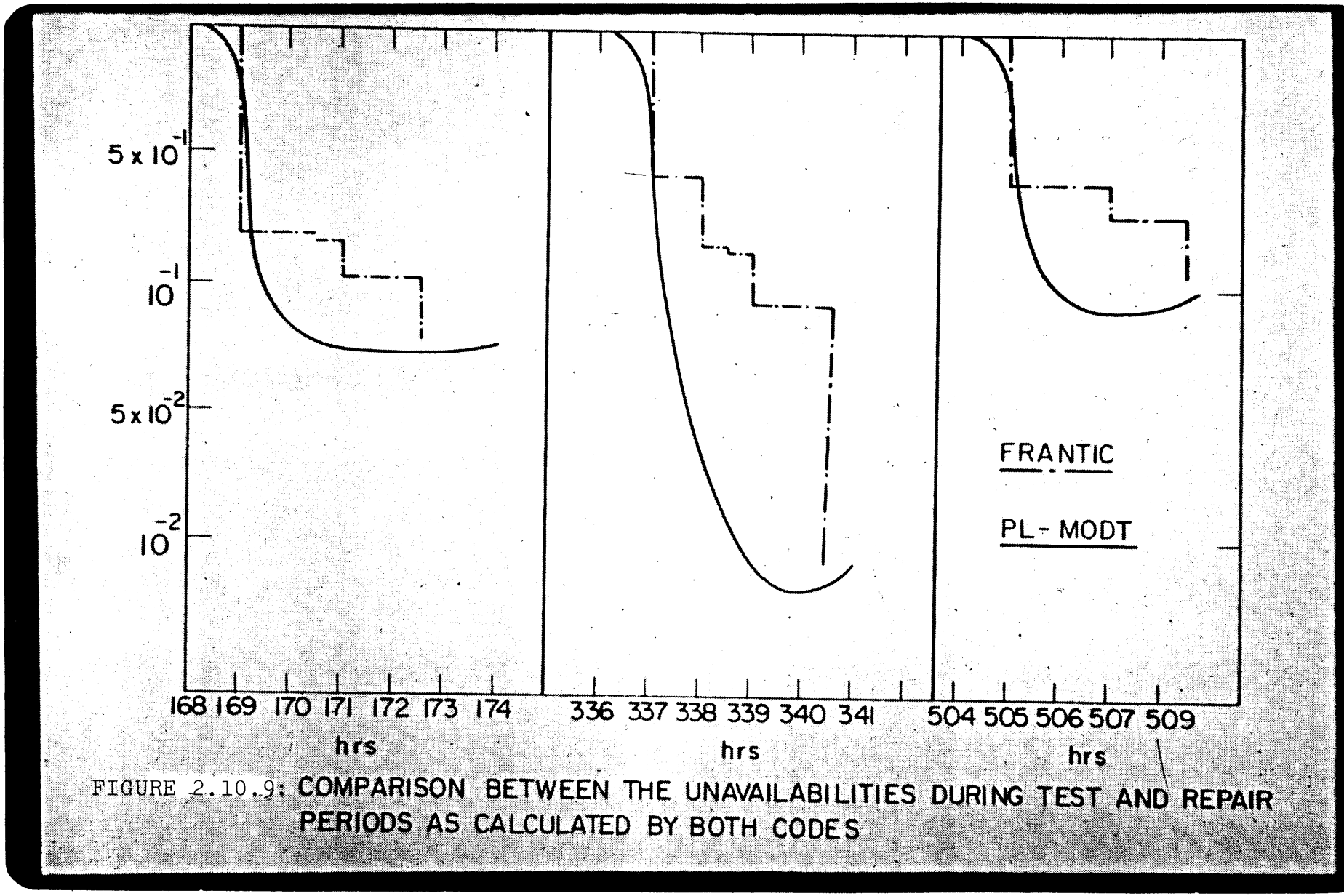


FIGURE 2.10.9: COMPARISON BETWEEN THE UNAVAILABILITIES DURING TEST AND REPAIR PERIODS AS CALCULATED BY BOTH CODES

not as pronounced as in Example 2.10.2 because more components are involved in the present example.

The CPU time for PL-MODT was 0.98 seconds for modularizing the tree, evaluating the unavailabilities for components, modules and the top event for 32 time steps and for determining the importances for various components and modules.

For the same tree and data, FRANTIC needed 1.12 seconds alone for calculating the system unavailability over the period of 180 days and to determine the mean unavailability of the system over this period. However, it should be noted here that PL-MODT would be an efficient and convenient code for evaluating large fault trees consisting of periodically tested components. Examples 2.10.2 and 2.10.3 showed that the code is also fairly fast for evaluating small fault trees as compared to other state-of-the-art computer codes.

2.10.4 Comments and Discussion

2.10.4.1 Differences Between FRANTIC and PL-MODT

The obvious difference between the two codes is that whereas PL-MODT is capable of analyzing large fault trees and evaluating them at the same time, the use of FRANTIC is mainly confined to the analysis of small systems for which the system unavailability function is known in advance. This off-line approach is not only time consuming but has the additional disadvantage that the user may introduce spurious errors.

In case that a system, for example, the Aux-Feed system, is to be evaluated in more detail than each valve, pump, and diesel would be further developed down to the level of subcomponents which are periodically tested. For this purpose, PL-MODT is especially suited.

The analysis of computer storage and CPU statistics for PL-MODT is underway.

2.10.4.2 Comments on the Vesely-Fussell Importance Measure

PL-MODT enables the user to select the option for the determination of the V.F. importance in steady-state and transient evaluations. As an example, Figure 2.10.10 shows the importance of the pump in the Aux-Feed system as a function of time. The importance stays about constant through the operational period. After a sudden increase in system unavailability due to the testing of the pump, the valves, and the diesel, the importance of the pump sharply decreases and increases again, once the inspection has been finished. Thereafter, it remains fairly constant during repair and operation. The opposite behavior can be seen in Figure 2.10.11 for the valves. The reason for this behavior is that the combination of valve 1, pump and valve 2 constitutes a prime module which is directly connected to the top event and thus its total importance is 1 over the whole operational period. The same holds for diesels 4 and 5. As a result, any increase in pump importance is accompanied at the same time by a decrease in valve importance.

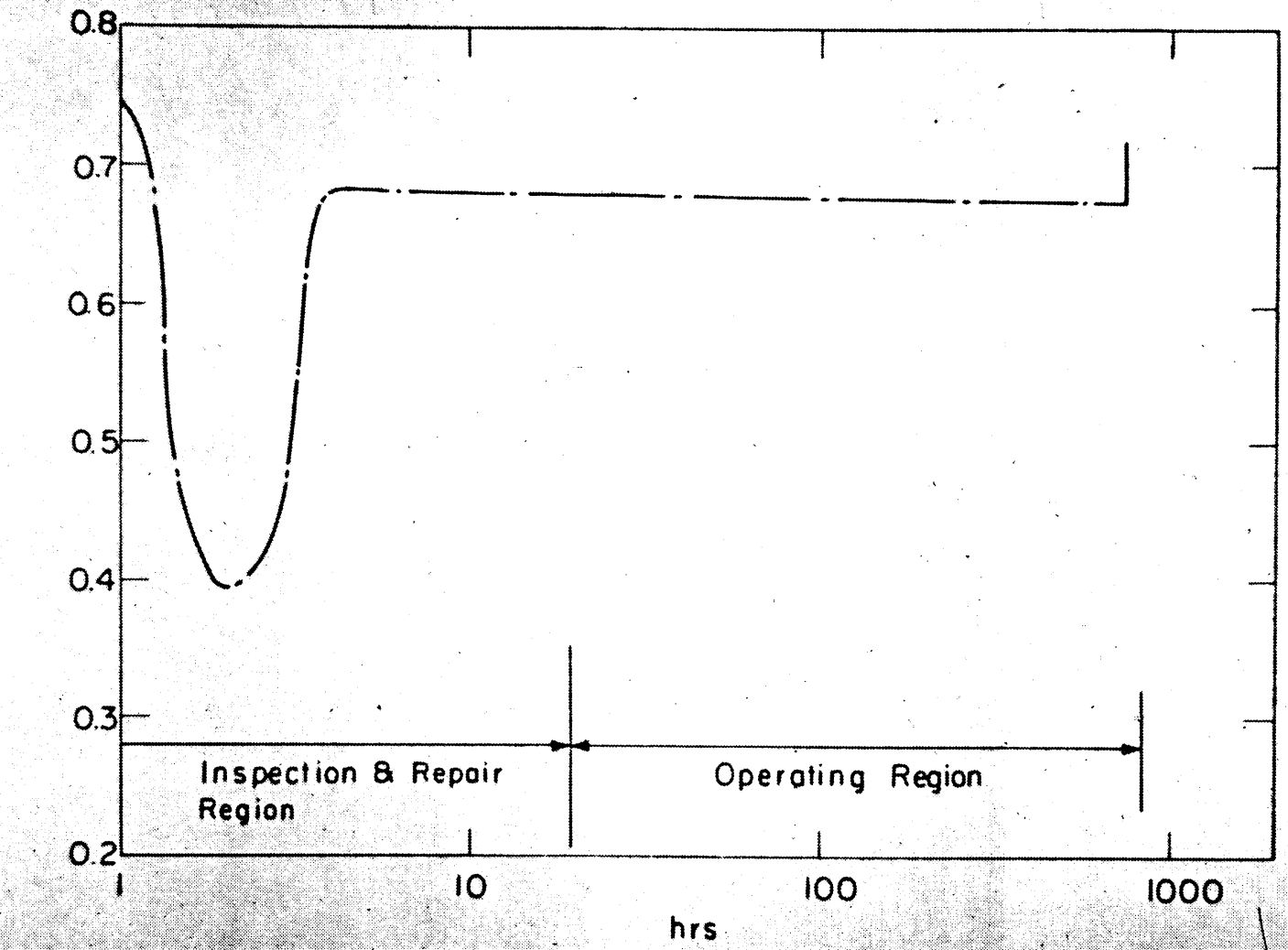
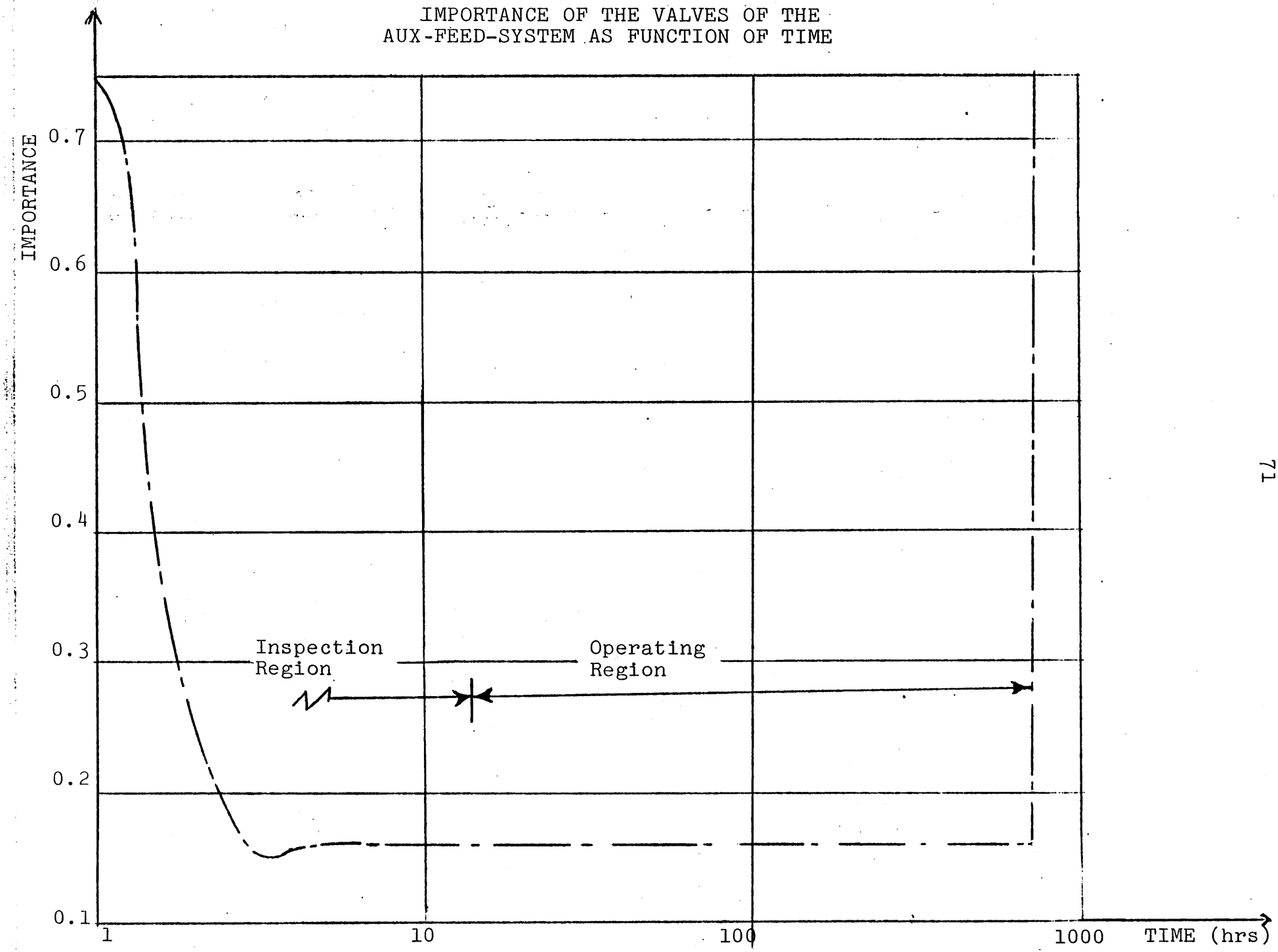


FIGURE 2.10.10: IMPORTANCE OF THE PUMP IN THE AUX-FEED SYSTEM AS FUNCTION OF TIME

FIGURE 2.10.11

IMPORTANCE OF THE VALVES OF THE
AUX-FEED-SYSTEM AS FUNCTION OF TIME



2.10.5 Comparison Between PL-MODT and PREP & KITT-1 (Class 2 and 3)

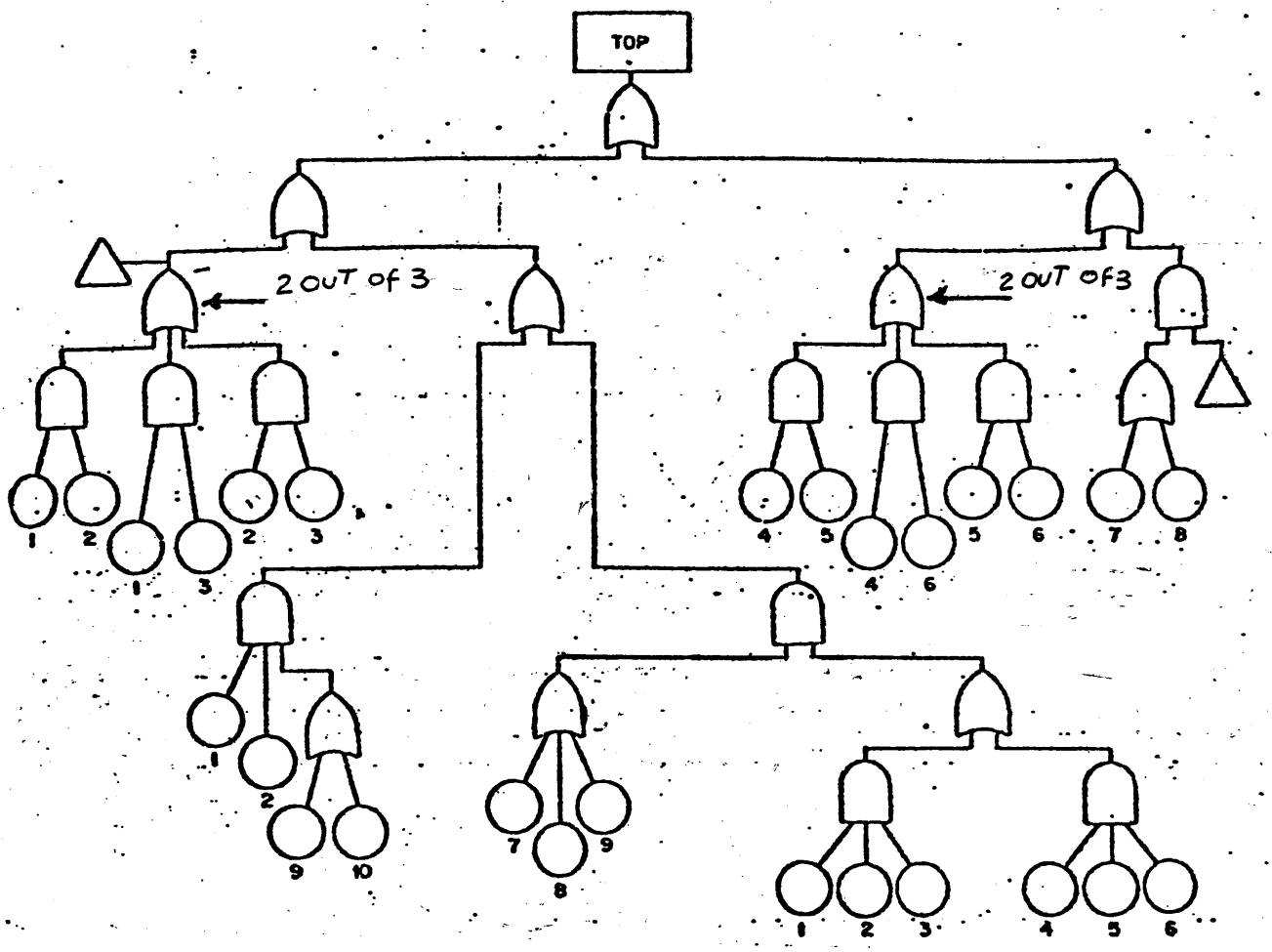
This comparison is based upon an example considered in Ref. [6]. The associated fault tree is shown in Figure 2.10.12. It has been analyzed and evaluated by PL-MODT for the data summarized in Table 2.10.3 by assuming all components show a time-dependent failure behavior but are non-repairable.

Figure 2.10.13 compares the results of the two codes. Those for PREP & KITT are taken from [6]. For the first 3000 hrs, PL-MODT calculates unavailabilities which are higher than those computed by PREP & KITT. Thereafter, the trend reverses and PREP & KITT gives higher values. Which code comes closer to the exact answer can only be answered by benchmarking these codes against a code which employs the Markovian approach.

The computation time for PL-MODT was 0.64 seconds for modularizing the tree, finding the unavailabilities for system, components and modules for 10 time steps. Advantage was taken of the fact that PL-MODT is capable of handling directly K-out-of-N gates. Therefore, the two 2/3 gates were inputted, rather than analyzed by the code. This approach naturally saves computer storage and computation time.

The same fault tree in Figure 2.10.12 was used to evaluate the top event unavailability for the case that components 1 to 6 in Figure 2.10.12 are repairable and 7 to 10 are non-repairable. The following data were used (Table 2.10.4).

FIGURE 2.10.12
FAULT TREE EXAMPLE GIVEN IN [6]



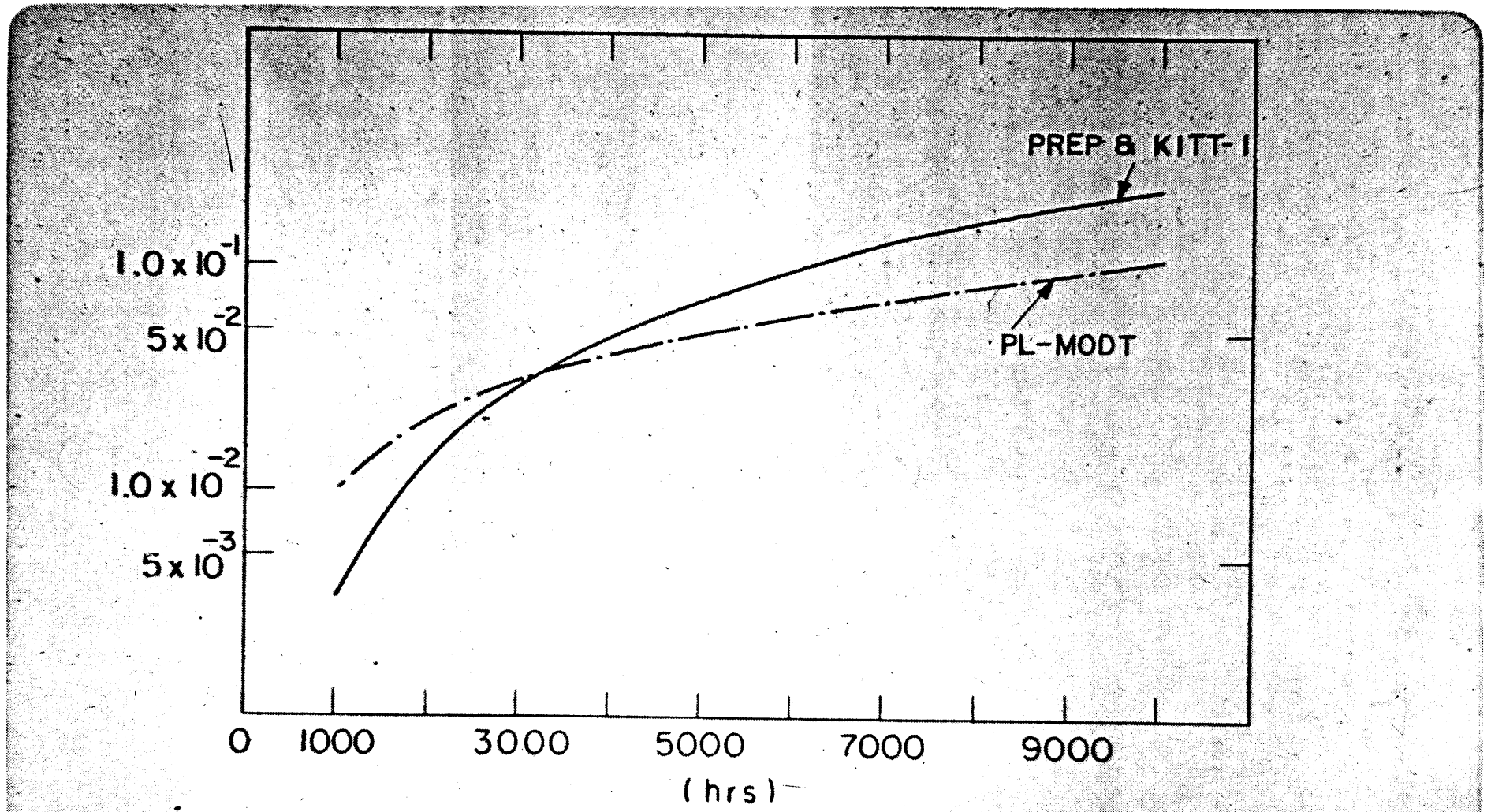


FIGURE 2.10.13: Comparison of the Time-Dependent Unavailabilities For the Sample Tree Given in Figure 2.10.12 as Calculated by KITT and PL-MODT

TABLE 2.10.3

PRIMARY FAILURE RATES FOR SAMPLE FAULT TREE SHOWN IN FIGURE 2.10.12

Primary Failure Index	$\lambda(hr^{-1})$
1	$2.6 \cdot 10^{-6}$
2	$2.6 \cdot 10^{-6}$
3	$2.6 \cdot 10^{-6}$
4	$3.5 \cdot 10^{-5}$
5	$3.5 \cdot 10^{-5}$
6	$3.5 \cdot 10^{-5}$
7	$3.5 \cdot 10^{-6}$
8	$5.0 \cdot 10^{-6}$
9	$8.0 \cdot 10^{-6}$
10	$8.0 \cdot 10^{-6}$

TABLE 2.10.4

FAILURE AND REPAIR RATES FOR SAMPLE TREE IN FIGURE 2.10.12

Primary Failure Index	(hr^{-1})	(hr^{-1})
1	2.6×10^{-6}	4.1×10^{-2}
2	2.6×10^{-6}	4.1×10^{-2}
3	2.6×10^{-6}	4.1×10^{-2}
4	3.5×10^{-5}	1.66×10^{-1}
5	3.5×10^{-5}	1.66×10^{-1}
6	3.5×10^{-5}	1.66×10^{-1}
7	5.0×10^{-6}	0
8	5.0×10^{-6}	0
9	8.0×10^{-6}	0
10	8.0×10^{-6}	0

First PREP & KITT were used to compare its results with PL-MODT. Next, FRANTIC results were used for the same example. The results are shown in Figure 2.10.14

Small differences exist due to different approximations used in these codes. However, these three codes give essentially the same asymptotic values for unavailabilities. The job run time for PL-MODT to calculate the top event unavailability for 15 time steps was 0.62 seconds.

