NASA
Technical
Paper
2764

1988

SURE Reliability Analysis

Program and Mathematics

Ricky W. Butler
and Allan L. White

Langley Research Center
Hampton, Virginia

NASA
National Aeronautics
and Space Administration

**Scientific and Technical
Information Division**

Contents

Introduction

A reliability analysis of a reconfigurable fault-tolerant computer system requires the determination of the deathstate probabilities of a stochastic reliability model. For more than a decade, automated tools (e.g., ARIES, SURF, and CARE III) have been developed to analyze such models. (See ref. 1.) Recently, a mathematical theorem was proven by White that enables the efficient computation of the deathstate probabilities of a large family of semi-Markov models that are useful for the reliability analysis of fault-tolerant architectures. (See ref. 2.) A major advantage of this new approach is that an arbitrary recovery transition can be handled. Consequently, a specific parametric form of the distribution, such as exponential or uniform, does not have to be assumed. This theorem served as the basis of the original version of the Semi-Markov Unreliability Range Evaluator (SURE). (See ref. 3.) After the development of the original SURE program, the mathematical technique was generalized by Lee (ref. 4) and White (ref. 5). The new mathematical results were used to produce version 2 of SURE which was documented in NASA TM-87593 (ref. 6). After the publication of TM-87593, the capabilities of the program have been further expanded. The following improvements have been made:

1. A simple method for specifying fast exponential transitions has been added
2. A new command to compute the "probabilistic OR" of the results from several "runs" has been added
3. The pruning algorithm has been made more efficient
4. The lower bound has been improved
5. A warning message for loop truncation has been added
6. The accuracy of the $Q(T)$ calculation is now reported for `QTCALC=1`

The SURE program capabilities, including these new features, are fully documented in this paper.

Both White's and Lee's methods provide a means for bounding the probability of entering a deathstate of a semi-Markov model using simple parameters of the model such as the means and variances of the transitions. Consequently, the SURE program computes an upper and lower bound on system reliability. Although an exact answer is not produced by the SURE program, the calculated bounds are close together for reliability models of ultrareliable systems—usually within 5 percent of each other as is shown. The advantage of the SURE technique is that the bounds are algebraic in form and, consequently, are computationally efficient. Very large and complex models can be analyzed by the program. Furthermore, the technique applies to the general class of semi-Markov models and thus does not impose restrictions on the type of architecture that can be analyzed. Of course, the practical utility of the tool is related to the closeness of the generated bounds.

Since the SURE program can handle any form of recovery process (i.e., any mathematical distribution of recovery time), the fault-handling process of a fault-tolerant computer system can be captured in a single transition. It is unnecessary to assume some underlying parametric form or a special model of fault-handling behavior. The results of recovery-process experimentation can be used directly in the SURE program, which only requires the mean and standard deviation of the observed recovery times. If the user desires to model the recovery process with a number of transitions (e.g., a detailed fault-handling model), the SURE program can still be used, but the user must supply values for all the transitions included in the model.

In this paper, the method of Markov/semi-Markov modeling is first reviewed. Second, the essential aspects of the new bounding theorems are presented. Third, the technique used by SURE to handle transient and intermittent fault models is given. Fourth, the tightness of the SURE bounds is discussed. Fifth, a detailed description of the user input language is given along with several illustrative interactive sessions. Finally, the mathematical derivation of the bounding theorem is presented in detail.

SURE Approach to Reliability Analysis

The SURE approach to the reliability analysis of a fault-tolerant computer system is an extension of the standard Markov modeling approach. Markov models have been used for many years to describe fault-tolerant systems. (See ref. 7, pp. 246–302.) However, many reliability analysts are unfamiliar with this technique, since fault-tree analysis has been sufficient for non-reconfigurable systems. In recent years, reconfigurable architectures which cannot be analyzed with fault trees have been designed and implemented. The more powerful Markov approach is used which captures the dynamic aspects of the system in a natural manner. The following section has been included to introduce the Markov modeling method along with the semi-Markov extensions.

Reliability Modeling of Computer System Architecture

Highly reliable systems must use parallel redundancy to achieve their fault tolerance since current manufacturing techniques cannot produce circuitry with adequate reliability. Furthermore, reconfiguration has been utilized in an attempt to increase the reliability of the system without the overhead of even more redundancy. Such systems exhibit behavior that involves both slow and fast processes, and when modeled stochastically, some state transitions are many orders of magnitude faster than others. The slower transitions correspond to fault arrivals in the system. The faster transition rates correspond to the system recovery from faults. If the states of the system are delineated properly, then the slow transitions can be obtained from field data and/or by using the MIL-STD-217D Handbook calculation. These transitions have been shown to be exponentially distributed for most electronic devices, which is assumed in the SURE program. (See ref. 7, pp. 31–42.) The system recovery processes can be measured experimentally by using fault injection. In a pure Markov model, the recovery process would typically be represented as a single exponential transition. However, experiments made by the Charles Stark Draper Laboratory, Inc., on the Fault-Tolerant Multiprocessor (FTMP) computer architecture have demonstrated (ref. 8) that these transitions are not exponential. In order to model the nonexponential behavior of these processes accurately, semi-Markov models are necessary. Once a system has been mathematically modeled and the state transitions determined, a computational tool such as SURE may be used to compute the probability of entering the deathstates (i.e., the states that represent system failure) within a specified mission time, for example, 10 hours.

Mathematical models of fault-tolerant systems must describe the processes that lead to system failure and the system fault-recovery capabilities. The first level of model granularity to consider is the unit of reconfiguration/redundancy in the system. In some systems this is as large as a complete processor with memory. In other systems, a smaller unit such as a CPU or memory module is appropriate. The states of the mathematical model are vectors of attributes such as the number of faulty units and the number of removed units. Certain states in the system represent system failure and others represent fault-free behavior or correct operation in the presence of faults.

A semi-Markov model of a triad of processors with one spare is given in figure 1. The outputs of the processors in the triad are voted in order to mask faults. (In this model it is assumed that the spare does not fail while inactive.) The horizontal transitions represent fault arrivals; these occur with exponential rate λ. The coefficients of λ represent the number of processors in the configuration that can fail. The vertical transitions represent recovery from a fault. The first recovery is accomplished by replacing the faulty processor with a spare. The second recovery is accomplished by degrading to a simplex processor. A recovery transition typically is not exponentially distributed and, consequently, must be described by a general distribution function $F(t)$ where

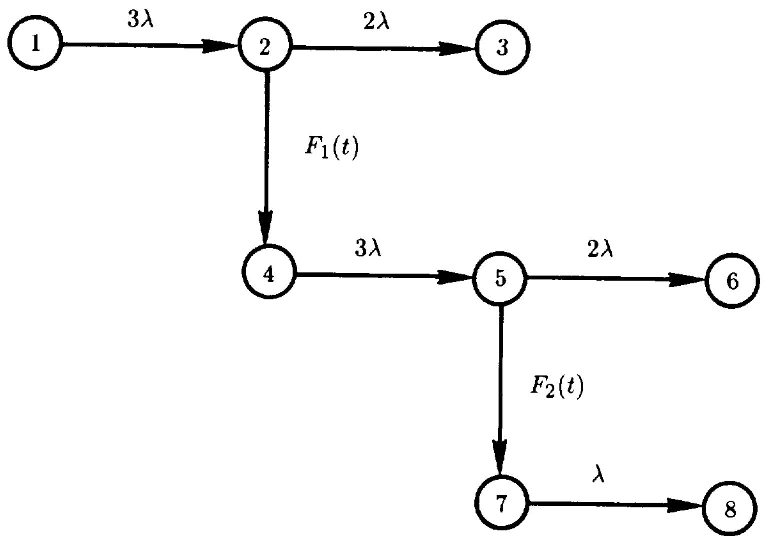


Figure 1. Semi-Markov model of triad with one spare.

$F(t)$ = Probability that the recovery occurs within t hours after the fault arrives

Throughout the paper, greek letters are used to represent the rates of exponential transitions, and roman letters are used to represent the distributions of the fast recovery transitions. In the model of figure 1, the two recovery processes are different; therefore, two different recovery distributions are necessary—$F_1(t)$ and $F_2(t)$. Since the system uses three-way voting for fault masking, there is a "race" between the occurrence of a second fault and the removal of the first. If the second fault wins the race, then system failure occurs (state 3).

The development of a reliability model of a large, complex system uses the same concepts that are used in the development of the model of the triad plus a spare. The two types of transitions—failure and recovery—are still used, but there often are many different types of failure and different recoveries for each type. Thus, there may be several failure transitions from a state, each representing a failure of a different part of the system. Likewise, some states are reached after a sequence of different failures, and thus, there are multiple recoveries from the state. In this situation, the response of the system to two simultaneous failures must be measured and included in the model.

Formerly, the numerical solution of a semi-Markov model was intractable; therefore, pure Markov models were typically used to model reconfigurable systems. Since the SURE program solves semi-Markov models, more realistic system models can now be used to calculate system reliability. Furthermore, since the mathematical bounds depend only upon the conditional means and standard deviations of the recovery transitions, distribution fitting is unnecessary. Given an empirical distribution of system recovery, the easily calculated sample means and standard deviations can be used directly.

SURE Program

The calculation of the probability of entering a deathstate of a Markov model requires the solution of a set of coupled differential equations. The solution of the more general semi-Markov model requires the numerical integration of a set of convolution integrals. Because of the large disparity between the rates of fault arrivals and system recoveries, models of fault-tolerant architectures inevitably lead to numerically stiff differential-integral equations. This problem along with the large computational cost of solving large state space problems has led to the use of exotic computational methods in recent reliability analysis tools such as CARE III and HARP. (See refs. 9 and 10.) In such programs, the problem is decomposed into a fault-handling model and a fault-occurrence model. Coverage parameters derived from the solution of the fault-handling model are inserted by various aggregation techniques into the fault-occurrence model in order to compute the system reliability. These aggregation techniques are based on the assumption that critical-pair failures are the dominant failure mode in the system. Unfortunately, such strategies reduce the class of architectures that can be modeled. Because SURE does not rely on the solution of differential equations, stiffness is not a problem—in fact, the "stiffer" the model, the more accurate the approximation technique. Furthermore, since the SURE program computes probabilities using algebraic formulas, large state spaces can be accommodated. Therefore, decomposition or aggregation techniques are unnecessary and have not been utilized. A simple model pruning technique, however, has been included in the SURE program for extremely large models that otherwise might require large computational resources.

The SURE program is based on a new method for computing the reliability of a fault-tolerant system. Two features of a fault-tolerant system have traditionally made this task difficult. First, the use of sophisticated digital processors has led to complex reconfiguration strategies which result in large, complex models. Unfortunately, one cannot arbitrarily ignore details when attempting to estimate the reliability of an ultra-reliable system. Second, the rate of recovery is many orders of magnitude faster than the fault-arrival process. This causes rapid growth in the error terms in numerical integration algorithms. The new mathematical theorem which SURE is based on provides a solution to both of these problems for systems with slow fault arrival processes and fast system recovery (i.e., a good fault-tolerant system). The theorem establishes that just the means and variances of the recovery times are sufficient information about the reconfiguration process in order to obtain tight bounds on the probability of system failure. The bounds consist of an algebraic factor using the means and variances of the system recoveries and a factor that is the solution of a nonstiff differential equation whose coefficients are the slow fault-occurrence rates. Thus, the theorem reduces the traditionally difficult problem to easily computed mathematics which provides the basis of the SURE program.

The input language to the SURE program is very simple. The input model is defined by listing all the transitions of the model. For example, the model of figure 1 is defined as follows:

```
LAMBDA = 1E-4;        (* Failure rate of a processor *)
MU1 = 2.7E-4;         (* Mean time to replace faulty processor w/ a spare *)
SIGMA1 = 1.4E-4;      (* Standard deviation of time to replace w/ a spare *)
MU2 = 9.2E-4;         (* Mean time to degrade to a simplex *)
SIGMA2 = 3.8E-4       (* Standard deviation of time to degrade to simplex *)

1,2 = 3*LAMBDA;
2,3 = 2*LAMBDA;
2,4 = <MU1,SIGMA1>;
4,5 = 3*LAMBDA;
5,6 = 2*LAMBDA;
5,7 = <MU2,SIGMA2>;
7,8 = LAMBDA;
```

The first five statements equate values to identifiers (i.e., symbolic names). The first identifier LAMBDA represents the processor failure rate. The next two identifiers MU1 and SIGMA1 are the mean and the standard deviation of the time to replace a faulty processor with a spare. The last two identifiers MU2 and SIGMA2 are the mean and standard deviation of the time to degrade to a simplex. Conveniently, the means and the standard deviations are the only information SURE needs about the nonexponential recovery processes. The final seven statements define the transitions of the model. If the transition is a slow fault arrival process then only the exponential rate must be provided. The last statement defines a transition from state 7 to state 8 with rate LAMBDA. If the transition is a fast recovery process then the mean and the standard deviation of the recovery time must be given. For example, the statement `2,4 = <MU1,SIGMA1>` above defines a transition from state 2 to state 4 with mean recovery time MU1 and standard deviation SIGMA1.

The SURE program is currently running under VMS 4.4 on VAX-11/750 and VAX-11/780 computers at the NASA Langley Research Center. The program has been designed with minimal usage of VMS specific constructs. Consequently, the program should be easy to transfer to other systems. The SURE program consists of three modules—the front end module, the computation module, and the graphics output module. The front end and computation modules are implemented in Pascal and should easily transfer to other machines. The graphics output module is written in FORTRAN but uses the graphics library TEMPLATE; this module can be used only by installations having this library. The SURE program can be installed and used without the graphics output module. Alternatively, this module can be rewritten using another graphics library. The SURE program is available from NASA's software dissemination center:

Computer Software Management and Information Center (COSMIC)
The University of Georgia
382 East Broad Street
Athens, GA 30602

The Fundamental SURE Mathematics

In this section, the mathematical theorems upon which the SURE program is based are presented in summary form. Two closely related theorems are implemented in the SURE program. One theorem enables the user to describe the system recovery processes in terms of means and variances. The other theorem enables the user to describe the system recovery processes in terms of means and percentiles. The SURE user is free to use either method he wishes. In the next two subsections, the two bounding theorems are discussed. A complete derivation of the theorem using means and variances is given in the section entitled "Derivation of Bounding Theorem." The second theorem can be proven with basically the same techniques used in the proof of the first theorem. For details the reader is referred to reference 4.

Bounds Based on Means and Variances

The theorem provides a means of bounding the probability of traversing a specific path in the model within the specified time. Since traversing different paths are disjoint events, the bounds for all the paths can be added together to get the bounds for the entire model. A simple semi-Markov model of the six-processor Software Implemented Fault-Tolerance (SIFT) computer system (ref. 11) is used to introduce the theorem. This model is illustrated in figure 2.

The horizontal transitions in the model represent fault arrivals which are assumed to be exponentially distributed and relatively slow. The vertical transitions represent system recoveries by reconfiguration, that is, removal of the faulty processor from the working set of processors. These transitions are assumed to be fast but can have arbitrary distribution. White's theorem requires only that the means and variances of the fast transitions and their transition probabilities be specified. The deathstates of the model are 4, 8, 11, 14, and 16. Deathstate 4 represents the case in which three processors out of six have failed before the system reconfigures. State 16 represents the case in which the system has been completely depleted of processors. The unreliability of the system is precisely the sum of the probabilities of entering each deathstate. The theorem is used to analyze every path from the start state to the deathstates. In the SIFT model the following paths must be considered:

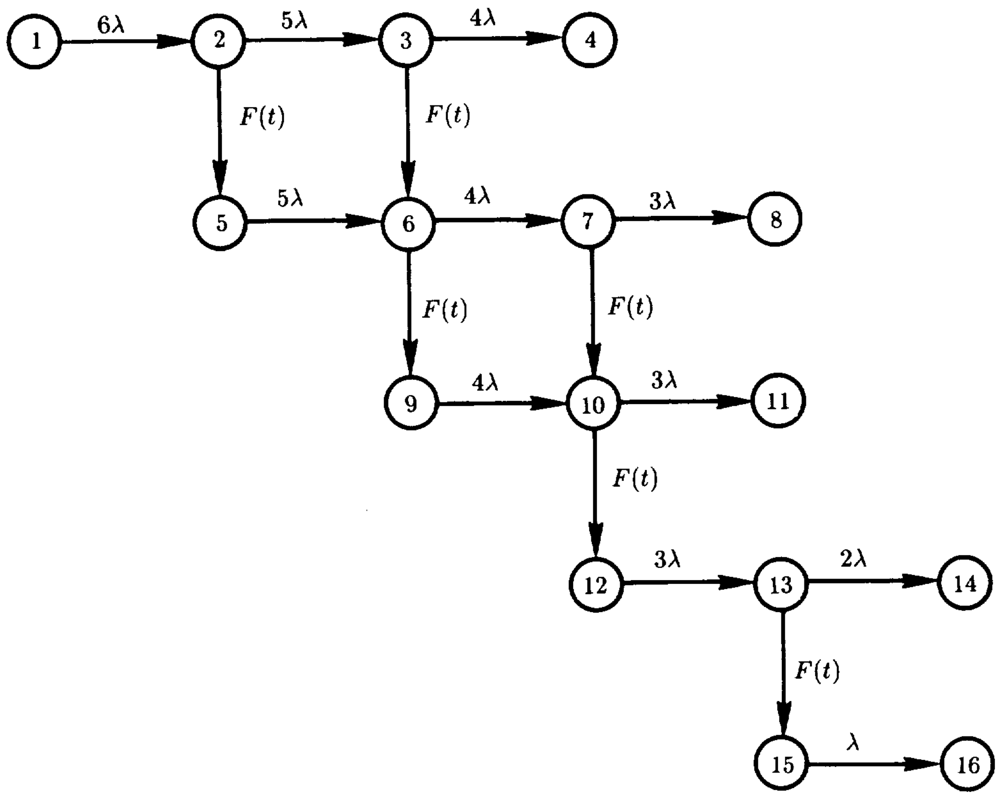


Figure 2. Semi-Markov model of SIFT.

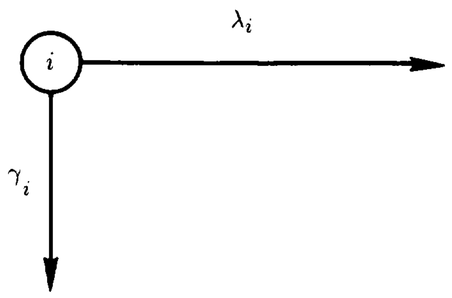
Path 1:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$
Path 2:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 8$
Path 3:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$
Path 4:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 11$
Path 5:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 11$
Path 6:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 11$
Path 7:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 11$
Path 8:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 14$
Path 9:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 14$
Path 10:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 14$
Path 11:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 14$
Path 12:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow 16$
Path 13:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow 16$
Path 14:	$1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow 16$
Path 15:	$1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow 16$

The number of paths can be enormous in a large model. The SURE computer program automatically finds all the paths in the model.

Path-Step Classification

Once a particular path has been isolated for analysis, the theorem is easily applied. In the analysis, each state along the path must first be classified into one of three classes which are distinguished by the type of transitions leaving the state. A state and all the transitions leaving it are referred to as a "path step." The transition on the path currently being analyzed is referred to as the "on-path transition." In the following sketches (sketches A–C), the on-path transition will always be the horizontal transition. (This is different from the previous sections where the horizontal transitions were fault arrivals and vertical transitions were recoveries.) The remaining transitions will be referred to as the "off-path transitions." The classification is made on the basis of whether the on-path and off-path transitions are slow (and hence also exponential) or fast. If there are no off-path transitions, the path step is classified as if it contained a slow off-path transition. Thus, the following classes of path steps are of interest.

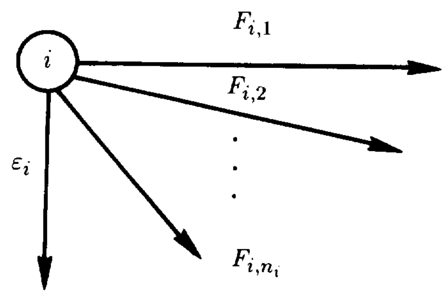
Class 1: slow on path, slow off path.



Sketch A

The rate of the on-path exponential transition is λ_i. (See sketch A.) There may be an arbitrary number of slow off-path transitions. The sum of their exponential transition rates is γ_i. If any of the off-path transitions are not slow, then the path step is in class 3. The path steps $1 \to 2$ and $5 \to 6$ in the SIFT model (fig. 2) are examples.

Class 2: fast on path, arbitrary off path.



Sketch B

The on-path transition must be fast in order for the path step to be in class 2. There may be an arbitrary number of slow or fast off-path transitions. As before, the slow off-path, exponential transitions can be represented as a single transition with a rate ε_i equal to the sum of all the slow off-path transition rates. (See sketch B.) The path steps $2 \to 5$ and $3 \to 6$ in the SIFT model (fig. 2) are examples. The distribution of the fast on-path transition is $F_{i,1}$. The distribution of time for the kth fast transition from state i is referred to as "$F_{i,k}$" (i.e., the probability that the next transition out of state i is into state k and that the transition occurs within time t is $F_{i,k}$). Three measurable parameters must be specified for each fast transition. These are the transition probability $\rho(F^*_{i,k})$, the conditional mean $\mu(F^*_{i,k})$, and the conditional variance $\sigma^2(F^*_{i,k})$, given that this transition occurs. Mathematically, these parameters are defined as follows:

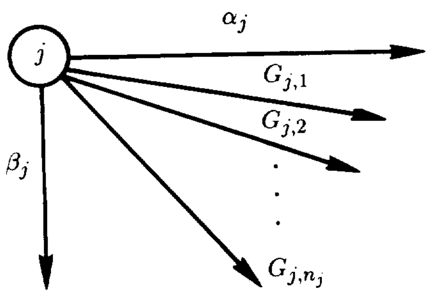
$$\rho(F^*_{i,k}) = \int_0^\infty \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

$$\mu(F^*_{i,k}) = \frac{1}{\rho(F^*_{i,k})} \int_0^\infty t \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

$$\sigma^2(F^*_{i,k}) = \frac{1}{\rho(F^*_{i,k})} \int_0^\infty t^2 \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t) - \mu^2(F_{i,k})$$

Experimentally, these parameters correspond to the fraction of times that a fast transition is successful and the mean and variance of the conditional distribution given that the transition occurs.[1] *The asterisk is used to indicate that the parameters are defined in terms of the conditional distributions.* It should be noted that these expressions are defined independently of the exponential transitions ε_j. *Consequently, the sum of the fast transition probabilities* $\sum \rho(F^*_{i,k})$ *must be 1.* In particular, if there is only one fast transition, its probability is 1 and the conditional mean is equivalent to the unconditional mean. (The SURE user does not have to deal explicitly with the unconditional distributions $F_{i,k}$. However, in order to develop the mathematical theory, they must be used.)

