NASA TM-86261

NASA Technical Memorandum 86261

The Semi-Markov Unreliability Range Evaluator (SURE) Program

Ricky W. Butler

July 1984

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

```
ENTER:
     SELECT     RN/NAS-TM-86261
     TERM IN SELECT COMMAND NOT IN DICTIONARY
================================================================================
  6      1       1 RN/NASA-TM-86261
            DISPLAY 06/6/1
      84N29640**#   ISSUE 19   PAGE 3093   CATEGORY 65   RPT#: NASA-TM-86261 NAS
      1.15:86261    84/06/00  33 PAGES   UNCLASSIFIED DOCUMENT
UTTL: The semi-Markov unreliability range evaluator program
AUTH: A/BUTLER, R. W.
CORP: National Aeronautics and Space Administration. Langley Research Center,
      Hampton, Va.    AVAIL.NTIS   SAP: HC A03/MF A01
MAJS: /*ARCHITECTURE (COMPUTERS)/*COMPUTER AIDED DESIGN/*COMPUTER PROGRAMS/*
      COMPUTER SYSTEMS DESIGN/*MARKOV PROCESSES/*RELIABILITY ANALYSIS
MINS: / COMPUTER SYSTEMS PERFORMANCE/ FAULT TOLERANCE/ MATHEMATICAL MODELS
ABA:  Author
```

INTRODUCTION

The traditional methods of computing the state probabilities of a semi-Markov model are either applicable to only a small subclass of models or must be programmed independently for each problem. Furthermore, these methods require the solution of equations which are extremely complex and hence utilize significant computer resources. Recently a new mathematical theorem was proven which enables the efficient computation of the death state probabilities of a large family of semi-Markov models useful for the analysis of the reliability of fault-tolerant architectures. (See ref. 1.) This theorem has been mechanized into a flexible interactive reliability tool--the Semi-Markov Unreliability Range Evaluator (SURE). The SURE program provides the capability for parametric analyses of candidate configurations of a computer system architecture and thus should serve well as a design tool and as a validation aid.

The SURE reliability tool applies to semi-Markov models with the following characteristics:

(1) The transitions of the model must fall into two classes--slow exponential transitions and fast general transitions.

(2) No circuit may exist in the graph structure of the model, i.e., it must be a pure death process. Thus, transient faults cannot be modeled or analyzed in SURE.

The SURE program consists of three modules--the front-end, the computation module, and graphics output module. The front-end and computation modules are implemented in Pascal and should easily "port" to other machines. The graphics output module is written in Fortran but uses the graphics library TEMPLATE. This module will only "port" to users with this library. The interface to this module is very simple and could easily be rewritten for other graphics systems.

The graphics output portion of the program was written by Mr. Dan Palumbo of Langley Research Center. The graphics module has been very useful in demonstrating the capabilities of the SURE system. The author is grateful to Mr. Palumbo for this useful contribution to the SURE program.

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 which involves both slow and fast processes. When these systems are modeled stochastically some state transitions are many orders of magnitude faster than others. The slower transitions correspond to fault arrivals in the system. If the states of the system are delineated properly, then the slow transitions can be obtained from field data and/or MIL STD 217D calculations. These transitions are known to be exponentially distributed. The faster transition rates correspond to the system's response to fault arrivals and can be measured experimentally using fault injection. (Experiments by Charles Stark Draper Laboratory, Inc., on the Fault-Tolerant Multiprocessor, FTMP, computer architecture have demonstrated that these transitions are not exponential; see ref. 2.) The primary problem is to properly model the system so as to facilitate the determination of these transitions. If the model is too coarse the transitions become experimentally unobservable. If the model is too detailed the number of transitions which must be measured can be exorbitant. 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 death states (i.e., the states which represent system failure) within a specified mission time, e.g., 10 hours.