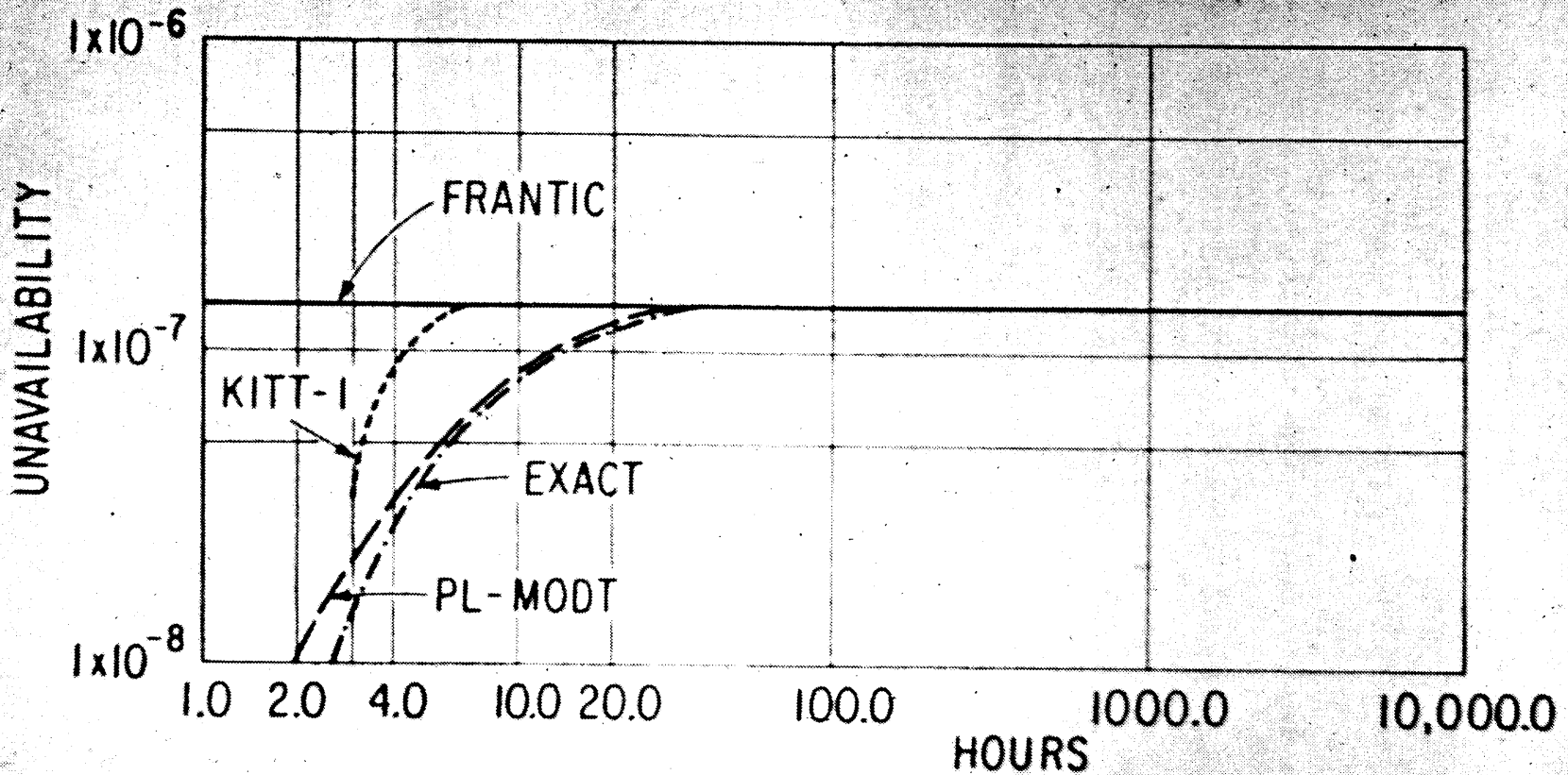


FIGURE 2.10.14: COMPARISON OF THE TIME-DEPENDENT UNAVAILABILITIES OF REPAIRABLE COMPONENTS FOR THE SAMPLE TREE GIVEN IN FIGURE 2.10.12 AS CALCULATED BY FRANTIC, PREP, KITT AND PL-MODT

REFERENCES

[1] J. Olmos and L. Wolf, A Modular Approach to Fault Tree and Reliability Analysis, Department of Nuclear Engineering, MIT, MITNE-209, August 1977.

[2] L. Caldorola, "Unavailability and Failure Intensity of Components", Nucl. Eng. Design, 44, 147 (1977).

[3] Reactor Safety Study, U.S. Atomic Energy Commission, Washington, Appendix II.

[4] W.E. Vesely and F.F. Goldberg, FRANTIC: A Computer Code for Time-Dependent Unavailability Analysis, NUREG-0193, October 1977.

[5] H.E. Lambert, Fault Trees for Decision Making in System Analysis, UTRL-51829, October 1975.

[6] W.E. Vesely, A Time-Dependent Methodology for Fault Tree Evaluation, Nuclear Engineering 13(1970), 337-360.

[7] M. Modarres and L. Wolf, PL-MODT: An Extended Version of the PL-MOD Code for the Modular, Transient Fault Tree Analysis and Evaluation, Department of Nuclear Engineering, MIT, Internal Report.

[8] M. Modarres and L. Wolf, "PL-MODT: A Modular Fault Tree Analysis and Transient Evaluation Code", Trans. Am. Nucl. Soc. 28, 510 (1978).

3. REDUCTION OF LARGE FAULT TREES BASED ON THE VESELY-FUSSELL IMPORTANCE MEASURES

3.1 Introduction

In this chapter the fault tree reduction method which is incorporated into the PL-MODT code is discussed. This reduction method is based on the Vesely-Fussell importance measures which are calculated by the code. A value called cut-off value is inputted and any higher order module, simple module, or component which has an importance less than this value is eliminated. The remaining part of the original fault tree is the reduced version. Essentially, it must have all of the characteristics of the original fault tree. The reduced version of the tree is very useful for further assessment of the fault tree, such as low order cut-set generation, test and maintenance consideration, common cause analysis,..., etc.. In the following sections, some discussions are provided to clarify the method of reduction and the cut-off range to be used.

3.2 Importance Measures and the Use of PL-MOD to Calculate the V-F Importance Measures

The code PL-MOD is able to calculate the importance measures for large fault trees very effectively and economically. Olmos and Wolf [1] have developed V-F importance measures to calculate importance of higher order modules in a fault tree.

For example, a higher order module is shown in Figure 3.2.1. To evaluate the V-F importance of modules such as the one shown in Figure 3.2.1, it follows that

$$\sigma_M = \beta(\sigma_1, \sigma_2, \ldots, \sigma_n) = \coprod_{\ell=1}^{N_k^j} \prod_{i \epsilon k_\ell} \sigma_i \qquad (3.2.1)$$

$$(i=1, 2, \ldots, n)$$

The probability that module σ_i will contribute to the failure of its parent module σ_M, given that the parent module has failed is given by

$$I^{VF}_{\beta,\sigma_j} = \frac{P(\beta_k^j(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1)}{P(\beta(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1)} \qquad (3.2.2)$$

now

$$P(\beta(\sigma_1, \sigma_2, \ldots, \sigma_n)) = h_{\sigma_M} \qquad (3.2.3)$$

and Eq. (3.2.1) implies that β_k^j is given by

$$\beta_k^j = \coprod_{\ell=1}^{N_k^j} \prod_{\substack{\ell \epsilon k_\ell \\ j \epsilon k_\ell}} \sigma_\ell \qquad (3.2.4)$$

Thus, the V.F. importance for module j with respect to the top event will be

$$I^{V.F.}_{\theta,\sigma_j} = I^{V.F.}_{\alpha,M} = \frac{P((\coprod_{\ell=1}^{N_k^j} \prod_{\substack{\ell \epsilon k_\ell \\ j \epsilon k_\ell}} \sigma_\ell) = 1)}{h_{\sigma_M}} \qquad (3.2.5)$$

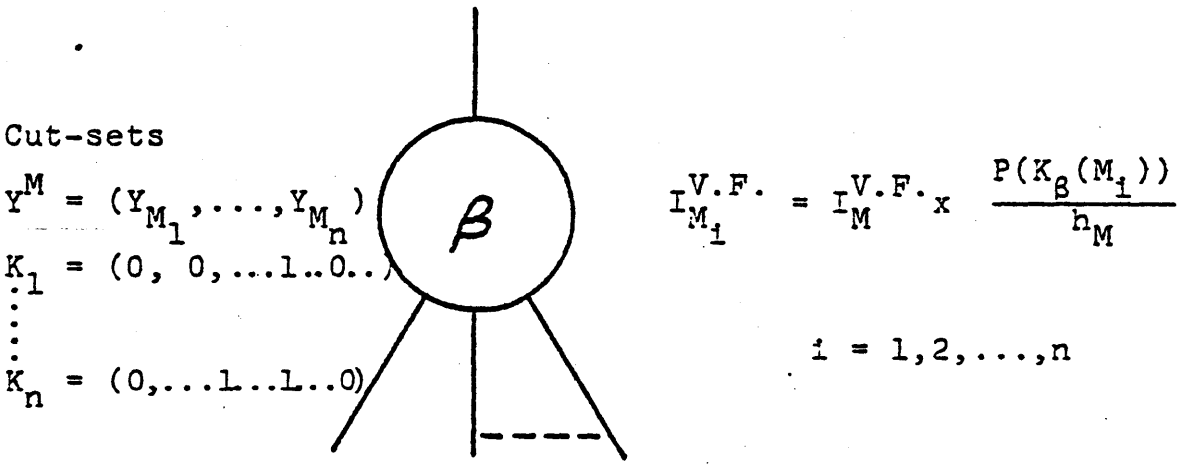


Figure 3.2.1: HIGHER ORDER PRIME GATE SUPER-MODULE

In the code PL-MOD the subroutine IMPORTANCE is constructed to evaluate the Vesely-Fussell importance ($I^{V.F.}$) for every modular and basic component in the fault tree. IMPORTANCE provides these quantities by starting from the top event and taking the top event importance to be equal to 1 (i.e., $I_{TOP}^{V.F.} = 1$), and by proceeding then to higher order modules, simple modular and finally basic components of the fault tree. For a simple AND module it follows from the foregoing that

$$I_{c_i}^{V.F.} = I_M^{V.F.} \quad (i=1, 2,..., n)$$

and

$$I_{M_i}^{V.F.} = I_M^{V.F.} \quad (i=1, 2,..., n)$$

For a simple OR module we have

$$I_{c_i}^{V.F.} = I_M^{V.F.} \cdot \frac{P_i}{P_M} \quad (i=1, 2,..., n)$$

$$I_{M_i}^{V.F.} = I_M^{V.F.} \frac{P_{M_i}}{P_M} \quad (i=1, 2,..., n)$$

For the case of higher order modular gates, the following equations are used in the IMPORTANCE subroutine

$$I_{r_i}^{V.F.} = I_M^{V.F.} \frac{\sum_{j, r_i \varepsilon k_j} P(k_j)}{P(M)} \qquad (3.2.6)$$

$$I_{M_i}^{V.F.} = I_M^{V.F.} \frac{\sum_{j, M_i \in k_j} P(k_j)}{P(M)} \qquad (3.2.7)$$

The numerator of Eqs. (3.2.6) and (3.2.7) is calculated in the IMPORTANCE subroutine, whereas all other probabilities are already calculated in the EXPECT subroutine.

The pressure tank system [1] will be analyzed here to show the procedure for V.F. importance calculations. Figure 3.2.2 shows the pressure tank system and Figure 3.2.3 shows its fault tree. This fault tree consists of 13 free components and one replicated component with no replicated module. Failure data for the fault tree in Figure 3.2.3 are given in Table 3.2.1. The different modules which exist in the pressure tank system fault tree are presented in Figure 3.2.3. To calculate the importance measures, the following calculations are performed in three steps. It should be recalled that all probabilities are calculated in the EXPECT subroutine prior to the use of the IMPORTANCE subroutine.

STEP 1 $\qquad I_{TOP}^{V.F.} = 1$

$$I_r^{V.F.} = \frac{P_r}{P(TOP)} = 2.49937 \times 10^{-1}$$

$$I_{M_1}^{V.F.} = \frac{P_{M_1}}{P(TOP)} = 7.500625 \times 10^{-1}$$

$$I_{M_4}^{V.F.} = I_{M_5}^{V.F.} = \frac{P_{M_4} P_{M_5}}{P(TOP)} = 4.49887 \times 10^{-10}$$

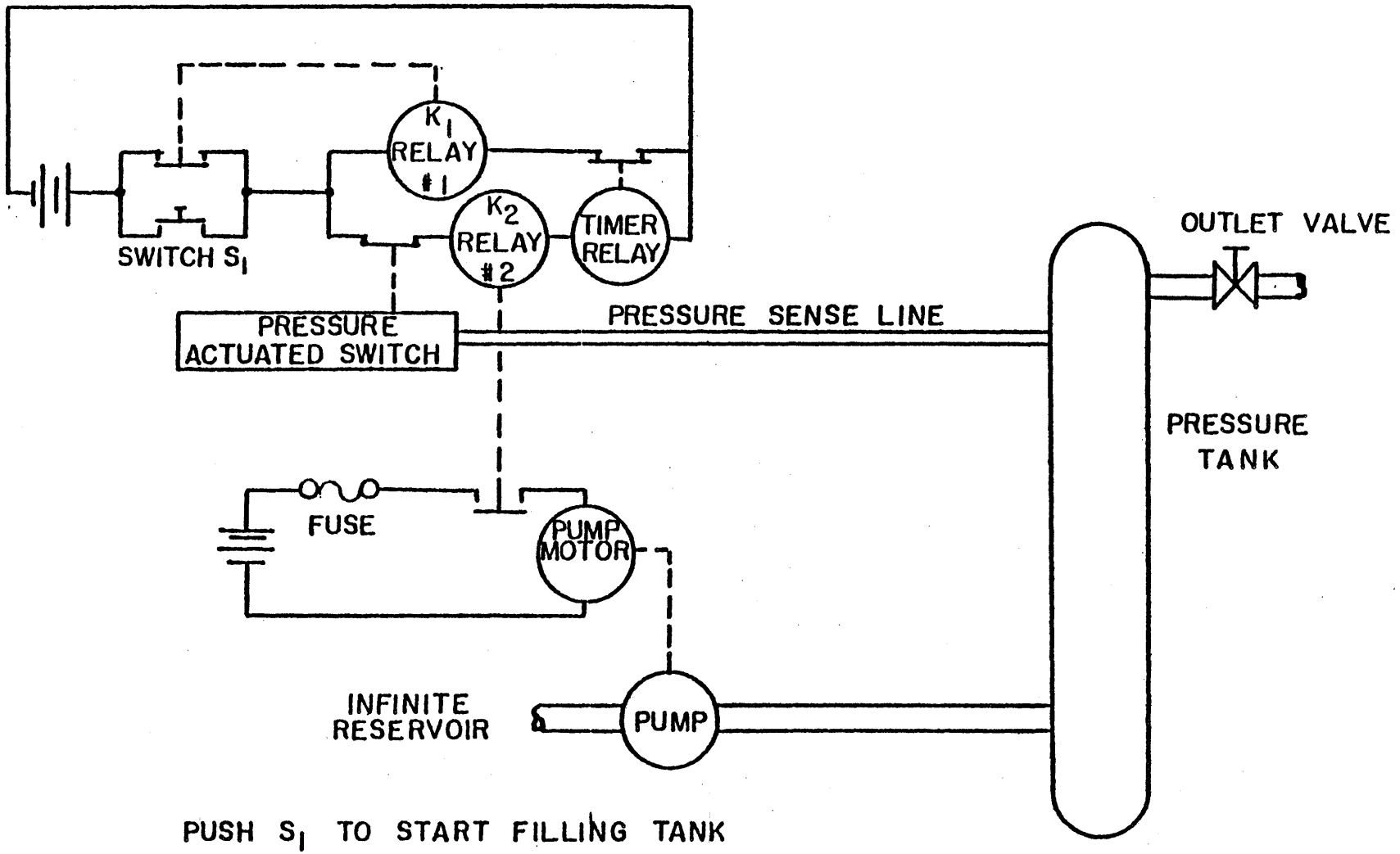


Figure 3.2.2: PRESSURE TANK EXAMPLE

MODULES IN THIS TREE

GATE CORRESPONDING TO A MODULE	TYPE OF MODULE	COMPONENTS AND MODLES ATTACHED TO THE MODULE
9	SYMMETRIC	11, 12, AND 13
4	NESTED	Mg
1	OR SIMPLE MODULE	1, 2, 3, 4, 30001
5	NESTED	5, 6, 7, 8, 9 AND 10

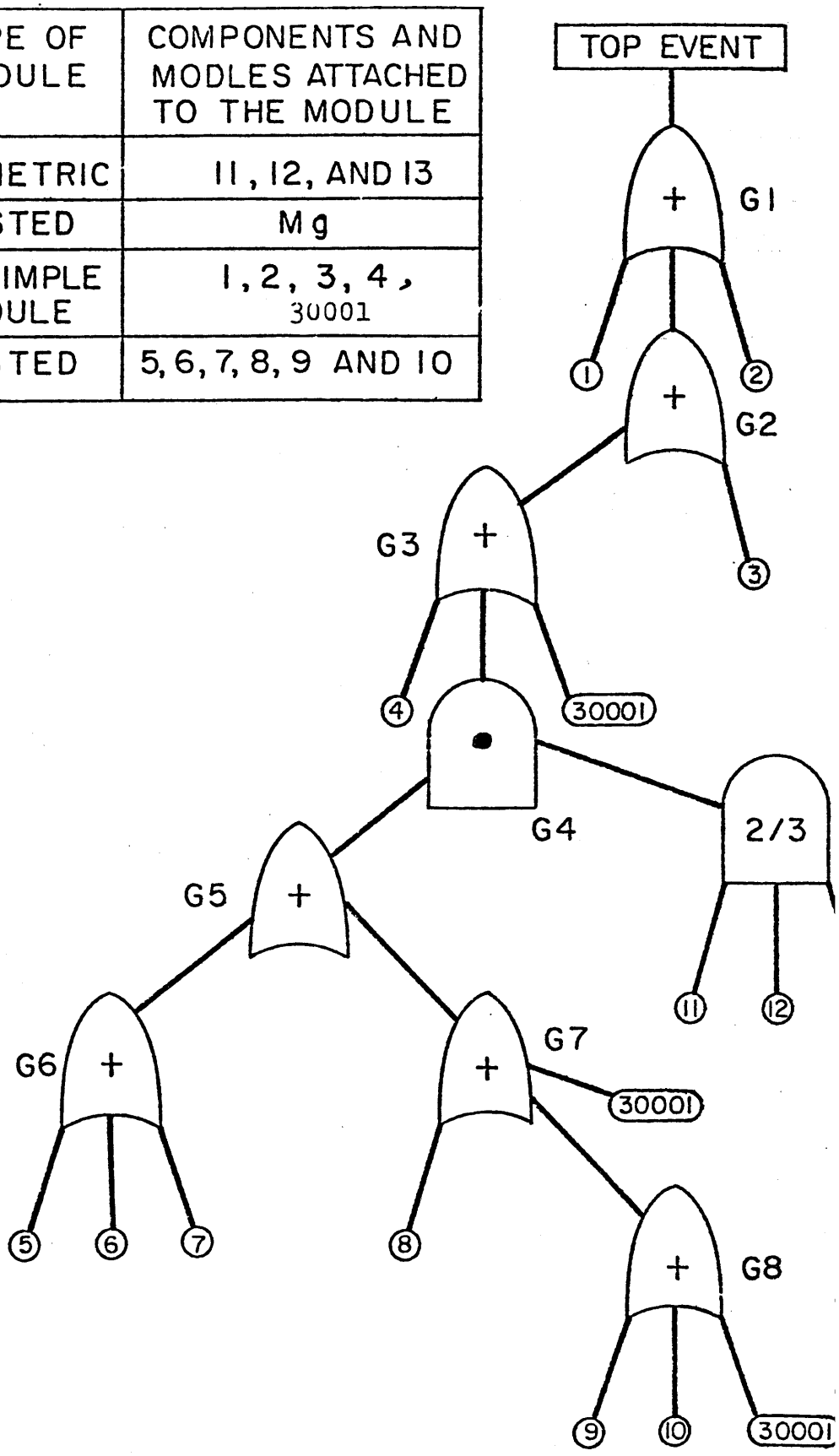


Figure 3.2.3:- Pressure Tank Rupture Fault Tree and Associated Modules (Failure Rate for Component 1, $\lambda_1 = 1 \times 10^{-8}$ hr-1; all others $\lambda = 1 \times 10^{-5}$ hr-1)

TABLE 3.2.1

PRESSURE TANK RUPTURE FAULT TREE FAILURE PROBABILITY DATA

Basic Event i	Event Description	Failure Rate (Per Loading Cycle)
1	Pressure Tank Faulure	10^{-8}
2	Secondary failure of Pressure Tank Due to Improper Selection	10^{-5}
3	Secondary failure of Pressure Tank Due to out-of-tolerance conditions	10^{-5}
4	K2 relay contacts fail to open	10^{-5}
5	S1 switch secondary failure	10^{-5}
6	S1 switch contacts fail to open	10^{-5}
7	External reset actuation force remains on switch S1	10^{-5}
8	K1 relay contacts fail to open	10^{-5}
9	Timer does not "time-off" due to improper setting	10^{-5}
10	Timer relay contacts fail to open	10^{-5}
11	Pressure switch not actuated by sensor 1	10^{-5}
12	Pressure switch not actuated by sensor 2	10^{-5}
13	Pressure switch not actuated by sensor 3	10^{-5}

Replicated Event i	Event Description	Failure Rate (Per Loading Cycle
(3000)1	Common Cause failure among relays K_1, K_2 and timer T	10^{-5}

$$I_1^{V.F.} = I_{M_1}^{V.F.} \frac{P_1}{P_{M_1}} = 2.49937 \times 10^{-4}$$

$$I_2^{V.F.} = I_3^{V.F.} = I_4^{V.F.} = I_{M_1}^{V.F.} \frac{10^{-5}}{P_{M_1}} = 2.49937 \times 10^{-1}$$

STEP 2

$$I_{M_9}^{V.F.} = I_{M_4}^{V.F.} = 4.49887 \times 10^{-10}$$

$$I_5^{V.F.} = I_6^{V.F.} = I_7^{V.F.} = I_8^{V.F.} = I_9^{V.F.} = I_{10}^{V.F.} =$$

$$= I_{M_5}^{V.F.} \cdot \frac{10^{-5}}{P_{M_5}} = 7.49812 \times 10^{-11}$$

STEP 3

$$I_{11}^{V.F.} = I_{12}^{V.F.} = I_{13}^{V.F.} = I_{M_9}^{V.F.} \times \frac{2(10^{-5})^2}{P_{M_9}} = 2.99924 \times 10^{-10}$$

Therefore, if we were to reduce this fault tree based on the above calculations for a cut-off value of 10^{-4}, we should perform the following procedure.