Class 3: slow on path, fast off path.



Sketch C

This class includes path steps with both slow and fast off-path transitions. The on-path transition must be slow. At least one off-path transition must be fast or the path step is in class 1. (See sketch C.) The path steps 2 → 3 and 7 → 8 in the SIFT model (fig. 2) are in this class. The slow on-path transition rate is α_j. The sum of the slow off-path transition rates is β_j. As in class 2, the transition probability $\rho(G^*_{j,k})$, the conditional mean $\mu(G^*_{j,k})$, and the conditional variance $\sigma^2(G^*_{j,k})$ must be given for each fast off-path transition with distribution $G_{j,k}$.[2]

Although the parameters described suffice to specify a class 3 path step to SURE, the mathematical theory is more easily expressed in terms of the holding time in the state. The holding time in a state is the time the system remains in the state before it transitions to some other state. The bounding theorem is expressed using a slightly different form of holding time

[1] In any experiment where competing processes in a system are studied, the observed empirical distributions are conditional. The time it takes a system to transition to the next state is only observed when that transition occurs.

[2] There really is no difference between transitions labeled with F and those labeled with G. The two different letters are used to help keep track of the context, i.e., whether the transition is a class 2 (labeled F) or class 3 (labeled G) in the current path. In either case, the SURE user supplies the conditional mean, the conditional standard deviation, and the transition probability.

which will be referred to as "recovery holding time" to prevent confusion. The recovery holding time is the holding time in the state with the slow exponential distributions removed. Since the slow exponential transitions occur at a rate many orders of magnitude less than the fast transitions, the recovery holding time is approximately equal to the holding time. Letting H_j represent the distribution of the recovery holding time in state j gives

$$H_j(t) = 1 - \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]$$

then the following parameters are used in the theorem:

$$\mu(H_j) = \int_0^\infty \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]\, dt$$

$$\sigma^2(H_j) = 2\int_0^\infty t \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]\, dt - \mu^2(H_j)$$

These parameters are the mean and the variance of the holding time in state j without consideration of the slow exponential transitions (i.e., with the slow exponential transitions removed). *These parameters do not have to be supplied to the SURE program.* The SURE program derives these parameters from the other available inputs—$\rho(G^*_{j,k})$, $\mu(G^*_{j,k})$, and $\sigma^2(G^*_{j,k})$—as follows:

$$\mu(H_j) = \sum_{k=1}^{n_j} \rho(G^*_{j,k})\, \mu(G^*_{j,k})$$

$$\sigma^2(H_j) = \left\{ \sum_{k=1}^{n_j} \rho(G^*_{j,k})\, [\sigma^2(G^*_{j,k}) + \mu^2(G^*_{j,k})] \right\} - \mu^2(H_j)$$

where $\rho(G^*_{i,k})$, $\mu(G^*_{j,k})$, and $\sigma^2(G^*_{j,k})$ are defined as

$$\rho(G^*_{i,k}) = \int_0^\infty \prod_{j \neq k} [1 - G_{i,j}(t)]\, dG_{i,k}(t)$$

$$\mu(G^*_{i,k}) = \frac{1}{\rho(G^*_{i,k})} \int_0^\infty t \prod_{j \neq k} [1 - G_{i,j}(t)]\, dG_{i,k}(t)$$

$$\sigma^2(G^*_{i,k}) = \frac{1}{\rho(G^*_{i,k})} \int_0^\infty t^2 \prod_{j \neq k} [1 - G_{i,j}(t)]\, dG_{i,k}(t) - \mu^2(G_{i,k})$$

These parameters are defined in exactly the same way as the class 2 path-step parameters.

Although the fast distributions are specified without consideration of the competing slow exponential transitions, the theorem gives bounds that are correct in the presence of such exponential transitions. The parameters were defined in this manner to simplify the process of specifying a model. Throughout the paper, the holding time in a state in which the slow transitions have been removed is referred to as "recovery holding time."

Summary of Information Needed by SURE Program

Although the path-step classification discussion in the previous section included a significant amount of detail in order to make the mathematical theory tractable, the amount of information needed by the program is quite small. The following parameters must be given for each type of path step:

Class 1 parameters:

$$\lambda_i = \text{rate of on-path exponential transition from state } i$$

$$\gamma_i = \text{sum of off-path exponential transition rates from state } i$$

Class 2 parameters:

$$\varepsilon_i = \text{sum of all slow off-path transition rates from state } i$$

$$\rho(F^*_{i,k}) = \text{probability that } k\text{th transition from state } i \text{ is successful}$$

$$\mu(F^*_{i,k}) = \text{conditional mean transition time of } k\text{th transition from state } i \text{ given that this transition occurs}$$

$$\sigma(F^*_{i,k}) = \text{conditional standard deviation of } k\text{th transition from state } i \text{ given that this transition occurs}$$

Class 3 parameters:

$$\alpha_j = \text{slow on-path transition rate from state } j$$

$$\beta_j = \text{sum of all slow off-path transition rates from state } j$$

$$\rho(G^*_{j,k}) = \text{probability that } k\text{th transition from state } j \text{ is successful}$$

$$\mu(G^*_{j,k}) = \text{conditional mean transition time of } k\text{th transition from state } j \text{ given that this transition occurs}$$

$$\sigma(G^*_{j,k}) = \text{conditional standard deviation of } k\text{th transition from state } j \text{ given that this transition occurs}$$

White's Multiple Recovery Theorem

With the previous classification, the bounding theorem can now be given. For convenience, when referring to a specific path in the model, the distribution of an on-path fast transition is indicated by a single subscript which specifies the source state. For example, if the transition with distribution $F_{j,k}$ is the on-path transition, then it can be referred to as F_j:

$$F_{j,k} = k\text{th fast transition from state } j$$
$$F_j = \text{on-path fast transition from state } j$$

Theorem [White]: The probability $D(T)$ of entering a particular deathstate within the mission time T, following a path with k class 1 path steps, m class 2 path steps, and n class 3 path steps, is bounded as follows:

$$\text{LB} \le D(T) \le \text{UB}$$

where

$$\text{UB} = Q(T) \prod_{i=1}^{m} \rho(F_i^*) \prod_{j=1}^{n} \alpha_j \mu(H_j)$$

$$\text{LB} = Q(T-\Delta) \prod_{i=1}^{m} \rho(F_i^*) \left[1 - \varepsilon_i \mu(F_i^*) - \frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}\right]$$

$$\times \prod_{j=1}^{n} \alpha_j \left\{\mu(H_j) - \frac{(\alpha_j + \beta_j)[\mu^2(H_j) + \sigma^2(H_j)]}{2} - \frac{\mu^2(H_j) + \sigma^2(H_j)}{s_j}\right\}$$

for all values of $r_i > 0$, $s_j > 0$; and

$$\Delta = r_1 + r_2 + \cdots + r_m + s_1 + s_2 + \cdots + s_n$$

and

Q(T) = probability of traversing the path consisting of the k class 1 path steps within time T

The theorem is true for any $r_i > 0$ and $s_j > 0$ provided that $\Delta < T$. Different choices of these parameters will lead to different bounds. The SURE program uses the following values of r_i and s_j:

$$r_i = \left\{2\text{T}\left[\mu^2(F_i^*) + \sigma^2(F_i^*)\right]\right\}^{1/3}$$

$$s_j = \left\{\frac{\text{T}[\mu^2(H_j) + \sigma^2(H_j)]}{\mu(H_j)}\right\}^{1/2}$$

The default values have been found to give very close bounds in practice, usually very near the optimal choice. A mathematical procedure for selecting the globally optimal values of r_i and s_j (i.e., leading to the closest bounds) has not been developed. However, the values used by the SURE program are shown to be near optimal in appendix A.

Two simple algebraic approximations for $Q(T)$ were given by White (ref. 2)—one that overestimates and one that underestimates, respectively:

$$Q(T) < Q_u(T) = \frac{\lambda_1 \lambda_2 \lambda_3 \cdots \lambda_k T^k}{k!}$$

$$Q(T) > Q_1(T) = Q_u(T) \left[1 - \frac{T}{k+1} \sum_{i=1}^{k} (\lambda_i + \gamma_i)\right]$$

Both $Q_u(T)$ and $Q_1(T)$ are close to $Q(T)$ as long as $T \sum (\lambda_i + \gamma_i)$ is small; that is, as long as the mission time is short compared with the average lifetime of the components. The SURE program uses the following slightly improved upper bound on $Q(T)$:

$$Q(T) < Q_u^*(T) = \frac{1}{|S|!} \prod_{i \in S} \lambda_i T$$

where

$$S = \{i \mid \lambda_i T < 1\}$$

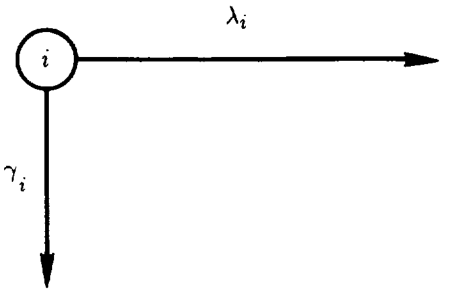
This bound is obtained by removing all the fast exponential transitions from the $Q(T)$ model. Since the path is shorter, the probability of reaching the deathstate is larger than the original $Q(T)$ model. These algebraic bounds on $Q(T)$ are used when the QTCALC option is set equal to 0. When the QTCALC = 1 option is used, a differential equation solver is used to calculate $Q(T)$ and $Q(T - \Delta)$. *If* `QTCALC = 2`, *then the SURE program automatically selects the most appropriate method.* This option is discussed in a subsequent section entitled "SURE User Interface."

Bounds Based on Means and Percentiles

Path-Step Classification

The path-step classification defined in the previous section is also useful for describing Lee's technique (ref. 4). The primary difference (from a user's perspective) between Lee's and White's techniques is the method of describing the fast transitions. Lee's technique utilizes the transition probabilities, the conditional mean holding time, as well as a user-chosen percentile of the recovery holding time distribution. The required information for each class is listed as follows:

Class 1: slow on path, slow off path.

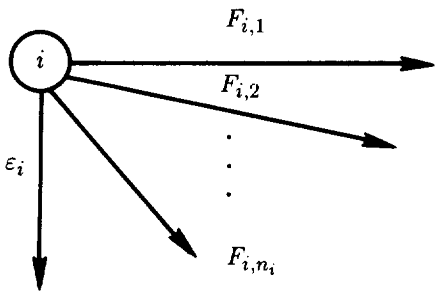


Sketch A (repeated)

λ_i = on-path exponential transition rate

γ_i = sum of off-path exponential transition rates

Class 2: fast on path, arbitrary off path.



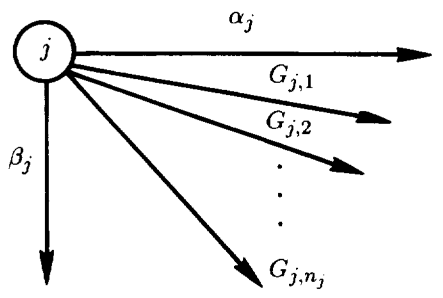
Sketch B (repeated)

$$\rho(F^*_{i,k}) = \int_0^\infty \prod_{j \neq k} [1 - F_{i,j}(t)]\, dF_{i,k}(t)$$

= probability that kth fast transition from state i is successful (mathematically and experimentally the same as in White's theory)

ε_i = sum of off-path exponential transition rates

Class 3: slow on path, fast off path.



Sketch C (repeated)

α_j = slow on-path transition rate

$\rho(G^*_{j,k})$ = probability that kth fast transition from state j is successful (this must be specified for all competing off-path fast transitions)

ξ_j = percentile point of distribution chosen by user; this can also be viewed as censoring point of experiment (i.e., longest time experimenter waits for a transition to occur)

$H_j(\xi_j)$ = probability that recovery holding time (i.e., with slow transitions removed) in state j is less than ξ_j

$\mu(\xi_j)$ = conditional mean of recovery holding time in source state j, given that fastest transition occurs before ξ_j

$$= \int_0^{\xi_j} \frac{t\, dH_j(t)}{H_j(\xi_j)}$$

where

$$H_j(t) = 1 - \prod_{k=1}^{n_j} [1 - G_{j,k}(t)]$$

β_j = sum of slow off-path transition rates

With the use of the above notation, Lee's theorem is easily stated in the following discussion.

Lee's Multiple Recovery Theorem

The probability $D(T)$ of entering a particular deathstate within the mission time T, following a path with k class 1 path steps, m class 2 path steps, and n class 3 path steps, is bounded as follows:

$$\text{LB} \le D(T) \le \text{UB}$$

where

$$\text{UB} = Q(T) \prod_{i=1}^{m} \rho(F_i^*) \prod_{j=1}^{n} \alpha_j \left\{ H_j(\xi_j)\, \mu(\xi_j) + [1 - H_j(\xi_j)] \left(\xi_j + \frac{1}{\alpha_j + \beta_j} \right) \right\}$$

$$\text{LB} = Q(T - \Delta) \prod_{i=1}^{m} [\exp(-\varepsilon_i \xi_i)]\, [\rho(F_i^*) + H_i(\xi_i) - 1]$$

$$\times \prod_{j=1}^{n} \alpha_j \exp[-(\alpha_j + \beta_j)\xi_j]\; \{H_j(\xi_j)\mu(\xi_j) + \xi_j[1 - H_j(\xi_j)]\}$$

$$\Delta = \sum_{i=1}^{m} \xi_i + \sum_{j=1}^{n} \xi_j$$

and

$Q(T)$ = probability of traversing path consisting of only class 1 path steps within time T

Choosing Between White's Method and Lee's Method

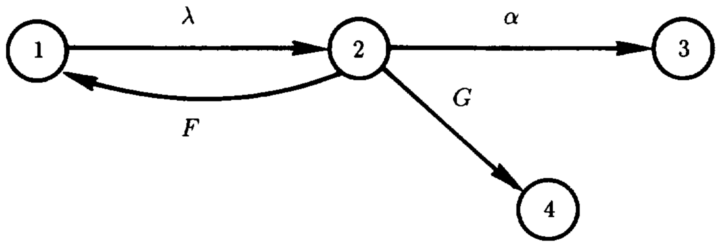
The user of the SURE program is free to use either White's method or Lee's method. In many ways the choice is merely a matter of taste. White's method appears to be more convenient for design studies in which properties of the fast distributions are assumed. Engineering judgment appears to be more skillful at predicting means and variances. Lee's method is especially adapted for the analysis of models for which experimental data are available. This method explicitly takes into consideration the problem of censored data. However, it is clear that either Lee's or White's method could be used for both design studies and experimental analyses.

Transient and Intermittent Models

The mathematical techniques developed by White and Lee do not explicitly accommodate semi-Markov models that are not pure-death processes. The problem with nonpure death process models is that the circuits in the graph structure of the model lead to an infinite sequence of paths of increasing length. Models which include transient or intermittent faults are typically not pure-death processes. The issues involved in using the SURE program to analyze such models are discussed in this section.

Transient Fault Models

Consider the following semi-Markov model (see sketch D) of a system susceptible to transient faults:



Sketch D

The parameter λ is the arrival rate of transient faults in the system. The duration of a transient fault is described by the distribution function $F(t)$, which competes with a system reconfiguration process with distribution $G(t)$. The loop in this model leads to an infinite sequence of paths:

$1 \rightarrow 2 \rightarrow 3$
$1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3$
$1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3$
$1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3$
.
.
.

However, the longer the path the less significant is its contribution to the probability of entering deathstate 3. If we let $P_3^{(k)}(T)$ be the probability of being in state 3 at time T after traversing the loop k times, then using White's theorem gives

$$P_3^{(k)}(T) = \frac{\alpha\mu(H)\ \rho^k(F^*)\ (\lambda T)^{k+1}}{(k+1)!}$$

where $\mu(H)$ is the mean of the recovery holding time in state 2. Then, the probability of being in state 3 at time T (by any path) is

$$P_3(T) = \sum_{k=0}^{\infty} P_3^{(k)}(T) = \left(\frac{\alpha}{\rho}\right) \mu(H)[\exp(\lambda\rho T) - 1]$$

where $\rho = \rho(F^*)$. This infinite series converges to an exponential function. The convergence of this series is very fast for $\lambda\rho T < 1$. Since λ is the rate of a slow transition, this relationship holds. Accurate values can be obtained using only two or three terms of the series. In general, the error in truncating the series after n terms (using Taylor's theorem) is less than

$$\frac{\exp(\lambda\rho T)(\lambda\rho T)^{n+1}}{(n+1)!}$$

The SURE program automatically unfolds a loop into a sequence of paths. The truncation point is user-specifiable via the TRUNC command. If

```
TRUNC = 4
```

then the sequence of paths is terminated after unfolding the loop four times. This is equivalent to truncating the series after the fourth term. The SURE program produces the following warning message when the value of TRUNC is likely to be too small:

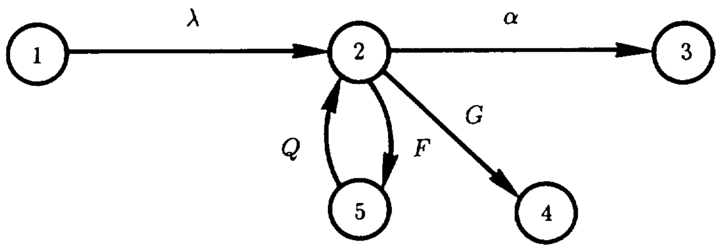
```
.. TRUNC TOO SMALL
```

It is recommended that the user try several values of TRUNC until convergence is certain.

In order to use SURE, the following parameters must be supplied to the program— $\mu(F^*)$, $\sigma(F^*)$, $\rho(F^*)$, $\mu(G^*)$, $\sigma(G^*)$, and $\rho(G^*)$. These parameters must either be calculated from some known distribution or be measured experimentally.

Intermittent Fault Models

Intermittent faults can be modeled in a similar manner. The major difference is that an intermittent fault does not totally disappear. (See sketch E.) Therefore, a benign state (state 5) with holding time $Q(t)$ is introduced in the model to represent the fault while it is not active.



Sketch E

Computationally, however, the problem is different. Since the loop in sketch E contains only fast transitions, the rate of convergence can be very slow. With White's upper bound, the probability of entering state 3 within time T is

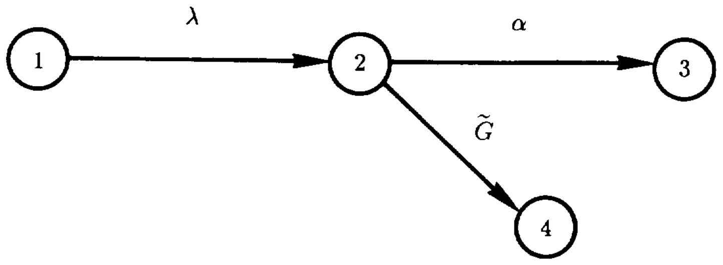
$$P_3(T) \approx \alpha\lambda\mu(H_2)T[1 + \rho(F^*) + \rho^2(F^*) + \rho^3(F^*) + \cdots]$$

For simplicity, suppose that F and G are exponentially distributed, then,

$$\rho(F^*) = \frac{\mu(G^*)}{\mu(F^*) + \mu(G^*)}$$

The percentage error in truncating the series after n terms is easily shown to be $[\rho(F^*)]^{n+1} \times 100\%$. If $\rho(F^*)$ is near 1 (i.e., when $\mu(G^*)$ is large relative to $\mu(F^*)$), then n must be large to get the percentage error acceptably low. For example, if $\mu(F^*) = 10$ and $\mu(G^*) = 1000$, then n (and thus TRUNC) must be equal to 300 in order to have a percentage error of 5 percent.

An alternative approach is recommended when convergence is slow. The model in sketch E can be collapsed into the following model (sketch F):



Sketch F

In this model, states 2 and 5 in sketch E have been aggregated into one state. The new recovery transition $\tilde{G}$ contains the total effect of the intermittent fault. Experimentally, $\tilde{G}$ represents the time to recover in the presence of the intermittent fault. If experimental data are not available, but a parametric form is postulated for the unconditional distributions F, G, and Q then the mean and variance of $\tilde{G}$ can be calculated. For example, if F, G, and Q are exponential, then

$$\mu(\tilde{G}) = [\mu(Q) + \mu(F)]\,\frac{\mu(G)}{\mu(F)}$$

$$\sigma^2(\tilde{G}) = \left[\mu(G) + \frac{\mu(Q)\mu(G)}{\mu(F)}\right]^2 + \frac{2\mu^2(Q)\mu(G)}{\mu(F)}$$

where $\mu(F)$, $\mu(G)$, and $\mu(Q)$ are the unconditional means of the transitions.

Open Issues

In this section, a general proof that convergence will always be obtained for arbitrary models with arbitrary loops has not been given. However, a general proof has not been attempted because of the large number of cases involved. Nevertheless, the SURE program provides a solution technique for models with loops if convergence occurs. The lack of convergence can be observed by increasing the TRUNC constant. A model for which the upper bound does not converge has not yet been found.

Tightness of the SURE Bounds

In this section, an informal argument is given to show why the SURE bounds are typically very close. The SURE program in no way depends on the arguments of this section. The purpose of this discussion is to present a formula for the relative difference between the bounds where the closeness of the bounds can be seen intuitively. Of course, the SURE user need only look at the output of his run to see if the bounds are close for his problem.