The accuracy of the computational analysis is strongly dependent on the correctness of the mathematical model. The absence of a critical transition from the model can often be far more devastating than a 100 percent error in the estimation of a recovery transition. Unfortunately, experimental validation of the model essentially requires "life-testing" type experiments which are impractical for ultrareliable systems. The only recourse is to rely on the careful scrutiny of the model by system experts to insure that the model correctly represents the behavior of the system. Consequently, it is essential that the assumptions of the modeling exercise be carefully enumerated.

The behavior of a fault-tolerant highly reliable system is extremely complex. Fortunately, most of the detailed instruction level activities of the system do not directly affect the system reliability. The mathematical models must capture the processes that lead to system failure and the system fault-recovery capabilities. The first level of model granularity to consider is thus 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 modules, the number of modules reconfigured out, etc. The transitions correspond to changes in these specified attributes of the system. Certain states in the system represent system failure and others represent fault-free behavior or correct operation in the presence of faults. The model chosen for the system must represent system failure properly. This is difficult because it is even difficult to define exactly what constitutes system failure. System failure is an extremely complex function of external events, software state, and hardware state. The modeler is forced to make either conservative or nonconservative assumptions about what is system failure. If one wishes to say that the reliability of the system is higher than a specific value then conservative assumptions are made. For example, in a TMR system of computers, the presence of two faulty computers is considered system failure whether or not the two faults are actually corrupting data in such a way as to defeat the voting system. If one wishes to say the reliability is not less than some value, then nonconservative assumptions are made. For example, the modeler assumes only certain parts of the system can fail. The problem is further compounded by the plethora of failure modes possible at the module level. The higher the level of model granularity, the more bizarre the failure modes. Typically one must at least consider the following classes of failures:

1. permanent
2. intermittent
3. transient.

But, the diversity within each of these classes is enormous. For example, the permanent class includes single-pin failures such as stuck-at-1s, stuck-at-0s,

inversions, etc. as well as multiple-pin failure modes. Intermittents can occur with arbitrary duration and frequency. Transients can affect multiple pins and last for arbitrary times. In addition, each class can have different arrival distributions and effects on the system. The SURE system can be used to solve models dealing with permanent failures only.

A semi-Markov model of a triad with one spare is given in figure 1. (In this model it is assumed that the spares do not fail while inactive.)

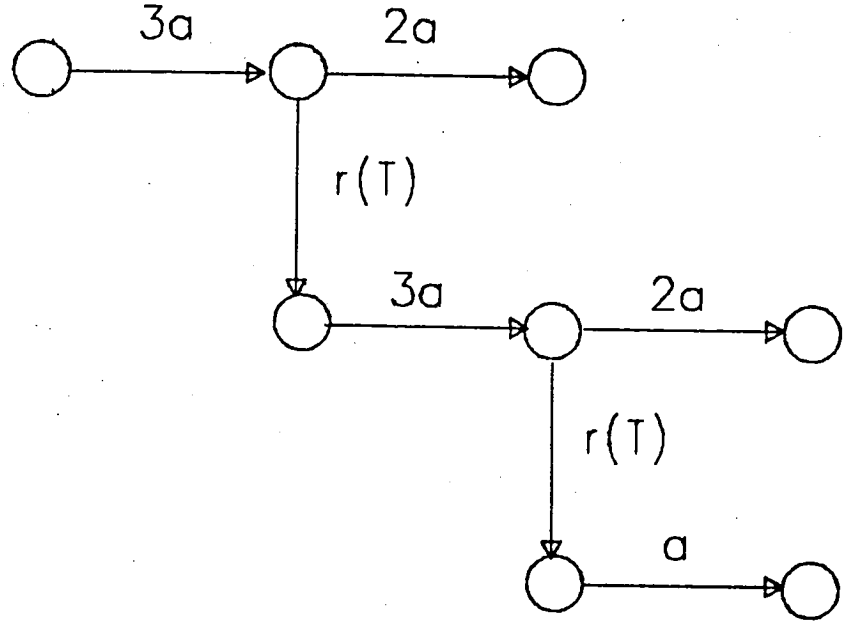


Figure 1. - Semi-Markov model of a triad with 1 spare.