STEP 1

$I_r^{V.F.} = 2.49937 \times 10^{-1}$ KEPT
(since larger than 10^{-4})

$I_{M_1}^{V.F.} = 7.500625 \times 10^{-1}$ KEPT

$I_{M_4}^{V.F.} = I_{M_5}^{V.F.} = 4.49887 \times 10^{-10}$ CANCELED

Therefore, M_9 and components 5, 6, 7, 8, 9, and 10 will be automatically canceled

$$I_1^{V.F.} = 2.49937 \times 10^{-4} \quad \text{KEPT}$$

$$I_2^{V.F.} = I_3^{V.F.} = I_4^{V.F.} = 2.49937 \times 10^{-1} \quad \text{KEPT}$$

From this discussion, the reduced version of the fault tree in Figure 3.2.3 is given in Figure 3.2.4. Naturally, the same procedure is adopted to reduce large fault trees by the use of the IMPORTANCE subroutine in the PL-MOD code. The method applied is presented in the next section.

3.3 Use of the Code PL-MOD to Reduce Large Fault Trees

As discussed in Section 3.2, the subroutine IMPORTANCE in PL-MOD provides the V-F importance measures for all of the fault tree components as well as simple and higher order modules. This subroutine is modified to a new form such that it accounts for the cut-off value to exclude those components and modules that have V-F importances less than this prescribed value. For any further evaluation, the new reduced form of the fault tree could be used.

Obviously, the cut-off value plays an important role in the reduction strategy. That is, different values of the cut-off value result in different reduced fault trees. Therefore, a great deal of attention must be paid to the selection of the cut-off value in order to achieve a desired and accurate form of the reduced fault tree.

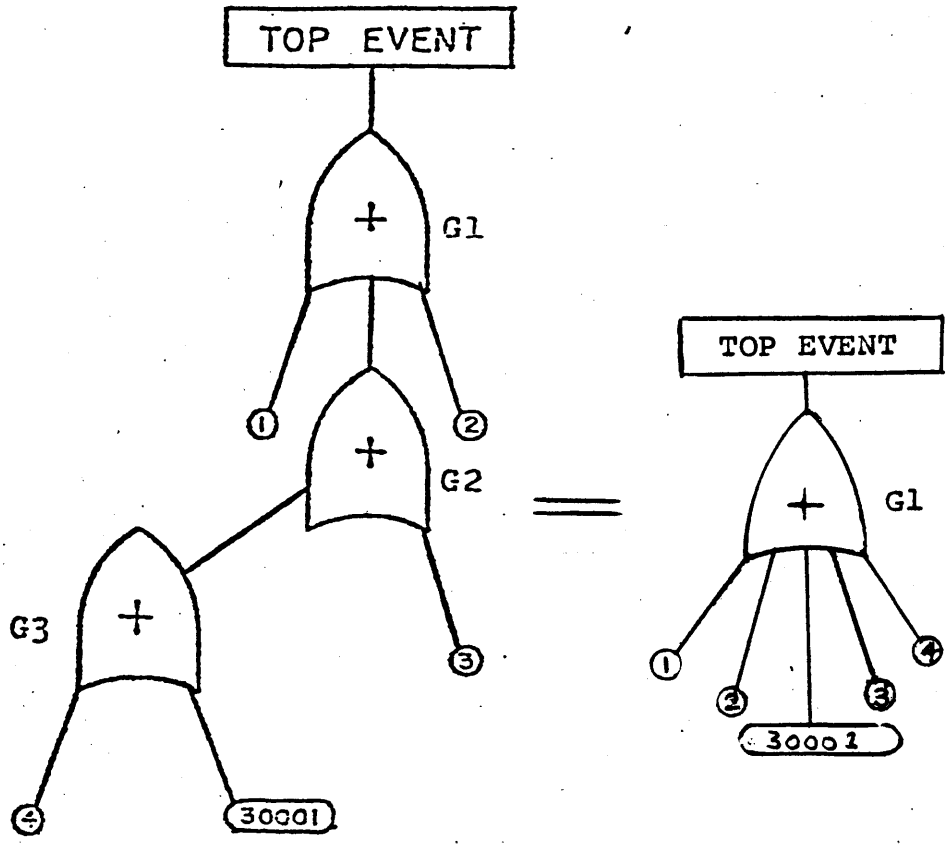


Figure 3.2.4

Reduced Fault Tree for the Pressure Tank Rupture for an Importance Cut-Off Value of 1×10^{-4}

The new form of the IMPORTANCE subroutine is linked with the code PL-MODT, so that for future developments of the code, the reduced version of fault trees will be used by the code to evaluate time-dependent behavior of the fault tree.

In the modified form of the IMPORTANCE subroutine, the top event importance will be set equal to 1. Starting from the top event it proceeds to the bottom. First, V-F importances of the higher modules and replicated events that are connected to the top event are calculated. Next, the code automatically removes all modules and replicated events whose V-F importances are less than the prescribed cut-off value. Each higher order module which is removed contains some other modules and free components that are attached to them. Therefore, there is no need to compute the V-F importance of any of these attached members of the removed higher order modules because their importances are always equal to or less than that of the parent module which by itself is lower than the cut-off value. If, however, the higher order module is not removed because of its importance being higher than the cut-off value, the code will proceed to calculate importances of simple modules and components attached to it, and to automatically remove components and simple modules with importance low enough to be cancelled.

Therefore, starting from the top event of an unreduced fault tree, all of the branches of the tree with low importances will be cut. Those lower order branches and leaves which show low importances compared to the cut-off value will be cancelled. The pruned tree is the reduced version of the original tree.

The pruning process is performed by setting the variable STATE(2,I) which is the measure of the V-F importance for components equal to zero. If the module is reduced, then automatically all components I connected to this module are cancelled (i.e., STATE(2,I) = 0).

For example, the removal of a simple AND module proceeds as follows:

```
IF(PROP.LIM=1 ε PROP.TIL(1)=0) THEN DO;
  PROP.REL(2)=0
GO TO EME 1;
(To calculate the importance of the NOT gate if any)
END;
ELSE PROP.REL(2)=PER.REL(2)
IF(PROP.REL(2) < CUT-OFF) THEN PROP.REL(2)=0;
DO IT=1 TO PROP.LIM;
STATE(2,PROP.T_IL(IT))=PROP.REL(2);
END;
```

Therefore, all of the components input to the AND gate will have an importance which is either equal to zero (i.e., cancelled) or equal to the importance of the AND module.

3.4 Reductions of LPRS and HPIS Fault Trees

In order to demonstrate the reduction process, the reduced fault tree of the Low Pressure Recirculation System presented in Appendix II of WASH-1400 is further reduced by using different values for the cut-off limit. The maximum change in the top event occurrence for the following range of cut-off values 10^{-2} cut-off 10^{-4} has been observed not to exceed 1% from its published version, i.e.,

$$P_{(TOP)_{reduced}} \equiv (1 \pm 0.01)\ P_{unreduced}$$

For example, for the LPRS with 6 replicated and 61 non-replicated components, the results for the various reduced versions which follow from the application of the different cut-off values are summarized in Table 3.4.1.

TABLE 3.4.1

Percentage Change in the Top Event Occurrence of Reduced Trees of LPRS for Different Cut-off Values

Cut-off Value	No. of Free Components Remained	No. of Replicated Components Remained	Percentage Change in the Top Event
10^{-5}	43	5	No change
10^{-4}	37	5	No change
10^{-3}	20	4	No change
10^{-2}	12	2	1.1%

It becomes obvious from Table 3.4.1 that for cut-off values in the range of 10^{-5} to 10^{-3} no change in the top event occurs and, therefore, one can safely use the upper bound (i.e., 10^{-3}) for the cut-off value in order to get the most reduced tree, thereby saving computation time. Even the use of 10^{-2} for the cut-off provides a still reasonable fault tree for the LPRS which results in a change of only 1.1% compared to the originally published version. Investigation of the reduced fault trees showed that they have essentially the same low order cut-sets as the original fault tree, unless components of a low order cut-set have very low probabilities so that they result in a small V-F importance.

It is very important to understand that a component with low probability or unavailability will not necessarily result in a low V-F importance. For example, for the pressure tank example Component 1 is found to have

$$I_r^{V.F.} = 2.49937 \times 10^{-4} \quad \text{and}$$

$$I_{5-10}^{V.F.} = 7.49812 \times 10^{-11}$$

However, $P_1=10^{-8}$ and $P_{5-10}=10^{-5}$, and, therefore, even though the replicated event has a probability of occurrence which is by 3 orders of magnitude less than that of components 5 through 10, its importance is 7 orders of magnitude larger. This means that

any reduction process which is solely based on the orders of event probabilities does not necessarily result in a meaningful reduced fault tree. To have the same order of magnitude for the importance of the replicated event one needs to reduce the probability of the replicated component down to $P_r=10^{-15}$ which results in

$$I_r^{V.F.} = 2.49937 \times 10^{-11}$$

From the above discussion it follows that the reduction schemes in the code PL-MODT do not only provide an excellent objective engineering judgment to reduce a fault tree, but it is also instrumental in improving the design objectives of the system under consideration.

The same kind of study is performed for the HPIS and for a cut-off value of 10^{-3}. The HPIS fault tree consists of 142 free and 13 replicated components. The reduction process resulted in 53 free and 9 replicated components. The top event remained unchanged and the low order cut-sets are almost the same.

Since there are no calculations or iterations involved in the reduction procedure, the computer cost increase is negligibly small compared to the modularization process. For example, for the LPRS fault tree, the CPU time for modularization and evaluation by the PL-MODT amounts to 0.46 seconds where it is about 0.47 seconds if a reduction with cut-off value equal to 10^{-2} is requested in addition to the above calculations. Although the saving does not become apparent for this steady-state example, it

should be pointed out that a substantial saving will result if one uses the reduced version for time-dependent fault tree analysis and Monte-Carlo simulations.

REFERENCES

[1] J. Olmos and L. Wolf, A Modular Approach to Fault Tree and Reliability Analysis, Department of Nuclear Engineering, MIT, MITNE-209, August 1977.

[2] H.E. Lambert, Fault Tree for Decision Making in System Analysis, UTRL-51829, October 1975.

[3] G. Apostolakis, Mathematical Methods of Probabilistic Safety Analysis, UCLA-ENG-7464, September 1974.

4. INCORPORATION OF A MONTE-CARLO SIMULATION PACKAGE INTO THE CODE PL-MOD

4.1 Introduction

It is a well-known fact that the Monte-Carlo simulation always involves many calculations of the same kind. That is to say that in order to calculate the top event of a fault tree, several hundred to several thousand simulations have to be performed on the accuracy desired. Each top event probability corresponds to a specific set of possible components' failure data. The code PL-MOD provides the modular cut-sets which in turn are very simple and efficient to use for the top event calculations.

The codes SAMPLE [3] and LIMITS [2] are two examples of codes which have recently been used for Monte-Carlo simulations. The SAMPLE code has been used in WASH-1400 to calculate reliability bounds. Both codes require a system function as input to identify the logical dependencies in the systems that are being analyzed. For very large fault trees, construction of this equation is a difficult process and most of the time serious errors may result from mistakes during the construction of this function. Therefore, for large fault trees it would seem more comfortable to use the fault tree itself as an input rather than the system equation derived from the cut-sets of the fault tree.

The code PL-MOD is used to calculate the modular cut-sets and the NUMERO subroutine in this code is modified such that it can handle fault trees consisting of steady-state components with

some uncertainties associated with their failure rates. The new code is called PL-MODMC, where MC stands for the Monte-Carlo package added to the PL-MOD code.

The failure rates of the components are assumed to be log-normally distributed. The same sorting routine which has been used in the code LIMITS is adopted here since it has been demonstrated that it is very fast and efficient. However, an efficient PL/1 random number generation is developed and used in PL-MODMC. This random number generator was found to be very simple and fast. The output of the PL-MODMC consists of the top event probability for any arbitrary set of confidence limits. Also, mean and point unavailabilities along with the top event standard deviation are also calculated. The minimum probability and maximum error for each confidence level will also be provided by PL-MODMC.

4.2 Mathematical Concepts of the Monte-Carlo Method in Fault Tree Analysis

As was discussed in the previous chapters, fixed values for failure rates and other data are commonly referred to as point values. In a probabilistic approach, because of the variations and uncertainties in the failure rates and other parameters, these quantities should be treated as random variables.

In the code PL-MODMC, the lognormal distribution serves as the basis of the uncertainty propagation. However, the use of other distributions in the code should be easily established.

The log-normal distribution is found to be more adequate and convenient to be used in fault tree analysis because the raw input data are sparse and the assessed ranges are large, having widths of one or two orders of magnitude. Also, examinations of the existing data showed that the log-normal distribution gives an adequate empirical fit.

The random variable t has a log-normal distribution if its logarithm follows a normal distribution. The distribution is skewed to the right. For example, having a possible range between t/f and t.f (f is a factor) for a log-normal distribution, this range transfers to logt $\pm$ logf which is a description of normally distributed data. Therefore, the log-normal distribution describes data which vary by factors. On the other hand, the normal distribution describes data which vary by additive or subtractive increments.

Most of the failure data can vary by factors. For example, a failure rate estimated at 10^{-6} could vary from 10^{-7} to 10^{-5} which is ($10^{-6}/10$) to (10^{-6}x10). The log-normal distribution has two parameters: μ specifying the distribution scale and σ specifying its shape. The probability density function (P.d.f) for a log-normal distribution is given by:

$$f(t) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp-\left[\frac{(Lnt - \mu)^2}{2\sigma^2}\right]; \quad t>0 \qquad (4.2.1)$$

From Eq. (4.2.1) the following parameters can be calculated:

Mode (the most probable value): $t_m = e^{\mu-\sigma^2}$

Median: $t_{0.5} = e^{\mu}$ or in terms of the upper and lower bounds

$$t_{0.5} = \sqrt{x_u \cdot x_L}$$

Mean: $\bar{t} = e^{\mu+\sigma^2/2}$

Variance: $v = e^{2\mu+\sigma^2}[(e^{\sigma^2}-1)]$

The Monte-Carlo technique by itself is very simple. Once the log-normal distribution parameters are known for all of the components of a fault tree, these values are then used to compute a point value for the top event probability using the cut-sets which have been provided by the analysis of the fault tree. The procedure is repeated for a large number of trials and the results are sorted to obtain an estimate of the system unavailability distribution.

One of the important factors in a Monte-Carlo simulation is its accuracy. The accuracy of an estimated distribution by sampling is discussed in [4].

It is known that

$$pr(|X(P) - P| \leq \varepsilon) = erf(\frac{t}{\sqrt{2}}) + R \qquad (4.2.2)$$

where

$$X(P) = M/N$$

N is the number of trials; M is the number of successes.

$$t = \varepsilon\sqrt{\frac{N}{pq}}$$

P = the p parameter of a binomial distribution

q = 1 - p

R = error associated with the probability measure which is given by

$$|R| \leq \frac{e^{-t^2/2}}{\sqrt{2\pi Npq}} + \frac{0.2+0.25\,|P-q|}{Npq} + e^{-(\frac{3}{2})\sqrt{Npq}} \qquad (4.2.3)$$

erf(t) = error function of a variable $t=\sqrt{\frac{2}{\pi}}\int_o^t e^{-\frac{u^2}{2}}.du$

For example, $pr(|X(0.95) - 0.951) \leq 0.01)$ means the probability that $X_{0.95}$ lies between the probability limits of 0.94 and 0.96. In other words, the value of X is associated with a confidence interval of $P\pm\varepsilon$, with a probability of erf $(\frac{t}{\sqrt{2}} + R)$, and as can be seen it is not dependent on the distribution but on the confidence level.

In the code PL-MODMC, the value of ε is calculated by subtracting the smallest confidence level from zero. For example, if the smallest confidence level is 0.5% (i.e., the largest confidence level is 99.5%), then the accuracy is ε=0.5. Therefore, the minimum probability would be

$$\text{erf}(\frac{t}{\sqrt{2}}) - \max(R)$$

For a large sample size N, it follows that

$$pr(|t_s - t_p| \leq \frac{1.36}{\sqrt{N}}) = 0.95$$

where t_s is the estimated distribution fractile, and t_p is the corresponding exact cumulative distribution value of the underlying population from which the sample was picked randomly. For example,

for a sample size of 2000 we will be 95% sure that the estimated distribution deviates by not more than 0.03 from the exact distribution.

4.3 The PL/1 Random Numbers Generator Used in the Code PL-MODMC

The task is to generate random numbers for the calculation of samples from the components' failure rate distributions. A PL/1 random number generator has been developed for this purpose and implemented into the PL-MODMC. The numbers generated by this generator are normally deviating about a specified value which the user must provide as an input. The Central Limit Theorem is applied to generate normal random numbers. Then, each of these random numbers is used to calculate the failure rates from specified log-normal distributions. The PL/1 listing of this random number generator procedure is given below.

```
RAND: PROCEDURE OPTIONS (MAIN);
DECLARE (IY,A,IY1,X) FLOAT DECIMAL(16);
DECLARE CEIL BUILTIN;
GET LIST (N,X,M);
IY=X/0.499977;
IY1=IY;
DO WHILE (M>0);
A=0;
DO I=1 TO N;
```

```
IY1=CEIL (IY1);
IY=IY-IY1+1;
A=A+IY;
IY1=IY*65539;
  IY=IY1;
  END;
PUT DATA (A)(SKIP (2), F(12,5));
  M=M-1;
  END;
  END RAND;
```

This random number generator is very fast, primarily because it is written in PL/1 language. The variable N denotes the approximation of the Central Limit Theorem. The variable X is any odd-starting number to generate random numbers. Finally, the variable M is the total number of random numbers to be generated. The computation time for generating 4000 random numbers by using the above procedure is 0.026 sec. of CPU time.

4.4 Description of the Code PL-MODMC

As was discussed before, the code PL-MODMC is developed to incorporate the capability of a Monte-Carlo simulation using the modular cut-sets that the code PL-MOD generates. In the modified form of the code, the subroutines IMPORTANCE and STATE-IN are not used. Therefore, the code PL-MODMC does not perform any importance calculations and automatic fault reductions. However,

it is acknowledged that it would be much simpler if the code would automatically reduce the tree and then perform a Monte-Carlo simulation. This option is not incorporated into the code yet, but it is hoped that it will be incorporated during the future research activities. This would not only reduce the computation time, but would also provide almost the same top event probability for different confidence levels of the original unreduced tree.

The Monte-Carlo code PL-MODMC has been developed in two steps:

Step 1 includes the development of the MONTCA subroutine.

Step 2 includes some minor changes in the NUMERO and EXPECT subroutines.

4.4.1 Step 1

The subroutine MONTCA consists of a special procedure to choose failure rate samples from each component log-normal distribution. First, any arbitrary combination of confidence levels is given by the user, and an array will be allocated to store them. Also, the total number of trials is provided by the user. Then, the computer calculates the maximum error and the minimum probabilities for each confidence level by using Eqs. (4.2.2) and (4.2.3).

Second, a special procedure enables the code to use point values of the failure rates to calculate the top event probability (Class 1 components in the PL-MODT code).

Third, the Monte-Carlo simulation starts by using the median and spread values given by the user whichare stored in the allocated array MEDIAN(I) and FEN2(I), respectively.

Finally, the following procedure similar to that of the SAMPLE code is employed:

```
  DO I=1 TO VEN;
[VEN is the total number of replicated and non-replicated
 events, i.e., VEN=FUN+DUN;]
IF (MEDIAN(I)=0) THEN FEN1(I)=0;
ELSE FEN1(I)=LOG(MEDIAN(I));
IF (FEN2(I)¬=0) THEN
FEN2(I)=LOG(FEN2(I))/1.64;
END;
XP1=SQRT(12/N);
(N is the same variable described in Sec. 4.3)
ALLOCATE TOP-P;
(to store different top event probabilities that are
 calculated in the EXPECT subroutine)
    .
    .  RANDOM GENERATOR
    .  procedure follows here
    .
XP1=XP1*(AA-0.5*N)*FEN2(I)+FEN1(I);
(AA calculated by the random number generator, it is
 identical to the variable A described in Section 4.3)
IF (XP1=0) THEN MEDIAN (I)=0;
ELSE MEDIAN(I)=EXP(XP1);
```

At this point, the values calculated at each trial will be assigned to the STATE and STATD arrays which are ALLOCATED before as follows:

```
DO K=1 TO FUN;
STATE (1,K)=MEDIAN(K);
END;
DO N=1 to DUN;
  J=N+FUN;
STATE (1,N)=MEDIAN(J);
END;
```

At this stage, the modified subroutine EXPECT is called to calculate the top event probability from the modular cut-sets by using the data obtained in the trials for the component unavailabilities that are stored in the MEDIAN array. Therefore, it follows:

```
CALL EXPECT;
TOP-P (ASACH)=REY;
ASACH=ASACH+1;
```

where REY is the top event probability that is assigned to this variable in the EXPECT subroutine. When ASACH exceeds the total number of trials requested by the user, the code calculates the mean value and the standard deviation of the top event probability from the TOP-P(ASACH).

At the end, the same sorting routine as utilized in the LIMITS code is used in the PL-MODMC with some changes to allow its formulation in PL/1 language. This sorting method has an empirical computer time requirement directly proportional to $n^{1.226}$. The method does not take up any additional computer core memory. When the sorting process is finished, the spread factors for the top event are calculated by simply using the following equations.

$$F_1 = \frac{P_{50}}{P_{05}} \tag{4.4.1}$$

$$F_2 = \frac{P_{95}}{P_{50}} \tag{4.4.2}$$

where P_{05}, P_{50}, and P_{95} are the top event probabilities which are calculated in the sorting process for 5%, 50% (which is also the median value for the top event) and 95% confidence levels. The Monte-Carlo calculation ends by printing the top event probabilities for different confidence levels along with the mean values, standard deviations, median values, 5% and 95% error factors (spread values), and the point unavailability.

4.4.2 Step 2

The subroutine EXPECT is used in PL-MODMC to calculate the top event probability for each trial. The same form of the EXPECT subroutine that is used in the PL-MODT has been modified to be used in this code.