The relative difference between the upper bound (UB) and the lower bound (LB) is

$$\frac{\text{UB} - \text{LB}}{\text{UB}} = \frac{Q(T) - Q(T-\Delta)}{Q(T)} \prod_{i=1}^{m} Z_i \prod_{j=1}^{n} Y_j$$

where

$$Z_i = 1 - \varepsilon_i \; \mu(F_1^*) - \frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}$$

$$Y_j = 1 - \frac{(\alpha_j + \beta_j)[\mu^2(H_j) + \sigma^2(H_j)]}{2\mu(H_j)} - \frac{\mu^2(H_j) + \sigma^2(H_j)}{s_j \; \mu(H_j)}$$

Since the fault arrivals ε_i, α_j, and β_j are small (e.g., 10^{-4}/hr) and the means and the standard deviations $\mu(H_j)$ and $\sigma(H_j)$ are small (e.g., 10^{-4} hr),

$$Z_i \approx 1$$

$$Y_j \approx 1$$

Thus,

$$\frac{\text{UB} - \text{LB}}{\text{UB}} \approx \frac{Q(T) - Q(T - \Delta)}{Q(T)}$$

Using the algebraic upper bound on $Q(T)$ and $Q(T - \Delta)$ gives

$$\frac{\text{UB} - \text{LB}}{\text{UB}} \approx 1 - \left[1 - \frac{\Delta}{T}\right]^k$$

$$\approx \frac{k\Delta}{T}$$

For recovery times on the order of 10^{-4} hour, the parameters r_i and s_j are on the order of 10^{-2} hour. Using fairly large values of Δ and k (the number of class 1 path steps in the path), namely, $\Delta = 10^{-1}$ and $k = 5$,

$$\frac{\text{UB} - \text{LB}}{\text{UB}} \approx \frac{k\Delta}{T} = \frac{0.5}{10} = 5\%$$

For smaller values of k and Δ, the relative error is smaller. From the above expressions for the relative error, it is obvious that as the failure rates or the recovery times increase, the bounds separate.

SURE User Interface

Basic Program Concept

The user of the SURE program must describe his semi-Markov model to the SURE program with a simple language for enumerating all the transitions of the model. The SURE user must first assign numbers to every state in the system. The semi-Markov model is then described by enumerating all the transitions. As described in the previous sections, each transition is classified as being either slow or fast. Consequently, there are two different statements used to enter transitions—one for slow transitions and the other for fast. If a transition is slow, then the following type of statement is used:

```
1,2 = 0.0001;
```

This defines a slow exponential transition from state 1 to state 2 with rate 0.0001. The program does not require any particular units, for example, hour^{-1} or sec^{-1}. However, the user must use consistent units. If the transition is fast, then either of two methods can be used to describe the transition. These methods correspond to White's and Lee's methods discussed previously. The following specifies a fast transition using White's method:

```
2,4 = < 1E-4, 1E-6, 1.0 >;
```

The numbers in the brackets (< >) correspond to the conditional mean, conditional standard deviation, and transition probability of the fast transition, respectively. Using Lee's method the same transition would be specified as

```
@2 = < 1E-4, 1E-3, 0.99 >;
2,4 = < 1.0 >;
```

The numbers in the brackets on the first line describe the holding time in state 2. The first number is the conditional mean. The next two numbers define a quantile of the recovery holding time distribution; that is, the probability that the recovery holding time is less than 1E-3 is 0.99. The number in the brackets on the second line is the probability that the transition from state 2 to state 4 succeeds over other competing fast transitions. Since there are no other competing transitions, this probability is 1.

Although the transition description statements described above are the key constructs of the SURE language, the flexibility of the SURE program has been increased by adding several features commonly seen in programming languages such as FORTRAN or Pascal. In the next section, the SURE input language is described in detail.

The SURE Input Language

The SURE input language includes two types of statements—model-definition statements and commands. These are described in detail in the next sections.

Model-Definition Syntax

Models are defined in SURE by enumerating all the transitions of the model.

Lexical details. The state numbers must be positive integers between 0 and the MAXSTATE implementation limit, usually 25 000. (This limit can be changed by redefining a constant in the SURE program and recompiling the SURE source.) The transition rates, conditional means and standard deviations, etc., are floating point numbers. The Pascal REAL syntax is used for these numbers. Thus, all the following would be accepted by the SURE program:

0.001
12.34
1.2E-4
1E-5

The semicolon is used for statement termination. Therefore, more than one statement may be entered on a line. Comments may be included any place that blanks are allowed. The notation "(*" indicates the beginning of a comment and "*)" indicates the termination of a comment. The following is an example of the use of a comment:

```
LAMBDA = 5.7E-4;    (* FAILURE RATE OF A PROCESSOR *)
```

If statements are entered from a terminal (instead of by the READ command described below), then the carriage return is interpreted as a semicolon. Thus, interactive statements do not have to be terminated by an explicit semicolon unless more than one statement is entered on the line.

The SURE program prompts the user for input by a line number followed by a question mark. For example,

```
1?
```

The number is a count of the syntactically correct lines entered into the system thus far plus the current one.

Constant definitions. The user may equate numbers to identifiers. Thereafter, these constant identifiers may be used instead of the numbers. For example,

```
LAMBDA = 0.0052;
RECOVER = 0.005;
```

Constants may also be defined in terms of previously defined constants:

```
GAMMA = 10*LAMBDA;
```

In general, the syntax is

```
"name" = "expression";
```

where `"name"` is a string of up to eight letters, digits, and underscores (_) beginning with a letter, and `"expression"` is an arbitrary mathematical expression as described in a subsequent section entitled "Expressions."

Variable definition. In order to facilitate parametric analyses, a single variable may be defined. A range is given for this variable. The SURE system computes the system reliability as a function of this variable. If the system is installed with the graphics module (to be described later), then a plot of this function can be obtained using the PLOT command. The following statement defines LAMBDA as a variable with range 0.001 to 0.009:

```
LAMBDA = 0.001 TO 0.009;
```

Only one such variable may be defined. A special constant, POINTS, defines the number of points over this range to be computed. The method used to vary the variable over this range can be either geometric or arithmetic and is best explained by example. Thus, suppose `POINTS = 4`, then

Geometric:

```
XV = 1 TO* 1000;
```

where the values of XV used would be 1, 10, 100, and 1000.

Arithmetic:

```
XV = 1 TO+ 1000;
```

where the values of XV used would be 1, 333, 667, and 1000.

*The * following the TO implies a geometric range. A TO+ or simply TO implies an arithmetic range.*

One additional option is available—the BY option. By following the above syntax with `BY "increment"`, the value of POINTS is automatically set such that the value is varied by adding or multiplying the specified amount. For example,

```
V = 1E-6 TO* 1E-2 BY 10;
```

sets POINTS equal to 5 and the values of V used would be 1E-6, 1E-5, 1E-4, 1E-3, and 1E-2. The statement

```
Q = 3 TO+ 5 BY 1;
```

sets POINTS equal to 3, and the values of Q used would be 3, 4, and 5.

In general, the syntax is

```
"var" = "expression" TO {"c"} "expression" { BY "increment" }
```

where "var" is a string of up to eight letters and digits beginning with a letter, "expression" is an arbitrary mathematical expression as described in the next section, and the optional "c" is a + or *. The BY clause is optional; if it is used, then "increment" is any arbitrary expression.

Expressions. When specifying transition or holding time parameters in a statement, arbitrary functions of the constants and the variable may be used. The following operators may be used:

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

The following standard functions may be used:

EXP(X)	exponential function
LN(X)	natural logarithm
SIN(X)	sine function
COS(X)	cosine function
ARCSIN(X)	arc sine function
ARCCOS(X)	arc cosine function
ARCTAN(X)	arc tangent function
SQRT(X)	square root

Both ( ) and [ ] may be used for grouping in the expressions. The following are permissible expressions:

```
2E-4
1.2*EXP(-3*ALPHA);
7*ALPHA + 12*LAMBDA;
ALPHA*(1+LAMBDA) + ALPHA**2;
2*LAMBDA + (1/ALPHA)*[LAMBDA + (1/ALPHA)];
```

Slow transition description. A slow transition is completely specified by citing the source state, the destination state, and the transition rate. The syntax is as follows:

```
"source", "dest" = "rate";
```

where "source" is the source state, "dest" is the destination state, and "rate" is any valid expression defining the exponential rate of the transition. The following are valid SURE statements:

```
PERM = 1E-4;
TRANSIENT = 10*PERM;

1,2 = 5*PERM;
1,9 = 5*(TRANSIENT + PERM);
2,3 = 1E-6;
```

In the notation of the previous section we have

```
i,j = $\lambda_i$;
```

Fast transition description. To enter a fast transition, the SURE user may use either of two methods—White's method or Lee's method—described in this section.

White's method: The following syntax is used for White's method.

```
"source" , "dest"  = < "mu", "sig" {, "frac" } >;
```

where

`"mu"`	= expression defining conditional mean transition time, $\mu(F^*)$
`"sig"`	= expression defining conditional standard deviation of transition time, $\sigma(F^*)$
`"frac"`	= expression defining transition probability, $\rho(F^*)$

and `"source"` and `"dest"` define the source and destination states, respectively. The third parameter `"frac"` is optional. If omitted, the transition probability is assumed to be 1.0, that is, only one fast transition. *In the notation of the previous section, we have*

```
i,j = < $\mu(F_i^*)$, $\sigma(F_i^*)$, $\rho(F_i^*)$ >;
```

All the following are valid (while in White's mode):

```
2,5 = <1E-5, 1E-6, 0.9>;

THETA = 1E-4;
5,7 = <THETA, THETA*THETA, 0.5>;
7,9 = <0.0001,THETA/25>;
```

Lee's method: To describe a fast transition using Lee's method, the following syntax is used:

```
"source" , "dest" =  < "frac" >;
@ "source"  =  < "hmu", "quant", "prob" >;
```

where

`"source"`	= source state
`"dest"`	= destination state
`"frac"`	= expression defining transition probability, $\rho(F^*)$
`"hmu"`	= expression defining conditional mean recovery holding time $\mu(\xi)$, given that holding time is less than `"quant"`
`"quant"`	= expression defining percentile or censoring point ξ
`"prob"`	= expression defining probability $H(\xi)$ that holding time in state is less than `"quant"`

In the notation of the previous section we have

```
j,k = < $\rho(F_j^*)$ >;
@j = < $\mu(\xi_j)$, $\xi_j$, $H_j(\xi_j)$ >;
```

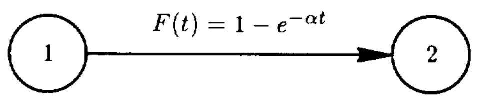
All the following are valid SURE statements (while in the Lee mode):

```
5,6 = <0.5>;
FRACT = 0.0 TO 0.5;
5,7 = <FRACT>;
5,8 = <0.5 - FRACT>;
@5 = < 0.00034, 0.003, (1.0-1E-4) >;
```

Although there may be many fast transitions from a state, the @ "source" statement should be issued only once for the state.

The SURE user must decide which method he will use before entering his model. Either the Lee method or White method may be used to describe the model, but both cannot be used at the same time. By default, the program assumes that the White method will be used. If Lee's method is desired, the LEE command must be issued prior to entering any fast transition.

FAST exponential transition description. Often when performing design studies, experimental data are unavailable for the fast processes of a system. In this case, one must assume some properties of the underlying processes. For simplicity, these fast transitions are often assumed to be exponentially distributed. However, it is still necessary to supply the conditional mean and standard deviation to the SURE program since they are fast transitions. If there is only one fast transition from a state, then these parameters are easy to determine. Suppose we have a fast exponential recovery from state 1 to state 2 with unconditional rate α:



The SURE input is simply

```
1,2 = < 1/α, 1/α, 1 >;
```

In this case, the conditional mean and standard deviation are equivalent to the unconditional mean and standard deviation. The above transition can be specified by using the following syntax:

```
1,2 = FAST α;
```

When multiple recoveries are present from a single state, then care must be exercised to properly specify the conditional means and standard deviations required by the SURE program. Suppose we have the model in figure 3, where the unconditional distributions are

$$F_1(t) = 1 - e^{-\alpha t}$$
$$F_2(t) = 1 - e^{-\beta t}$$
$$F_3(t) = 1 - e^{-\gamma t}$$

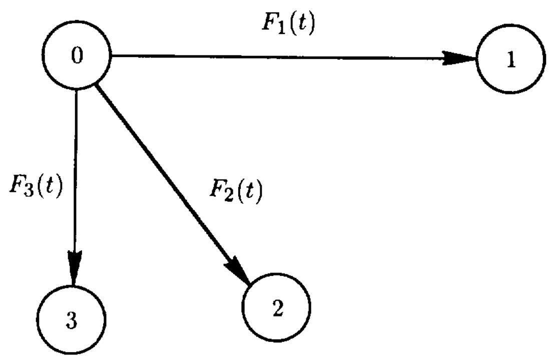


Figure 3. Model of three competing fast transitions.

The SURE input describing the model section in figure 3 is

```
0,1 = < 1/(α+β+γ), 1/(α+β+γ), α/(α+β+γ) >;
0,2 = < 1/(α+β+γ), 1/(α+β+γ), β/(α+β+γ) >;
0,3 = < 1/(α+β+γ), 1/(α+β+γ), γ/(α+β+γ) >;
```

Note that the conditional means and standard deviations are not equal to the unconditional means and standard deviations (e.g., the conditional mean transition time from state 0 to 1 is not equal to $1/\alpha$). The following can be used to define the model of figure 3:

```
0,1 = FAST α;
0,2 = FAST β;
0,3 = FAST γ;
```

The SURE program automatically calculates the conditional parameters from the unconditional rates α, β, and γ. *The FAST exponential capability can only be used in conjunction with the WHITE method of specifying recovery transitions.* The user may mix FAST exponential transitions with other general transitions. However, care must be exercised in specifying the conditional parameters of the nonexponential fast recoveries in order to avoid inconsistencies. Appendix A discusses this problem and gives the details on the formulas used by the SURE program to compute the conditional parameters for fast exponentials. Potential users of the FAST exponential capability should read appendix B.

SURE Commands

Two types of commands have been included in the user interface. The first type of command is initiated by one of the following reserved words:

EXIT	READ	INPUT	LEE	RUN	SHOW	IF
CALC	ORPROB	DISP	SAVE	GET	PLOT	

The second type of command is invoked by setting one of the following special constants:

AUTOFAST	ECHO	LIST	POINTS	PRUNE	QTCALC	START
TIME	TRUNC	WARNDIG				

equal to one of its predefined values.

EXIT command. The EXIT command causes termination of the SURE program.

READ command. A sequence of SURE statements may be read from a disk file. The following interactive command reads SURE statements from a disk file named SIFT.MOD:

```
READ SIFT.MOD;
```

If no file name extent is given, the default extent `.MOD` is assumed. A user can build a model description file by using a text editor and use this command to read it into the SURE program.

INPUT command. This command increases the flexibility of the READ command. Within the model description file created with a text editor, INPUT commands can be inserted that will prompt for values of specified constants while the model file is being processed by the READ command. For example, the command

```
INPUT LVAL;
```

will prompt the user for a number as follows:

```
LVAL?
```

and a new constant LVAL is created that is equal to the value input by the user. Several constants can be interactively defined using one statement as in the following example:

```
INPUT X, Y, Z;
```

LEE command. The LEE command prepares the program to receive fast transition commands according to Lee's syntax. By default, the program expects fast transitions to be described in White's format. The syntax of the LEE command is

```
LEE;
```

The LEE command must be issued prior to entering any fast transitions. The FAST exponential syntax cannot be used in LEE mode.

RUN command. After a semi-Markov model has been fully described to the SURE program, the RUN command is used to initiate the computation:

```
RUN;
```

The output is displayed on the terminal according to the LIST option specified. If the user wants the output written to a disk file instead, the following syntax is used:

```
RUN "outname";
```

where the output file "outname" may be any permissible VAX VMS file name. Two positional parameters are available on the RUN command. These parameters enable the user to change the value of the special constants POINTS and LIST in the RUN command. For example,

```
RUN (30,2) OUTFILE.DAT
```

is equivalent to the following sequence of commands:

```
POINTS = 30;
LIST = 2;
RUN OUTFILE.DAT
```

Each parameter is optional so the following are acceptable:

```
RUN(10);       (* Change POINTS to 10 then run *)
RUN(,3);       (* Change LIST to 3 and run *)
RUN(20,2);     (* Change POINTS to 20 and LIST to 2 then run *)
```

After a run is completed, the SURE program clears all the transition, constant, and variable definitions, returning the program state to its original state. However, throughout the session, the output of each RUN is stored internally. The results of prior RUN commands are available in special variables which can be referenced in future model descriptions or in a CALC command. The syntax is as follows:

```
#L1      lowerbound for RUN #1 (no variable)
#U2      upperbound for RUN #2 (no variable)
#1       upperbound for RUN #1 (no variable)

#L1[3]   lowerbound for third value of variable on run #1
#U2[1]   upperbound for first value of variable on run #2
```

SHOW command. The value of a constant or variable may be displayed by the following command:

```
SHOW ALPHA;
```

Information about a transition may also be displayed by the SHOW command. For example, information concerning the transition from state 654 to state 193 is displayed by the following

```
SHOW 654-193;
```

If the model is described with Lee's method, the information about a state holding time may be displayed. For example, state 12 holding time characteristics are listed in response to

```
SHOW 12;
```

More than one constant, variable, holding time, or transition may be shown at one time:

```
SHOW ALPHA, 12-13, BETA, 123;
```

IF command. The IF statement provides a "conditional assembly" capability to the SURE program. The statement following the THEN reserved word is only processed if the preceding Boolean expression is true. The syntax of this statement is:

```
IF "expression" "bool-op" "expression" THEN "statement";
```

where

```
"bool-op" is one of the following operators: =  <  <=  >  >=
```

The following session illustrates this command:

```
$ SURE

   1? X = 1; Y = 2;
   2? IF X = 1 THEN Y = 3;
      Y       CHANGED TO 3.00000E+00
   3? SHOW Y;
      Y        = 3.00000E+00
   4? IF Y > X THEN 1,2 = 1E-4;
   5? SHOW 1-2;
      TRANSITION 1 →  2: EXPONENTIAL RATE = 1.00000E-4;
   6? IF X < 0 THEN 2,3 = 1E-3;
   7? SHOW 2-3
      TRANSITION 2 →  3 NOT FOUND
   8? EXIT
```

CALC command. For convenience, a calculator function has been included. This command allows the user to obtain the value of an arbitrary expression. For example, if the following commands are entered:

```
X = 1.6E-1;
CALC  (X-.12)*EXP(-0.001) + X**3;
```

the system responds with

```
= 4.405601999335E-02
```

If a variable has been defined prior to issuing the CALC function, the expression is computed as a function of the variable over the specified range. The PLOT command can be used after the CALC command to obtain a plot of the function. This feature is illustrated in example 10 of the section entitled "Examples." The output can be sent to a disk file instead of the terminal by using the following syntax:

```
CALC "expression" TO "filename";
```

where `"filename"` is the name of the destination file.

ORPROB command. A common complaint about the Markov approach to modeling is the rapid growth in state space size as the complexity of a system is increased. For large, complex interdependent systems, this is often unavoidable. But, systems which consist of several isolated subsystems can be analyzed easily by using the additive law of probability.

Suppose the probabilities that subsystem 1 and subsystem 2 fail within the mission time are P_1 and P_2, respectively. If these subsystems fail independently, the probability of system failure P_{sys} can be calculated as follows:

$$P_{sys} = P_1 + P_2 - (P_1)(P_2)$$

If there are failure dependencies between the subsystems, then a single model must be used.

The ORPROB command lists all the previous run output results and then computes the probabilistic OR of the previous runs. See example 8 in the section entitled "Examples." The PLOT command may be used to plot the results of the ORPROB command. If the variable feature of SURE is used and LIST = 1, then the ORPROB command does not list out the answers from the previous runs. Only the probabilistic OR for each value of the variable is given. If LIST = 2 is set prior to issuing ORPROB, then a detailed list of all the outputs from the previous runs, along with the probabilistic OR of the runs for each value of the variable, is given.

AUTOFAST constant. If the special constant AUTOFAST is set equal to 1, then the reserved word FAST does not have to be used before a rate expression to indicate that the transition is fast. The program automatically decides if the rate is fast with respect to the mission time. If the product of the transition rate and the mission time (TIME) is greater than 100, then the transition is treated as FAST and the conditional means and standard deviations are automatically calculated just as if FAST had been explicitly specified. Otherwise, the transition is treated as a slow transition. The default value of AUTOFAST is 0 which implies *no* automatic conversion to FAST.

ECHO constant. The ECHO constant can be used to turn off the echo when reading a disk file. The default value of ECHO is 1, which causes the model description to be listed as it is read. (See example 3 in the section entitled "Example SURE Sessions.")

LIST constant. The amount of information output by the program is controlled by this command. Four list modes are available as follows:

`LIST = 0;` No output is sent to the terminal, but the results can still be displayed using the PLOT command

`LIST = 1;` Only the upper and lower bounds on the probability of total system failure are listed; this is the default

`LIST = 2;` The probability bounds for each deathstate in the model are reported along with the totals

`LIST = 3;` Every path in the model is listed and its probability of traversal; the probability bounds for each deathstate in the model are reported along with the totals

If a variable is defined and `LIST=1` is specified, then the summary statistics are only given for the value of the variable for which the bounds had the worst accuracy. (See example 12 in the section entitled "Examples.") If `LIST >= 2` then the summary statistics are given for each value of the variable.

POINTS constant. The POINTS constant specifies the number of points to be calculated over the range of the variable. The default value is 25. If no variable is defined, then this specification is ignored.