The horizontal transitions represent fault arrivals. These occur with exponential rate "a." The coefficients of "a" represent the number of processors in the configuration which can fail. The vertical transitions represent recovery from a fault. These transitions may have arbitrary distribution and hence the rate is time dependent: r(T). (This time must be local time, i.e., the time since entering the current state, in order to preserve the semi-Markov property of the system.) Since the system uses 3-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.

A NEW MATHEMATICAL RESULT

A recently proven mathematical theorem enables a quick analysis of a large class of semi-Markov models. This theorem was proven by Mr. Allan White of Kentron, Inc. under contract to NASA Langley Research Center and will thus be referred to as White's Theorem. (See ref. 1.) In this section White's Theorem will be discussed but not proven. The reader is referred to reference 1 for details of the proof.

White's Theorem involves a graphical analysis of a semi-Markov model. The theorem provides a means of bounding the probability of traversing a specific path in the model within the specified time. By applying the theorem to every path of the model, the probability of the system reaching any death state can be determined within usually very close bounds. A simple semi-Markov model of the 6-processor SIFT (see ref. 3) computer system will be used to introduce the theorem. This model is illustrated in figure 2.

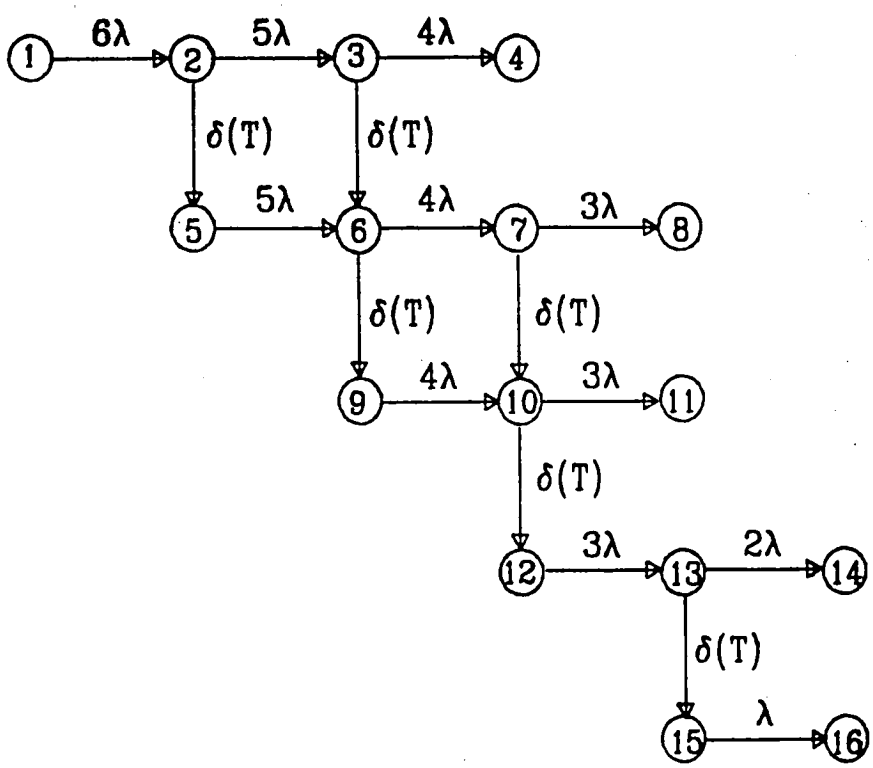


Figure 2. - Semi-Markov model of SIFT.