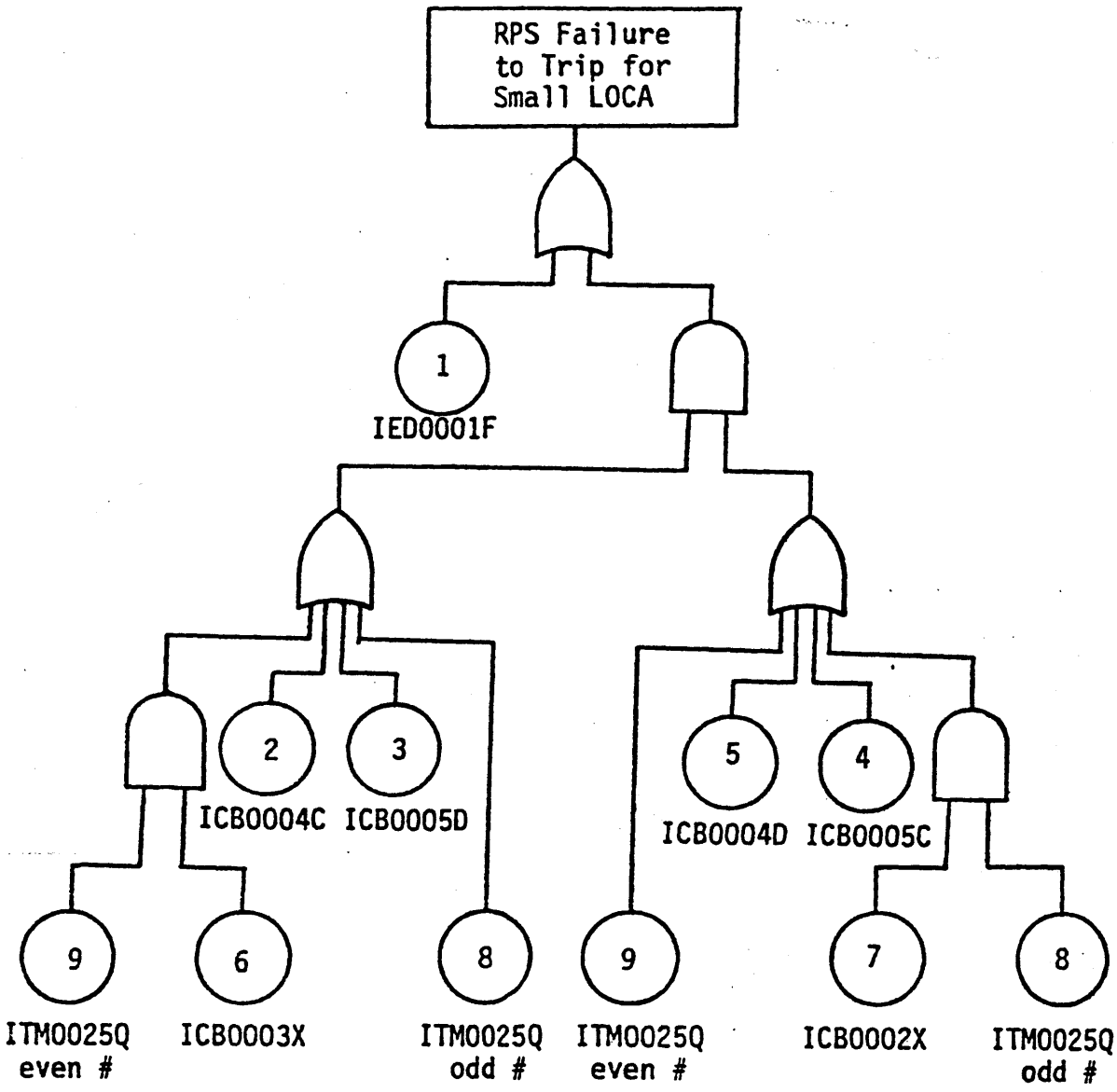
The new form of EXPECT is designed such that there is no printout of probabilities for modules and the top event at the end of each trial. Rather, only top events will be stored in the TOP-P (ASACH) array. Also, some features are added to the subroutine to utilize all spaces used to store different variables that are not necessary for a Monte-Carlo calculation, such as probabilities or different higher-order modules that are automatically calculated in each trial by the EXPECT subroutine.

4.5 Examples

Two examples are provided in this section: first, the same reduced fault tree given in the LIMITS code [2] for a Reactor Protection System; second, the LPRS fault tree given in WASH-1400 is used to calculate the probability of the top event for different confidence levels.

For the reduced fault tree of the Reactor Protection System (Figure 4.5.1), the CPU times for the three codes LIMITS, SAMPLE, and PL-MODMC are found and presented in Figure 4.5.2. The probabilities calculated by the PL-MODMC agree well with the other two codes. The results for the fault tree in Figure 4.5.1 and failure data in Table 4.5.1 are given in Table 4.5.2. Slight differences from those given in [2] and in Table 4.5.2 are because of the different techniques that are employed. For instance, random numbers are generated by slightly different techniques.

It should be noticed that CPU time is comparatively higher for PL-MODMC than for the LIMITS code. This is because of the



IED0001F: Failure of suff. no. of rods to drop when power is removed.
ICB0004C: Breaker BYA fails closed
ICB0005D: Breaker RTA fails to open
ICB0005C: Breaker BYB fails closed
ICB0004D: Breaker RTB fails to open
ICB0003X: Breaker BYA closed due to test and maintenance
ICB0002X: Breaker BYB closed due to test and maintenance
ITM0025Q (odd #): Train 'A' logic fault
ITM0025Q (even #): Train 'B' logic fault

Figure 4.5.1: Reduced Fault Tree of the Reactor Protection System for a PWR [2].

Event	Failure Rate ($Hr.^{-1}$)	Fault Exposure Time (Hr.)	Unavailability q_i	Error Factor
IED0001F			$1.7X10^{-5}$	10
ICB0004C	$1.0X10^{-6}$	360	$3.6X10^{-4}$	3
ICB0005D			$1.0X10^{-3}$	3
ICB0005C	$1.0X10^{-6}$	360	$1.0X10^{-3}$	3
ICB0004D			$3.6X10^{-4}$	3
ICB0003X			$6.1X10^{-3}$	4
ICB0002X			$6.1X10^{-3}$	4
ITM0025Q				
Odd #			$9.7X10^{-4}$	10
Even #			$9.7X10^{-4}$	10

Table 4.5.1: Failure Data for the Reactor Protection System of a Pressurized Water Reactor [2]

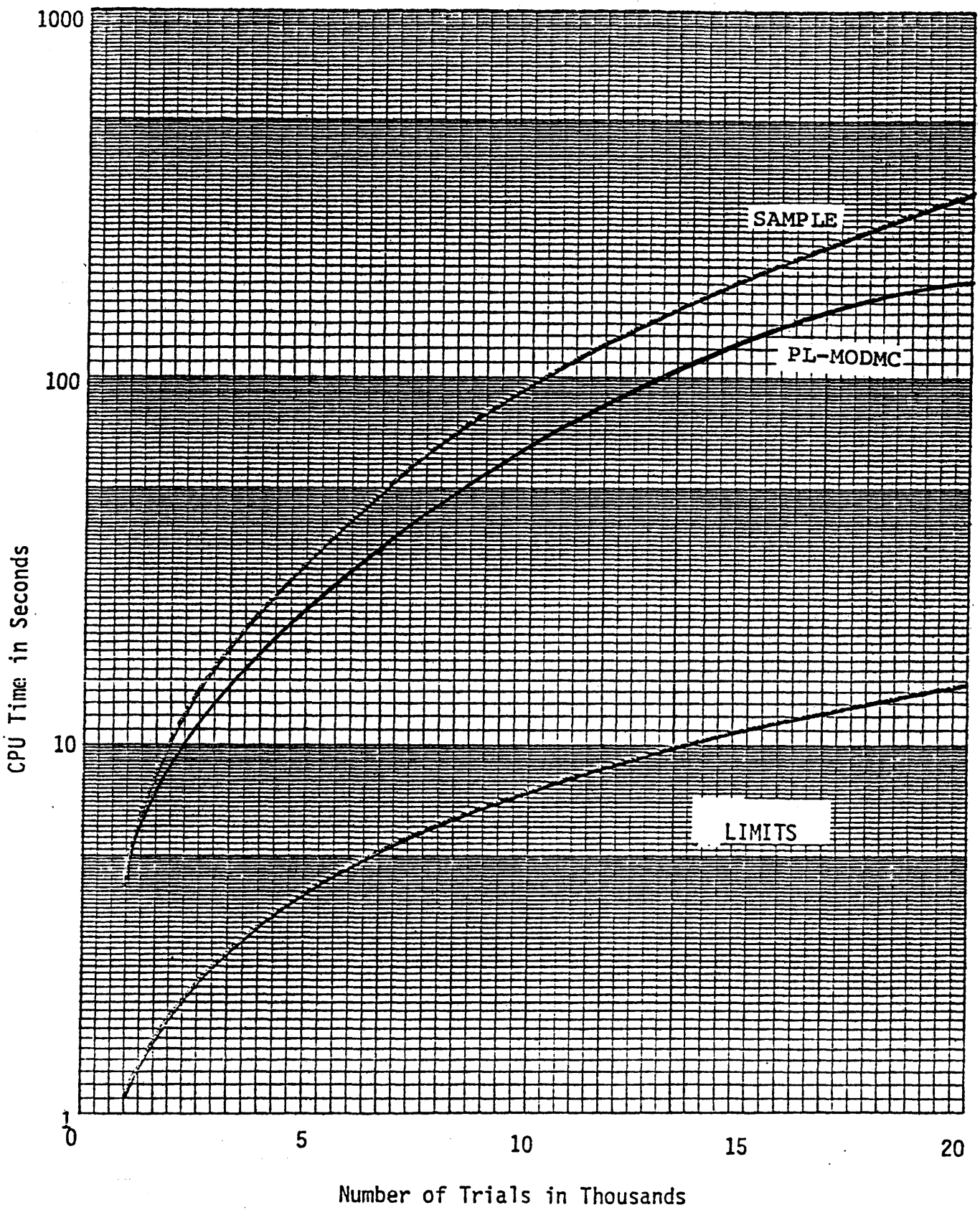


Figure 4.5.2: Execution Times of the LIMITS, SAMPLE & PL-MODMC Codes

MEAN PROB= 9.813583E-05 STANDARD DEVIATION= 1.264061E-04

ERROR FACTOR(5%) = 3.62884 ERROR FACTOR(95%) = 4.65287 MEDIAN PROB= 6.260029E-05

CONFIDENCE LEVEL	PROBABILITY	MAX ERROR	MIN PROBABILITY
0.50000	9.086117E-06	1.855380E-02	9.814456E-01
1.00000	1.053545E-05	9.118769E-03	9.905009E-01
2.50000	1.384564E-05	6.371949E-03	9.700879E-01
5.00000	1.725076E-05	8.733936E-03	8.865085E-01
10.00000	2.237782E-05	1.027986E-02	7.511274E-01
15.00000	2.654296E-05	1.026612E-02	6.676319E-01
20.00000	3.167291E-05	9.981245E-03	6.132597E-01
30.00000	4.048581E-05	9.428132E-03	5.501711E-01
40.00000	5.006256E-05	9.084430E-03	5.204290E-01
50.00000	6.260029E-05	8.947823E-03	5.115520E-01
60.00000	7.702695E-05	9.084430E-03	5.204290E-01
70.00000	9.864400E-05	9.428132E-03	5.501711E-01
80.00000	1.306296E-04	9.981245E-03	6.132597E-01
85.00000	1.594959E-04	1.026612E-02	6.676319E-01
90.00000	2.061887E-04	1.027986E-02	7.511274E-01
95.00000	2.912714E-04	8.733936E-03	8.865085E-01
97.50000	3.989008E-04	6.371949E-03	9.700879E-01
99.00000	6.001061E-04	9.118769E-03	9.905009E-01
99.50000	7.802579E-04	1.855380E-02	9.814456E-01

Table 4.5.2: Reactor Protection System Top Event Unavailability for 5000 Trials Using PL-MODMC

fact that the calculation time of PL-MODMC includes the modularization and the minimal modular cut-set generation. Therefore, the curves in Figure 4.5.2 do not represent a real comparison between PL-MODMC and the other two codes. Although the modularization needs only a small fraction of the time consumed in the PL-MODMC, the CPU time is primarily due to the establishment of several structures and calling some subroutines at each individual trial. A more honest comparison between the codes would be obtained by adding the CPU time consumed by the fault tree analysis code such as MOCUS to calculate the cut-sets prior to the use of LIMITS or SAMPLE codes to the CPU times consumed by the LIMITS and SAMPLE codes.

From the discussion above, it becomes apparent that the use of the code PL-MODMC for a Monte-Carlo analysis of very large fault trees may be very economical. It has also been found that the CPU time in PL-MODMC is more sensitive to the number of trials rather than the size of the tree.

The LPRS fault tree with 61 free components and 6 replicated components were input into the PL-MODMC and 1200 trials were simulated. The results are presented in Table 4.5.3. The computation time including the modularization is 52 seconds. In general, the results agree well with those given in WASH-1400 as can be seen by the comparison presented in Table 4.5.4.

Table 4.5.3: LPRS Top Event Unavailability for 1200 Trials Using PL-MODMC

MEAN PROB= 1.410070E-02 STANDARD DEVIATION= 7.909754E-03

ERROR FACTOR(5%)= 2.13365 ERROR FACTOR(95%)= 2.21611 MEDIAN PROB= 1.243022E-02

CONFIDENCE LEVEL	PROBABILITY	MAX ERROR	MIN PROBABILITY
0.50000	3.575392E-03	1.893921E-01	7.656468E-01
1.00000	4.353750E-03	1.224911E-01	7.222902E-01
2.50000	5.138319E-03	8.370483E-02	5.512648E-01
5.00000	5.825788E-03	6.371158E-02	4.198762E-01
10.00000	6.662294E-03	4.763011E-02	3.150179E-01
15.00000	7.352761E-03	4.019815E-02	2.677403E-01
20.00000	8.052737E-03	3.585986E-02	2.404665E-01
30.00000	9.505350E-03	3.113346E-02	2.112458E-01
40.00000	1.096696E-02	2.891828E-02	1.982517E-01
50.00000	1.243022E-02	2.810337E-02	1.945992E-01
60.00000	1.385169E-02	2.891828E-02	1.982517E-01
70.00000	1.591348E-02	3.113346E-02	2.112458E-01
80.00000	1.868913E-02	3.585986E-02	2.404665E-01
85.00000	2.036301E-02	4.019815E-02	2.677403E-01
90.00000	2.358368E-02	4.763011E-02	3.150179E-01
95.00000	2.754679E-02	6.371158E-02	4.198762E-01
97.50000	3.381504E-02	8.370483E-02	5.512648E-01
99.00000	3.927734E-02	1.224911E-01	7.222902E-01
99.50000	4.923805E-02	1.893921E-01	7.656468E-01

TABLE 4.5.4

Comparison of the LPRS Fault Tree Simulation Using PL-MODMC And the Results Calculated by the SAMPLE Code in WASH-1400

Confidence Level	WASH-1400 (SAMPLE)	PL-MODMC
50%	1.3×10^{-2}	1.24×10^{-2}
5%	4.4×10^{-3}	5.82×10^{-3}
95%	3.1×10^{-2}	2.75×10^{-2}

REFERENCES

[1] J. Olmos and L. Wolf, "A Modular Approach to Fault Tree and Reliability Analysis", MITNE-209, Dept. Nucl. Engng., MIT (August 1977).

[2] Y.L. Lee and S.L. Salem, "Probability Intervals for Reliability of Complex Systems Using Monte-Carlo Simulation", UCLA-ENG-7758 (December 1977).

[3] "Reactor Safety Study", WASH-1400, U.S. Nuclear Regulatory Commission (October 1975).

[4] D.K. Lloyd and M. Lipow, "Reliability: Management, Methods, and Mathematics", 2nd Ed. (1977).

[5] M. Modarres and L. Wolf, "PL-MODT: A Modular Approach to Fault Tree and Reliability Analysis", ANS Meeting, San Diego, June 1978.

5. CONCLUSIONS AND RECOMMENDATIONS

During this research period, the code PL-MOD has been extended to include some additional features such as time-dependent basic events, automatic fault tree reduction, and Monte-Carlo simulation.

The original algorithm to derive a fault tree's modular composition directly from its diagram was already reported by Olmos and Wolf [1]. The procedure consists of piecewise collapsing and modularizing portions of the tree, until eventually the fault tree structure is described as a set of modular equations recursively relating the top tree event to its basic component inputs. The structural representation of fault trees containing replicated events was shown to necessitate the use of higher-order gate modules. A Boolean vector representation was chosen to express the family of minimal cut-sets corresponding to a higher-order gate. The code PL-MOD is written in PL/1 in order to take advantage of the list processing capabilities available in this computer language. For instance, extensive use is made of based structure pointer variables and dynamic storage allocation. Moreover, the manipulation of Boolean state vectors, which requires the treatment of higher-order modular structures, is conveniently performed by using bit-string variables.

In a second step, the code PL-MODT has been developed to handle time-dependent events and automatic fault tree reduction

based on the importance measures of different modules and components in the tree. Four classes of components are used to include the time-dependent behavior of basic free and replicated components. These classes of components are as follows:

a) Class 1 components, which are components with a finite time-independent probability of occurrence.

b) Class 2 components, which are non-repairable components where the failure rates are time-independent.

c) Class 3 components, which are repairable components where the failure and repair rates are time-dependent.

d) Class 4 components, which are repairable components, whose failures are detected upon periodic inspection.

The code PL-MODT employs the same modularization algorithm utilized in the code PL-MOD, but includes the aforementioned four classes of components. Various approximations are employed in this code to make the time-dependent calculations fast and economical. Some useful features such as the calculation of the mean unavailability and the option of different time step mesh sizes during different periods in life have been incorporated into PL-MODT. The upper bound cut-set probability approximation is changed to the prediction of exact values, which enables the PL-MODT code now to handle large unavailabiltiy values (i.e., unavailabilities close to 1).

It is found that the PL-MODT calculations are performed efficiently, economically, and accurately. Benchmarking tests against the codes KITT and FRANTIC have clearly demonstrated these advantages. It should be noticed that the use of the KITT and FRANTIC codes is not as straightforward as the application of PL-MODT because prior to the use of KITT, the code PREP must be employed in order to find low-order cut-sets of the fault tree. Similarly, in FRANTIC, a system equation (QS) has to be supplied by the user which indicates the logical dependencies of input components. This equation usually becomes quite complicated for large fault trees. In contrast to these procedures, in PL-MODT only the fault tree structure and components' failure characteristics are inputted. Thus, appreciable savings in computation time and manpower result when large fault trees are to be analyzed by PL-MODT.

A reduction scheme of a tree is incorporated into the PL-MODT code to provide an objective judgment for the reduction of a large fault tree, such that the probability of occurrence of the top event is essentially maintained. For this purpose, the cut-off value is defined. It provides a criterion for eliminating those modules and components whose importances are lower than this prespecified value. In the reduction analysis of fault trees such as for the LPRS and the HPIS, it is found that cut-off values in the range of 10^{-2} to 10^{-3} provide the simplest and yet still accurate reduced versions of the trees. It is demonstrated that the use of the reduced version of a large fault tree enhances the

understanding and processing of the tree for further analysis (i.e., Monte-Carlo analysis, time-dependent analysis, etc.).

Due to uncertainties associated with the failure data of fault tree components, it is important to implement a Monte-Carlo analysis to propagate input uncertainties up to the top event. For this purpose, the code PL-MODMC was developed. It enables the user to work with large fault trees. The code has been benchmarked against the well-known SAMPLE code and the recently developed LIMITS code. The comparison showed good agreement for the sample cases considered. The codes SAMPLE and LIMITS require the generation of fault tree cut-sets prior to their Monte-Carlo simulations (i.e., using either the codes PREP or MOCUS), whereas PL-MODMC uses the modular minimal cut-sets that are already generated before calling the MONTCA subroutine which is described in Chapter 4.

Although the codes PL-MOD, PL-MODT, and PL-MODMC have been proven to be valuable tools for various aspects of fault tree analysis and evaluation, they all have the same restriction in common to only handle replicated modular gates, i.e., replicated gates representing a supercomponent event independent from all other gates in the tree. However, in general, replicated gates may exist which do not represent only a supercomponent event. A study has been recently initiated to eliminate this restriction. Its removal will significantly enhance the capabilities of the code. It is recommended that after the removal of the aforementioned restriction the following features be incorporated into the code.

i) Generation of all simple minimal cut-sets from the fault tree's modular cut-sets of up to an order specified by the user. Appendix B summarizes the efforts undertaken in this direction thus far. The comparison with the specially designed fault tree analysis code, FTAP (UCB), is indeed very encouraging.

ii) Application of these simple cut-sets to generate the system equation (QS) and incorporation of this equation into the PL-MODT and PL-MODMC codes to save computation time, especially for PL-MODMC.

iii) Common cause analysis should be effectively performed by the modular decomposition approach. For instance, by generating different modular tree representations associated with postulated common cause failure modes being considered, one would be able to access the contribution of the common cause failure to the top event.

iv) Derivation and implementation of a unique and meaningful importance measure for periodically tested components. This would avoid the abrupt change in the unavailability in the transition between the different periods experienced by using the V-F importance measure.

v) Extension and application of the Monte-Carlo simulation to time-dependent problems, where repair rates and test intervals are considered to be random variables in addition to the failure rates. Here, the modular decomposition approach offers unique savings in computation time because it operates on numbers of

modular cut-sets which are by orders of magnitude smaller than the number of minimal cut-sets commonly applied. Thus, realistic uncertainty propagation for time-dependent problems seems to be in reach in the near future.

REFERENCES

[1] J. Olmos and L. Wolf, "A Modular Approach to Fault Tree and Reliability Analysis", Dept. of Nucl. Engng., MIT, MITNE-209, August 1977.

[2] M. Modarres and L. Wolf, "PL-MODT: A Modular Fault Tree Analysis and Transient Evaluation Code", Trans. Am. Nucl. Soc., 28, 510 (1978).

[3] M. Modarres and L. Wolf, "Automatic Reduction of Complex Fault Trees by PL-MODT". To be presented at the ANS Winter Meeting, Washington, 1978.

APPENDIX A

USERS MANUAL FOR THE CODES PL-MODT AND PL-MODMC

A.1 Introduction

This manual describes the input for the code PL-MOD [1] for the modular fault tree analysis and steady state reliability analysis as well as for the extended version PL-MODT [2] which accounts for time-dependent processes and automatic fault tree reduction now without using the concept of minimal cut-sets.

Both versions use the language PL/1 and the most recent IBM PL/1 compiler with optimization on the 370/168. For these reasons, the effectiveness and operation of these codes seem to be highly system-dependent.

It should be pointed out that PL-MODT is merely an extension of the PL-MOD code, and thus relies upon the same modularization procedure. The user has the option to either select a steady state or time-dependent calculation once the modules have been determined by the code. Therefore, this manual is equally applicable for the original version and for the most recent one. In Section A.2 input to the code PL-MODT is presented, and in Section A.3 input to the code PL-MODMC is described.

In what follows, each card group is identified by a special name in order to more easily comprehend the meaning of this group. Furthermore, the variable names and their meanings are given. Special notes will provide extra information where needed to support the user in setting up his own problem.

The input to the code is FORMAT free. The only requirement is that the data, when punched on the same card, be separated by at least one blank space or a comma.

To make the manual more clear at the end (Section A.4), some sample problems are presented, showing different inputs for PL-MODT and PL-MODMC codes.

A.2 Description of the Code PL-MODT

1. Card groups I through VIII describe fault tree logic following these card groups. Any one of the following card group sets described in Parts 3 through 5 plus the card group in Part 2 could be used.
2. Card groups IX and X are control cards for the type of time-dependent analysis to be used.
3. Card groups X through XIII are for the analysis of Class 1 components (time-independent PL-MOD case).
4. Card groups XIV through XVIII are for the analysis of Class 2 components.
5. Card groups XIX through XXIII are for the analysis of Class 3 components.
6. Card groups XXIV through XXVIII are for the analysis of Class 4 components, or a combination of the three time-dependent components.
7. Card group XXIX is the reduction option card.

INPUT DESCRIPTION

Title Card

CARD GROUP I: TITLE CARD

No. of Variable	Variable	Entry
1	TITLE	A set of characters enclosed by a single quote marks

NOTE: Number of characters must be equal to or less than 71.

Calculated Options

CARD GROUP II: RELIABILITY PARAMETER OPTION

No. of Variable	Variable	Entry
1	DEL	Number of items to be computed if DEL = 1 only reliabilities or unavailabilities are calculated. IF DEL = 2 reliabilities and importances are calculated.

NOTE: When periodically tested or repairable (revealed fault) components are considered, it is recommended to set DEL=1 because the importance measure built into the code seemingly loses its meaning under these conditions.

When automatic reduction of the tree is desired, the value of DEL must be equal to 2 (i.e., DEL=2).

Fault Tree Characteristics

CARD GROUP III: GATES

No. of Variable	Variable	Entry
1	GUM	Total number of fault tree gates

NOTE: GUM includes all AND, OR, and k-out-of-N gates but excludes replicated gates (modules). The replicated gates in the original tree will be considered as a replicated component or module.

CARD GROUP IV: REPLICATED MODULES

No. of Variable	Variable	Entry
1	RMOD	Total number of replicated modules = 0 for no replications

NOTE: RMOD does not include replicated components. However, replicated components may be represented in a replicated module, i.e., component is replicated within this module.