QTCALC constant. The value of the QTCALC constant determines the numerical method used to compute $Q(T)$—the probability of traversing the class 1 transitions within time T. If QTCALC = 0, the program uses White's algebraic formulas for $Q(T)$. If QTCALC = 1, the program uses an exponential matrix solver to calculate $Q(T)$ rather than the algebraic approximations. This method is slower but is much more accurate when the mission time is long. The mathematical basis of the matrix exponential algorithm is described in appendix C. The default value of QTCALC is 2, which specifies that the program should automatically select the appropriate $Q(T)$ algorithm on a path-by-path basis. The following rule is used by the program when QTCALC equals 2:

```
IF $(T - \Delta) \sum_{i=1}^{k} (\lambda_i + \gamma_i) \; / \; (k+1) < .1$ THEN
        the algebraic formula (QTCALC=0) is used
ELSE
        the matrix exponential solver (QTCALC=1) is used
```

The SURE program indicates when the exponential matrix solver is used by writing **<ExpMat>** in the comments field of the output. (See example 4.) The program also writes the following statement in the summary statistics:

```
Q(T) ACCURACY >= x DIGITS
```

If the accuracy of the numerical method is less than seven digits or LIST is greater than 2, the following is written into the comments field:

```
<ExpMat - x,y>
```

where x is the number of digits accuracy in the lower bound and y is the number of digits accuracy in the upper bound.

PRUNE and WARNDIG constant. The time required to analyze a large model can often be greatly reduced by model pruning. It is essential that this be done carefully in order to maintain accuracy. The SURE user specifies the level of pruning desired using the PRUNE constant. A path is traversed by the SURE program until the probability of reaching the current point on the path falls below the pruning level. For example, if PRUNE = 1E-14 and the upper bound falls below 1E-14 at any point on the path, the analysis of the path is terminated and its contribution to the deathstate probabilities is not included in the final results. The sum of all the occupancy probabilities of the pruned states is given in the following format:

```
SUM OF PRUNED STATES PROBABILITY < x
```

Clearly, the probability of reaching a deathstate by continuing along this path must be less than this sum. The error resulting from this pruning method is therefore less than this sum. The SURE program will warn the user if the pruning process resulted in an upper bound with less than WARNDIG digits of accuracy. In other words, the warning message

```
PRUNING TOO SEVERE
```

is given if

```
SUM OF PRUNED STATES PROBABILITY > (Pf / 10^WARNDIG)
```

where P_f is the upper bound on system failure. This warning message is very conservative. Typically, the accuracy is far greater than is guaranteed by this test. The default value of PRUNE is 0.0. The default value of WARNDIG is 2.

For very large models, it is recommended that the user start with a very large value of PRUNE (e.g., 1E-10) and decrease the value (e.g., to 1E-15) until the message PRUNING TOO SEVERE disappears.

START constant. The START constant is used to specify the start state of the model. If the START constant is not used, the program will use the source state (i.e., the state with no transitions into it) of the model (if one exists). If there is no source state in the model, the program will use the first state entered as the start state. If no start state is specified and there are two or more source states, an error message is issued. The program arbitrarily chooses one of the source states as the start state and proceeds.

TIME constant. The TIME constant specifies the mission time. For example, if the user sets TIME = 1.3, the program computes the probability of entering the deathstates of the model within time 1.3. The default value of TIME is 10. All parameter values must be in the same units as the TIME constant.

TRUNC and WARNDIG constant. The TRUNC constant sets the number of times the program will unfold a loop in the graph structure of the model. The default value is 3. The SURE program issues the following warning:

```
TRUNC TOO SMALL
```

when it detects that the truncation error could lead to less than WARNDIG accuracy. The default value of WARNDIG is 2. Also, whenever the accuracy is less than seven digits accuracy, the following statement is written in the summary statistics:

```
ACCURACY MAY BE LESS THAN 6 DIGITS DUE TO LOOP TRUNCATION
```

The issues involved in the analysis of a model with loops are discussed in the section entitled "Transient and Intermittent Models."

SURE Graphics

Although the SURE program is easily used without graphics output, many users desire the increased user friendliness of the tool when assisted by graphics. The Langley AIRLAB contains four color graphics monitors (and TEMPLATE support software) enabling the full utilization of the graphics capability of SURE. However, the version of SURE available from COSMIC does not contain the graphics software. The SURE program can plot the probability of system failure as a function of any model parameter as well as display the semi-Markov models in a graphical form. The output from several SURE runs can be displayed together in the form of contour plots. Thus, the effect on system reliability of two model parameters can be illustrated on one plot. The generation of a graphical picture of the semi-Markov model can be directed by user input or left completely to the SURE program.

Plotting Results of SURE Runs

After a RUN, CALC, or ORPROB command, the PLOT command can be used to plot the output on the graphics display. The syntax is

```
PLOT <op>, <op>, ... <op>
```

where <op> are plot options. Any TEMPLATE "USET" or "UPSET" parameter can be used, but the following are the most useful:

`XLOG`	plot X-axis using logarithmic scale
`YLOG`	plot Y-axis using logarithmic scale
`XYLOG`	plot both X- and Y-axes using logarithmic scales
`NOLO`	plot X- and Y-axes with normal scaling
`XLEN=5.0`	set X-axis length to 5.0 in.
`YLEN=8.0`	set Y-axis length to 8.0 in.
`XMIN=2.0`	set x-origin 2 in. from left side of screen
`YMIN=2.0`	set y-origin 2 in. above bottom of screen

The PLOTINIT and PLOT+ commands are used to display multiple runs on one plot. A single run of SURE generates unreliability as a function of a single variable. To see the effect of a second variable (i.e., display contours of a three-dimensional surface) the PLOT+ command is used. The PLOTINIT command should be called before performing the first SURE run. This command defines the 2d variable (i.e., the contour variable):

```
PLOTINIT BETA;
```

This defines BETA as the 2d independent variable. Next, the user must set BETA to its first value. After the run is complete, the output is plotted by using the PLOT+ command. The parameters of this command are identical to the PLOT command. The only difference is that the results are saved, so they can be displayed in conjunction with subsequent results. Next, BETA must be set to a second value, another SURE run made, and PLOT+ must be called again. This time both outputs are displayed together. Up to 10 such runs can be displayed together.

Graphical Display of Models

In order to obtain a graphical display of the semi-Markov model being processed, the user must issue the DISP command

```
DISP;
```

prior to entering any transition commands. This command causes the system to prompt for the state locations while the model is being defined. The user indicates by joystick input where each state of the model should be located. The system automatically pans as the model exceeds the current scope of the screen. Once the user indicates where each state should be placed, the program automatically draws all the transitions and labels them. The DISP command is more fully explained in the following section. The user may store the state location information on disk by using the SAVE command. For example, the current state location information is written to file SIFT.MEG by the following command:

```
SAVE SIFT
```

State location information may be retrieved from a disk file by using the GET command. If state location has been stored on disk file FTMP.MEG in a prior SURE session, then the following command will retrieve this information:

```
GET FTMP
```

An abbreviation can be used if the location information is on a file with the same VMS file name (except the extent) as the command file that describes the model. For example, the commands `GET TRIPLEX.MEG; READ TRIPLEX.MOD` may be abbreviated as

```
READ TRIPLEX*;
```

The extent names must be .MOD for the file containing the model commands and .MEG for the file containing the state locations on the graphics display in order for this abbreviation technique to work.

The SCAN and ZOOM commands may be used to peruse the model. The joystick button is used to end the ZOOM and SCAN commands. Each of the special graphics commands is described in the subsequent sections.

DISP command. The DISP command initializes the model display capability of the SURE program. After this command is issued, the SURE program displays every transition it processes on the graphics device. The states of the model are represented by circles containing the number of the state. The transitions are represented by lines connecting the states. (See example 2 of the section entitled "Examples.") The determination of the best place to locate a state in the model (i.e., where to put the node of the graph) is a difficult problem (even for a human). A simplistic heurism is included in the SURE program to aid the user in positioning a state with the "wand joystick." This heurism can be utilized in two different ways—fully automatic or manual. In the fully automatic mode, the program places the state without prompting the user for joystick input. However, for complex models the picture is often quite ugly, with transition lines crossing in many places. In the manual mode the program selects a position and sets the cross hairs at that location. If the user likes the location, he need only press the wand button. Otherwise, the position can be changed with the joystick prior to hitting the button. If fully automatic state location is desired, the user issues the following command:

```
DISP*
```

If the manual mode is desired the command

```
DISP
```

is used.

The length of the transition selected by the heurism can be specified with parameters on the DISP command. By default the length in both the x- and y-directions is set at 2 in. If the default value is not desired, the lengths can be changed as shown

```
DISP 2.5, 5.6
```

This sets the x-length to 2.5 in. and the y-length to 5.6 in.

Finally the DISP command can be used to generate a hard copy of the screen on the plotter via the following syntax:

```
DISP COPY
```

GET and SAVE commands. Once the locations of the states have been established by using either the manual joystick input method or the automatic heuristic method, this information can be saved on a file with the SAVE command. The syntax is simply

```
SAVE "filename"
```

where "`filename`" is in VMS file syntax. In future sessions this information can be retrieved with the GET command:

```
GET "filename"
```

If no VMS `filename` extent is given, the program assumes it to be .MEG by default. The format of the file is simple and can be edited using a text editor if desired. The format is three columns of numbers, with each row defining a particular state location. The first column contains the state numbers, the second column contains the x-coordinates, and the third column contains the y-coordinates, for example:

```
30    1.250000          118.7500
31    4.250000          118.7500
32    7.250000          118.7500
20    4.250000          115.7500
21    7.250000          115.7500
```

If a row is deleted by the editor and if this file is used in a later session (i.e., using the GET command), only the deleted state location will have to be entered via the joystick.

CLEAR command. The CLEAR command erases all transitions and state locations from internal memory. However, the CLEAR* erases only the state locations specified as parameters plus all the transitions. For example,

```
CLEAR* 3,7
```

erases all the transitions but retains all state locations except 3 and 7. The user can then reissue the READ* (or DISP; READ) command and the program will only prompt for states 3 and 7. All the other states are located in the same place they were in the previous display.

ZOOM and SCAN commands. The SCAN command causes the graphics view to pan across the model. The direction of the pan is in the direction the joystick is turned. When the final position is selected, the wand button can be pressed to terminate the pan.

The ZOOM command causes the graphics display to "zoom in" or "zoom away" from the model. If the wand is pushed forward, the zoom is inward; if the wand is pulled backward the zoom is away from the model. This process is also terminated by pressing the wand button. At this time the program asks if a hard copy on the plotter is desired:

```
HARD COPY?  (YES=1, NO=0)
```

After this the user is asked to select a new center point around which the program will reexpand the model to its normal size. This is accomplished by using the joystick and wand button as in the scan mode.

SCREEN constant. The size of the display screen can be specified with the SCREEN constant. The default size is 10 by 10 in. The display area is always square; however, the size of the square can be changed. For example, if a 6-in. screen is desired the following command should be issued prior to the DISP command:

```
SCREEN = 6;
```

GREEK constant. The GREEK constant specifies whether constants with greek names such as LAMBDA, GAMMA, PHI, or RHO should be displayed as greek characters on the display monitor (e.g., as λ, γ, etc.). IF GREEK = 1 then this translation process is performed. If GREEK = 0 then this translation is not done. The default setting is GREEK = 1. Sometimes it is desired to display the model without the transitions labeled at all. This can be accomplished by setting GREEK = −1.

Example SURE Sessions

Outline of a Typical Session

The SURE program was designed for interactive use. The following method of use is recommended (see example 2):

1. Create a file of SURE commands using a text editor describing the semi-Markov model to be analyzed.

2. Start the SURE program and use the READ command to retrieve the model information from this file.
3. Then, various commands may be used to change the values of the special constants, such as LIST, POINTS, QTCALC, and TRUNC, as desired. Altering the value of a constant identifier does not affect any transitions entered previously even though they were defined with a different value for the constant. The range of the variable may be changed after transitions are entered.
4. Enter the RUN command to initiate the computation.

Examples

The following examples illustrate interactive SURE sessions. For clarity, all user inputs are given in lowercase letters.

Example 1

This session illustrates direct interactive input and the type of error messages given by SURE:

```
$ sure

  SURE V5.2     NASA Langley Research Center

  1? lambda = 1e-5;
  2? 1,2 = 6*lambda;
  3? 2,3 = 5*lamba;
                 ^ IDENTIFIER NOT DEFINED
  3? 2,3 = 5*lambda;
  4? show 2-3;
     TRANSITION 2 -> 3: RATE = 5.00000E-5
  5? 2,4 = <1e-4,1e-5>;
  6? 4,5 = 2*lambda;
  7? list = 2;
  8? time = 1;
  9? run

 DEATHSTATE      LOWERBOUND    UPPERBOUND     COMMENTS       RUN #1
 ----------      -----------   -----------    ----------------------
          3      2.93992E-13   3.00000E-13
          5      5.95908E-10   6.00000E-10

 TOTAL           5.96202E-10   6.00300E-10

 *** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN
 2 PATH(S) PROCESSED
 0.100 SECS. CPU TIME UTILIZED

 10? exit
```

The warning message indicates that a syntax error was encountered by the program. If a user receives this message, he should check his input file to make sure that the model description is correct. In this example, since the syntax error was corrected in the next line, the model was correct. A complete list of program-generated error messages is given in appendix D.

Since LIST = 2, upper and lower bounds are given for each deathstate as well as the total. The mission time is set to 1 in statement 8. If this statement was omitted, the program would use 10 by default.

Example 2

The following session indicates the normal method of using SURE. Prior to this session, a text editor has been used to build file TRIADP1.MOD. This file contains a description of a triad system with one spare. The system uses threefold redundancy to mask single processor faults. If a spare is available the system replaces a faulty processor with the spare. If no spare is available the system degrades to a simplex. For simplicity the means and standard deviations of both types of recovery are assumed to be the same—RECOVER and STDEV, respectively. The program displays the contents of the files as it is read (with the READ command). Input lines, which are read, are labeled with a line number followed by a colon. The file TRIADP1.MEG was created by the SAVE command in a previous session.

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? read triadp1*;

  2: LAMBDA = 1E-6 TO* 1E-2;
  3: RECOVER = 2.7E-4;
  4: STDEV = 1.3E-3;
  5: 1,2 = 3*LAMBDA;
  6: 2,3 = 2*LAMBDA;
  7: 2,4 = <RECOVER,STDEV>;
  8: 4,5 = 3*LAMBDA;
  9: 5,6 = 2*LAMBDA;
 10: 5,7 = <RECOVER,STDEV>;
 11: 7,8 = LAMBDA;
 12: POINTS = 10;
 13: TIME = 6;

 14? run

     LAMBDA       LOWERBOUND     UPPERBOUND     COMMENTS      RUN #1
  -----------   -----------   -----------   ---------------------
  1.00000E-06   9.40296E-15   1.00441E-14
  2.78256E-06   7.71327E-14   8.22407E-14
  7.74264E-06   6.90469E-13   7.33127E-13
  2.15444E-05   7.35487E-12   7.75250E-12
  5.99484E-05   1.00201E-10   1.04754E-10
  1.66810E-04   1.70631E-09   1.77475E-09
  4.64159E-04   3.31737E-08   3.45029E-08
  1.29155E-03   6.81859E-07   7.14440E-07
  3.59382E-03   1.41321E-05   1.51683E-05
  1.00000E-02   2.83744E-04   2.92932E-04   <ExpMat>

  3 PATH(S) PROCESSED
  Q(T) ACCURACY >= 14 DIGITS
  0.400 SECS. CPU TIME UTILIZED

15? plot ylog
16? exit
```

Figure 4 illustrates the model displayed on the output graphics device (defined in file TRIADP1.MEG). The plot in figure 5 was generated from this run by the "`plot ylog`" command. The `<ExpMat>` comment indicates that the exponential matrix algorithm was used to calculate $Q(T)$ for LAMBDA = 1E-2. The accuracy of this calculation was 14 digits.

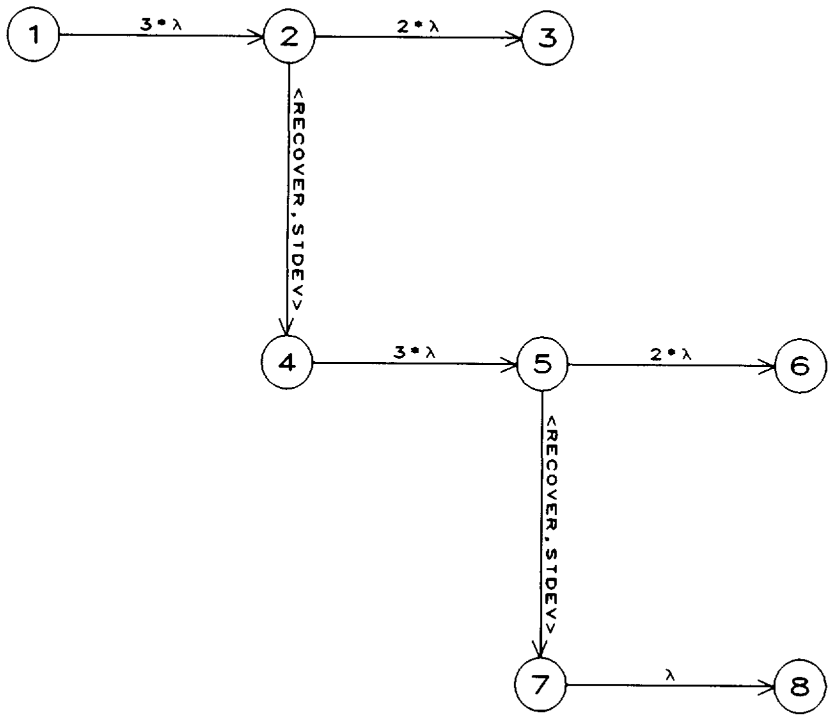


Figure 4. Semi-Markov model from example 2.

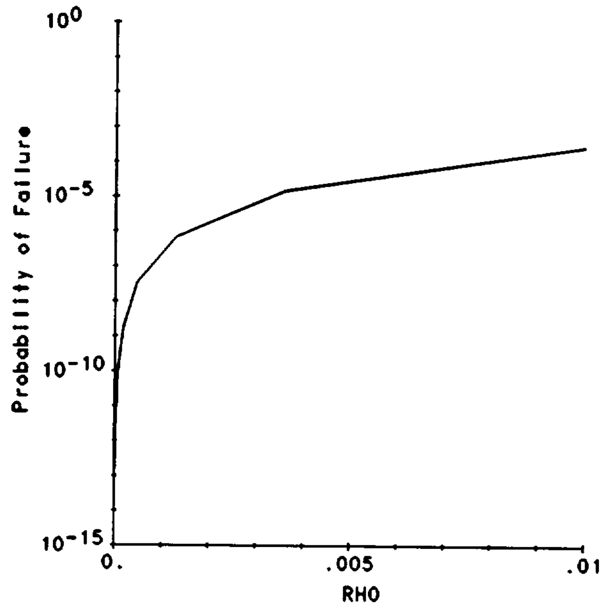


Figure 5. SURE output from example 2.

Example 3

The following interactive session illustrates the use of the ECHO constant. This constant is used when the model description file is large and one desires that the model input not be listed on the terminal as it is read by the SURE program.

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? echo = 0;
  2? read ftmp2.mod;

 26? run

   LAMBDA        LOWERBOUND     UPPERBOUND     COMMENTS      RUN #1
 -----------    -----------    -----------    ----------------------
 1.00000E-04    4.88265E-10    5.02254E-10
 2.00000E-04    1.95291E-09    2.01807E-09
 3.00000E-04    4.39357E-09    4.56112E-09
 4.00000E-04    7.80964E-09    8.14516E-09
 5.00000E-04    1.22003E-08    1.27841E-08
 6.00000E-04    1.75647E-08    1.84919E-08
 7.00000E-04    2.39013E-08    2.52827E-08
 8.00000E-04    3.12090E-08    3.31707E-08
 9.00000E-04    3.94859E-08    4.21702E-08
 1.00000E-03    4.87302E-08    5.22958E-08

 7 PATH(S) PROCESSED
 0.550 SECS. CPU TIME UTILIZED

 27? exit
```

Example 4

This interactive session illustrates how SURE can be used to obtain system unreliability as a function of mission time.

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? read ftmp9

  2: LAMBDA = 5E-4;             (* PERMANENT FAULT RATE *)
  3: STDEV = 3.6E-4;            (* STAN. DEV. OF RECOVERY DISTRIBUTION *)
  4: RECOVER = 2.7E-4;          (* MEAN OF RECOVERY DISTRIBUTION *)
  5: TIME = 0.1 TO* 1000 BY 10;
  6: 1,2 = 9*LAMBDA;
  7: 2,3 = 2*LAMBDA;
  8: 2,4 = <RECOVER,STDEV>;
  9: 4,5 = 9*LAMBDA;
 10: 5,6 = 2*LAMBDA;
 11: 5,7 = <RECOVER,STDEV>;
 12: 7,8 = 6*LAMBDA;
 13: 8,9 = 2*LAMBDA;
 14: 8,10 = <RECOVER,STDEV>;
 15: 10,11 = 6*LAMBDA;
 16: 11,12 = 2*LAMBDA;
```

```
17: 11,13 = <RECOVER,STDEV>;
18: 13,14 = 6*LAMBDA;
19: 14,15 = 2*LAMBDA;
20: 14,16 = <RECOVER,STDEV>;
21: 16,17 = 3*LAMBDA;
22: 17,18 = 2*LAMBDA;
23: 17,19 = <RECOVER,STDEV>;
24: 19,20 = 1*LAMBDA;
25: START = 1;

26? qtcalc = 0;                  (* use algebraic Q(T) calculation *)
27? run

   TIME          LOWERBOUND     UPPERBOUND     COMMENTS      RUN #1
 -----------    -----------    -----------    ----------------------
 1.00000E-01    1.01365E-10    1.21527E-10
 1.00000E+00    1.14931E-09    1.21774E-09
 1.00000E+01    1.19341E-08    1.24261E-08
 1.00000E+02    1.20763E-07    1.59925E-07 .. Q(T) INACCURATE
 1.00000E+03    0.00000E+00    8.03688E-02 .. Q(T) INACCURATE

7 PATH(S) PROCESSED
0.390 SECS. CPU TIME UTILIZED

28? exit
```

The Q(T) INACCURATE message indicates that the QTCALC = 0 option is inaccurate for the last two values of TIME in this problem. The computation should be rerun with QTCALC=1 or QTCALC=2 (the default value). The next session shows the result of rerunning this problem with QTCALC = 2.