The horizontal transitions in the model represent fault arrivals. These are assumed to be exponentially distributed and relatively slow. The vertical transitions represent system recoveries by reconfiguration, i.e., 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 mean and variance of the recovery time distribution be specified. The death states of the model are states 4, 8, 11, 14 and 16. Death state 4 represents the case where three processors out of six have failed before the system reconfigures. State 16 represents the case where the system has been completely depleted of processors. The unreliability of the system is precisely the sum of the probabilities of entering each death state. White's Theorem analyzes every path to each death state individually. In the SIFT model the following paths must be considered:

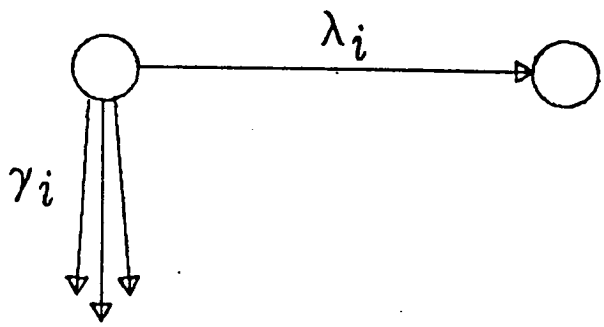
path 1: 1 -> 2 -> 3 -> 4.
path 2: 1 -> 2 -> 3 -> 6 -> 7 -> 8.
path 3: 1 -> 2 -> 5 -> 6 -> 7 -> 8.
path 4: 1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 11.
path 5: 1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 11.
path 6: 1 -> 2 -> 3 -> 6 -> 9 -> 10 -> 11.
path 7: 1 -> 2 -> 5 -> 6 -> 9 -> 10 -> 11.
path 8: 1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14.
path 9: 1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14.
path 10: 1 -> 2 -> 3 -> 6 -> 9 -> 10 -> 12 -> 13 -> 14.
path 11: 1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 14.
path 12: 1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16.
path 13: 1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16.
path 14: 1 -> 2 -> 3 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16.
path 15: 1 -> 2 -> 5 -> 6 -> 7 -> 10 -> 12 -> 13 -> 15 -> 16.

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

Once a particular path has been isolated for analysis, White's Theorem is

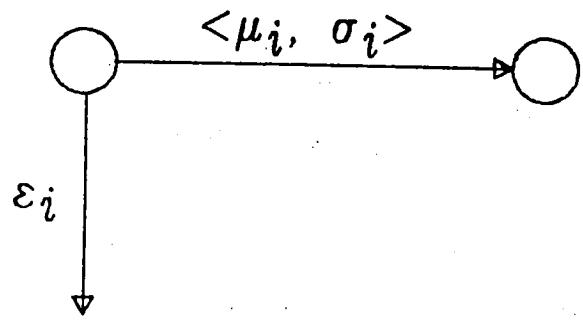
easily applied. Each step along the path must first be classified into one of three classes. These classes are distinguished by the type of transitions leaving the state. Each state along with the transitions leaving it will be referred to as a path step. Only one transition can be on the current path under analysis. This will be referred to as the on-path transition. 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 in the path-step this is classified as a slow off-path transition. Thus the following classes of path-steps are of interest:

Class 1: (SLOW ON-PATH, SLOW OFF-PATH)



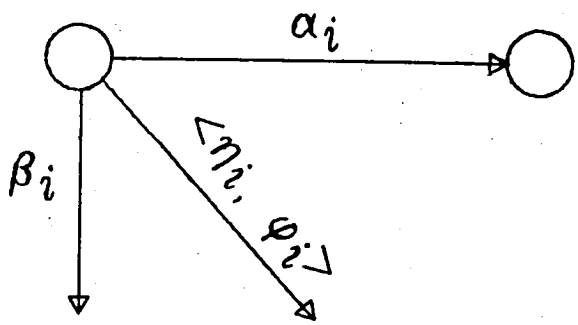
There may be an arbitrary number of slow off-path transitions. If any of the off-path transitions are not slow, then the path-step is in Class 3 below. The path-steps 1 -> 2 and 5 -> 6 in the SIFT model are examples. The transition rate of the on-path transition is λ_i and the sum of the off-path transitions is γ_i.