CARD GROUP V: TREE STRUCTURE

No. of Variable	Variable	Entry
1	I	Gate number
2	AGIN(I)	Total number of gate inputs to gate number I
3	ALIL(I)	Total number of leafs which are input to gate number I
4	ALIR(I)	Total number of replicated leafs which are input to gate number I = 0 no replicated leaf

NOTE: I = 1, 2, ... , GUM

I includes all gates and associated modules, i.e., replicated or non-replicated modules in the tree.

CARD GROUP VI: REPLICATED MODULES

No. of Variable	Variable	Entry
1	TRIM(I)	Name of replicated leaf associated with a module
2	TRIN(I)	Number of replicated gate

NOTE: I = 1, 2, ..., RMOD

If RMOD = 0 on card group IV, then card group VI is to be skipped. If replicated gates exist then TRIM(I) = A9BCD,

where

A : Total number of occurrences of the specific replicated module

Nine →9 : Replication of gate or module

BCD : Number of replicated components associated with this module in the fault tree.

CARD GROUP VII: REPLICATED LEAVES

No. of Variable	Variable	Entry
1	NOR	Total number of replicated leaf inputs = 0 no replicated leaf inputs

CARD GROUP VIII: GATE STRUCTURE

No. of Variable	Variable	Entry
1	NAME	Gate Number
2	VALUE	= 1 for AND gate = 2 for OR gate = KON for K-out-of N gate
3	GIN	Total number of gate inputs to this gate
4	PIT(I) I = 1,2,...,GIN	I-th gate input For GIN = 0, then PIT(I) = 0
5	LIL	Total number of free leaf inputs
6	TIL(I) I = 1,2,...,LIL	I-th free leaf input For LIL = 0, then TIL(I) = 0
7	LIR	Total number of replicated leaf inputs
8	TIR(I) I = 1,2,...,LIR	I-th replicated leaf input For LIR = 0, then TIR(I) = 0

NOTE: LIT(I) = $\overline{\text{AOBCD}}$

where A: total number of occurrences of this component

$\overline{\text{BCD}}$: number of replicated component

For dual replicated components:

LIR(I) = $\overline{\text{A1BCD}}$ when ON

= $\overline{\text{A2BCD}}$ when OFF

Numerical Evaluations

CARD GROUP IX: EVALUATION OPTION

No. of Variable	Variable	Entry
1	FOX	= 0 numerical evaluation is not desired = 1 numerical evaluation is desired

NOTE: If FOX = 0, then all of the following card groups can be deleted.
If the former version, PL-MOD is to be run, only card groups XI, XII and XIII should be used after this card group.
If PL-MODT is to be run, card group X must be included and the procedure thereafter.

CARD GROUP X: CALCULATION OPTION

No. of Variable	Variable	Entry
1	ESF	= 1 for steady state calculation = 2 for time-dependent calculation of non-repairable components = 3 for time-dependent calculation of repairable components = 4 for time-dependent calculation of periodically tested components, or the combination of periodically tested components and time-dependent non-repairable or repairable nontested components.

NOTE: For ESF=1, only card groups XI, XII, XIII and XXIX are needed in what follows.

For ESF=2, only card groups XIV, XV, XVI, XVII, XVIII and XXIX are needed.

For ESF=3, only card groups XIX, XX, XXI, XXII, XXIII, and XXIX are needed.

(cont.)

CARD GROUP X: CALCULATION OPTION (cont.)

For ESF = 4, only card groups XXIV, XXV, XXVI, XXVII, XXVIII and XXIX are needed.
If none of these options is desired, any number other than 1, 2, 3, 4 suffices, and all of the following card groups should be deleted.

CARD GROUP XI: LEAF INPUT

No. of Variable	Variable	Entry
1	FUN	Total number of free leaf inputs
2	DUN	Total number of replicated leaf inputs = 0, no replicated components or modules.

CARD GROUP XII: COMPONENT RELIABILITY

No. of Variable	Variable	Entry
1	I	Number of free component
	I = 1, 2, ... FUN	
2	STATE(1,I)	Probability of occurrence of the I-th free input

CARD GROUP XIII: REPLICATED COMPONENT RELIABILITY

No. of Variable	Variable	Entry
1	I	Number of replicated component
2	STATD(1,I)	Probability associated with the I-th replicated component = 0 if I-th component is associated with a replicated module

CARD GROUP XIV: TIME STEP CHANGES

No. of Variable	Variable	Entry
1	MOH	Number of regions where time step size changes = 1 for no changes in time step size

NOTE: This card group must be supplied for ESF = 2 (see card group X).

CARD GROUP XV: TIME STEP INPUT

No. of Variable	Variable	Entry
1	DEL9T(1,I)	Time step associated with the I-th region
2	AUN(1,I) I = 1,2,...,MOH	Time interval for which time step size is applied

NOTE: This card group must be supplied for ESF = 2 (see card group X).

CARD GROUP XVI: LEAF INPUT

No. of Variable	Variable	Entry
1	FUN	Total number of free leaf inputs
2	DUN	Total number of replicated leaf inputs = 0 no replicated components or modules

NOTE: This card group must be supplied for ESF = 2 (see card group X).

CARD GROUP XVII: FREE COMPONENT FAILURE RATE

No. of Variable	Variable	Entry
1	I	Number of free component
2	STATT(1,I)	Failure rate $\lambda(hr^{-1})$ associated with the I-th free component

NOTE: This card group must be supplied for ESF = 2 (see card group X).

CARD GROUP XVIII: REPLICATED COMPONENT FAILURE RATE

No. of Variable	Variable	Entry
1	I	Number of replicated component
2	STATS(1,I)	Failure rate $\lambda(hr^{-1})$ associated with the I-th replicated component = 0 if I is a replicated component associated with a replicated module.

NOTE: This card group must be supplied for ESF = 2 (see card group X).

CARD GROUP XIX: TIME STEP CHANGES

No. of Variable	Variable	Entry
1	MOH	Number of regions where time step size changes. = 1 for no changes in time step size.

NOTE: This card group must be supplied for ESF = 3 (see card group X).

CARD GROUP XX: TIME STEP INPUT

No. of Variable	Variable	Entry
1	DEL9T(1,I)	Time step associated with the Ith region
2	AUN(1,I) I = 1, 2,...,MOH	Time step interval for which time step size is applied

NOTE: This card group must be supplied for ESF=3 (see card group X).

CARD GROUP XXI: LEAF INPUT

No. of Variable	Variable	Entry
1	FUN	Total number of free leaf inputs
2	DUN	Total number of replicated leaf inputs = 0 no replicated components or modules

NOTE: This card group must be supplied for ESF = 3 (see card group X).

CARD GROUP XXII: FREE COMPONENT FAILURE AND REPAIR RATES

No. of Variable	Variable	Entry
1	I	Number of free components
2	STATT(1,I)	Failure rate $\lambda(hr^{-1})$ associated with the Ith free component
3	STATTE(1,I)	Repair rate $\mu(hr^{-1})$ associated with the Ith free component

NOTE: This card group must be supplied for ESF = 3 (see card group X).

CARD GROUP XXIII: REPLICATED COMPONENT FAILURE AND REPAIR RATES

No. of Variable	Variable	Entry
1	I	Number of replicated component
2	STATS(1,I)	Failure rate $\lambda(hr^{-1})$ associated with the Ith replicated component =0 if I is a replicated component associated with a module.
3	STATED(1,I)	Repair rate $\mu(hr^{-1})$ associated with the Ith replicated component

NOTE: If STATS(1,I)=0 (i. e., the component I is associated with a replicated module), then STATED(1,I)=1. This card group must be supplied for ESF = 3 (see card group X).

CARD GROUP XXIV: TIME STEP CHANGES

No. of Variable	Variable	Entry
1	MOH	Number of regions where time step size changes = 1 for no changes in time step size

NOTE: This card group must be supplied for ESF = 4 (see card group X).

CARD GROUP TIME STEP INPUT

No. of Variable	Variable	Entry
1	DEL9T(1, I)	Time step associated with the I-th region
2	AUN (1,I) I = 1,2,...MOH	Time interval for which time step size is applied

NOTE: This card group must be supplied for ESF = 4 (see card group X).

CARD GROUP XXVI LEAF INPUT

No. of Variable	Variable	Entry
1	FUN	Total number of free leaf inputs
2	DUN	Total number of replicated leaf inputs = 0 no replicated components or modules

NOTE: This card group must be supplied for ESF = 4 (see card group X).

CARD GROUP XXVII: FREE COMPONENT TEST DATA

No. of Variables	Variables	Entry
1	I	Number of free component
2	STATT(1,I)	Failure rate $\lambda(hr^{-1})$ associated with I-th free component
3	ETTA(1,I)	Time between inspections (hrs).
4	TTETA(1,I)	Inspection time (hrs).
5	TAVV(1,I)	Repair time (hrs).
6	FIRTM(1,I)	Time of first inspection (hrs).
7	QUZR(1,I)	Override probability

NOTE: If the component is not inspected, then ETTA(1,I) = 0, TTETA(1,I) = 0, TAVV(1,I) = 0, FIRT(1,I) = 0, and QUZR(1,I) = 0. In this case the component will be considered as being a nonrepairable one. If the component is repairable but not tested (revealed fault), then ETTA(1,I) =0, TTETA(1,I)=repair rate (hr^{-1}), TAVV(1,I)=0, FIRT(1,I)=0 and QUZR (1,I)=0.

CARD GROUP XXVIII: REPLICATED COMPONENT TEST DATA

No. of Variable	Variable	Entry
1	I	Number of replicated component
2	STATS(1,I)	Failure rate $\lambda(hr^{-1})$ associated with this component
3	ETTAD(1,I)	Time between inspections (hrs)
4	TTETAD	Inspection time (hrs).
5	TAVV(1,I)	Repair time (hrs).
6	FIRTMD(1,I)	Time of first inspection (hrs).
7	QUZR(1,I)	Override probability

NOTE: If the component is not inspected then ETTAD(1,I) = 0, TTETAD(1,I) = 0, TAVVD(1,I) = 0, FIRTMD(1,I) = 0, and QUZRD(1,I) = 0.

If the component is associated with a replicated module then STATS (1,I) = 0, ETTAD(1,I) = 0, TTETAD(1,I) = 0,TAVVD(1,I) = 0, FIRTMD(1,I) = 0, and QUZRD(1,I) = 0. In this case the component is considered as being a nonrepairable one.

If the component is repairable but not tested (revealed fault), then ETTAD(1,I)=0, TTETAD(1,I) =repair rate (hr^{-1}), TTETAD(1,I)=0, TAVVD(1,I) =0,FIRTMD(1,I)=0, and QUZRD(1,I)=o.

CARD GROUP XXIX: REDUCTION OPTION CARD

No. of Variable	Variable	Entry
1	CUT-OFF	The cut-off value. Any component having an importance less than this value will be cancelled from the fault tree.

NOTE: If DEL=1, then this card group is not needed since no importance calculation is performed.
If DEL=2 and CUT-OFF=0, then no reduction process will be performed. Only the importances of modules and components in the fault tree will be calculated.
If DEL=2 and CUT-OFF>0, then the reduction procedure will be followed.

A.3 Description of the Code PL-MODMC

1. Card groups I through VIII are the same for the codes PL-MODT and PL-MODMC. These card groups are described in Section A.2 of this manual.
2. Card groups VIII through XIII are input data for a Monte-Carlo simulation.

CARD GROUP IX: CONFIDENCE LEVEL DATA

No. of Variable	Variable	Entry
1	ROM	Total number of different confidence levels that the top event probability should be evaluated for.
2	I	Number of confidence level.
3	CONPNT(I) I=1,2,...,ROM	Confidence level (e.g., CONPNT(I)=40 means the I-th confidence level is 40%.

NOTE: In this code, the accuracy is equal to the lowest confidence level (CONPNT(1)), i.e., if CONPNT (1)=0.5, then accuracy=0.5.

CARD GROUP X: RANDOM GENERATOR INITIATING NUMBER

No. of Variable	Variable	Entry
1	IPP	Any odd number to start generating random numbers.

CARD GROUP XI: SIMULATION CONTROL CARD

No. of Variable	Variable	Entry
1	FUN	Total number of free leaf inputs
2	DUN	Total number of replicated leaf inputs=0 for no replicated component or module.
3	NRAND	Total number of trials to be used in the simulation
4	NTERM	Number of terms to be used in the Central Limit Theorem approximation

CARD GROUP XII: FREE COMPONENT FAILURE AND ERROR SPREAD DATA

No. of Variable	Variable	Entry
1	I	Number of free component.
2	MEDIAN (I) I=1,2,...,FUN	Failure rate $\lambda(hr^{-1})$ associated with the I-th free component
3	FEN2(I) I=1,2,...,FUN	Error factor associated with the I-th free component

NOTE: If MEDIAN(I)=0, then FEN2(I) must be equal to zero, i.e., FEN2(I)=0.

CARD GROUP XIII: REPLICATED COMPONENT FAILURE RATE AND ERROR SPREAD DATA

No. of Variable	Variable	Entry
1	I	Number of replicated component
2	MEDIAN(J) J=FUN+1,...,VEN	Failure rate $\lambda(hr^{-1})$ associated with the I-th replicated component
3	FUN2(J) J=FUN+1,...,VEN	Error factor associated with the I-th replicated component

NOTE:
1. J=I+FUN
2. If MEDIAN(I)=0, then FEN2(I) must be equal to zero, i.e., FEN2(I)=0
3. VEN=FUN+DUN

A.4 Sample Fault Tree

For a fault tree given in Figure A.4.1, the input data are provided as follows:

The input data are given for this fault tree for the four classes of components and also for the Monte-Carlo simulation of the tree.

Table A.1 presents input data for the logic of the tree. Therefore, this part of the data is the same for both PL-MODT and PL-MODMC.

Table A.2 presents the rest of the input data following data set given in Table A.1, if the tree consists of only Class 1 components (i.e., PL-MOD steady state case).

Table A.3 presents the rest of the input data following data set given in Table A.1, if the tree consists of only Class 2 components.

Table A.4 presents the rest of the input data following data set given in Table A.1, if the tree consists of only Class 3 components.

Table A.5 presents the rest of input data following data set in Table A.1, if the tree consists of Class 4 components, or combination of different time-dependent classes of components.

Table A.6 presents the rest of input data following data set given in Table A.1, if the tree is to be simulated by the Monte-Carlo code.

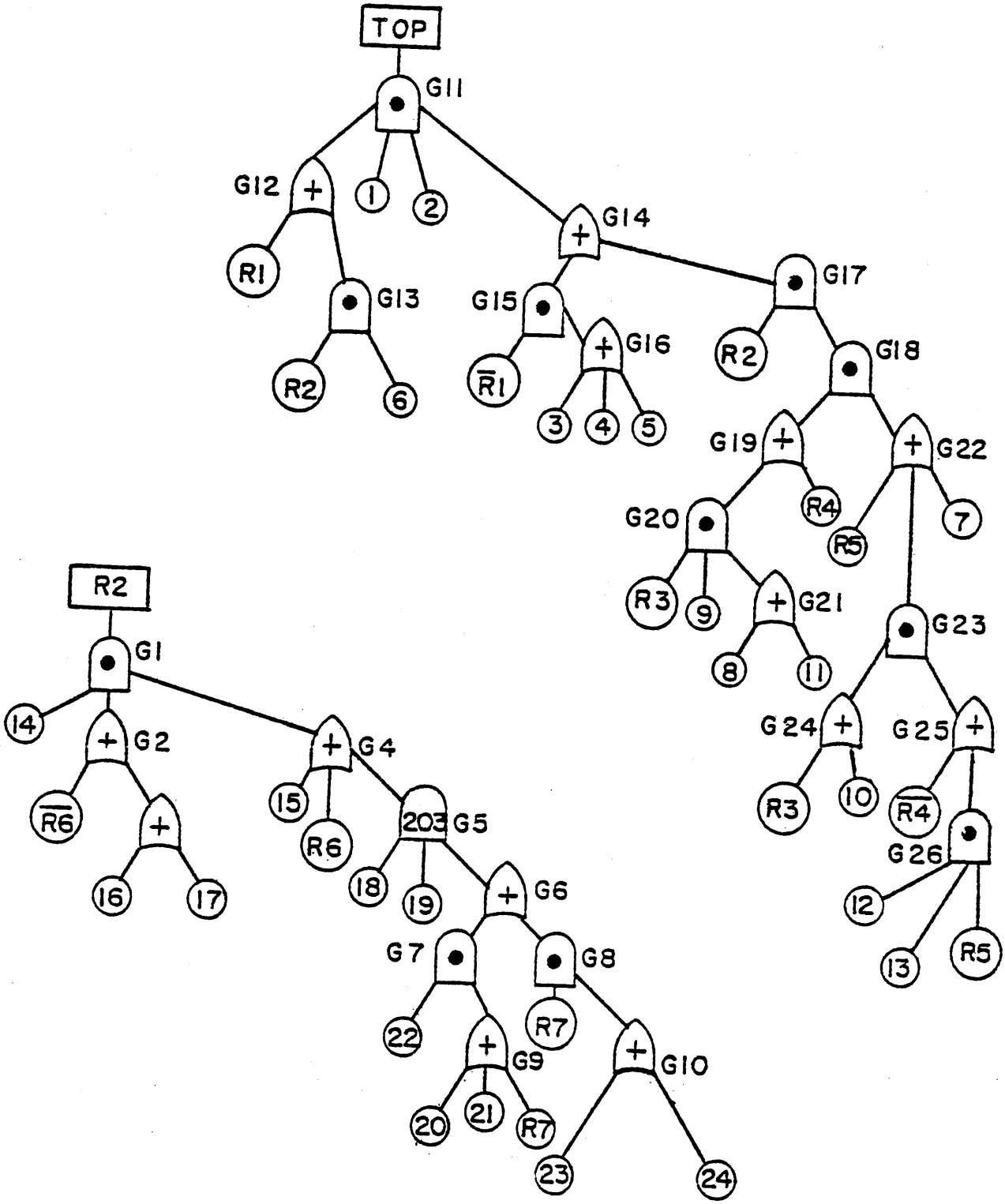


FIGURE A.1 SAMPLE FAULT TREE

TABLE A.1

GENERAL DATA SET REPRESENTING
THE LOGIC OF THE TREE

```
'SAMPLE PROB_EM'
2  26  1
1  2  1  0
2  1  0  1
3  0  2  0
4  1  1  1
5  1  2  0
6  2  0  0
7  1  1  0
8  1  0  1
9  0  2  1
10  0  2  0
11  2  2  0
12  1  0  1
13  0  1  1
14  2  0  0
15  1  0  1
16  0  3  0
17  1  0  1
18  2  0  0
19  1  0  1
20  1  1  1
21  0  2  0
22  1  1  1
23  2  0  0
24  0  1  1
25  1  0  1
26  0  2  1
29002  1
7
1  1  2  2  4  1  14  0  0
2  2  1  3  0  0  1  22006
3  2  0  0  2  15  17  0  0
4  2  1  5  1  15  1  21006
5  203  1  6  2  18  19  0  0
6  2  2  7  8  0  0  0  0
7  1  1  9  1  22  0  0
8  1  1  10  0  0  1  20007
9  2  0  0  2  20  21  1  20007
10  2  0  0  2  23  24  0  0
11 1  2  12  14  2  1  2  0  0
12  2  1  13  0  0  1  21001
13  1  0  0  1  5  1  29002
14  2  2  15  17  0  0  0  0
15  1  1  16  0  0  1  22001
16  2  0  0  3  3  4  5  0  0
17  1  1  18  0  0  1  29002
18  1  2  19  22  0  0  0  0
19  2  1  20  0  0  1  21004
20  1  1  21  1  9  1  20003
21  2  0  0  2  8  11  0  0
22  2  1  23  1  7  1  20005
23  1  2  24  25  0  0  0  0
24  2  0  0  1  10  1  20003
25  2  1  26  0  0  1  22004
26  1  0  0  2  12  13  1  20005
1
```

TABLE A.2

DATA SET FOLLOWING TABLE A.1 FOR ONLY CLASS 1 COMPONENTS (PL-MOD CASE)

```
1
24   7
1   1.0E-01
2   1.0E-01
3   1.0E-02
4   1.0E-02
5   1.0E-02
6   1.0E-01
7   1.0E-03
8   0.5E-03
9   0.5E-03
10   2.0E-05
11   2.0E-04
12   2.0E-04
13   6.1E-04
14   6.1E-04
15   8.1E-04
16   .6E-02
17   .6E-02
18   3.0E-01
19   3.0E-01
20   2.5E-04
21   2.5E-04
22   2.5E-04
23   2.5E-04
24   1.0E-02
1   1.0E-01
2   0
3   1.0E-02
4   1.5E-03
5   1.5E-03
6   1.2E-01
7   1.2E-01
      1.0E-03
```

TABLE A.3

DATA SET FOLLOWING TABLE A.1 FOR ONLY CLASS 2 COMPONENTS

```
2
3
18   180
24   240
36   720
24   7
1   1.0E-01
2   1.0E-01
3   1.0E-02
4   1.0E-02
5   1.0E-02
6   1.0E-01
7   1.0E-03
8   0.5E-03
9   0.5E-03
10   2.0E-05
11   2.0E-04
12   2.0E-04
13   6.1E-04
14   6.1E-04
15   8.1E-04
16   .6E-02
17   .6E-02
18   3.0E-01
19   3.0E-01
20   2.5E-04
21   2.5E-04
22   2.5E-04
23   2.5E-04
24   1.0E-02
1   1.0E-01
2   0
3   1.0E-02
4   1.5E-03
5   1.5E-03
6   1.2E-01
7   1.2E-01
      1.0E-03
```

TABLE A.4

DATA SET FOLLOWING TABLE A.1 FOR ONLY CLASS 3 COMPONENTS

```
3
2
2   24
24   720
24   7
1   1.0E-01      1.0E-01
2   1.0E-01      1.0E-01
3   1.0E-02      1.0E-01
4   1.0E-02      1.0E-01
5   1.0E-02      1.0E-01
6   1.0E-01      1.2E-01
7   1.0E-03      1.2E-01
8   0.5E-03      1.2E-01
9   0.5E-03      1.2E-01
10   2.0E-05     1.0E-02
11   2.0E-04     1.0E-02
12   2.0E-04     1.0E-02
13   6.1E-04     1.0E-02
14   6.1E-04     1.0E-02
15   8.1E-04     1.0E-01
16   .6E-02      1.0E-01
17   .6E-02      1.0E-01
18   3.0E-01     5.0E-01
19   3.0E-01     1.0E-01
20   2.5E-04     1.0E-01
21   2.5E-04     1.0E-01
22   2.5E-04     1.0E-02
23   2.5E-04     1.0E-02
24   1.0E-02     1.0E-02
1   1.0E-01       1.0E-1
2   0             0
3   1.0E-02      1.0E-01
4   1.5E-03      1.0E-01
5   1.5E-03      1.0E-01
6   1.2E-01      1.0E-01
7   1.2E-01      1.0E-01
      1.0E-03
```

TABLE A.5

DATA SET FOLLOWING TABLE A.1 FOR ONLY CLASS 4 COMPONENTS OR COMBINATION OF ALL TIME-DEPENDENT CLASSES OF COMPONENTS

```
4
5
12  240
1  5
24  360
2  8
36  1440
24  7
1  1.0E-01   360  2  15  360  1
2  1.0E-01   400  3  18  700  1
3  1.0E-02   720  1.5  19  1440  1
4  1.0E-02   150  1  10  150  1
5  1.0E-02   1200  5  25  1200  .5
6  1.0E-01   600  1  20  600  1
7  1.0E-03   720  1.5  19  800  0.5
8  0.5E-03   120  1  12  120  1
9  0.5E-03    450  3  12.5  600  1
10  2.0E-05  720  1  22  720  1
11  2.0E-04  300 2.5  20  300  0.1
12  2.0E-04  0  1.0E-01  0  0  0
13  6.1E-04    0  0  0  0  0
14  6.1E-04     0  1.5E-01  0  0  0
15  8.1E-04  720  1  12  720  1
16  .6E-02     400  2  15  400  1
17  .6E-02      0  0  0  0  0
18  3.0E-01   120  1  12  120  1
19  3.0E-01     0  0  0  0  0
20  2.5E-04  840  2.6  20  1440  0.5
21  2.5E-04    140  1.5  19  280  1
22  2.5E-04    720  1.5  12  840  1
```

TABLE A.5 (CONT.)