```
$ sure

 SURE V5.2    NASA Langley Research Center

 1? echo = 0; read ftmp9;

26? qtcalc = 2;
27? run

   TIME          LOWERBOUND     UPPERBOUND     COMMENTS      RUN #1
 -----------    -----------    -----------    ----------------------
 1.00000E-01    1.01365E-10    1.21500E-10
 1.00000E+00    1.14931E-09    1.21500E-09
 1.00000E+01    1.19341E-08    1.21487E-08
 1.00000E+02    1.25980E-07    1.26748E-07    <ExpMat>
 1.00000E+03    7.68092E-03    7.69898E-03    <ExpMat>

7 PATH(S) PROCESSED
Q(T) ACCURACY >= 11 DIGITS
1.820 SECS. CPU TIME UTILIZED

28? exit
```

Example 5

This example illustrates the use of SURE to solve a model of a triplex system with transient and permanent faults. The permanent faults arrive at rate LAMBDA and the transient faults arrive at rate GAMMA. In the presence of a single fault the system degrades to a simplex at rate DELTA. The operating system sometimes improperly degrades in the presence of a transient fault. This occurs at rate PHI. This model contains a loop, and therefore, it is necessary to use the TRUNC feature of SURE. In this example, the TRUNC feature is used:

```
$ sure

 SURE V5.2   NASA Langley Research Center

 1? read 3trans*

 2: LAMBDA = 1E-4;               (* FAULT ARRIVAL RATE *)
 3: INPUT DELTA;                 (* RECOVERY RATE *)
    DELTA? 1800
 4: GAMMA = 10*LAMBDA;           (* TRANSIENT FAULT RATE *)
 5: RHO = 1 TO* 1E7 BY 10;       (* RATE OF DISAPPEARANCE OF TRANSIENT FAULTS *)
 6: PHI = DELTA;                 (* RATE TRANSIENTS RECONFIGURED OUT *)
 7: T = RHO + DELTA;
 8:
 9: 1,2 = 3*LAMBDA;
10: 2,3 = 2*LAMBDA + 2*GAMMA;
11: 2,4 = <1/DELTA,1/DELTA,1.0>;
12: 4,5 = LAMBDA + GAMMA;
13: 1,6 = 3*GAMMA;
14: 6,1 = <1/T,1/T,RHO/T>;
15: 6,4 = <1/T,1/T,PHI/T>;
16: 6,7 = 2*LAMBDA + 2*GAMMA;

17? trunc=3; warndig = 6;
18? run

*** START STATE ASSUMED TO BE 1

DELTA = 1.800E+03

    RHO          LOWERBOUND    UPPERBOUND     COMMENTS      RUN #1
 -----------    -----------   -----------    ---------------------
 1.00000E+00    1.77763E-04   1.81450E-04
 1.00000E+01    1.76971E-04   1.80639E-04
 1.00000E+02    1.69461E-04   1.72945E-04
 1.00000E+03    1.20686E-04   1.23038E-04 .. TRUNC TOO SMALL
 1.00000E+04    4.12797E-05   4.20342E-05 .. TRUNC TOO SMALL
 1.00000E+05    1.92296E-05   1.96140E-05 .. TRUNC TOO SMALL
 1.00000E+06    1.66248E-05   1.69692E-05 .. TRUNC TOO SMALL
 1.00000E+07    1.63596E-05   1.66999E-05 .. TRUNC TOO SMALL

 12 PATH(S) PROCESSED
 1 LOOP(S) TRUNCATED AT DEPTH 3
 ACCURACY MAY BE LESS THAN 5 DIGITS DUE TO LOOP TRUNCATION
 3.170 SECS. CPU TIME UTILIZED

18? plot xylog
19? disp copy
```

Figure 6 illustrates the model displayed on the output graphics device. (Note that the Greek words in the model description are displayed as Greek characters in the graphics output.) The plot in figure 7 was generated from this run.

To make sure that the truncation error is insignificant, the model is reprocessed with TRUNC = 4:

```
20? echo = 0;
21? read 3trans;
    DELTA? 1800

37? trunc=4
38? run

*** START STATE ASSUMED TO BE 1

DELTA = 1.800E+03

   RHO              LOWERBOUND     UPPERBOUND      COMMENTS     RUN #1
 -----------      -----------    -----------     ---------------------
 1.00000E+00      1.77763E-04    1.81450E-04
 1.00000E+01      1.76971E-04    1.80639E-04
 1.00000E+02      1.69461E-04    1.72945E-04
 1.00000E+03      1.20686E-04    1.23038E-04
 1.00000E+04      4.12797E-05    4.20342E-05
 1.00000E+05      1.92296E-05    1.96140E-05
 1.00000E+06      1.66249E-05    1.69692E-05
 1.00000E+07      1.63596E-05    1.66999E-05

 16 PATH(S) PROCESSED
 1 LOOP(S) TRUNCATED AT DEPTH 4
 3.220 SECS. CPU TIME UTILIZED

19? exit
```

It can be seen that truncation error is insignificant.

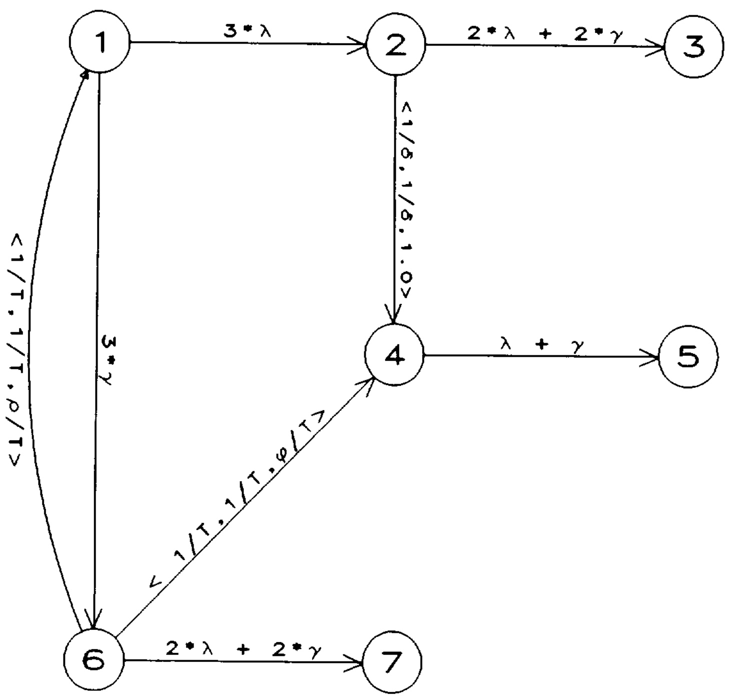


Figure 6. Semi-Markov model from example 5.

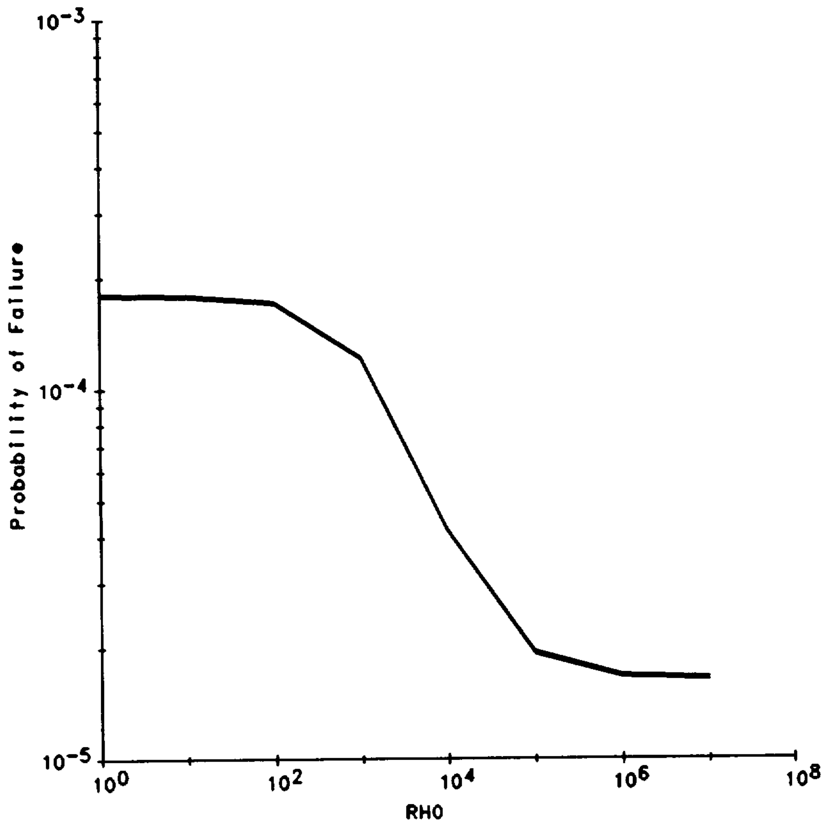


Figure 7. SURE output from example 5.

Example 6

This example illustrates the use of SURE in Lee's mode. The same model as used in example 5 is used here. However, the information given for the fast recovery transitions is different. In the presence of a permanent fault, the system degrades to a simplex. The mean degradation time is 1/DELTA. The probability that the degradation process takes more than QUANT2 hours is QPROB2. In the presence of a transient fault, the system degrades to a simplex with probability PHI/(PHI+RHO) and returns to the fault-free state with probability RHO/(RHO+PHI). The probability that this requires more than QUANT6 hours is QPROB6.

```
$ sure

 SURE V5.2    NASA Langley Research Center

 1? read leem

 2: LEE;
 3: LAMBDA = 1E-4;              (* FAULT ARRIVAL RATE *)
 4: DELTA = 1800.0;             (* MEAN RECOVERY TIME *0)
 5: GAMMA = 10*LAMBDA;          (* TRANSIENT FAULT RATE *)
 6: RHO = 1 TO* 1E7 BY 10;      (* RECOVERY RATE FROM TRANSIENT FAULT *)
 7: PHI = DELTA;                (* RATE TRANSIENTS RECONFIGURED OUT *)
 8: T = RHO + PHI;
 9: QUANT2 = 1E-2;
10: QPROB2 = 1.0 - EXP(-DELTA*QUANT2);
11: TIME = 10;
12: 1,2 = 3*LAMBDA;
13: 2,3 = 2*LAMBDA + 2*GAMMA;
14: @2 = <1/DELTA, QUANT2, QPROB2>;
```

```
15: 2,4 = <1.0>;
16: 4,5 = LAMBDA + GAMMA;
17: 1,6 = 3*GAMMA;
18: QUANT6 = 1E-2;
19: QPROB6 = 1.0 - EXP(-T*QUANT6);
20: @6 = <1/T,QUANT6,QPROB6>;
21: 6,1 = <RHO/T>;
22: 6,4 = <PHI/T>;
23: 6,7 = 2*LAMBDA + 2*GAMMA;

24? run

*** START STATE ASSUMED TO BE 1
----- LEE STATISTICAL ANALYSIS MODE -----

   RHO             LOWERBOUND     UPPERBOUND      COMMENTS     RUN #1
-----------       -----------    -----------     ---------------------
1.00000E+00       1.78430E-04    1.81450E-04
1.00000E+01       1.77632E-04    1.80639E-04
1.00000E+02       1.70066E-04    1.72945E-04
1.00000E+03       1.20986E-04    1.23038E-04
1.00000E+04       4.13316E-05    4.20343E-05
1.00000E+05       1.92859E-05    1.96141E-05
1.00000E+06       1.66853E-05    1.69692E-05
1.00000E+07       1.64206E-05    1.67000E-05

12 PATH(S) PROCESSED
1 LOOP(S) TRUNCATED AT DEPTH 3
3.750 SECS. CPU TIME UTILIZED
```

Example 7

This example illustrates the use of Lee's method to model a system with two possible recoveries from a fault. In this model, the system recovers from a fault by bringing in a (nonfailed) spare 90 percent of the time and degrades to a simplex 10 percent of the time.

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? lee;
  2? lambda = 1e-4;          (* Failure rate of a processor *)
  3? pr1 = 0.90;             (* Probability recovery is by sparing *)
  4? mu = 2e-4;              (* Mean recovery time *)
  5? 1,2 = 3*lambda;
  6? 2,3 = 2*lambda;
  7? 2,4 = <pr1>;
  8? 4,5 = 3*lambda;
  9? 5,6 = 2*lambda;
 10? 2,7 = <1-pr1>;
 11? 7,8 = lambda;
 12? @2 = <mu,2*mu,1.0>;      (* No observed recoveries greater than 2*MU*)
 13? list=2;
 14? run

 ----- LEE STATISTICAL ANALYSIS MODE -----
```

```
        DEATHSTATE      LOWERBOUND     UPPERBOUND    COMMENTS      RUN #1
        ----------     ------------   ------------  ----------------------
             3         1.19815E-10    1.20000E-10
             6         2.72421E-09    2.73000E-09
             8         1.34809E-07    1.35000E-07

          TOTAL        1.37653E-07    1.37850E-07

        3 PATH(S) PROCESSED
        0.120 SECS. CPU TIME UTILIZED
```

Example 8

The following session illustrates the use of the ORPROB command:

```
        $sure

          SURE V5.2     NASA Langley Research Center

          1? 1,2 = 1E-4;
          2? run

                              LOWERBOUND     UPPERBOUND     COMMENTS      RUN #1
          ------------       ------------   ------------   ----------------------
                              9.99500E-04    1.00000E-03

          1 PATH(S) PROCESSED
          0.070 SECS. CPU TIME UTILIZED

          3? 2,4 = 1E-5;
          4? run

                              LOWERBOUND     UPPERBOUND     COMMENTS      RUN #2
          ------------       ------------   ------------   ----------------------
                              9.99500E-05    1.00000E-04

        1 PATH(S) PROCESSED
        0.050 SECS. CPU TIME UTILIZED

        5? 1,2 = 2.5E-4;
        6? run

                              LOWERBOUND     UPPERBOUND     COMMENTS      RUN #3
          ------------       ------------   ------------   ----------------------
                              2.49687E-03    2.50000E-03

        1 PATH(S) PROCESSED
        0.040 SECS. CPU TIME UTILIZED

        7? orprob

            RUN #          LOWERBOUND      UPPERBOUND
        ----------        ------------    ------------
             1            9.99500E-04     1.00000E-03
             2            9.99500E-05     1.00000E-04
             3            2.49687E-03     2.50000E-03
        ----------        ------------    ------------
         OR PROB =        3.59352E-03     3.59715E-03

        8? exit
```

Example 9

In this example a model of a triad with spares is investigated. When an active processor fails, a spare processor is brought into the configuration to replace the faulty one. If a spare fails, the fault remains undetectable until it is brought into the active configuration. For simplicity the time required to replace a faulty processor with a spare and the degradation time are assumed to be exponentially distributed. Therefore, the FAST exponential specification method can be used:

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? read undet

  2: LAMBDA = 1E-4;                    (* Failure rate of a processor *)
  3: DELTA = 1E4;                      (* Rate of sparing *)
  4: DEGRATE = 1E4;                    (* Rate of degrading to a simplex *)
  5: PSI = 1E-6 TO* LAMBDA BY 10;      (* Failure rate of spares *)
  6: 1,2 = 3*LAMBDA;
  7: 2,3 = 2*LAMBDA;
  8: 1,7 = PSI;
  9: 2,4 = FAST DELTA;
 10: 2,8 = PSI;
 11: 4,5 = 3*LAMBDA;
 12: 5,6 = 2*LAMBDA;
 13: 5,10 = FAST DEGRATE;
 14: 7,8 = 3*LAMBDA;
 15: 8,9 = 2*LAMBDA;
 16: 8,5 = FAST DELTA;
 17: 10,11 = LAMBDA;

 18? run

    PSI          LOWERBOUND   UPPERBOUND    COMMENTS     RUN #1
  -----------   -----------  -----------   ----------------------
  1.00000E-06   1.55410E-09  1.56509E-09
  1.00000E-05   1.59876E-09  1.61010E-09
  1.00000E-04   2.04524E-09  2.06016E-09

 9 PATH(S) PROCESSED
 0.180 SECS. CPU TIME UTILIZED
```

Example 10

This example shows how the CALC function can be used in conjunction with the PLOT commands to obtain plots of mathematical functions. The plots produced from this session are shown in figures 8 through 11.

```
$ sure

  SURE V5.2    NASA Langley Research Center

  1? X = 1E-2 TO 2;
  2? POINTS = 500; LIST = 0;
  3? CALC SIN(1/X);
  4? PLOT                                   (* Figure 8 *)
  5? DISP COPY
     HARD COPY IN PROGRESS
  6? CALC SIN(1/X)*X
  7? PLOT                                   (* Figure 9 *)
  8? DISP COPY
     HARD COPY IN PROGRESS
  9? CLEAR
```

```
10? X = .1 TO 10;
12? POINTS = 500; LIST = 0;
13? PI = 3.14159265;
14? CALC SIN(2*PI*X);
15? PLOT                                        (* Figure 10 *)
16? DISP COPY
    HARD COPY IN PROGRESS
17? CALC EXP(-X)*SIN(2*PI*X)
18? PLOT                                        (* Figure 11 *)
19? DISP COPY
    HARD COPY IN PROGRESS
20? EXIT
```

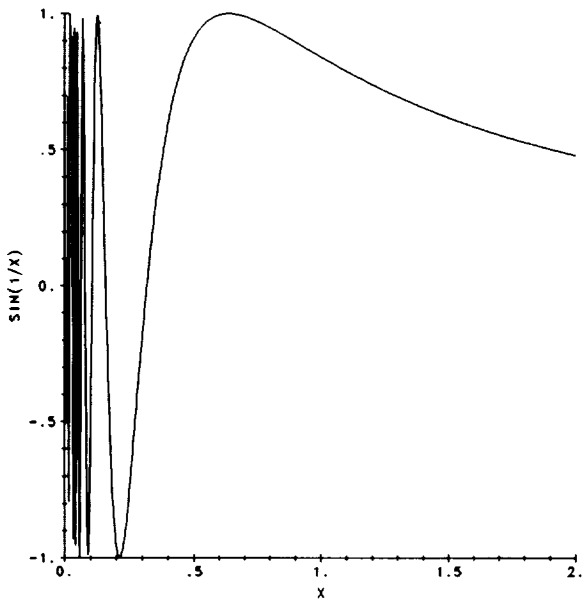


Figure 8. Plot of SIN(1/X).

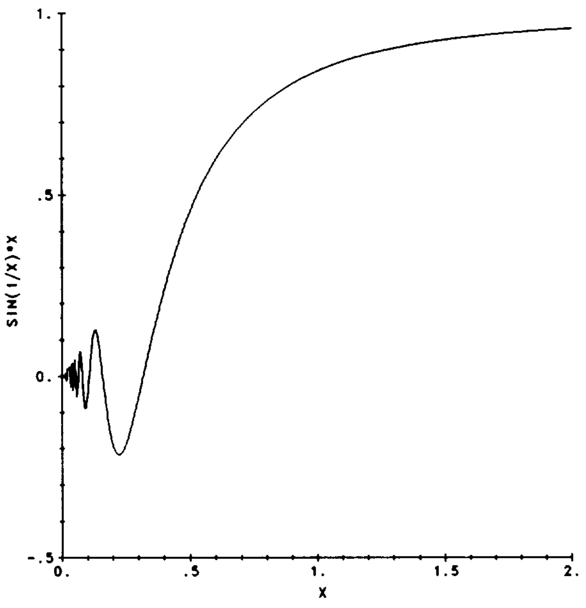


Figure 9. Plot of SIN(1/X)*x.

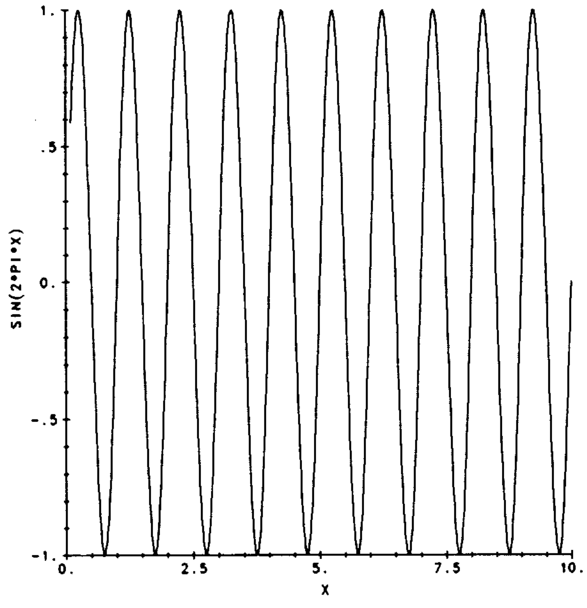


Figure 10. Plot of SIN(2*PI*X).

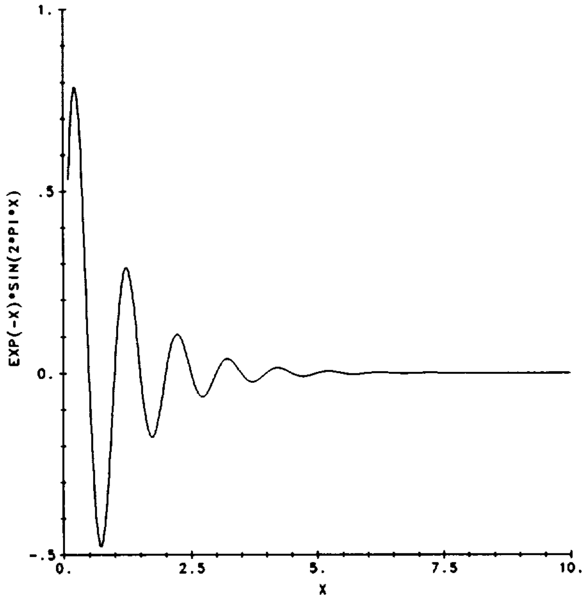


Figure 11. Plot of EXP(−X)*SIN(2*PI*X).

Example 11

This example illustrates the use of the IF command to analyze the probability of system failure of a N-multiply redundant (NMR) system as a function of N:

```
$ sure

  SURE V5.2     NASA Langley Research Center

  1? read nmr

  2: LAMBDA = 1E-4;
  3: N = 3 TO 15 BY 2;
  4: 1,2 = N*LAMBDA;
  5: IF N > 2 THEN 2,3 = (N-1)*LAMBDA;
  6: IF N > 4 THEN 3,4 = (N-2)*LAMBDA;
  7: IF N > 6 THEN 4,5 = (N-3)*LAMBDA;
  8: IF N > 8 THEN 5,6 = (N-4)*LAMBDA;
  9: IF N > 10 THEN 6,7 = (N-5)*LAMBDA;
 10: IF N > 12 THEN 7,8 = (N-6)*LAMBDA;
 11: IF N > 14 THEN 8,9 = (N-7)*LAMBDA;

 12? run

   N              LOWERBOUND    UPPERBOUND    COMMENTS      RUN #1
 -----------     -----------   -----------   ---------------------
 3.00000E+00     2.99500E-06   3.00000E-06
 5.00000E+00     9.97000E-09   1.00000E-08
 7.00000E+00     3.48460E-11   3.50000E-11
 9.00000E+00     1.25265E-13   1.26000E-13
 1.10000E+01     4.58634E-16   4.62000E-16
 1.30000E+01     1.70098E-18   1.71600E-18
 1.50000E+01     6.36922E-21   6.43500E-21

 1 PATH(S) PROCESSED
 1.990 SECS. CPU TIME UTILIZED
```

Example 12

In this example the use of the PRUNE constant is demonstrated.