23	2.5E-04	0	1.0E-02	0	0	0
24	1.0E-02	720	1.5	19	1440	1
1	1.0E-01	0	0	0	0	0
2	0	0	0	0	0	0
3	1.0E-02	450	2	16	720	1
4	1.5E-03	0	0	0	0	0
5	1.5E-03	1440	3	28	1440	1
6	1.2E-01	720	1.5	19	720	1
7	1.2E-01	440	1.5	20	440	1
	1.0E-03					

```
21
1 0.5 2 1 3 2.5 4 5 5  10 6 15 7 20 8 25 9 30 10 40 11 50 12 60 13 70 14 75
15 80 16 85 17 90  18 95 19 97.5 20 99 21 99.5
1331
24  7  5000  12
1  1.0E-01   10
2  1.0E-01   5
3  1.0E-02   3
4  1.0E-02   10
5  1.0E-02   7
6  1.0E-01   5
7  1.0E-03   3
8  0.5E-03   8
9  0.5E-03   5
10  2.0E-05  10
11  2.0E-04  10
12  2.0E-04  8
13  6.1E-04  6
14  6.1E-04  3
15  8.1E-04  7
16  .6E-02   10
17  .6E-02   5
18  3.0E-01  10
19  3.0E-01  6
20  2.5E-04  12
21  2.5E-04  8
22  2.5E-04  6
23  2.5E-04  8
24  1.0E-02  6
1  1.0E-01   2
2  0        0
3  1.0E-02   3
4  1.5E-03   5
5  1.5E-03   3
6  1.2E-01  2
7  1.2E-01   5
```

TABLE A.6

DATA SET FOLLOWING TABLE A.1 FOR A MONTE-CARLO SIMULATION

APPENDIX B

MIN-CUT-SET GENERATION OF COMPLEX FAULT TREES BY USING PL-MODT

B.1 Introduction

It was found that in large fault trees, some replicated components exist within a replicated module that are repeated elsewhere in the fault tree. As was discussed in Chapter 5, the code PL-MODT is actually not able to handle automatically this type of fault tree. Therefore, a method was developed to qualitatively solve this problem with PL-MODT and hence to generate minimal simple cut-sets.[1]

In essence, this method consists of a treatment of higher-order modules of the fault tree as supercomponents by neglecting the presence of replications outside the domain of the replicated module. The replacement of these replicated modules by a replicated component will provide a basis for starting the analysis. For example, the SNM fault tree [2], which is used as an example, is solved in the next section. It was found that there are four replicated modules that can be replaced by an imaginary supercomponent. After this replacement, the original fault tree becomes substantially smaller with only 26 replicated and non-replicated events being present. The modular cut-sets

---

[1] Simple cut-sets consist of only simple components and not modules.

can simply be found by PL-MODT for the original tree as well as the superevents (i.e., the replicated modules). Due to the replacement of replicated modules, it was discovered that some replicated components will appear only once within the original fault tree or replicated modules (i.e., replication presents elsewhere in the tree, but repeated only one time within the replicated module's domain). This special case was treated by simply connecting a replicated pair to the top event of the module via an arbitrary OR gate. Figure B.1 demonstrates this procedure.

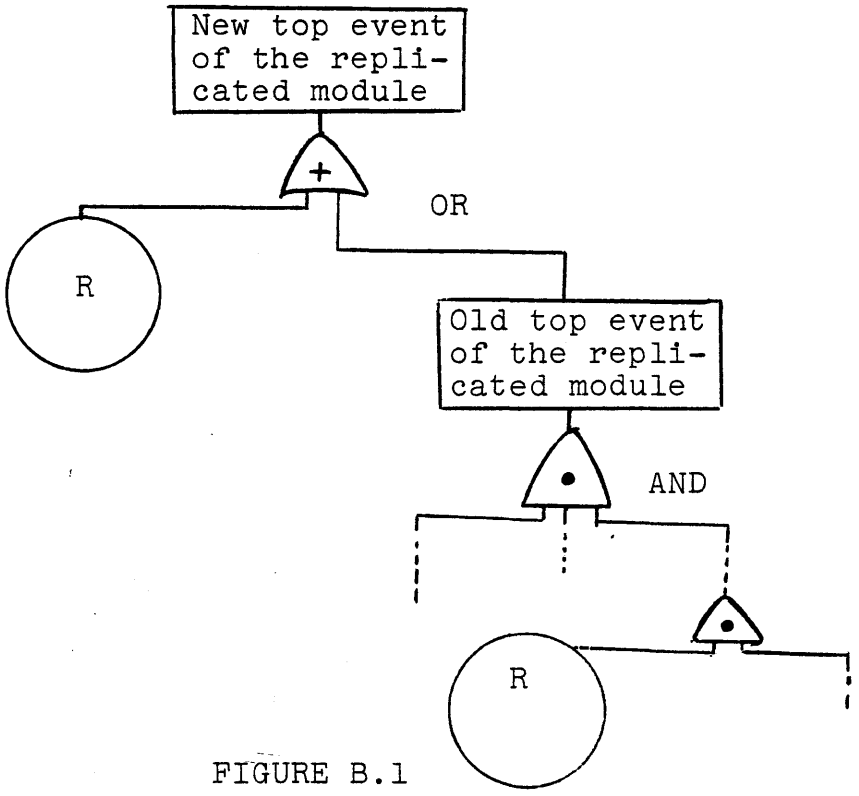


FIGURE B.1
ADDITION OF A REPLICATED PAIR TO A FAULT TREE

This enables the PL-MODT code to find the modular cut-sets of the replicated module with an additional order one cut-set consisting of only the replicated component R that is eliminated from the list of cut-sets. The rest of the cut-sets are the exact modular cut-sets of the replicated module.

These modular cut-sets are then used to generate simple cut-sets. The method is very simple. Modular cut-sets are only composed of some replicated component inputs and some simple OR and AND modules. An OR simple module of 10 component inputs consists of 10 order one cut-sets, and an AND simple module of 10 component inputs consists of only 1 order ten cut-set. By removing each module in the modular cut-sets by its associated simple cut-sets, one is able to find all of the simple cut-sets in the replicated module. This procedure cannot presently be performed by the code, and thus some of the lower-order cut-sets of the replicated modules were found by hand off-line.

The modular cut-sets of the original tree are used to determine the simple minimal cut-sets. These modular cut-sets consist of replicated modules (i.e., the superevents whose simple cut-sets are found before), replicated components and simple modules. In a similar way, by assigning corresponding cut-sets from the replicated modules and from the simple modules, one is able to generate all of the cut-sets of the fault tree.

Replicated events which are repeated outside the domain of the replicated module are found in some of the cut-sets of the tree. These replicated components would appear more than one

time within some of the generated simple cut-sets and, therefore, the presence of only one of them is sufficient whereas the rest of them must be eliminated from the cut-sets.

Finally, from these simple cut-sets only minimal cut-sets should be selected by using the same procedure which is already applied in the codes PL-MOD and PL-MODT to find minimal modular cut-sets.

The method described above has been applied to determine the cut-sets of the SNM fault tree and some of the simple cut-sets are derived. These cut-sets are presented in Table B.12. To generate all of the cut-sets, the problem becomes trivial and the use of computer is hardly recommended. A subroutine is therefore necessary to be incorporated into the PL-MODT code for the generation of cut-sets. The cut-sets found in Table B.12 are the lowest-order cut-sets in the SNM fault tree.

B.2 Example

The code PL-MODT was used in order to find the modular minimal cut-sets for the Test Bed Design Fault Tree for SNM Diversion at Pump Washout Line [2]. This fault tree consists of 125 gates, 95 simple components and 18 replicated components. Also, 9 replicated gates (modules) were presented. (See Figure B.2, Page 205).

First, higher order modular cut-sets were found. These cut-sets contain some modules that contain many cut-sets of lower order. Second, minimal cut-sets for these higher-order modules were found. Therefore, combination of these cut-sets would provide probably most of the presented simple cut-sets.

Table B.1 identifies different components in this fault tree. In order to simplify the cut-sets, Table B.2 gives the identification of higher order modules that are presented in this fault tree; then Table B.3 lists all of the modular minimal cut-sets in terms of those higher modules in Table B.2.

In Table B.4 are listed minimal cut-sets presented in the module G_{15}.

TABLE B.4

MINIMAL CUT-SETS FOR THE MODULE 15

Cut-Set No.	1	2
1	7	
2	R_{11}	
3	9	
4	8	
5	6	R_{13}

In Table B.5 we have identified the components and modules presented in the higher-order module 50 (G_{50}). The cut-sets for this higher-order module are listed in Table B.6. Note that the

TABLE B.1

DESCRIPTION OF THE COMPONENTS IN THE TEST BED DESIGN FAULT TREE, IN PL-MODT AND FTAP

PL-MODT	FTAP	PL-MODT	FTAP	PL-MODT	FTAP	PL-MODT	FTAP
1	AN0DE1	31	NOMCCAS	61	TGTTKFIL	91	GBOFAIL
2	CBE1HIT	32	MDCCASIT	62	V7130	92	GBOSLOW
3	TPEQMM	33	MDCCASHI	63	PRMODE	93	GARFAIL
4	CBEEBHIT	34	MDCCASIR	64	MPU709	94	GARSLOW
5	CBEEAHIT	35	MDCCASIT	65	V7190	95	T7221545
6	SRCB	36	ADV200MT	66	T701GT45	R_1	SRMSOLN
7	ALARMLX	37	ADV1KGMT	67	V7220	R_2	SRDUEST
8	ALARMIR	38	SRMDW	68	LS722IR	R_3	T722LT15
9	ALARMIT	39	TDR2-G	69	LS722IT	R_4	APM-AL
10	GFAIL13	40	SRMDS	70	LS722CIR	R_5	CCASOK
11	GAMB01LX	41	CBE2RHIT	71	LS722CIT	R_6	AMP-AR
12	GAMB01HI	42	CBE2LHIT	72	V7010	R_7	AMP-BO
13	GAMB01IR	43	GAMB02HI	73	TLEVEL-1	R_8	TDR1-G
14	GAMB01IT	44	GAMB02LX	74	SENSLOWM	R_9	TPNESM
15	SRGAMB01	45	GAMB02IR	75	TK1FILL	R_{10}	TPNETM
16	CSNMRMAA	46	GAMB02IT	76	DPCELLIR	R_{11}	CDLI
17	GCCASF	47	SRGAMB02	77	DPCELLIT	R_{12}	SRLS701
18	GCCASLOW	48	RDCCASLX	78	TGTPROMD	R_{13}	TDR3-G
19	ALCCAS	49	RDCCASHI	79	T722GT45	R_{14}	MC1-OPO
20	AA-DHIT	50	RDCCASIT	80	LABFALSE	R_{15}	SRLS722
21	SRCDALRM	51	RDCCASIR	81	SENSLOWP	R_{16}	T701LT15
22	CDALRMLX	52	MC2-OPO	82	TLTLAB	R_{17}	ANODE14
23	CDALRMIR	53	SRRDCCAS	83	T7011545	R_{18}	ASUBHNO
24	CDALRMIT	54	APM-B1	84	TGTLAB		
25	ACCAS	55	AFILLCON	85	LS701IR		
26	WEIGHTOR	56	ANODE706	86	LS701IT		
27	WEIGHTIT	57	CONAT706	87	LS701CIR		
28	WEIGHTIR	58	V290	88	LS701CIT		
29	WEGT3	59	CV436IR	89	T722GT15		
30	SRWEIGHT	60	CV436IT	90	T701GT15		

Table B.2

HIGHER ORDER MODULE OF THE TEST BED DESIGN FAULT TREE

No.	Name of the module, replicated component, or simple component in PL-MODT	Corresponding name in the original fault tree	Corresponding name in the FTAP
1	G^M_{24}	$\overline{\text{MPU2}}$	---
2	G^M_{50}	$\overline{\text{MPU5}}$	---
3	G^M_{15}	INADEQUATE RESPONSE FROM MC SYSTEM WHEN CRASH BAR EE_B IS HIT	---
4	G^M_{81}	INADEQUATE GUARD RESPONSE IN AREA AR WHEN ADV/MAN CROSSES PM - AR	---

TABLE B.3
ORIGINAL TREE MODULAR MINIMAL-CUT-SETS

Cut Set No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	R_4	R_{10}	G_{24}	1										
2	R_4	R_7	G_{50}	R_9	2	G_{15}	1							
3	R_4	R_7	G_{50}	R_{10}	2	G_{15}	1							
4	R_{17}	R_6	R_7	G_{50}	R_9	10	16	2	13					
5	R_{17}	R_6	R_7	G_{50}	R_9	10	16	2	14					
6	R_{17}	R_6	R_7	G_{50}	R_9	10	16	2	12					
7	R_{17}	R_6	R_7	G_{50}	R_9	10	16	2	11					
8	R_{17}	R_6	R_7	G_{50}	R_9	R_{11}	10	16	2					
9	R_{17}	R_6	R_7	G_{50}	R_9	R_{13}	10	16	2	15				
10	R_{17}	R_6	R_7	G_{50}	R_{10}	2	G_{15}	5	G_{81}	4				
11	R_{17}	R_6	R_7	G_{50}	G_{15}	R_{11}	5	G_{81}	4	41	42			
12	R_{17}	R_6	R_7	G_{50}	R_{11}	R_{13}	G_{15}	5	G_{81}	4	41	42	47	
13	R_{17}	R_6	R_7	G_{50}	G_{15}	5	G_{81}	4	41	42	46	45	43	44

TABLE B.5

IDENTIFICATION OF DIFFERENT MODULES IN THE PRIME MODULE G_{50}

PL-MODT	THE ORIGINAL TREE
M_{58}	INADEQUATE MC RESPONSE WHEN VALVE 722-1 OPEN
M_{62}	INADEQUATE MC RESPONSE TO CHANGE IN PU MASS IN TANK 1, $MPUTK_1$ (-1)
M_{67}	INADEQUATE MC RESPONSE WHEN VALVE 701-1 OPEN

TABLE B.6
MINIMAL CUT-SETS FOR MODULE 50

Cut-Set No.									
1	M_{50}	R_{11}							
2	M_{50}	R_8	74	R_{15}	R_{12}				
3	M_{50}	R_8	75	R_{15}	R_{12}				
4	M_{50}	R_8	76	R_{15}	R_{12}				
5	M_{50}	R_8	R_{13}	R_{15}	R_{12}	R_1	78		
6	M_{50}	R_8	R_{18}	R_{13}	R_{15}	R_{12}	R_2	84	
7	M_{50}	R_8	R_{18}	R_{15}	R_{12}	80	81	82	
8	M_{50}	R_8	R_{18}	68	R_{12}	80	81	82	
9	M_{50}	R_8	R_{18}	69	R_{12}	80	81	82	
10	M_{50}	R_8	R_{18}	70	R_{12}	80	81	82	
11	M_{50}	R_8	R_{18}	85	R_{15}	80	81	82	
12	M_{50}	R_8	R_{18}	86	R_{15}	80	81	82	
13	M_{50}	R_8	R_{18}	87	R_{15}	80	81	82	
14	M_{50}	R_8	R_{18}	88	R_{15}	80	81	82	
15	M_{50}	R_8	R_{18}	71	R_{12}	80	81	82	
16	M_{50}	R_8	R_{13}	85	R_{15}	R_1	78		
17	M_{50}	R_8	R_{13}	86	R_{15}	R_1	78		
18	M_{50}	R_8	R_{13}	87	R_{15}	R_1	78		
19	M_{50}	R_8	R_{13}	88	R_{15}	R_1	78		
20	M_{50}	R_8	R_{13}	68	R_{12}	R_2	84		
21	M_{50}	R_8	R_{13}	69	R_{12}	R_2	84		
22	M_{50}	R_8	R_{13}	70	R_{12}	R_2	84		
23	M_{50}	R_8	R_{13}	71	R_{12}	84			
24	R_8	R_{18}	R_{13}	68	R_{12}	R_2	84	M_{50}	

(CONT.)

Table B.6
(cont.)
MINIMAL CUT-SETS FOR MODULE 50

Cut-Set No.		
25	R_8 R_{18} R_{13} 69 R_{12} R_2 84 M_{50}	
26	R_8 R_{18} R_{13} 70 R_{12} R_2 84 M_{50}	
27	R_8 R_{18} R_{13} 71 R_{12} R_2 84 M_{50}	
28	R_8 R_{18} R_{13} 85 R_{15} R_2 84 M_{50}	
29	R_8 R_{18} R_{13} 86 R_{15} R_2 84 M_{50}	
30	R_8 R_{18} R_{13} 87 R_{15} R_2 84 M_{50}	
31	R_8 R_{18} R_{13} 88 R_{15} R_2 84 M_{50}	
32	M_{58} M_{67} M_{62} M_{50}	←Contains 64 simple cut-sets
33	R_8 M_{67} M_{62} R_{15} M_{50}	←Contains 16 simple cut-sets
34	R_8 M_{58} M_{62} R_{12} M_{50}	←Contains 16 simple cut-sets
35	R_{13} M_{58} M_{67} R_1 78 M_{50}	←Contains 16 simple cut-sets
36	R_{18} R_{13} M_{58} M_{67} R_2 84 M_{50}	←Contains 16 simple cut-sets
37	R_{18} M_{58} M_{67} 80 81 82 M_{50}	←Contains 16 simple cut-sets
	TOTAL SIMPLE CUT-SETS 175 X 2 = 350	

following components are always required to fail in order to have the failure of module 50:($\overline{MPU5}$), 55, 56, 57, 54, 58, 61, 63, 62, 64, 65, 67, 72, 73, and (59 or 60). We call, for simplicity, all of these components module 50 (M50).

Cut-sets in Table B.6 are on the order of between 14 and 20. It should be noted that the total number of cut-sets for the higher-order module (G_{50}) is 350. For the modules 58, 62 and 67, the cut-sets are given in Table B.7. All of the cut-sets are on an order of 1 (e.g., failure of each component 68, 69, 60, or 71 would cause the failure of the module 58).

TABLE B.7

CUT-SETS FOR MODULES 58, 62, and 67

Module / Cut-Set Order	1	2	3	4
M_{58}	68	69	70	71
M_{62}	74	75	76	77
M_{67}	85	86	87	88

For higher-order module G_{81} the cut-sets were found. Table B.8 gives an identification of components in the cut-sets and their corresponding name in the fault tree. Table B.9 lists all of the cut-sets for this prime module.

TABLE B.8

IDENTIFICATION OF DIFFERENT MODULES IN THE PRIME MODULE 81

PL-MODT	THE ORIGINAL TREE
M_{11}	NO RESPONSE FROM MC SYSTEM
M_8	NO MC RESPONSE FROM MC SYSTEM WHEN VALVE 701 OPEN
M_{10}	NO MC RESPONSE FROM ESTIMATION WHEN VALVE 722 OPEN

TABLE B.9

CUT-SETS FOR THE MODULE 81

Cut-sets						
1	R_{15}	R_{12}	79	66	R_1	1
2	R_{15}	R_{12}	79	66	R_1	2
3	R_{15}	R_{12}	79	66	R_2	1
4	R_{15}	R_{12}	79	66	R_2	2
5	M_{11}	M_8	M_{10} ← Containing 112 cut-sets			
6	R_{11}					
7	R_{15}	M_{11}	M_{10} ← Containing 28 cut-sets			
8	R_{12}	M_{11}	M_8	79 ←Containing 28 cut-sets		
9	R_{15}	R_{12}	80	95	79	
10	R_{15}	R_{12}	81	95	79	
11	R_{15}	R_{12}	82	95	79	
12	R_{15}	R_{12}	74	95	79	
13	R_{15}	R_{12}	76	95	79	
14	R_{15}	R_{12}	77	95	79	
15	R_{15}	R_{12}	75	95	79	
	TOTAL SIMPLE CUT-SETS 180					

As can be seen from Table B.9, the total number of cut-sets for higher-order module 81 is 180 simple cut-sets. These cut-sets are of an order between 1 and 6.

Now, from Table B.3 all of the simple cut-sets for modular cut-sets 2 through 13 could be found by simply assigning the corresponding simple cut-sets of different modules from Tables B.4, B.6, and B.9. The total number of simple cut-sets in these 12 modular cut-sets is 1,265,600.

For the first modular cut-set in Table B.3, we have to find simple cut-sets for the module G_{24}. These cut-sets were found and Table B.10 describes different components of these cut-sets and their corresponding name in PL-MODT. Table B.11 lists all of the cut-sets presented in the module G_{24}, based on the names of the modules given in Table B.10. As could be seen from Tables B.10 and B.11, all of the cut-sets contain module G_{50} which by itself has 350 different cut-sets (see Table B.6). Some other higher-order modules are also contained in most of the cut-sets in Table B.11. Therefore, a very high rough estimate of the number of simple cut-sets in Table B.11 could be about 6 million. By assigning different simple cut-sets of module G_{24} into the first modular cut-set in Table B.3, we would be able to find the rest of the cut-sets for the fault tree.

It should be noticed, however, that not all of these simple cut-sets are minimal for the modular cut-set number 1. Therefore, it is not possible to estimate the exact number of minimal simple cut-sets in this fault tree. In the future development of the

PL-MODT code, the combination of these tables will be done by computer and nonminimal cut-sets will be eliminated automatically.

By the combination of Tables B.4, B.6, B.9 and B.11, we have found some of these simple cut-sets. These cut-sets are listed in Table B.12.

TABLE B.10

IDENTIFICATION OF COMPONENTS IN THE MODULE 24

Module Number as Listed in Table B.11	Type of Module	Components in this Module
1	AND	R_5
2	AND	R_{14}
3	AND	R_{13}
4	AND	R_{15}
5	AND	R_{12}
6	AND	R_{11}
7	AND	G_{78}, G_{50}, R_7, R_9
8	OR	48, 49, 50, 51
9	OR	31, 32, 33, 34, 35
10	--	--
11	--	--
12	AND	52, 53
13	AND	36, 38, 39
14	AND	37, 40
15	AND	19, 20
16	AND	25
17	OR	17, 18
18	--	--
19	OR	22, 23, 24
20	OR	26, 27, 28
21	OR	R_1, R_2
22	OR	68, 69, 70, 71
23	OR	85, 86, 87, 88
24	OR	74, 75, 76, 77
25	AND	21
26	AND	30, R_8, 29
27	AND	89
28	AND	90

(CONTINUED)

Table B.10

(cont.)

IDENTIFICATION OF COMPONENTS IN THE MODULE 24

Module Number as Listed in Table B.11	Type of Module	Components in this Module
29	AND	R_3
30	AND	R_{16}
31	AND	R_{18}
32	OR	80, 81, 82

TABLE B.11
(Next 14 Pages)

LISTS OF ALL THE MIN-CUT-SETS IN THE PRIME MODULE G_{24}

(In this table, different components of each cut set are numbered according to those modules listed in Table B.10.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	7	8	9	15	17	19	21								
2	1	7	8	9	16	17	20	21								
3	1	2	7	8	13	15	17	19	21							
4	1	2	7	8	13	16	17	20	21							
5	1	2	3	7	8	14	15	17	19	21						
6	1	2	3	7	8	14	16	17	20	21						
7	1	6	7	9	15	17	21									
8	1	6	7	9	16	17	20	21								
9	1	2	6	7	13	15	17	21								
10	1	2	6	7	13	16	17	20	21							
11	1	2	3	6	7	14	15	17	21							
12	1	2	3	6	7	14	10	17	20	21						
13	1	2	3	7	9	12	15	17	19	21						
14	1	2	3	7	9	12	16	17	20	21						
15	1	2	3	7	12	13	15	17	19	21						
16	1	2	3	7	12	13	16	17	20	21						
17	1	2	3	7	12	14	15	17	19	21						
18	1	2	3	7	12	14	16	17	20	21						
19	1	7	8	9	15	18	19	22	23	24						
20	1	7	8	9	16	18	20	22	23	24						
21	2	7	7	13	13	15	18	19	22	23	24					
22	1	2	7	8	13	16	18	20	22	23	24					
23	1	2	3	7	8	14	15	18	19	22	23	24				

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
47	1	2	3	7	8	9	10	16	17	21						
48	1	2	3	7	12	14	15	18	22	23	24	25				
49	1	2	7	8	9	16	17	21	26							
50	1	2	7	8	9	16	18	22	23	24	26					
51	1	2	7	8	13	16	17	21	26							
52	1	2	7	8	13	16	18	22	23	24	26					
53	1	2	3	7	8	14	16	17	21	26						
54	1	2	3	7	8	14	16	18	22	23	24	26				
55	1	2	6	7	9	16	17	21	26							
56	1	2	6	7	9	16	18	26								
57	1	2	6	7	13	16	17	21	26							
58	1	2	6	7	13	16	18	26								
59	1	2	3	6	7	14	16	17	21	26						
60	1	2	3	6	7	14	16	18	26							
61	1	2	3	7	9	12	16	17	21	26						
62	1	2	3	7	9	12	16	18	22	23	24	26				
63	1	2	3	7	12	13	16	17	21	26						
64	1	2	3	7	12	13	16	18	22	23	24	26				
65	1	2	3	7	12	14	16	17	21	26						
66	1	2	3	7	12	14	16	18	22	23	24	26				
67	1	4	7	8	9	15	17	19	27							
68	1	3	4	7	8	9	15	17	25	27						
69	1	4	7	8	9	14	15	18	27							

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
70	1	2	4	7	8	9	16	17	26	27						
71	1	2	4	7	8	13	15	17	19	27						
72	1	2	3	4	7	8	13	15	17	25	27					
73	1	2	4	7	8	13	16	17	20	27						
74	1	2	4	7	8	13	16	17	26	27						
75	1	2	3	4	7	8	14	15	17	19	27					
76	1	2	3	4	7	8	14	15	17	25	27					
77	1	2	3	4	7	8	14	16	17	20	27					
78	1	2	3	4	7	8	14	16	17	26	27					
79	1	4	6	7	9	15	17	27								
80	1	4	6	7	9	16	17	20	27							
81	1	2	4	6	7	9	16	17	26	27						
82	1	2	4	6	7	13	15	17	27							
83	1	2	4	6	7	13	16	17	20	27						
84	1	2	4	6	7	13	16	17	26	27						
85	1	2	3	4	6	7	14	15	17	27						
86	1	2	3	4	6	7	14	16	17	20	27					
87	1	2	3	4	6	7	14	16	17	26	27					
88	1	2	3	4	7	9	12	15	17	19	27					
89	1	2	3	4	7	9	12	15	17	25	27					
90	1	2	3	4	7	9	12	16	17	20	27					
91	1	2	3	4	7	9	12	16	17	25	27					

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
92	1	2	3	4	7	12	13	15	17	19	27					
93	1	2	3	4	7	12	13	15	25	27	17					
94	1	2	3	4	7	12	13	16	17	27						
95	1	2	3	4	7	12	13	16	17	26	27					
96	1	2	3	4	7	12	14	15	17	27	19					
97	1	2	3	4	7	12	14	15	17	25	27					
98	1	2	3	4	7	12	14	16	17	20	27					
99	1	2	3	4	7	12	14	16	17	26	27					
100	1	5	7	8	9	15	17	28	19							
101	1	3	5	7	8	9	15	17	25	28						
102	1	5	7	8	9	16	17	20	28							
103	1	2	5	7	8	9	16	17	26	28						
104	1	2	5	7	8	13	15	17	28							
105	1	2	3	5	7	8	13	15	17	25	28					
106	1	2	5	7	8	13	16	17	20	28						
107	1	2	5	7	8	13	16	17	26	28						
108	1	5	7	8	14	15	17	19	28							
109	1	2	3	5	7	8	14	15	17	25	28					
110	1	2	3	5	7	8	14	16	17	20	28					
111	1	2	3	5	7	8	14	16	17	26						
112	1	5	6	7	9	15	17	28								
113	1	5	6	7	9	16	17	20	28							

	1	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16
114	1	2	5	6	7	9	16	17	26	28						
115	1	2	5	6	7	13	15	17	28							
116	1	2	5	6	7	13	16	17	20	28						
117	1	2	5	6	7	13	16	17	26	28						
118	1	2	3	5	6	7	14	15	17	28						
119	1	2	3	5	6	7	14	16	17	20	28					
120	1	2	3	5	6	7	14	16	17	26	28					
121	1	2	3	5	7	9	12	15	17	19	28					
122	1	2	3	5	7	9	12	15	17	25	28					
123	1	2	3	5	7	9	12	16	17	20	28					
124	1	2	3	5	7	9	12	16	17	26	28					
125	1	2	3	5	7	12	13	15	17	19	28					
126	1	2	3	5	7	12	13	15	17	25	28					
127	1	2	3	5	7	12	13	16	17	20	28					
128	1	2	3	5	7	12	13	16	17	26	28					
129	1	2	3	5	7	12	14	15	17	19	28					
130	1	2	3	5	7	12	14	15	17	28	25					
131	1	2	3	5	7	12	14	16	17	20	28					
132	1	2	3	5	7	12	14	16	17	26	28					
133	1	4	7	8	9	15	18	19	23	24	29					
134	1	3	4	7	8	9	15	18	23	24	25	29				
135	1	4	7	8	9	16	18	20	23	24	29					

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
136	1	2	4	7	8	9	16	18	23	24	26	29				
137	1	2	4	7	8	13	15	18	19	23	24	29				
138	1	2	3	4	7	8	13	15	18	23	24	25	29			
139	1	2	4	7	8	13	16	18	20	23	24	29				
140	1	2	4	7	8	13	16	18	23	24	26	29				
141	1	2	3	4	7	8	14	15	18	19	23	24	29			
142	1	2	3	4	7	8	14	15	18	23	24	25	29			
143	1	2	3	4	7	8	14	16	18	20	23	24	29			
144	1	2	3	4	7	8	14	16	18	23	24	26	29			
145	1	2	3	4	7	9	12	15	18	19	23	24	29			
146	1	2	3	4	7	9	12	15	18	23	24	25	29			
147	1	2	3	4	7	9	12	16	18	20	23	24	29			
148	1	2	3	4	7	9	12	16	18	23	24	26	29			
149	1	2	3	4	7	12	13	15	18	19	23	24	29			
150	1	2	3	4	7	12	13	15	18	23	24	25	29			
151	1	2	3	4	7	12	13	16	18	20	23	24	29			
152	1	2	3	4	7	12	13	16	18	23	24	26	29			
153	1	2	3	4	7	12	14	15	18	19	23	24	29			
154	1	2	3	4	7	12	14	15	18	23	24	25	29			
155	1	2	3	4	7	12	14	16	18	23	24	20	29			
156	1	2	3	4	7	12	14	16	18	23	24	26	29			
157	1	5	7	8	9	15	18	14	22	24	30					

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
158	1	3	5	7	8	9	15	18	22	24	25	30				
159	1	3	4	5	7	8	9	15	18	24	25	29	30			
160	1	5	7	8	9	16	18	20	22	24	30					
161	1	4	5	7	8	9	16	18	20	24	29	30				
162	1	2	5	7	8	9	16	18	22	24	26	30				
163	1	2	4	5	7	8	9	16	18	24	26	29	30			
164	1	2	5	7	8	13	15	18	22	24	30	19				
165	1	2	4	5	7	8	13	15	18	19	24	29	30			
166	1	2	3	5	7	8	13	15	18	12	24	25	30			
167	1	2	3	4	5	7	8	13	15	18	24	25	29	30		
168	1	2	5	7	8	13	16	18	20	22	24	25	29	30		
169	1	2	4	5	7	8	13	16	18	20	24	29	30			
170	1	2	5	7	8	13	16	18	22	24	26	30				
171	1	2	4	5	7	8	13	16	18	24	26	29	30			
172	1	2	3	5	7	8	14	15	18	19	22	24	30			
173	1	2	3	4	5	7	8	14	15	18	19	24	29	30		
174	1	2	3	5	7	8	14	15	18	22	24	25	30			
175	1	2	3	4	5	7	8	14	15	18	24	25	29	30		
176	1	2	3	5	7	8	14	16	18	20	22	24	30			
177	1	2	3	4	5	7	8	14	16	18	20	24	29	30		
178	1	2	3	5	7	8	14	16	18	22	24	26	30			
179	1	2	3	4	5	7	8	14	16	18	24	26	29	30		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
180	1	2	3	5	7	9	12	15	18	19	22	24	30			
181	1	2	3	4	5	7	9	12	15	18	19	24	29	30		
182	1	2	3	5	7	9	12	15	18	22	24	25	30			
183	1	2	3	4	7	9	12	15	18	24	25	5	29	30		
184	1	2	3	5	7	9	12	16	18	20	22	24	30			
185	1	2	3	4	5	7	9	12	16	18	20	24	29	30		
186	1	2	3	4	5	7	9	12	16	18	22	24	26	30		
187	1	2	3	4	5	7	9	12	16	18	24	26	29	30		
188	1	2	3	5	7	12	13	15	18	19	22	24	30			
189	1	2	3	4	5	7	12	13	15	18	19	24	29	30		
190	1	2	3	5	7	12	13	15	18	22	24	25	30			
191	1	2	3	4	5	7	12	13	15	18	24	25	29	30		
192	1	2	3	5	7	12	13	15	18	20	22	24	30			
193	1	2	3	4	5	7	12	13	16	18	20	24	29	30		
194	1	2	3	5	7	12	13	16	18	22	24	26	30			
195	1	2	3	4	5	7	12	13	16	18	24	26	29	30		
196	1	2	3	5	7	12	14	15	18	19	22	24	30			
197	1	2	3	4	7	12	14	15	18	19	24	29	5	30		
198	1	2	3	5	7	12	14	15	18	22	24	25	30			
199	1	2	3	4	5	7	12	14	15	18	24	25	29	30		
200	1	2	3	5	7	12	14	16	18	20	22	24	30			
201																

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
202	1	2	3	4	5	7	12	14	16	18	20	24	29	30		
203	1	2	3	5	7	12	14	16	18	22	24	26	30			
204	1	2	3	4	5	7	12	14	16	18	24	26	29	30		
205	1	7	8	9	15	18	19	22	31	32						
206	1	5	7	8	9	15	18	19	22	30	31	32				
207	1	4	7	8	9	15	18	19	23	29	31	32				
208	1	4	5	7	8	9	15	18	19	29	30	31	32			
209	1	3	7	8	9	15	18	22	23	25	31	32				
210	1	3	5	7	8	9	15	18	22	25	30	31	32			
211	1	3	4	7	8	9	15	18	23	25	29	31	32			
212	1	3	4	5	7	8	9	15	18	25	29	30	31	32		
213	1	7	8	9	16	19	20	22	23	31	32					
214	1	5	7	8	9	16	18	20	22	30	31	32				
215	1	4	7	8	9	16	18	20	23	29	31	32				
216	1	4	5	7	8	9	16	18	20	29	30	31	32			
217	1	2	7	8	9	16	18	22	26	21	32					
218	1	2	5	7	8	9	16	18	22	26	30	31	32			
219	1	2	4	7	8	9	16	18	23	26	29	31	32			
220	1	2	4	5	7	8	9	16	18	26	29	30	31	32		
221	1	2	7	8	13	15	18	19	22	31	32					
222	1	2	5	7	8	13	15	18	19	22	30	31	32			
223	1	2	4	7	8	13	15	18	19	23	29	31	32			

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
224	1	2	4	5	7	8	13	15	18	19	29	30	31	32		
225	1	2	3	7	8	13	15	18	22	23	25	31	32			
226	1	2	3	5	7	8	13	15	18	22	25	30	31	32		
227	1	2	3	4	7	8	13	15	18	23	25	29	31	32		
228	1	2	3	4	5	7	8	13	15	18	25	29	30	31	32	
229	1	2	7	8	13	16	18	21	22	23	31	32				
230	1	2	5	7	8	13	16	18	20	22	30	31	32			
231	1	2	4	7	8	13	16	18	20	23	29	31	32			
232	1	2	4	5	7	8	13	16	18	20	29	30	31	32		
233	1	2	7	8	13	16	18	22	23	26	31	32				
234	1	2	5	7	8	13	16	18	22	25	29	30	31			
235	1	2	4	7	8	13	16	18	23	26	29	31	32			
236	1	2	4	5	7	8	13	16	18	26	29	30	31	32		
237	1	2	3	7	8	14	15	18	19	22	23	31	32			
238	1	2	3	5	7	8	14	15	18	19	22	30	31	32		
239	1	2	3	4	7	8	14	15	18	19	23	29	31	32		
240	1	2	3	4	5	7	8	14	15	18	20	29	30	31	32	
241	1	2	3	7	8	14	15	18	22	23	25	31	32			
242	1	2	3	5	7	8	14	15	18	22	24	30	31	32		
243	1	2	3	4	7	8	14	15	18	23	25	29	31	32		
244	1	2	3	4	5	7	8	14	15	18	25	29	30	31	32	
245	1	2	3	7	8	14	16	18	20	22	23	31	32			

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
246	1	2	3	5	7	8	14	16	18	20	22	30	31	32		
247	1	2	3	4	7	8	14	16	18	20	23	29	31	32		
248	1	2	3	4	5	7	8	14	16	18	20	29	30	31	32	
249	1	2	3	7	8	14	16	18	22	23	26	31	32			
250	1	2	3	5	7	14	16	18	22	26	30	31	32			
251	1	2	3	4	7	8	14	16	18	23	26	29	31	32		
252	1	2	3	4	5	7	8	14	16	18	26	29	30	31	32	
253	1	2	3	7	9	12	15	18	19	22	23	31	32			
254	1	2	3	5	7	9	12	15	18	19	22	30	31	32		
255	1	2	3	4	7	9	12	15	18	19	23	29	31	32		
256	1	2	3	4	5	7	9	12	15	18	19	29	30	31	32	
257	1	2	3	7	9	12	15	18	22	23	25	31	32			
258	1	2	3	5	7	9	12	15	18	22	25	30	31	32		
259	1	2	3	4	7	9	12	15	18	23	25	29	31	32		
260	1	2	3	4	5	7	9	12	15	18	25	29	30	31	32	
261	1	2	3	7	9	12	16	18	20	22	23	31	32			
262	1	2	3	5	7	9	12	16	18	20	22	30	31	32		
263	1	2	3	4	7	9	12	16	18	20	23	29	31	32		
264	1	2	3	4	5	7	9	12	16	18	20	29	30	31	32	
265	1	2	3	7	9	12	16	18	22	23	26	31	32			
266	1	2	3	5	7	9	12	16	18	22	30	31	32			
267	1	2	3	4	7	9	12	16	18	23	26	29	31	32		

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
268	1	2	3	4	5	7	9	12	16	18	26	29	30	31	32	
269	1	2	3	7	12	13	15	18	19	22	23	31	32			
270	1	2	3	5	7	12	13	15	18	19	22	30	31	32		
271	1	2	3	4	7	12	13	15	18	19	23	29	31	32		
272	1	2	3	4	5	7	12	13	15	18	19	29	30	31	32	
273	1	2	3	7	12	13	15	18	22	23	25	31	32			
274	1	2	3	5	7	12	13	15	18	22	25	30	31	32		
275	1	2	3	4	7	12	13	15	18	23	25	29	31	32		
276	1	2	3	4	5	7	12	13	15	18	25	29	30	31	32	
277	1	2	3	7	12	13	16	18	20	22	23	31	32			
278	1	2	3	5	7	12	13	16	18	20	22	30	31	32		
279	1	2	3	4	7	12	13	16	18	23	29	31	32	20		
280	1	2	3	4	5	7	12	13	16	18	20	29	30	31	32	
281	1	2	3	7	12	13	16	18	22	23	26	31	32			
282	1	2	3	5	7	12	13	16	18	22	26	30	31	32		
283	1	2	3	4	7	12	13	16	18	23	26	29	31	32		
284	1	2	3	4	5	7	12	13	16	18	26	29	30	31	32	
285	1	2	3	7	12	14	15	18	19	22	23	31	32			
286	1	2	3	5	7	12	14	15	18	19	22	30	31	32		
287	1	2	3	4	7	12	14	15	18	19	23	29	31	32		
288	1	2	3	4	5	7	12	14	15	18	19	29	30	31	32	
289	1	2	3	7	12	14	15	18	22	23	25	31	32			

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
290	1	2	3	5	7	12	14	15	18	22	25	30	31	32		
291	1	2	3	4	7	12	14	15	18	23	25	29	31	32		
292	1	2	3	4	5	7	12	14	15	18	25	29	30	31	32	
293	1	2	3	7	12	14	16	18	20	22	23	31	32			
294	1	2	3	5	7	12	14	16	18	20	22	30	31	32		
295	1	2	3	4	7	12	14	16	18	20	23	29	31	32		
296	1	2	3	4	5	7	12	14	16	18	20	29	30	31	32	
297	1	2	3	7	12	14	16	18	22	23	26	31	32			
298	1	2	3	5	7	12	14	16	18	22	26	30	31	32		
299	1	2	3	4	7	12	14	16	18	23	26	29	31	32		
300	1	2	3	4	5	7	12	14	16	18	26	29	30	31	32	

TABLE B.12

SOME SIMPLE CUT-SETS OF THE ORIGINAL FAULT TREE

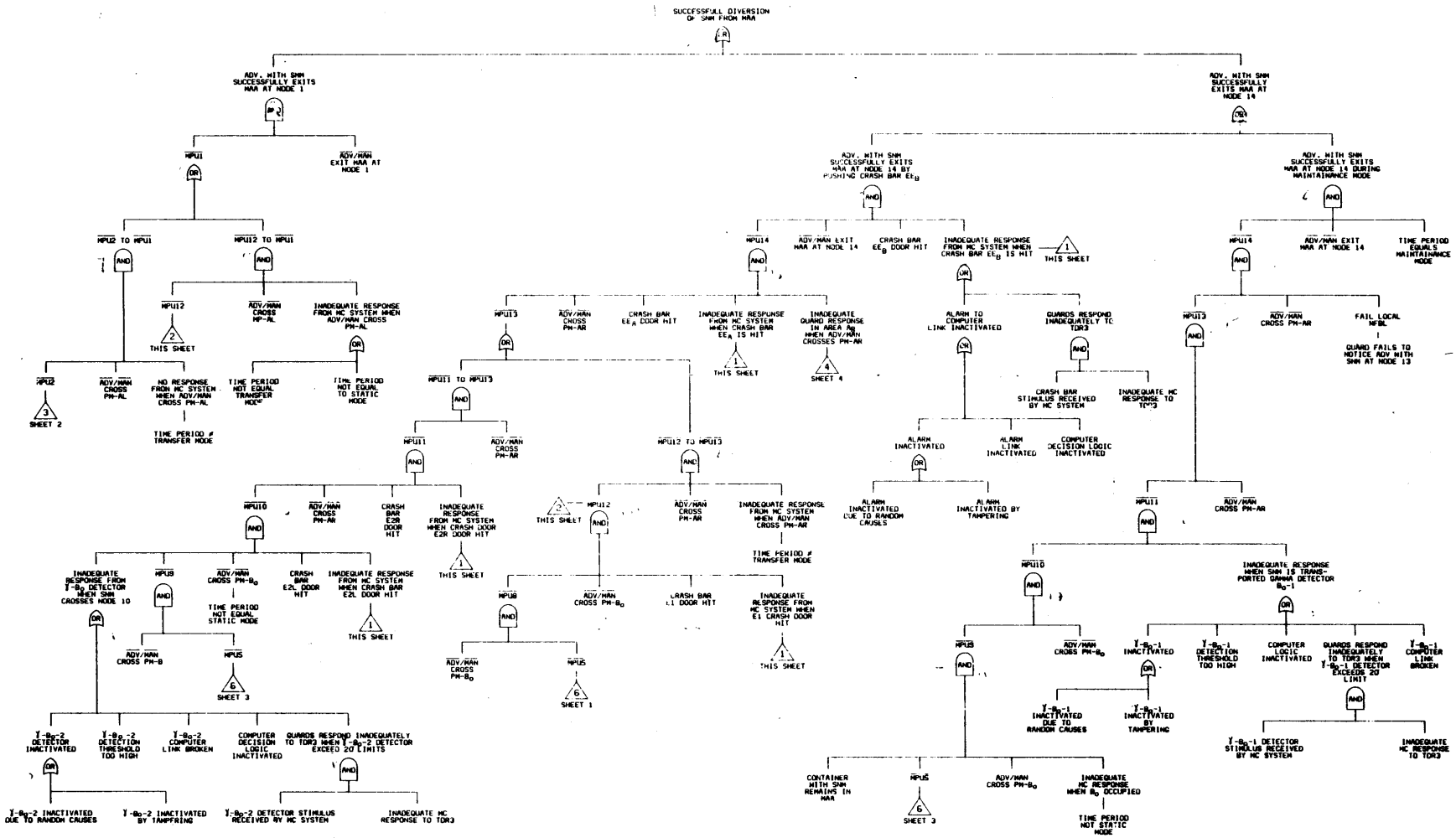
Cut-Set No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	R_4	R_7	R_9	2	1	55	56	57	54	58	59	61	63	62	69	65	67	72	73	R_{11}			
2	R_4	R_7	R_9	2	1	55	56	57	54	58	60	61	63	62	69	65	67	72	73	R_{11}			
3	R_4	R_7	R_{10}	2	1	55	56	57	54	58	60	61	63	62	69	65	67	72	73	R_{11}			
4	R_4	R_7	R_{10}	2	1	55	56	57	54	58	59	61	63	62	69	65	67	72	73	R_{11}			
5	R_{17}	R_6	R_7	55	56	57	54	58	59	61	63	62	69	65	67	72	73	R_{11}	R_9	10	16	2	13
6	R_{17}	R_6	R_7	55	56	57	54	58	59	61	63	62	69	65	67	72	73	R_{11}	R_9	10	16	2	14
7	R_{17}	R_6	R_7	55	56	57	54	58	60	61	63	62	69	65	67	72	73	R_{11}	R_9	10	16	2	14

Only 4 simple cut-sets of an order 20 exists.
These cut-sets are listed above.