```
$ sure

  SURE V5.2     NASA Langley Research Center

  1? read px

  2: 1,2 = 1E-7;
  3: 1,3 = 1E-2;
  4: 3,4 = <1E-2,1E-2>;
  5: 3,9 = 1E-6;  4,5 = 1E-3;  5,6 = 1E-3;  6,7 = 1E-5;
  6: PRUNEPOW = 7 TO 12 BY 1;
  7: PRUNE = 10**(-PRUNEPOW);
  8: WARNDIG = 4;

  9? run;
  PRUNEPOW       LOWERBOUND    UPPERBOUND     COMMENTS       RUN #1
 ------------   ------------  ------------   ----------------------
8.00000E+00    9.50000E-07   1.00000E-06 .. PRUNING TOO SEVERE
9.00000E+00    9.50000E-07   1.00000E-06 .. PRUNING TOO SEVERE
1.00000E+01    9.50868E-07   1.00100E-06
1.10000E+01    9.50906E-07   1.00104E-06
1.20000E+01    9.50906E-07   1.00104E-06

3 PATH(S) PROCESSED
2 PATH(S) PRUNED AT LEVEL  1.00000E-08
SUM OF PRUNED STATES PROBABILITY <  1.04167E-09
1.170 SECS. CPU TIME UTILIZED
```

The summary statistics refer to the worst case—that is, where the pruning is the most severe (PRUNE = 1E-8).

Derivation of Bounding Theorem

Mathematical Preliminaries

The proof of the upper and lower bound theorem requires four items that are elementary but that are not always covered in introductions to probability. They are the Markov inequality, the moments (e.g., mean and variance) as integrals of 1 minus the distribution function, the density functions for independent competing events, and the convolution formula for independent sequential events. These four topics are developed below assuming a background that includes an understanding of probability as the integral of a density function and the concept that the joint density function for independent events is the product of the individual density functions. All the distributions are holding-time distributions, which means all the densities are concentrated on the positive real axis. That is, if $f(t)$ is a density function, then $f(t) = 0$ for $t < 0$.

The notation used throughout is $\mu(H)$ and $\sigma^2(H)$ for the mean and variance of the distribution H and h as the density for H.

Markov's inequality is

$$1 - H(c) = \int_c^\infty h(t)\ dt \leq \frac{\mu^2(H) + \sigma^2(H)}{c^2}$$

for $c > 0$. The derivation is

$$\begin{aligned}\int_c^\infty h(t)\ dt &\leq \int_c^\infty \frac{t^2}{c^2} h(t)\ dt \\ &\leq \frac{1}{c^2}\int_0^\infty t^2\ h(t)\ dt \\ &= \frac{\mu^2(H) + \sigma^2(H)}{c^2}\end{aligned}$$

The first two moments as integrals of 1 minus the distribution function are

$$\int_0^\infty [1 - H(t)]\ dt = \mu(H) = \int_0^\infty t\ h(t)\ dt$$

$$2\int_0^\infty t[1 - H(t)]\ dt = \mu^2(H) + \sigma^2(H) = \int_0^\infty t^2\ h(t)\ dt$$

The equalities hold in the sense that if the improper integral on one side exists then the improper integral on the other side exists and the two are equal. For the derivation, perform an integration by parts to get

$$\int_0^b t^{k-1}[1 - H(t)]\, dt = \left.\frac{t^k[1 - H(t)]}{k}\right|_0^b + \frac{1}{k}\int_0^b t^k\, h(t)\, dt$$

First, suppose $\int_0^\infty t^k\, h(t)\, dt$ is finite and choose $\varepsilon > 0$. Since the improper integral converges, there exists an M such that if $x \geq M$ then

$$\begin{aligned}\varepsilon &> \int_x^\infty t^k\, h(t)\, dt \\ &\geq x^k \int_x^\infty h(t)\, dt \\ &\geq x^k[1 - H(x)]\end{aligned}$$

Therefore $t^k[1 - H(t)]$ goes to zero as t goes to infinity. Hence if the kth moment exists then the integral of 1 minus the distribution function exists and the two are equal.

Next, suppose that $\int_0^\infty t^{k-1}[1 - H(t)]\, dt$ is finite. The integration by parts formula says that

$$\frac{1}{k}\int_0^b t^k\, h(t)\, dt \leq \int_0^b t^{k-1}[1 - H(t)]\, dt$$

for all b. Therefore, the kth moment is finite. As before, if the kth moment is finite, then $t^k[1 - H(t)]$ goes to zero as t goes to infinity. Hence, if the integral of 1 minus the distribution function exists, then the moment exists and the two are equal.

The derivation of the densities for independent competing events is illustrated in figure 12.

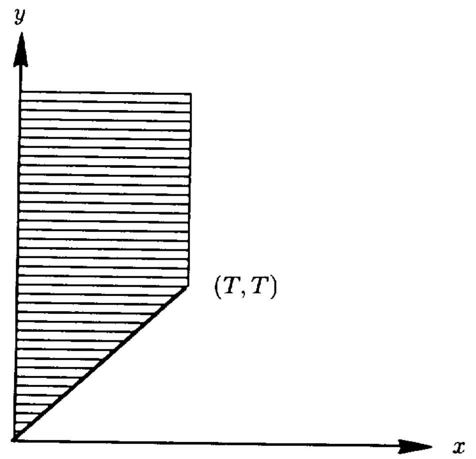


Figure 12. Graph for independent competing events.

Let event A have density $f(x)$ and event B have density $g(y)$. The probability that A occurs before B and before time T is given by the shaded area in figure 12, where $x \leq T$ and $x \leq y$. The integral of the shaded area is

$$\int_0^T \int_x^\infty f(x)\, g(y)\, dy\, dx = \int_0^T f(x)\, [1 - G(x)]\, dx$$

which means that the density of event A occurring before event B and before time T is $f(x)\ [1 - G(x)]$. This density is likely to be defective; that is, its integral is less than one.

Let $\rho(A)$ be the probability that event A occurs before event B. Then

$$\rho(A) = \int_0^\infty f(x)\ [1 - G(x)]\ dx$$

The conditional probability that event A occurs before time T given that event A occurs before event B is

$$\frac{1}{\rho(A)} \int_0^T f(x)\ [1 - G(x)]\ dx$$

The conditional kth moment of A, given that event A occurs before event B, is

$$\frac{1}{\rho(A)} \int_0^\infty x^k\ f(x)\ [1 - G(x)]\ dx$$

If event A is competing against events $B_1, B_2, ..., B_n$ with densities $g_1, g_2, ..., g_n$ then the density for event A occurring first is $f(x)\ [1 - G_1(x)] ... [1 - G_n(x)]$.

The derivation of the density for independent sequential events is illustrated in figure 13.

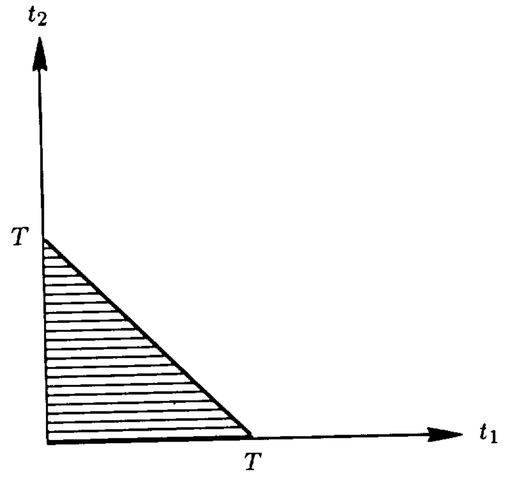


Figure 13. Graph for sequential independent events.

As before let events A and B have densities f and g. The probability that the occurrence times for both events A and B sum to less than T is given by the shaded area, where $t_1 + t_2 \leq T$. The integral of the shaded area is

$$\int_0^T \int_0^{T-t_1} f(t_1)\ g(t_2)\ dt_2\ dt_1 = \int_0^T \int_0^{T-t_2} g(t_2)\ f(t_1)\ dt_1\ dt_2$$

If there are n independent sequential events with densities $f_1, f_2, ..., f_n$, then the probability that the sum of all their occurrence times is less than or equal to T is

$$\int_0^T \int_0^{T-t_1} ... \int_0^{T-t_1-...-t_{n-1}} f_n(t_n)\ ...\ f_2(t_2)\ f_1(t_1)\ dt_n\ ...\ dt_2\ dt_1$$

This presentation has been made in terms of the Riemann integral where all the distributions have density functions. These results remain true for the more general Riemann-Stieltjes integral which can handle a wider variety of distributions such as instantaneous jumps. The bounds are derived for the more general case. Given a distribution H, the notation for the event occurring between time t_1 and time t_2 as a Riemann-Stieltjes integral is

$$\int_{t_1}^{t_2} dH(x)$$

If H is differentiable with $H'(x) = h(x)$ then

$$\int_{t_1}^{t_2} dH(x) = \int_{t_1}^{t_2} h(x)\, dx$$

Proof of Theorem

This section derives the upper and lower bounds presented in the previous section. The objective of the bounds is to reduce the computational burden in reliability analysis by means of a qualitative result. Initially two features of the reconfiguration process pose difficult descriptive and numerical problems. The first is that a sophisticated digital architecture has a complicated fault detection and system reconfiguration procedure. When trying to establish high reliability, none of the details can be arbitrarily ignored. The second is that system recovery is much faster than fault occurrence. An explicit set of equations describing both is numerically stiff.

The theorem proved in this section provides a solution to both these problems for systems with low fault occurrence rates and quick recovery—a class of systems that designers are currently trying to produce. The theorem establishes that just the means and variances of the recovery times are sufficient information about the reconfiguration process to obtain tight bounds on the probability of system failure. Furthermore, the formulas for the bounds consist of a factor involving the means and variances times a quantity that is the solution of a differential equation where the coefficients are the low failure rates. Since the failure rates do not differ much in magnitude, the differential equation is numerically stable. Hence a difficult descriptive and numerical problem is reduced to one with familiar statistics (means and variances) and a tractable differential equation.

The differential equation is tractable enough that for a large number of cases its solution has easy algebraic upper and lower bounds. These are derived below. The original probability bounds together with the quick bounds for the differential equation are referred to as the "algebraic bounds" for system failure. The SURE program automatically selects the appropriate method and informs the user when the differential equation package option is used.

A general path in a semi-Markov reliability model is shown in figure 14. The following notation applies to this path:

A_k	state in general path where only exiting transitions are low rate failure transitions
λ_k	successful (on-path) failure transition out of A_k
γ_k	sum of unsuccessful (off-path) failure transitions out of A_k
B_i	state in general path where successful (on-path) transition is fast recovery transition that competes against other fast recovery transitions and against low rate failure transitions
$F_{i,1}$	successful recovery transition out of state B_i
$F_{i,m}$	for $m > 1$, unsuccessful recovery transition out of B_i
F_i^*	conditional distribution of the successful transition, $F_{i,1}$, when it competes against unsuccessful recovery transitions, $F_{i,m}$ where $m > 1$

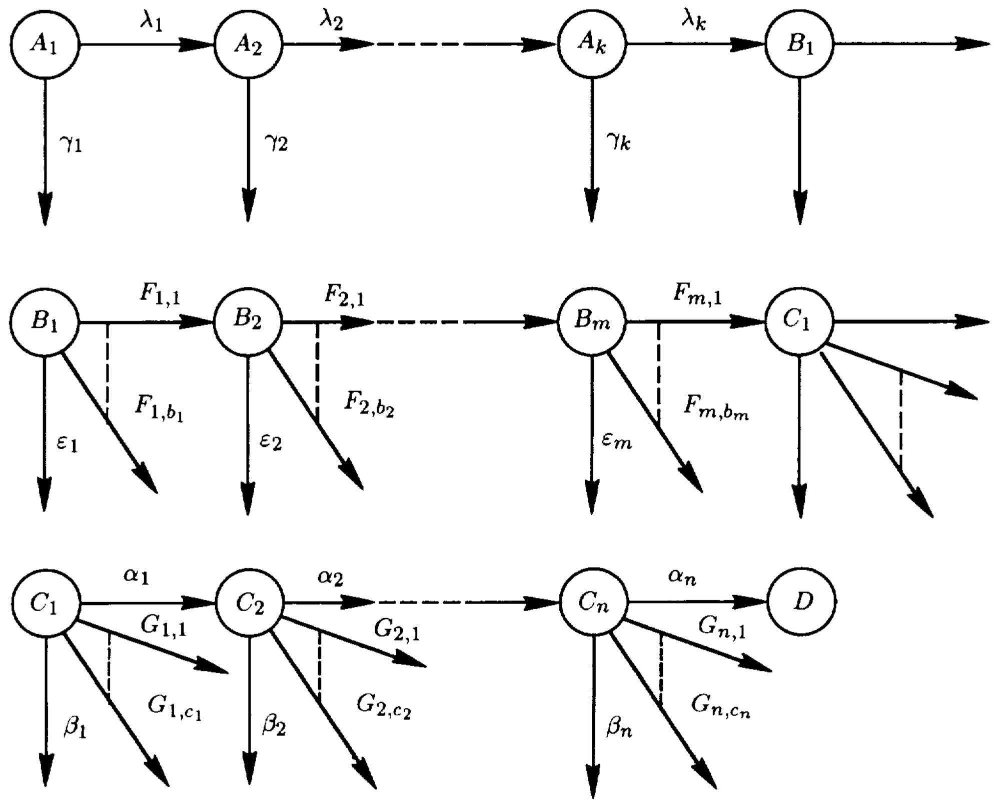


Figure 14. General path in a semi-Markov model.

ε_i	sum of rates of low-rate failure transitions out of state B_i
C_j	state in general path where successful transition is low rate failure transition that competes against fast recovery transitions and other low rate failure transitions
α_j	rate of successful low rate failure transition out of C_j
$G_{j,n}$	distribution of unsuccessful fast recovery transition out of state C_j
H_j	distribution of holding time in state C_j considering only fast recovery exiting transitions, $G_{j,n}$
β_j	sum of rates of unsuccessful low rate failure transitions out of state C_j
D	absorbing state for entire general path
Q	absorbing state for subpath consisting of states with only low rate exiting transitions
q	density function for distribution Q

The global time independence of a semi-Markov model permits the rearrangement of states on the path for notational and computational convenience. Using the terminology of the previous sections, the first line consists of k class 1 states, the second of m class 2 states, and the third of n class 3 states. Figure 15 displays the k class 1 states of figure 14.
The notation is

$D(T)$	= probability of traversing path in figure 14 by time T
$Q(T)$	= probability of traversing path in figure 15 by time T

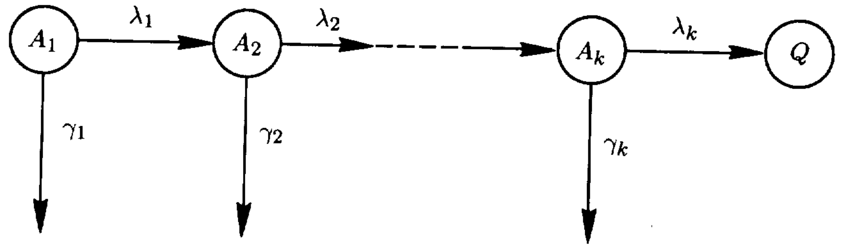


Figure 15. Constant rate part of path.

$\rho(F_i^*)$ = probability that transition $F_{i,1}$ is successful when competing against other recovery transitions

$$= \int_0^\infty [1 - F_{i,2}(t)]...[1 - F_{i,b_i}(t)]\, dF_{i,1}(t)$$

$\mu(F_i^*)$ = first conditional moment of $F_{i,1}$

$$= \frac{1}{\rho(F_i^*)} \int_0^\infty t[1 - F_{i,2}(t)]...[1 - F_{i,b_i}(t)]\, dF_{i,1}(t)$$

$\mu^2(F_i^*) + \sigma^2(F_i^*)$ = second conditional moment of $F_{i,1}$

$$= \frac{1}{\rho(F_i^*)} \int_0^\infty t^2[1 - F_{i,2}(t)]...[1 - F_{i,b_i}(t)]\, dF_{i,1}(t)$$

$\mu(H_j)$ = first moment of holding time in state C_j considering only recovery transitions

$$= \int_0^\infty [1 - G_{j,1}(t)]...[1 - G_{j,c_j}(t)]\, dt$$

$\mu^2(H_j) + \sigma^2(H_j)$ = second moment of holding time in state C_j considering only recovery transitions

$$= 2\int_0^\infty t[1 - G_{j,1}(t)]...[1 - G_{j,c_j}(t)]\, dt$$

The integrands for the probabilities and moments of F_i^* are the densities for an event competing against other independent events. The justification of the integrand for the moments of the H_j's is as follows. First, the holding time is the same as the leaving time. Let W be the distribution for the leaving time, and let $G_1, G_2, ..., G_j$ be the distributions for the exiting transitions. Then

$$1 - W(t) = [1 - G_1(t)]...[1 - G_j(t)]$$

since the product on the right is the probability that no exiting event has occurred which is equal to the probability that the state has not been left. The moments of the holding time are given as integrals of 1 minus the distribution function W.

The expressions for the probabilities and moments do not include the competing failure rate transitions. The formulas for the bounds take this exclusion into account and give correct bounds in the presence of device failure rates. This approach is taken because the measurement of the recovery processes of a system is usually made on prototype systems whose failure rate is not representative of a production system. By decoupling the specification of the recovery process parameters from the failure parameters, these processes can be measured and studied independently. The statistician, however, sees the recovery transition as always competing against device failure and wants expressions that reflect what is actually observed. These expressions and the resulting (slightly different) bounds are covered in another publication. (See ref. 12.) The numerical differences between the different versions are negligible.

Derivation of Bounds for a Simple Case

The derivation of the theorem is first given for a simple case. In the next subsection, the general proof is presented. Consider the reliability model in figure 16. The probability of arriving in the first coverage failure state, the state on the far right in the first row, by time T is

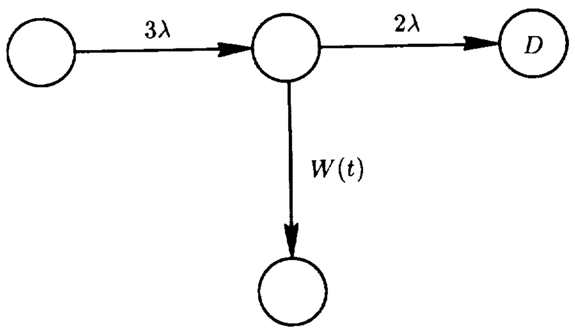


Figure 16. Failure state D in reliability model.

$$D(T) = \int_0^T 3\lambda e^{-3\lambda x} \int_0^{T-x} 2\lambda e^{-2\lambda y}[1 - W(y)]\, dy\, dx$$

where

$$W(y) = \int_0^y w(t)\, dt$$

Certainly,

$$D(T) \leq \int_0^T 3\lambda e^{-3\lambda x} \int_0^{\infty} 2\lambda e^{-2\lambda y}[1 - W(y)]\, dy\, dx$$

Since $w(t)$ is a fast transition, $W(t)$ goes to 1 extremely fast which means $[1 - W(t)]$ goes to 0 extremely fast. Hence, there is very little difference between the two iterated integrals. Writing the latter iterated integral as a product, pulling the 2λ outside the second integral, and replacing $e^{-2\lambda y}$ by an upper bound of 1 give

$$\begin{aligned} D(T) &\leq \left\{ \int_0^T 3\lambda e^{-3\lambda x}\, dx \right\} \left\{ 2\lambda \int_0^{\infty} [1 - W(y)]\, dy \right\} \\ &= \left\{ \int_0^T 3\lambda e^{-3\lambda x}\, dx \right\} \{2\lambda\mu\} \end{aligned}$$

where μ is the mean of the transition $w(t)$.

To illustrate the origin of the lower bound, suppose the operating time T is 1 hour and $w(t)$ has a mean of 1 sec. Let Δ be a time of 1 min. Then

$$D(T) \geq \int_0^{T-\Delta} 3\lambda e^{-3\lambda x} \int_0^{\Delta} 2\lambda e^{-2\lambda y}[1 - W(y)]\, dy\, dx$$

It can be seen that this lower bound is close to the upper bound. First, integrating from 0 to $T - \Delta$ is little different from integrating from 0 to T if T is 1 hour and Δ is 1 min. Second, integrating from 0 to Δ is little different from integrating from 0 to infinity if $w(t)$ has a mean of 1 sec, providing $w(t)$ has a small variance. (In fact, the expression that replaces the second integral involves the variance.)

In terms of pictures, the exact formula for $D(T)$ is the convolution integral over the triangle shown in figure 17. The upper bound is the integral over the superscribed infinite rectangle shown in figure 18. The lower bound is the integral over the inscribed rectangle shown in figure 19.

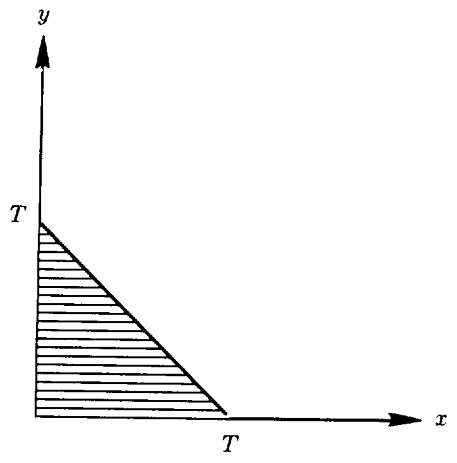


Figure 17. Convolution triangle.

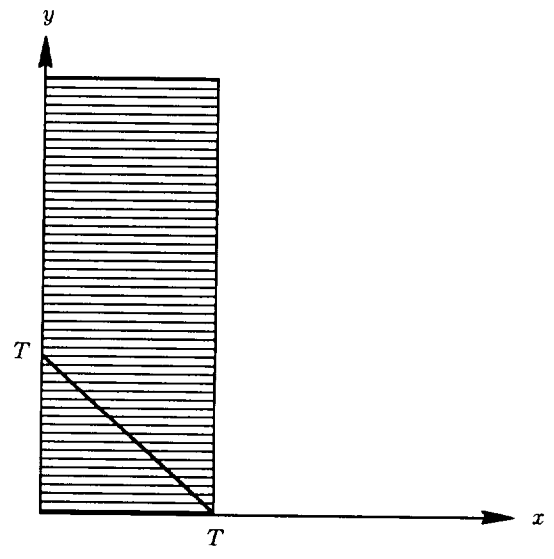


Figure 18. Upper bound rectangle.

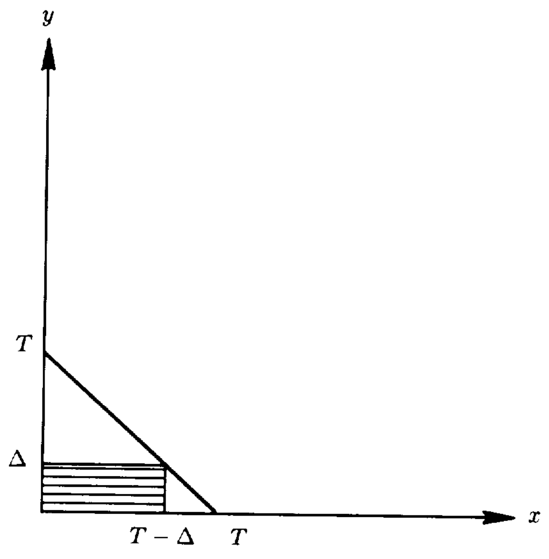


Figure 19. Lower bound rectangle.

There are two comments about the quantity Δ. First, the choice is flexible, and only a little work has been done on optimizing the chosen value. (See appendix A.) Second, the value of Δ increases for a path as the number of states with fast transitions increases, and a larger Δ increases the distance between the upper and lower bounds. In general, however, paths with many states are less likely to be traversed than paths with fewer states. As a result, the large value of Δ for long paths contributes little to the overall error when estimating reliability.

Proof of General Theorem

The proof of the general theorem is simply a multidimensional version of the previous argument.

Theorem (White). Let

$$r_1,\ r_2,\ \ldots,\ r_m,\ s_1,\ s_2,\ \ldots,\ s_n \geq 0$$
$$\Delta = r_1 + r_2 + \ldots + r_m + s_1 + s_2 + \ldots + s_n$$
$$\Delta \leq T$$

Then, with the assumptions and notation as previously described,

$$\text{LB} \leq D(T) \leq \text{UB}$$

where

$$\text{UB} = Q(T) \prod_{i=1}^{m} \rho(F_i^*) \prod_{j=1}^{n} \alpha_j \mu(H_j)$$

$$\text{LB} = Q(T-\Delta) \prod_{i=1}^{m} \rho(F_i^*) \left[1 - \varepsilon_i \mu(F_i^*) - \frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}\right]$$
$$\times \prod_{j=1}^{n} \alpha_j \left\{\mu(H_j) - \frac{(\alpha_j+\beta_j)[\mu^2(H_j)+\sigma^2(H_j)]}{2} - \frac{\mu^2(H_j)+\sigma^2(H_j)}{s_j}\right\}$$

Four lemmas. The following lemmas are used in the proof of the bounding theorem:

(*i*) $\displaystyle\int_0^\infty \exp(-\varepsilon_i x_i)[1 - F_{i,2}(x_i)]\ldots[1 - F_{i,b_i}(x_i)]\ dF_{i,1}(x_i) \leq \rho(F_i^*)$

(*ii*) $\displaystyle\int_0^\infty \alpha_j \exp(-\alpha_j y_j - \beta_j y_j)[1 - G_{j,1}(y_j)]\ldots[1 - G_{j,c_j}(y_j)]\ dy_j \leq \alpha_j \mu(H_j)$

(*iii*) $\displaystyle\int_0^{r_i} \exp(-\varepsilon_i x_i)[1 - F_{i,2}(x_i)]\ldots[1 - F_{i,b_i}(x_i)]\ dF_{i,1}(x_i)$

$$\geq \rho(F_i^*)\left[1 - \varepsilon_i \mu(F_i^*) - \frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}\right]$$

(*iv*) $\displaystyle\int_0^{s_j} \alpha_j \exp(-\alpha_j y y_j - \beta_j y_j)[1 - G_{j,1}(y_j)]\ldots[1 - G_{j,c_j}(y_j)]\ dy_j$

$$\geq \alpha_j \left\{\mu(H_j) - \frac{(\alpha_j+\beta_j)[\mu^2(H_j)+\sigma^2(H_j)]}{2} - \frac{\mu^2(H_j)+\sigma^2(H_j)}{s_j}\right\}$$

Proof of lemmas. Assertions (i) and (ii) follow from the inequality $e^{-a} \leq 1$ for $a \geq 0$. Assertions (iii) and (iv) use the equation

$$\int_0^c = \int_0^\infty - \int_c^\infty$$

the inequalities

$$1 - a \leq e^{-a} \leq 1 \quad (a \geq 0)$$

and Markov's inequality.

To prove lemma (iii) note that the integral is bigger than or equal to

$$\int_0^\infty (1 - \varepsilon_i x_i)[1 - F_{i,2}(x_i)]...[1 - F_{i,b_i}(x_i)]\, dF_{i,1}(x_i)$$
$$- \int_{r_i}^\infty [1 - F_{i,2}(x_i)]...[1 - F_{i,b_i}(x_i)]\, dF_{i,1}(x_i)$$
$$= \int_0^\infty [1 - F_{i,2}(x_i)]...[1 - F_{i,b_i}(x_i)]\, dF_{i,1}(x_i)$$
$$- \varepsilon_i\ \rho(F_i^*)\frac{1}{\rho(F_i^*)}\int_0^\infty x_i[1 - F_{i,2}(x_i)]...[1 - F_{i,b_i}(x_i)]\, dF_{i,1}(x_i)$$
$$- \rho(F_i^*)\frac{1}{\rho(F_i^*)}\int_{r_i}^\infty [1 - F_{i,2}(x_i)]...[1 - F_{i,b_i}(x_i)]\, dF_{i,1}(x_i)$$

which is bigger than or equal to

$$\rho(F_i^*) - \varepsilon_i\ \rho(F_i^*)\ \mu(F_i^*) - \rho(F_i^*)\frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}$$

when the last integral is replaced by Markov's inequality. The $\rho(F_i^*)$ factors appear because the last two integrals must be divided by $\rho(F_i^*)$ in order to get the conditional density.

To prove lemma (iv), note that the integral is bigger than or equal to

$$\alpha_j \int_0^\infty [1 - (\alpha_j + \beta_j)y_j][1 - G_{j,1}(y_j)]...[1 - G_{j,c_j}(y_j)]\, dy_j$$
$$- \alpha_j \int_{s_j}^\infty [1 - G_{j,1}(y_j)]...[1 - G_{j,c_j}(y_j)]\, dy_j$$

The first integral is a multiple of the first moment minus a multiple of the second moment. The integrand in the last integral is equal to 1 minus the probability of being in state C_j at time y_j and by Markov's inequality is less than or equal to $[\mu^2(H_j) + \sigma^2(H_j)]/y_j^2$. The indefinite integral of $1/y_j^2$ is $-1/y_j$, and its evaluation from s_j to infinity is $1/s_j$. Hence the integral in lemma (iv) is bigger than or equal to

$$\alpha_j\ \mu(H_j) - \frac{\alpha_j(\alpha_j + \beta_j)[\mu^2(H_j) + \sigma^2(H_j)]}{2} - \frac{\alpha_j[\mu^2(H_j) + \sigma^2(H_j)]}{s_j}$$

Proof of bounding theorem. Let $q(t)$ be the density function for traversing the path in figure 15 by time t. The probability of reaching state D in figure 14 before time T is given by the convolution integral for sequential independent events

$$D(T) = \int_0^T q(t) \int_0^{T-t} \exp(-\varepsilon_i x_i)[1 - F_{i,2}(x_i)]...[1 - F_{i,b_1}(x_i)]$$

$$\vdots$$

$$\int_0^{T-t-x_1-...-x_{m-1}} \exp(-\varepsilon_m x_m)[1 - F_{m,2}(x_m)]...[1 - F_{m,b_m}(x_m)]$$

$$\int_0^{T-t-x_1-...-x_m} \alpha_1 \exp[-(\alpha_1 + \beta_1)y_1][1 - G_{1,2}(y_1)]...[1 - G_{1,c_1}(y_1)]$$

$$\vdots$$

$$\int_0^{T-t-x_1-...-y_{n-1}} \alpha_n \exp[-(\alpha_n + \beta_n)y_n][1 - G_{n,1}(y_n)]...[1 - G_{n,c_n}(y_n)]$$

$$dy_n...dy_1 \; dF_{m,1}(x_m)...dF_{1,1}(x_1) \; dt$$

Working with just the limits of integration

$$\int_0^{T-\Delta} \int_0^{r_1} \cdots \int_0^{r_m} \int_0^{s_1} \cdots \int_0^{s_n} \leq D(T) \leq \int_0^T \int_0^\infty \cdots \int_0^\infty \int_0^\infty \cdots \int_0^\infty$$

where $\Delta = r_1 + r_2 + ... + r_m + s_1 + s_2 + ... + s_n$.

The theorem is proved by applying the inequalities in the proposition to the integrals in the above inequality for $D(T)$.

Algebraic Bounds for $Q(T)$

Convenient bounds for $Q(T)$, the probability of traversing the path in figure 15 by time T, are

$$\frac{\lambda_1...\lambda_k T^k}{k!}\left[1 - \frac{(\lambda_1 + \gamma_1 + ... + \lambda_k + \gamma_k)T}{k+1}\right] \leq Q(T) \leq \frac{\lambda_1...\lambda_k T^k}{k!}$$

These bounds are tight if $(\lambda_1 + \gamma_1 + ... + \lambda_k + \gamma_k)T$ is small.

The derivation of the upper bound is easy. The probability $Q(T)$ is given by the convolution integral

$$\begin{aligned}Q(T) &= \int_0^T \lambda_1 \exp[-(\lambda_1 + \gamma_1)t_1]... \int_0^{T-t_1-...-t_{k-1}} \lambda_k \exp[-(\lambda_k + \gamma_k)t_k]\; dt_k...dt_1 \\ &\leq \lambda_1...\lambda_k \int_0^T ... \int_0^{T-t_1-...-t_{k-1}} dt_k...dt_1 \\ &= \lambda_1...\lambda_k \frac{T^k}{k!}\end{aligned}$$

The lower bound requires the preliminary result that

$$\int_0^T t(T-t)^n\; dt = \frac{T^{n+2}}{(n+2)(n+1)}$$

which can be obtained from the integration by parts formula

$$\begin{aligned}\int_0^T t(T-t)^n\; dt &= \frac{-t(T-t)^{n+1}}{n+1}\Bigg|_0^T + \int_0^T \frac{(T-t)^{n+1}}{n+1} dt \\ &= \quad 0 \quad + \frac{T^{n+2}}{(n+2)(n+1)}\end{aligned}$$

The derivation of the lower bound proceeds by induction. The first step is trivial. The inductive step is

$$\begin{aligned}&\int_0^T \lambda_1 \exp[-(\lambda_1 + \gamma_1)t_1] \int_0^{T-t_1} \lambda_2 \exp[-(\lambda_2 + \gamma_2)t_2]... \int_0^{T-t_1-...-t_{k-1}} \lambda_k \exp[-(\lambda_k + \gamma_k)t_k]\; dt_k...dt_1 \\ &\geq \int_0^T \lambda_1 \exp[-\lambda_1 + \gamma_1)t_1]\left\{\frac{\lambda_2...\lambda_k(T-t_1)^{k-1}}{(k-1)!}\left[1 - \frac{(\lambda_2 + \gamma_2 + ... + \lambda_k + \gamma_k)(T-t_1)}{k}\right]\right\} dt_1 \\ &= \int_0^T \lambda_1 \exp[-(\lambda_1 + \gamma_1)t_1]\frac{\lambda_2...\lambda_k(T-t_1)^{k-1}}{(k-1)!}\; dt_1 \\ &\quad - \int_0^T \lambda_1 \exp[-(\lambda_1 + \gamma_1)t_1]\frac{\lambda_2...\lambda_k(\lambda_2 + \gamma_2 + ... + \lambda_k + \gamma_k)(T-t_1)_k}{k!}\; dt_1 \\ &\geq \int_0^T \lambda_1[1 - (\lambda_1 + \gamma_1)t_1]\frac{\lambda_2...\lambda_k(T-t_1)^{k-1}}{(k-1)!}\; dt_1 \\ &\quad - \int_0^T \lambda_1 \frac{\lambda_2...\lambda_k(\lambda_2 + \gamma_2 + ... + \lambda_k + \gamma_k)(T-t_1)^k}{k!}\; dt_1 \\ &= \lambda_1...\lambda_k\frac{T^k}{k!} - \lambda_1...\lambda_k(\lambda_1 + \gamma_1)\frac{T^{k+1}}{(k+1)k(k-1)!} \\ &\quad - \lambda_1...\lambda_k(\lambda_2 + \gamma_2 + ... + \lambda_k + \gamma_k)\frac{T^{k+1}}{(k+1)!} \\ &= \lambda_1...\lambda_k\frac{T^k}{k!}\left[1 - \frac{(\lambda_1 + \gamma_1 + ... + \lambda_k + \gamma_k)T}{k+1}\right]\end{aligned}$$

which is the lower bound.

Concluding Remarks

The SURE program is a flexible, user-friendly reliability analysis tool. The program provides a rapid computational capability for semi-Markov models useful in describing the fault-handling behavior of fault-tolerant computer systems. The only modeling restriction imposed by the program is that the nonexponential recovery transitions must be fast in comparison to the mission time—a desirable attribute of all fault-tolerant systems. The SURE reliability analysis method utilizes a fast bounding theorem based on means and variances and a fast bounding theorem based on means and percentiles. These bounding theorems enable the calculation of upper and lower bounds on system reliability. The upper and lower bounds are typically within about 5 percent of each other. Since the computation method is extremely fast, large state spaces are not a problem.

NASA Langley Research Center
Hampton, Virginia 23665-5225
November 24, 1987

Appendix A

Basis for SURE's Lower Bound Parameters

The lower bound in White's theorem contains several free parameters—the s_j and r_i parameters. The theorem is true for all values of $s_j > 0$ and $r_i > 0$. Although a technique to choose the globally optimal value of these parameters has not been developed, the values used by SURE,

$$r_i = \{2T[\mu^2(F_i^*) + \sigma^2(F_i^*)]\}^{1/3}$$

$$s_j = \left\{T\frac{[\mu^2(H_j) + \sigma^2(H_j)]}{\mu(H_j)}\right\}^{1/2}$$

can be shown to be nearly optimal. In this section, this is demonstrated.

We will consider paths with only one class 2 or one class 3 path step and derive the optimal value for such paths.

Class 2 Path Step

Suppose we have a path with only one class 2 path step and no class 3 path steps. The lower bound would be

$$\mathrm{LB} = Q(T - r_i)\ \rho(F_i^*)\left[1 - \varepsilon_i\mu(F_i^*) - \frac{\mu^2(F_i^*) + \sigma^2(F_i^*)}{r_i^2}\right]$$

For simplicity, the following abbreviations are used:

$$\rho = \rho(F_i^*)$$

$$\mu = \mu(F_i^*)$$

$$m_2 = \mu^2(F_i^*) + \sigma^2(F_i^*)$$

$$r = r_i$$

$$\varepsilon = \varepsilon_i$$

Thus, we have

$$\mathrm{LB} = Q(T - r)\ \rho\left[1 - \varepsilon\mu - \frac{m_2}{r^2}\right]$$

If there are k class 1 path steps, then the above expression can be written by using White's algebraic $Q(T - \Delta)$ formula as

$$\mathrm{LB} = (T - r)^k A\ \rho\left[1 - \varepsilon\mu - \frac{m_2}{r^2}\right]$$

for some constant A. Taking the derivative of LB with respect to r gives

$$\mathrm{LB}'(r) = -k(T - r)^{k-1} A\ \rho\left[1 - \varepsilon\mu - \frac{m_2}{r^2}\right] + (T - r)^k A\ \rho\left[\frac{2m_2}{r^3}\right]$$

Setting $\mathrm{LB}'(r)$ equal to zero gives

$$-k\left(1 - \varepsilon\mu - \frac{m_2}{r^2}\right) + (T - r)\left(\frac{2m_2}{r^3}\right) = 0$$

Since $\varepsilon\mu$ is virtually 0,

$$kr^3 + (2-k)m_2 r - 2Tm_2 = 0$$

This cubic equation has one real root and two complex roots. The real root is given by

$$r = \left[-\frac{b}{2} + \left(\frac{b^2}{4} + \frac{a^3}{27}\right)^{1/2}\right]^{1/3} + \left[-\frac{b}{2} - \left(\frac{b^2}{4} + \frac{a^3}{27}\right)^{1/2}\right]^{1/3}$$

where

$$a = \frac{(2-k)m_2}{k}$$

$$b = \frac{-2Tm_2}{k}$$

Since a^3 is small with respect to b^2 (i.e., $\frac{a^3}{b^2} \approx \frac{m_2}{T^2}$) the above root is approximately

$$r = \left(\frac{2Tm_2}{k}\right)^{1/3}$$

For computational efficiency the SURE program always uses $k = 1$. Note that for systems using three-way voting, the paths contributing the most to the probability of system failure contain only one class 1 path step. In such models, $k = 1$ is the best choice. Furthermore, since r is insensitive to small changes in k, the use of $k = 1$ leads to a lower bound very close to the optimal one. Using $k = 1$,

$$r = (2Tm_2)^{1/3}$$

Class 3 Path Step

Suppose we have a path with only one class 3 path step and no class 2 path steps. The lower bound would be

$$\text{LB} = Q(T - s_j)\alpha\left\{\mu(H_j) - [\mu^2(H_j) + \sigma^2(H_j)]\left(\frac{\alpha_j + \beta_j}{2} + \frac{1}{s_j}\right)\right\}$$

For simplicity, the following abbreviations are used:

$$\mu = \mu(H_j)$$

$$m_2 = \mu^2(H_j) + \sigma^2(H_j)$$

$$s = s_j$$

$$\alpha = \alpha_j$$

$$\beta = \beta_j$$

Thus we have

$$\text{LB} = Q(T - s)\alpha\left[\mu - m_2\left(\frac{\alpha + \beta}{2} + \frac{1}{s}\right)\right]$$

If there are k class 1 path steps then the above expression can be written by using White's algebraic formula for $Q(T - \Delta)$ as

$$\text{LB} = (T - s)^k A\ \alpha\left[\mu - \frac{(\alpha + \beta)m_2}{2} - \frac{m_2}{s}\right]$$

for some constant A. Taking the derivative of LB with respect to r gives

$$\mathrm{LB}'(s) = -k(T-s)^{k-1}A\;\alpha\left[\mu - \frac{(\alpha+\beta)m_2}{2} - \frac{m_2}{s}\right] + A(T-s)^k\alpha\left[\frac{m_2}{s^2}\right]$$

Setting $\mathrm{LB}'(s)$ equal to 0 gives

$$-k\left[\mu - \frac{(\alpha+\beta)m_2}{2} - \frac{m_2}{s}\right] + (T-s)\left(\frac{m_2}{s^2}\right) = 0$$

Since $(\alpha+\beta)m_2$ is approximately 0,

$$\mu k s^2 - (k-1)m_2 s - Tm_2 = 0$$

For computational efficiency the SURE program always uses $k = 1$. Note that for systems using three-way voting, the paths contributing the most to the probability of system failure contain only one class 1 path step. As before, $k = 1$ is the best choice for such systems:

$$\mu s^2 - Tm_2 = 0$$

Thus

$$s = \left(\frac{Tm_2}{\mu}\right)^{1/2}$$

Appendix B

Derivation of SURE Parameters for Models With Fast Exponentials Competing With a General Transition

This appendix gives the mathematical basis for the SURE program's calculation of the parameters needed by White's theorem for models with fast exponentials competing with a general transition. The semi-Markov model of figure 20 is analyzed.

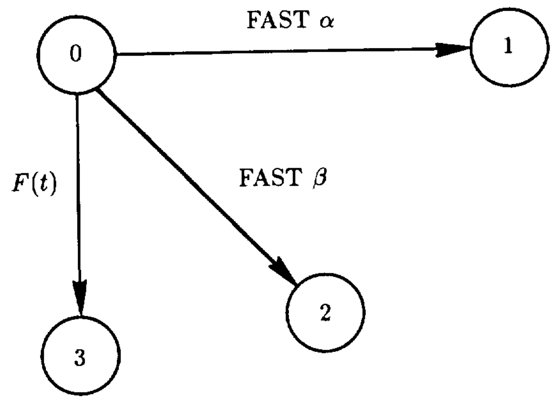


Figure 20. Model with fast exponential transitions.

The α and β transitions are exponential but fast. Therefore, it is necessary to compute the transition probabilities and the mean and variance of the holding time in state 0 along with each transition's conditional mean and variance. The following mathematics derives the formulas used by the SURE program to determine these parameters when the user specifies his model as follows:

$0,1 = \text{FAST}\ \alpha;$
$0,2 = \text{FAST}\ \beta;$
$0,3 = <\mu(F^*),\ \sigma(F^*),\ \rho(F^*)>$

The asterisk is used to indicate that the parameters are defined in terms of the conditional distributions as discussed previously. The holding time in state 0, $\mu(H_o)$, is

$$\mu(H_o) = \int_0^\infty \exp[-(\alpha + \beta)t][1 - F(t)]\, dt$$

$$= \int_0^\infty \exp[-(\alpha + \beta)t]\, dt - \int_0^\infty \exp[-(\alpha + \beta)t]\, F(t)\, dt$$

$$= \frac{1}{\alpha+\beta} - \int_0^\infty \exp[-(\alpha + \beta)t]\, F(t)\, dt$$

Integration by parts	
$u = F(t)$	$dv = \exp[-(\alpha + \beta)t]\, dt$
$du = f(t)\, dt$	$v = -\dfrac{1}{\alpha + \beta}\exp[-(\alpha + \beta)t]$

(Equation continued on next page)

$$= \frac{1}{\alpha + \beta} - \frac{F(t)}{\alpha + \beta} \exp[-(\alpha + \beta)t]\Big|_0^\infty - \int_0^\infty \frac{1}{\alpha + \beta} \exp[-(\alpha + \beta)t]\, f(t)\, dt$$
$$= \frac{1}{\alpha + \beta} - \frac{1}{\alpha + \beta} \int_0^\infty \exp[-(\alpha + \beta)t]\, f(t)\, dt$$
$$= \frac{1}{\alpha + \beta}[1 - \rho(F^*)]$$

(Note: Clearly, $\rho(F^*)$ must be less than 1 in order to have a positive mean holding time.) The second moment of the holding time is

$$\mu^2(H_o) + \sigma^2(H_o) = 2\int_0^\infty t \exp[-(\alpha + \beta)t]\, [1 - F(t)]\, dt$$
$$= 2\int_0^\infty t \exp[-(\alpha + \beta)t]\, dt - 2\int_0^\infty t \exp[-(\alpha + \beta)t]\, F(t)\, dt$$
$$= \frac{2}{(\alpha + \beta)^2} - 2\int_0^\infty t \exp[-(\alpha + \beta)t]\, F(t)\, dt$$

Integration by parts

$$u = F(t) \qquad dv = t \exp[-(\alpha + \beta)t]\, dt$$
$$du = f(t)\, dt \quad v = \frac{-t \exp[-(\alpha + \beta)t]}{\alpha + \beta} - \frac{1}{(\alpha + \beta)^2} \exp[-(\alpha + \beta)t]$$

$$= \frac{2}{(\alpha + \beta)^2} - 2\int_0^\infty \frac{t}{\alpha + \beta} \exp[-(\alpha + \beta)t]\, f(t)\, dt - 2\int_0^\infty \frac{1}{(\alpha + \beta)^2}$$
$$\times \exp[-(\alpha + \beta)t]\, f(t)\, dt$$
$$= \frac{2}{(\alpha + \beta)^2} - \frac{2}{\alpha + \beta}\rho(F^*)\, \mu(F^*) - \frac{2}{(\alpha + \beta)^2}\rho(F^*)$$
$$= \frac{2}{(\alpha + \beta)^2}\{1 - \rho(F^*)[1 + (\alpha + \beta)\, \mu(F^*)]\}$$