Figure B.2

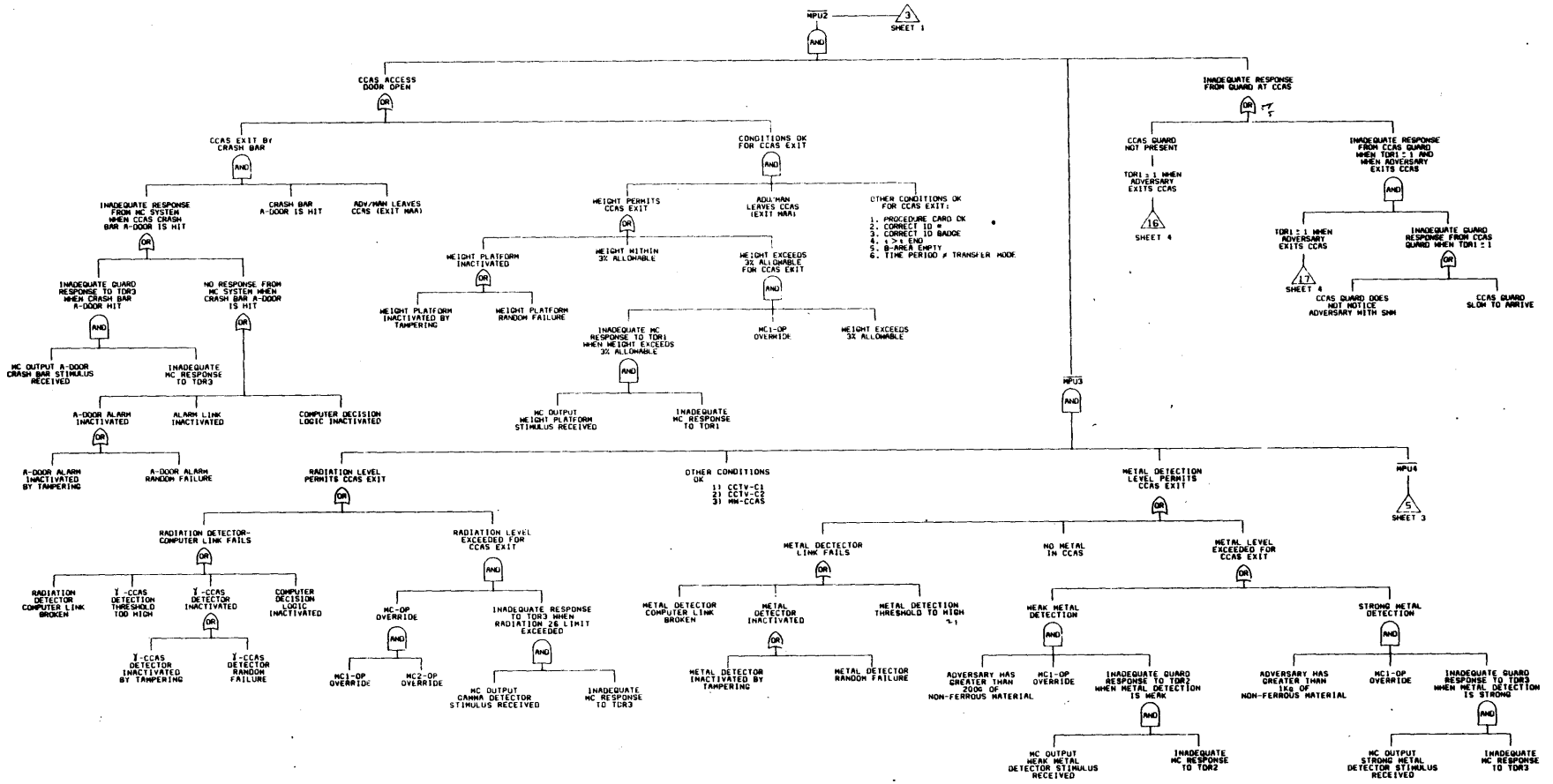
TEST BED DESIGN
FAULT TREE : OR SNM DIVERSION
AT PUMP WASHOUT LINE (706)

SHEET 1/5

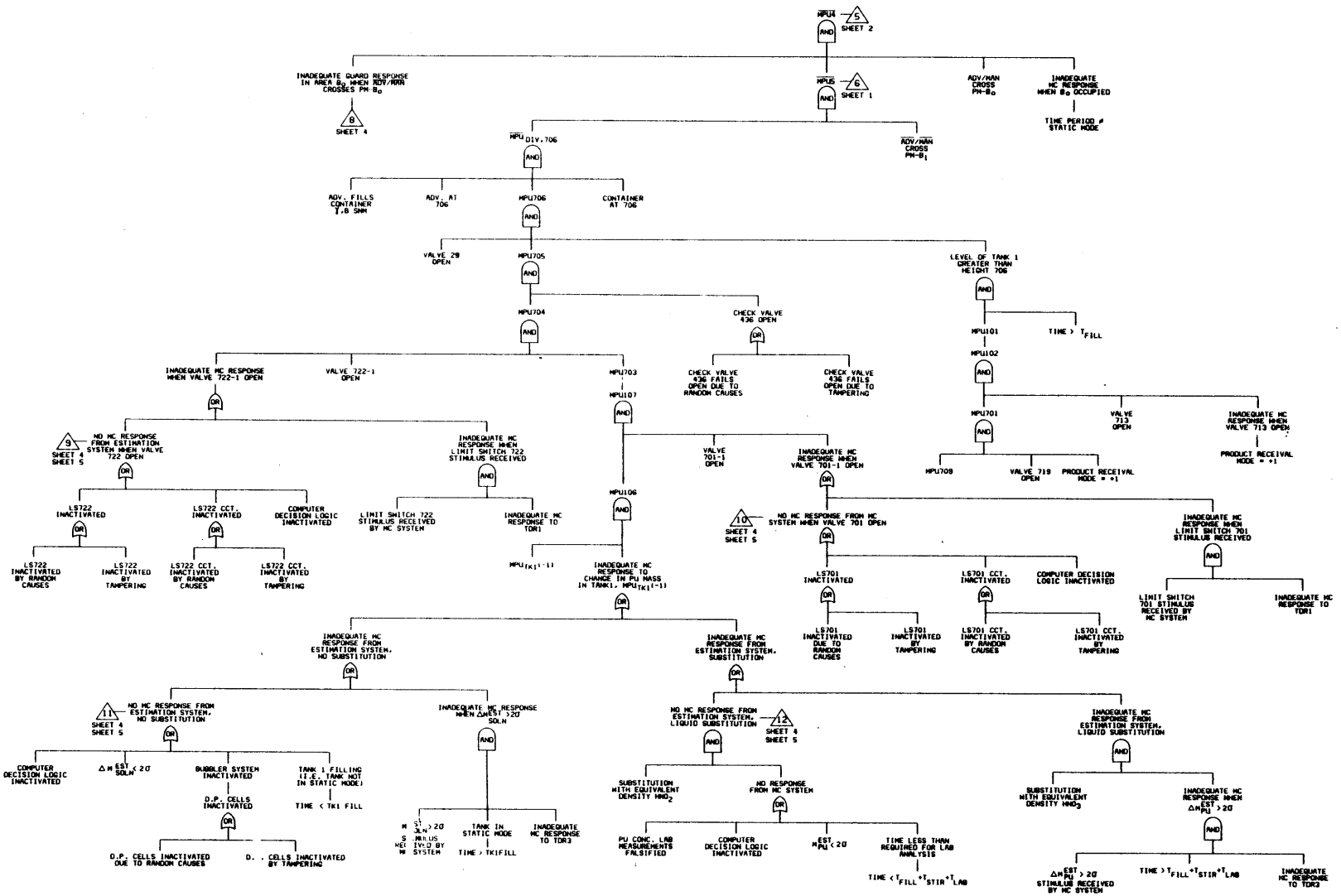


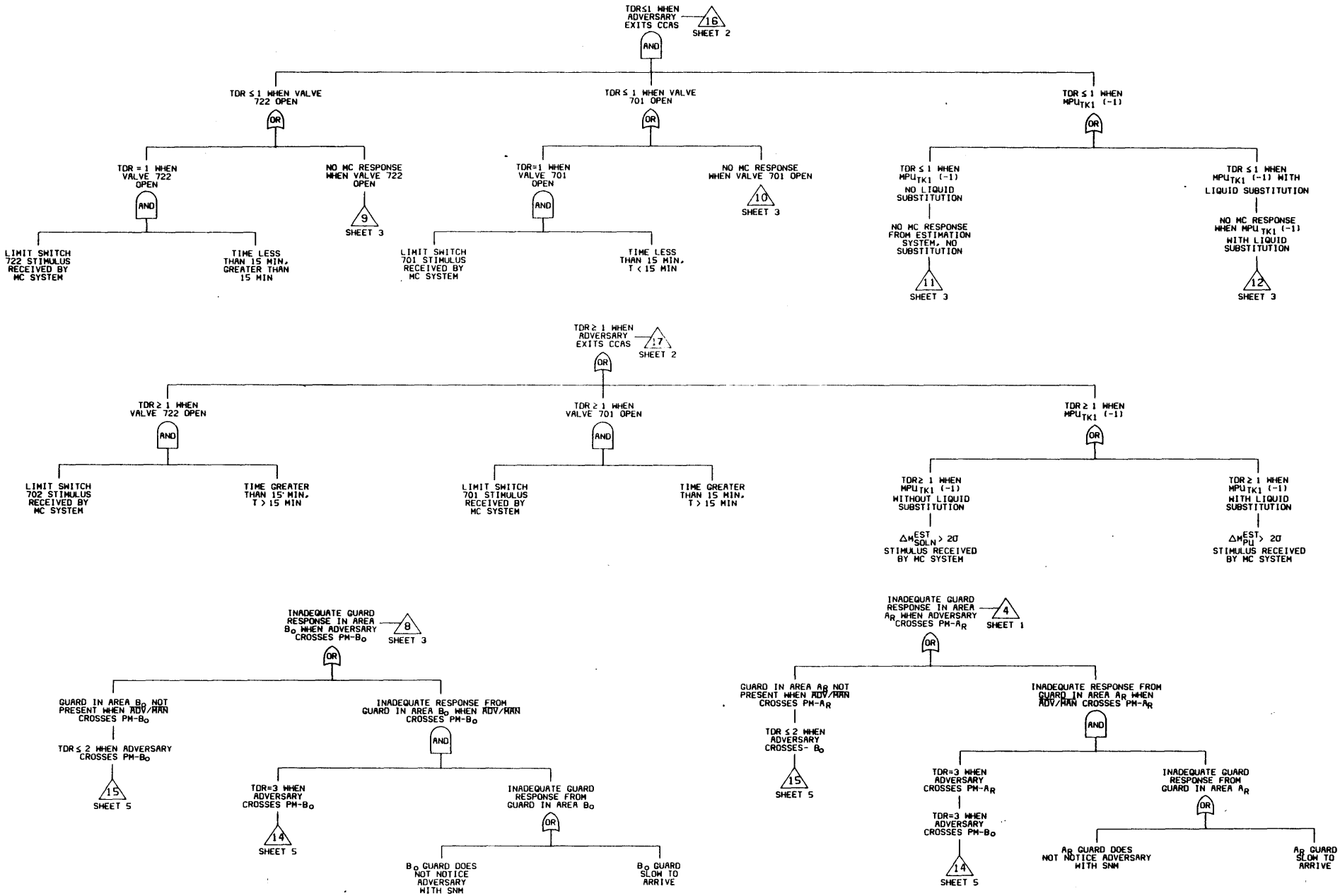
TEST BED DESIGN
FAULT TREE FOR SNM DIVERSION
AT PUMP WASHOUT LINE (706)

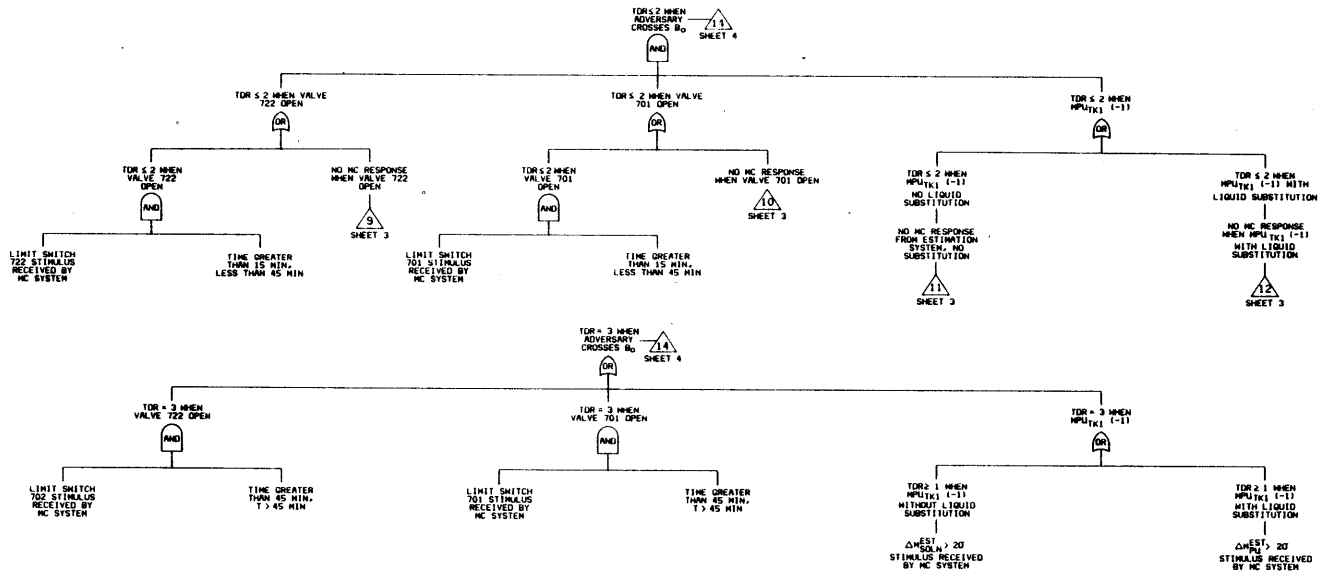
SHEET 2/5



SHEET 3/5


MPU1
5
SHEET 2
AND
INADEQUATE GUARD RESPONSE IN AREA B0 WHEN ADV/MAN CROSSES PM B0
8
SHEET 4
MPU5
6
SHEET 1
ADV/MAN CROSS PM-B0
INADEQUATE MC RESPONSE WHEN B0 OCCUPIED
TIME PERIOD # STATIC MODE
MPU DIV. 706
ADV/MAN CROSS PM-B1
ADV. FILLS CONTAINER T,B SNM
ADV. AT 706
MPU706
CONTAINER AT 706
VALVE 29 OPEN
MPU705
LEVEL OF TANK 1 GREATER THAN HEIGHT 706
MPU704
CHECK VALVE 436 OPEN
MPU101
TIME > TFILL
INADEQUATE MC RESPONSE WHEN VALVE 722-1 OPEN
VALVE 722-1 OPEN
MPU703
CHECK VALVE 436 FAILS OPEN DUE TO RANDOM CAUSES
CHECK VALVE 436 FAILS OPEN DUE TO TAMPERING
MPU102
MPU701
VALVE 713 OPEN
INADEQUATE MC RESPONSE WHEN VALVE 713 OPEN
MPU709
VALVE 719 OPEN
PRODUCT RECEIVAL MODE = +1
PRODUCT RECEIVAL MODE = +1
9
SHEET 4
SHEET 5
NO MC RESPONSE FROM ESTIMATION SYSTEM WHEN VALVE 722 OPEN
INADEQUATE MC RESPONSE WHEN LIMIT SWITCH 722 STIMULUS RECEIVED
MPU107
VALVE 701-1 OPEN
INADEQUATE MC RESPONSE WHEN VALVE 701-1 OPEN
LS722 INACTIVATED
LS722 CCT. INACTIVATED
COMPUTER DECISION LOGIC INACTIVATED
LIMIT SWITCH 722 STIMULUS RECEIVED BY MC SYSTEM
INADEQUATE MC RESPONSE TO TDR1
MPU106
10
SHEET 4
SHEET 5
NO MC RESPONSE FROM MC SYSTEM WHEN VALVE 701 OPEN
INADEQUATE MC RESPONSE WHEN LIMIT SWITCH 701 STIMULUS RECEIVED
LS722 INACTIVATED BY RANDOM CAUSES
LS722 INACTIVATED BY TAMPERING
LS722 CCT. INACTIVATED BY RANDOM CAUSES
LS722 CCT. INACTIVATED BY TAMPERING
MPUTK1(-1)
INADEQUATE MC RESPONSE TO CHANGE IN PU MASS IN TANK1, MPUTK1(-1)
LS701 INACTIVATED
LS701 CCT. INACTIVATED
COMPUTER DECISION LOGIC INACTIVATED
LIMIT SWITCH 701 STIMULUS RECEIVED BY MC SYSTEM
INADEQUATE MC RESPONSE TO TDR1
LS701 INACTIVATED DUE TO RANDOM CAUSES
LS701 INACTIVATED BY TAMPERING
LS701 CCT. INACTIVATED BY RANDOM CAUSES
LS701 CCT. INACTIVATED BY TAMPERING
INADEQUATE MC RESPONSE FROM ESTIMATION SYSTEM, NO SUBSTITUTION
INADEQUATE MC RESPONSE FROM ESTIMATION SYSTEM, SUBSTITUTION
11
SHEET 4
SHEET 5
NO MC RESPONSE FROM ESTIMATION SYSTEM, NO SUBSTITUTION
INADEQUATE MC RESPONSE WHEN ΔMEST PU > 2σ SOLN
NO MC RESPONSE FROM ESTIMATION SYSTEM, LIQUID SUBSTITUTION
12
SHEET 4
SHEET 5
INADEQUATE MC RESPONSE FROM ESTIMATION SYSTEM, LIQUID SUBSTITUTION
COMPUTER DECISION LOGIC INACTIVATED
ΔMEST PU < 2σ SOLN
BUBBLER SYSTEM INACTIVATED
TANK 1 FILLING (I.E. TANK NOT IN STATIC MODE)
D.P. CELLS INACTIVATED
TIME < TK1 FILL
SUBSTITUTION WITH EQUIVALENT DENSITY HNO3
NO RESPONSE FROM MC SYSTEM
SUBSTITUTION WITH EQUIVALENT DENSITY HNO3
INADEQUATE MC RESPONSE WHEN ΔMEST PU > 2σ
D.P. CELLS INACTIVATED DUE TO RANDOM CAUSES
D.P. CELLS INACTIVATED BY TAMPERING
ΔMEST PU > 2σ SOLN
TANK IN STATIC MODE
INADEQUATE MC RESPONSE TO TDR3
TIME > TK1FILL
PU CONC. LAB MEASUREMENTS FALSIFIED
COMPUTER DECISION LOGIC INACTIVATED
MEST PU < 2σ
TIME LESS THAN REQUIRED FOR LAB ANALYSIS
TIME < TFILL + TSTIR + TLAB
ΔMEST PU > 2σ STIMULUS RECEIVED BY MC SYSTEM
TIME > TFILL + TSTIR + TLAB
INADEQUATE MC RESPONSE TO TDR3


TDR≤1 WHEN ADVERSARY EXITS CCAS
16
SHEET 2
AND
TDR ≤ 1 WHEN VALVE 722 OPEN
OR
TDR = 1 WHEN VALVE 722 OPEN
AND
LIMIT SWITCH 722 STIMULUS RECEIVED BY MC SYSTEM
TIME LESS THAN 15 MIN, GREATER THAN 15 MIN
NO MC RESPONSE WHEN VALVE 722 OPEN
9
SHEET 3
TDR ≤ 1 WHEN VALVE 701 OPEN
OR
TDR=1 WHEN VALVE 701 OPEN
AND
LIMIT SWITCH 701 STIMULUS RECEIVED BY MC SYSTEM
TIME LESS THAN 15 MIN, T < 15 MIN
NO MC RESPONSE WHEN VALVE 701 OPEN
10
SHEET 3
TDR ≤ 1 WHEN MPU TK1 (-1)
OR
TDR ≤ 1 WHEN MPU TK1 (-1) NO LIQUID SUBSTITUTION
NO MC RESPONSE FROM ESTIMATION SYSTEM, NO SUBSTITUTION
11
SHEET 3
TDR ≤ 1 WHEN MPU TK1 (-1) WITH LIQUID SUBSTITUTION
NO MC RESPONSE WHEN MPU TK1 (-1) WITH LIQUID SUBSTITUTION
12
SHEET 3
TDR ≥ 1 WHEN ADVERSARY EXITS CCAS
17
SHEET 2
OR
TDR ≥ 1 WHEN VALVE 722 OPEN
AND
LIMIT SWITCH 702 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 15 MIN, T > 15 MIN
TDR ≥ 1 WHEN VALVE 701 OPEN
AND
LIMIT SWITCH 701 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 15 MIN, T > 15 MIN
TDR ≥ 1 WHEN MPU TK1 (-1)
OR
TDR ≥ 1 WHEN MPU TK1 (-1) WITHOUT LIQUID SUBSTITUTION
ΔM EST SOLN > 2σ STIMULUS RECEIVED BY MC SYSTEM
TDR ≥ 1 WHEN MPU TK1 (-1) WITH LIQUID SUBSTITUTION
ΔM EST PU > 2σ STIMULUS RECEIVED BY MC SYSTEM
INADEQUATE GUARD RESPONSE IN AREA Bo WHEN ADVERSARY CROSSES PM-Bo
8
SHEET 3
OR
GUARD IN AREA Bo NOT PRESENT WHEN ADV/MAN CROSSES PM-Bo
TDR ≤ 2 WHEN ADVERSARY CROSSES PM-Bo
15
SHEET 5
INADEQUATE RESPONSE FROM GUARD IN AREA Bo WHEN ADV/MAN CROSSES PM-Bo
AND
TDR=3 WHEN ADVERSARY CROSSES PM-Bo
14
SHEET 5
INADEQUATE GUARD RESPONSE FROM GUARD IN AREA Bo
OR
Bo GUARD DOES NOT NOTICE ADVERSARY WITH SNM
Bo GUARD SLOW TO ARRIVE
INADEQUATE GUARD RESPONSE IN AREA AR WHEN ADVERSARY CROSSES PM-AR
4
SHEET 1
OR
GUARD IN AREA AR NOT PRESENT WHEN ADV/MAN CROSSES PM-AR
TDR ≤ 2 WHEN ADVERSARY CROSSES- Bo
15
SHEET 5
INADEQUATE RESPONSE FROM GUARD IN AREA AR WHEN ADV/MAN CROSSES PM-AR
AND
TDR=3 WHEN ADVERSARY CROSSES PM-AR
TDR=3 WHEN ADVERSARY CROSSES PM-Bo
14
SHEET 5
INADEQUATE GUARD RESPONSE FROM GUARD IN AREA AR
OR
AR GUARD DOES NOT NOTICE ADVERSARY WITH SNM
AR GUARD SLOW TO ARRIVE


TDR≤2 WHEN ADVERSARY CROSSES B_0
11
SHEET 4
AND
TDR ≤ 2 WHEN VALVE 722 OPEN
OR
TDR ≤ 2 WHEN VALVE 701 OPEN
OR
TDR ≤ 2 WHEN MPU_{TK1} (-1)
OR
TDR ≤ 2 WHEN VALVE 722 OPEN
AND
NO MC RESPONSE WHEN VALVE 722 OPEN
9
SHEET 3
LIMIT SWITCH 722 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 15 MIN, LESS THAN 45 MIN
TDR ≤ 2 WHEN VALVE 701 OPEN
AND
NO MC RESPONSE WHEN VALVE 701 OPEN
10
SHEET 3
LIMIT SWITCH 701 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 15 MIN, LESS THAN 45 MIN
TDR ≤ 2 WHEN MPU_{TK1} (-1) NO LIQUID SUBSTITUTION
NO MC RESPONSE FROM ESTIMATION SYSTEM, NO SUBSTITUTION
11
SHEET 3
TDR ≤ 2 WHEN MPU_{TK1} (-1) WITH LIQUID SUBSTITUTION
NO MC RESPONSE WHEN MPU_{TK1} (-1) WITH LIQUID SUBSTITUTION
12
SHEET 3
TDR = 3 WHEN ADVERSARY CROSSES B_0
14
SHEET 4
OR
TDR = 3 WHEN VALVE 722 OPEN
AND
LIMIT SWITCH 702 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 45 MIN, T > 45 MIN
TDR = 3 WHEN VALVE 701 OPEN
AND
LIMIT SWITCH 701 STIMULUS RECEIVED BY MC SYSTEM
TIME GREATER THAN 45 MIN, T > 45 MIN
TDR = 3 WHEN MPU_{TK1} (-1)
OR
TDR ≥ 1 WHEN MPU_{TK1} (-1) WITHOUT LIQUID SUBSTITUTION
$\Delta H^{EST}_{SOLN} > 2\sigma$ STIMULUS RECEIVED BY MC SYSTEM
TDR ≥ 1 WHEN MPU_{TK1} (-1) WITH LIQUID SUBSTITUTION
$\Delta H^{EST}_{PU} > 2\sigma$ STIMULUS RECEIVED BY MC SYSTEM

REFERENCES

[1] J. Olmos and L. Wolf, "A Modular Approach to Fault Tree and Reliability Analysis", Dept. of Nucl. Engng., MIT, MITNE-209, August 1977.

[2] H. Lambert, private communication, March 1978.