(Note: The user must exercise care when mixing FAST exponentials with other general recoveries to prevent an inconsistent specification. It is necessary that $\rho(F^*)[1 + (\alpha + \beta)\, \mu(F^*)]$ be less than or equal to 1 in order for the second moment to be nonnegative.) The probability that the α transition is successful is

$$\rho(\alpha^*) = \int_0^\infty \alpha \exp[-(\alpha + \beta)t]\, [1 - F(t)]\, dt$$
$$= \alpha\, \mu(H_o)$$

The conditional mean time from state 0 to 1 given that this transition is successful is

$$\mu(\alpha^*) = \frac{1}{\rho(\alpha^*)} \int_0^\infty t\alpha \exp[-(\alpha + \beta)t][1 - F(t)]\, dt$$
$$= \frac{\alpha}{2\rho(\alpha^*)} \left\{2\int_0^\infty t \exp[-(\alpha + \beta)t][1 - F(t)]\, dt\right\}$$
$$= \frac{\alpha}{2\rho(\alpha^*)} \left[\mu^2(H_o) + \sigma^2(H_o)\right]$$

The conditional second moment of the transition time from state 0 to 1 given that this transition is successful is

$$\mu^2(\alpha^*) + \sigma^2(\alpha^*) = \frac{1}{\rho(\alpha^*)} \int_0^\infty t^2 \alpha \exp[-(\alpha + \beta)t][1 - F(t)]\, dt$$

$$= \frac{1}{\rho(\alpha^*)} \left\{ \int_0^\infty t^2 \alpha \exp[-(\alpha + \beta)t]\, dt - \int_0^\infty t^2 \alpha \exp[-(\alpha+\beta)t]\, F(t)\, dt \right\}$$

$$= \frac{1}{\rho(\alpha^*)} \left\{ \frac{\alpha}{\alpha+\beta} \frac{2}{(\alpha+\beta)^2} - \int_0^\infty t^2 \alpha \exp[-(\alpha + \beta)t]\, F(t)\, dt \right\}$$

$$\boxed{\begin{array}{l} \text{Integration by parts} \\ \quad u = F(t) \qquad dv = \alpha t^2 \exp[-(\alpha + \beta)t]\, dt \\ \quad du = f(t)\, dt \quad v = -\alpha \exp[-(\alpha + \beta)t] \left[\frac{t^2}{\alpha+\beta} + \frac{2t}{(\alpha+\beta)^2} + \frac{2}{(\alpha+\beta)^3} \right] \end{array}}$$

$$= \frac{1}{\rho(\alpha^*)} \frac{2\alpha}{(\alpha+\beta)^3} - [0] - \int_0^\infty \alpha \exp[-(\alpha + \beta)t] \left[\frac{t^2}{\alpha+\beta} + \frac{2t}{(\alpha+\beta)^2} \right.$$

$$\left. + \frac{2}{(\alpha+\beta)^3} \right] f(t)\, dt$$

$$= \frac{1}{\rho(\alpha^*)} \left[\frac{2\alpha}{(\alpha+\beta)^3} - \frac{\alpha}{\alpha+\beta} \int_0^\infty t^2 \exp[-(\alpha + \beta)t]\, f(t)\, dt \right.$$

$$- \frac{2\alpha}{(\alpha+\beta)^2} \int_0^\infty t \exp[-(\alpha + \beta)t]\, f(t)\, dt$$

$$\left. - \frac{2\alpha}{(\alpha+\beta)^3} \int_0^\infty \exp[-(\alpha + \beta)t]\, f(t)\, dt \right]$$

$$= \frac{1}{\rho(\alpha^*)} \left\{ \frac{2\alpha}{(\alpha+\beta)^3} - \frac{\alpha}{\alpha+\beta} \rho(F^*) \left[\mu^2(F^*) + \sigma^2(F^*)\right] - \frac{2\alpha}{(\alpha+\beta)^2} \rho(F^*)\, \mu(F^*) \right.$$

$$\left. - \frac{2\alpha}{(\alpha+\beta)^3} \rho(F^*) \right\}$$

$$= \frac{2}{\alpha+\beta} \frac{\alpha}{\rho(\alpha)(\alpha+\beta)^2} \left\{1 - \rho(F^*)[(\alpha+\beta)\, \mu(F^*) + 1]\right\}$$

$$- \frac{\alpha}{\rho(\alpha^*)(\alpha+\beta)} \rho(F^*) \left[\mu^2(F^*) + \sigma^2(F^*)\right]$$

$$= \frac{2}{\alpha+\beta} \mu(\alpha^*) - \frac{\alpha}{\rho(\alpha^*)(\alpha+\beta)} \rho(F^*) \left[\mu^2(F^*) + \sigma^2(F^*)\right]$$

$$= \frac{1}{\alpha+\beta} \left\{ 2\mu(\alpha^*) - \frac{\alpha}{\rho(\alpha^*)} \rho(F^*) \left[\mu^2(F^*) + \sigma^2(F^*)\right] \right\}$$

(Note: The user must exercise care when mixing FAST exponentials with other general recoveries to prevent an inconsistent specification. It is necessary that

$$2\mu(\alpha^*) - \frac{\alpha}{\rho(\alpha^*)} \rho(F^*) \left[\mu^2(F^*) + \sigma^2(F^*)\right] \geq 0$$

in order for the second moment to be nonnegative.) The generalization to more than one general fast transition (say $F_1, F_2, ..., F_n$) and more than two fast exponentials (say $\lambda_1, \lambda_2, ..., \lambda_m$) can be obtained by applying the following substitutions in the above formulas:

$$\rho(F^*) \longrightarrow \sum_{i=1}^{n} \rho(F_i^*)$$

$$\mu(F^*) \longrightarrow \frac{\sum_{i=1}^{n} \rho(F^*)\ \mu(F_i^*)}{\sum_{i=1}^{n} \rho(F_i^*)}$$

$$\sigma^2(F^*) + \mu^2(F^*) \longrightarrow \frac{\sum_{i=1}^{n} \rho(F_i^*) \left[\mu^2(F_i^*) + \sigma^2(F_i^*)\right]}{\sum_{i=1}^{n} \rho(F_i^*)}$$

$$(\alpha + \beta) \longrightarrow \sum_{i=1}^{m} \lambda_i$$

Appendix C

Mathematical Basis of the QTCALC=1 Algorithm

The probability $Q(T)$ can be solved with the use of an exponential matrix algorithm. (See ref. 13.) If A represents the transition matrix of the Markov process associated with just the n class 1 transitions and N/D represents the rational fraction which occupies the 9th diagonal position in the Páde table for $exp(Z)$, then

$$N = N(Z) = \sum_{0}^{9} c_i Z^i$$

$$D = D(Z) = \sum_{0}^{9} c_i (-Z)^i$$

where

$$c_i = \frac{(18 - i)!\ 9!}{18!\ i!\ (9 - i)!}$$

The algorithm used to compute $E = \exp(A * T)$ when `QTCALC=1` is specified is:

1. Compute $C = A * T$
2. Find $s = Max$ {Binary exponents of components of C}
3. If $s > 1$ then $B \longleftarrow C * 2^{-s}$ else $B \longleftarrow C$
4. Compute $N(B)$ and $D(B)$
5. Compute $E = D^{-1}N$
6. If $s > 1$ then perform $E \longleftarrow E * E$ s times

Appendix D

Error Messages

The following error messages are generated by the SURE system. These are listed in alphabetical order:

ARGUMENT TO EXP FUNCTION MUST BE < 8.80289E+01—The argument to the EXP function is too large.

ARGUMENT TO LN OR SQRT MUST BE > 0—The LN and SQRT functions require positive arguments.

ARGUMENT TO STANDARD FUNCTION MISSING—No argument was supplied for a standard function.

COMMA EXPECTED—Syntax error; a comma is needed.

CONSTANT EXPECTED—Syntax error; a constant is expected.

DELTA > TIME—The value of Δ used in the lower bound (i.e, $Q(T-\Delta)$ is larger than the mission time. This can lead to a very poor lower bound. This is usually caused by using the fast transition specification method to describe a slow transition (i.e., a very slow recovery transition).

DIVISION BY ZERO NOT ALLOWED—A division by 0 was encountered when evaluating the expression.

ERROR OPENING FILE – <vms status>—The SURE system was unable to open the indicated file.

FILE NAME EXPECTED—Syntax error; the file name is missing.

FILE NAME TOO LONG—File names must be 80 or less characters.

"id" CHANGED TO x—The value of the identifier "id" is being changed to x.

"id" CHANGED TO x TO y—The range of the variable "id" is being changed.

"id" NOT FOUND—The system is unable to SHOW the identifier since it has not yet been defined.

IDENTIFIER EXPECTED—Syntax error; identifier expected here.

IDENTIFIER NOT DEFINED—The identifier entered has not yet been defined.

ILLEGAL CHARACTER—The character used is not recognized by SURE.

ILLEGAL INPUT VALUE—A nonnumeric character was entered in response to the INPUT command prompt.

ILLEGAL STATEMENT—The command word is unknown by the system.

INPUT ALREADY DEFINED AS THE VARIABLE—An attempt was made to input a value for an identifier that was already defined as the variable.

INPUT LINE TOO LONG—The command line exceeds the 100-character limit.

INTEGER EXPECTED—Syntax error; an integer is expected.

LEE @ REQUIRES THREE PARAMETERS—The @ statement requires three parameters in the LEE mode.

MORE THAN ONE SOURCE STATE IN MODEL—The model entered by the user has more than one source state (i.e., a state with no transitions into it). If a start state has been specified by a START command, it is used. Otherwise, the program arbitrarily chooses a start state.

MUST BE IN "READ" MODE—The INPUT command can be used only in a file processed by a READ command.

NO RUNS MADE YET—The ORPROB command was called before any runs were made.

NUMBER TOO LONG—Only 15 digits/characters allowed per number.

ONLY 1 VARIABLE ALLOWED—Only one variable can be defined per model.

ONLY 100 VARIABLE RESULTS STORED—The ORPROB command can only process the first 100 values of the variable per run.

PRUNING TOO SEVERE—The specified level of pruning is too large to guarantee that the bounds have WARNDIG digits of accuracy.

Q(T) INACCURATE—The entered mission time is too large for the default value of QTCALC. Therefore, the upper and lower bounds are very far apart. Set QTCALC equal to 1.

Q(T) $\sim$ x DIGITS—The matrix exponential algorithm cannot guarantee more than x digits accuracy in the Q(T) calculation.

RATE TOO FAST—The upper and lower bounds are valid, but, an exponential transition in the model is too fast to permit close upper and lower bounds.

REAL EXPECTED—A floating point number is expected here.

RECOVERY TOO SLOW—The upper and lower bounds are valid, but a nonexponential transition in the model is too slow to permit close upper and lower bounds.

SEMICOLON EXPECTED—Syntax error; a semicolon is needed.

START STATE ASSUMED TO BE x—There was no source state in the model and no start was specified via a START command so the program arbitrarily selected x as the start state.

ST. DEV TOO BIG—The standard deviation of a fast distribution is too large to permit close upper and lower bounds; however, the bounds are valid.

SUB-EXPRESSION TOO LARGE, i.e., > 1.70000E+38—An overflow condition was encountered when evaluating the expression.

THIS CONSTRUCT NOT PERMITTED IN LEE MODE—This construct is not allowed while in the LEE mode.

THIS CONSTRUCT NOT PERMITTED IN WHITE MODE—This construct is not allowed while in the WHITE mode.

TRANSITION NOT FOUND—The system is unable to SHOW the transition because it has not yet been defined.

TRUNC TOO SMALL—The value of TRUNC is probably not large enough to guarantee that the upper bound is valid for this model. The user should rerun the model with a higher value of TRUNC.

VMS FILE NOT FOUND—The file indicated on the READ command is not present on the disk. (Note: make sure your default directory is correct.)

0 STATES IN MODEL—The RUN command found no states in the model.

*** ERROR: HOLDING TIME AT x NOT DEFINED—The holding time information (required in LEE mode) for state x has not yet been provided.

*** ERROR: INCONSISTENT SPECIFICATION OF FAST TRANSITIONS AT STATE n—When mixing FAST exponentials with a general fast transition (i.e., using conditional parameters) from a state it is possible to do so in an inconsistent manner. The following conditions must be satisfied in order to have a consistent specification (see appendix A):

$$1 - \rho(F^*)[1 + (\alpha + \beta)\ \mu(F^*)] > 0$$

$$2\mu(\alpha) - \frac{\alpha}{\rho(\alpha)}\ \rho(F^*)[\mu^2(F^*) + \sigma^2(F^*)] \geq 0$$

This error message indicates that one of these conditions has been violated at state n.

*** ERROR: INSTANTANEOUS TRANSITION AT STATE n—One of the transitions from state n has been defined with a mean of zero.

*** ERROR: SUM OF EXITING PROBABILITIES IS NOT 1 AT STATE n—The sum of the transition probabilities of the fast transitions from state n does not add up to 1.

*** ERROR: THE FAST EXPONENTIALS HAVE ZERO PROBABILITY OF OCCURRENCE AT STATE n—State n containing mixed fast transition specifications (i.e., some described by FAST exponentials and some by conditional parameters) has been overspecified such that the FAST exponential recoveries have zero probability of occurrence. This occurs when the sum of the transition probabilities of the transitions described by conditional parameters is 1.

*** ILLEGAL STATE NUMBER—The state number is negative or greater than the maximum state limit (Default = 10 000, set at SURE compilation time).

*** STATE x HOLDING TIME ALREADY ENTERED—The LEE-mode, holding-time information for state x has already been entered.

*** THE *CALC* EXPRESSION MUST BE ON 1 LINE—The mathematical expression processed by the CALC function must fit on one line. Constant subexpressions can be defined prior to the CALC function and used to simplify the CALC expression.

*** TRANSITION X → Y ALREADY ENTERED—The user is attempting to reenter the same transition again.

*** VARIABLES INCONSISTENT BETWEEN RUNS—The ORPROB command cannot process the preceding runs since they did not use the same variable or the same values of the variable.

*** WARNING: REMAINDER OF INPUT LINE IGNORED—Any commands that followed the READ command on the same line were ignored.

*** WARNING: RUN-TIME PROCESSING ERRORS—Computation overflow occurred during execution.

*** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN— Syntax errors were present during the model description process.

*** WARNING: VARIABLE CHANGED!—If previous transitions have been defined using a variable and the variable name is changed, inconsistencies can result in the values of the transitions.

= EXPECTED—Syntax error; the = operator is needed.

> EXPECTED—Syntax error; the closing bracket > is missing.

) EXPECTED—A right parenthesis is missing in the expression.

] EXPECTED—A right bracket is missing in the expression.

References

1. Geist, Robert M.; and Trivedi, Kishor S.: Ultrahigh Reliability Prediction in Fault-Tolerant Computer Systems. *IEEE Trans. Comput.*, vol. C-32, no. 12, Dec. 1983, pp. 1118–1127.
2. White, Allan L.: *Upper and Lower Bounds for Semi-Markov Reliability Models of Reconfigurable Systems.* NASA CR-172340, 1984.
3. Butler, Ricky W.: *The Semi-Markov Unreliability Range Evaluator (SURE) Program.* NASA TM-86261, 1984.
4. Lee, Larry D.: *Reliability Bounds for Fault-Tolerant Systems With Competing Responses to Component Failures.* NASA TP-2409, 1985.
5. White, Allan L.: *Synthetic Bounds for Semi-Markov Reliability Models.* NASA CR-178008, 1985.
6. Butler, Ricky W.: *The SURE Reliability Analysis Program.* NASA TM-87593, 1986.
7. Siewiorek, Daniel P.; and Swarz, Robert S.: *The Theory and Practice of Reliable System Design.* Digital Press, c.1982.
8. Lala, Jaynarayan H.; and Smith, T. Basil, III: *Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer. Volume III—FTMP Test and Evaluation.* NASA CR-166073, 1983.
9. Trivedi, Kishor; Dugan, Joanne Bechta; Geist, Robert; and Smotherman, Mark: Modeling Imperfect Coverage in Fault-Tolerant Systems. *The Fourteenth International Conference on Fault-Tolerant Computing—FTCS 14, Digest of Papers,* 84CH2050-3, IEEE, 1984, pp. 77–82.
10. Bavuso, S. J.; and Petersen, P. L.: *CARE III Model Overview and User's Guide (First Revision).* NASA TM-86404, 1985.
11. Goldberg, Jack; Kautz, William H.; Melliar-Smith, P. Michael; Green, Milton W.; Levitt, Karl N.; Schwartz, Richard L.; and Weinstock, Charles B.: *Development and Analysis of the Software Implemented Fault-Tolerance (SIFT) Computer.* NASA CR-172146, 1984.
12. White, Allan L.: Reliability Estimation for Reconfigurable Systems With Fast Recovery. *Microelectron. & Reliab.*, vol. 26, no. 6, 1986, pp. 1111–1120.
13. Ward, Robert C.: Numerical Computation of the Matrix Exponential With Accuracy Estimate. *SIAM J. Numer. Analys.*, vol. 14, no. 4, Sept. 1977, pp. 600–615.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TP-2764	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle SURE Reliability Analysis—Program and Mathematics		5. Report Date March 1988
		6. Performing Organization Code
7. Author(s) Ricky W. Butler and Allan L. White		8. Performing Organization Report No. L-16263
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No. 505-66-21-01
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		13. Type of Report and Period Covered Technical Paper
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

The SURE program is a new reliability analysis tool for ultrareliable computer system architectures. The program is based on computational methods recently developed at the NASA Langley Research Center. These methods provide an efficient means for computing accurate upper and lower bounds for the deathstate probabilities of a large class of semi-Markov models. Once a semi-Markov model is described by using a simple input language, the SURE program automatically computes the upper and lower bounds on the probability of system failure. A parameter of the model can be specified as a variable over a range of values directing the SURE program to perform a sensitivity analysis automatically. This feature, along with the speed of the program, makes it especially useful as a design tool.

17. Key Words (Suggested by Authors(s)) Reliability analysis, Semi-Markov models, Fault tolerance, Reliability modeling, Markov models	18. Distribution Statement Unclassified—Unlimited; Subject Category 65

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 74	22. Price A04