Class 2: (FAST ON-PATH, SLOW OFF-PATH)



Once again there may be an arbitrary number of slow off-path transitions, ε_i. As before, the off-path exponential transitions can be represented as a single transition with a rate equal to the sum of all the off-path transition rates. The path-steps 2 -> 5 and 3 -> 6 in the SIFT model are examples. The mean and standard deviation of the on-path recovery-time distribution will be referred to as μ_i and σ_i, respectively.

Class 3: (SLOW ON-PATH, FAST OFF-PATH)



This class includes path-steps with both slow and fast off-path transitions. However, only one off-path transition may be fast. The path steps 2 -> 3 and 7 -> 8 in the SIFT model are in this class. The slow on-path transition rate is α_i. The slow off-path transition rates are β_i and the mean and standard deviation of the fast off-path recovery time distribution are η_i and ϕ_i, respectively.

The class FAST ON-PATH, FAST OFF-PATH is not included since the theorem does not apply to models that contain a state with more than one fast recovery-type transition leaving it.

Classical semi-Markov theory has shown that the rearrangement of the path-steps does not affect the probability of entering the death state of the path within a specific time. Thus, the path can be decomposed into two subpaths--the first subpath containing only class 1 path-steps, the second subpath containing only class 2 and class 3 path-steps. The probability of leaving the first subpath by the mission time, T, is of special importance in the analysis. This probability is referred to as E(T). Since this subpath consists only of states with exponential transitions, it is pure Markov.

With the above classification, White's Theorem can now be given:

White's Theorem: If E(T) is the probability of leaving the pure Markov subpath by time T, the on-path recovery time distributions have means μ_i and variances σ_i^2 for $1 < i < m$, the off-path recovery transitions have means η_j and variances ϕ_j^2 for $1 < j < n$, and the slow transition rates, α_j, β_j, and ε_i, are as defined above, then the probability of entering the death state by time T, D(T), is bounded as follows:

$$LB < D(T) < UB$$

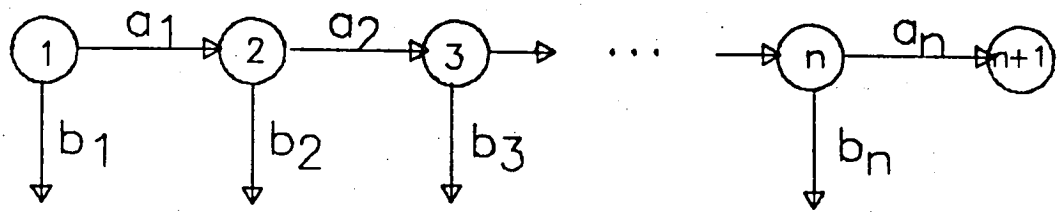
where

$$UB = E(T)\{\prod_{j=1}^{n} \alpha_j \eta_j\}$$

$$LB = E(T-\Delta)\{\prod_{i=1}^{m} \left[1-\varepsilon_i\mu_i - \frac{(\sigma_i^2+\mu_i^2)}{\mu_i}\right] \prod_{j=1}^{n}\alpha_j \left[\eta_j - \frac{(\alpha_j+\beta_j)(\phi_j^2+\eta_j^2)}{2} - \frac{(\phi_j+\eta_j)^2}{\eta_j^{\frac{1}{2}}}\right]\}$$

and $\Delta = \mu_1^{\frac{1}{2}} + \mu_2^{\frac{1}{2}} + \ldots + \mu_m^{\frac{1}{2}} + \eta_2^{\frac{1}{2}} + \ldots + \eta_n^{\frac{1}{2}}$

White's Theorem gives the upper and lower bounds in terms of E(T) defined earlier. Before illustrating the use of the theorem with an example, two simple algebraic approximations for E(T) will be given--one which overestimates and one which underestimates. (See ref. 1.) Suppose we have the following pure Markov submodel:



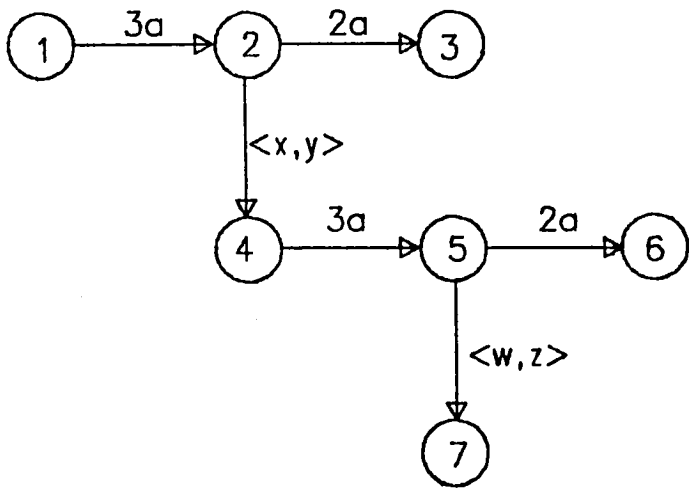
The following algebraic expressions hold, where E(T) is the probability of entering the state n+1 by time T.

$$E(T) \leq E_u(T) = \frac{a_1 a_2 a_3 \ldots a_n T^n}{n!}$$

$$E(T) \geq E_l(T) = E_u(T) \left[1 - T/(n+1) \sum_{i=1}^{n} (a_i + b_i)\right]$$

Furthermore, both $E_u(T)$ and $E_l(T)$ are usually very close to E(T). (See ref. 1.)

To see how White's Theorem is used, consider the following portion of the model:



There is only one path to state 3: 1 -> 2 -> 3. The first path step 1 -> 2 is a class 1 path-step and hence contributes to E(t). The second path-step is class 3 (α_i = 2a, η_i = x, β_i = 0, ϕ_i = y). Thus we have:

$$UB = E_u(T)(2ax)$$

$$LB = E_l(T-\Delta)\ [\ 2a\ (x - 2a(y^2 + x^2)/2 - (y^2 + x^2)/\ x^{1/2})\ ]$$

where

$$E_u(T) = 3aT$$

$$E_l(T-\Delta) = 3a(T-\Delta)\ [\ 1 - (T-\Delta)a/2]$$

$$\Delta = x^{1/2}$$

There is also only 1 path to state 6: 1 -> 2 -> 4 -> 5 -> 6. The path-steps 1 -> 2 and 4 -> 5 are class 1. The path-step 2 -> 4 is class 2 (i.e., $\epsilon_i = 2a$, $\mu_i = x$, $\sigma_i = y$) and the path step 5 -> 6 is class 3 (i.e. $\alpha_j = 2a$, $\beta_j = 0$, $\eta_j = w$, $\phi_j = z$). Thus,

$$UB = E_u(T)(2aw)$$

$$LB = E_l(T-\Delta)[\ 1 - 2ax - (y^2+x^2)/x\ ][\ 2a(w - a(z^2+w^2) - (z^2+w^2)/w^{1/2})\ ]$$

where

$$E_u(T) = (3a)(3a)T^2/2! = 9a^2T^2/2$$

$$E_l(T-\Delta) = [\ (3a)(3a)(T-\Delta)^2/2!\ ]\ [\ 1 - (T-\Delta)(3a + 3a)/3\ ]$$

$$= 9a^2(T-\Delta)^2/2 - 9a^3(T-\Delta)^3$$

and

$$\Delta = x^{1/2} + w^{1/2}$$

THE SURE PROGRAM USER INTERFACE

Basic Program Concept

Understanding the details of the above theory is not necessary to use the SURE program. The user of the program need only be able to describe the semi-Markov model of the system to the SURE program and enter values for the transitions in the model. All of the computations described above are performed automatically by the program. The SURE program utilizes a very simple command-style language for description of the semi-Markov model. This language will be discussed in detail in this section.

The SURE user must first assign numbers to every state in the system. The semi-Markov model is then described by enumerating all of the transitions. First, if the transition is slow then the following syntax 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, e.g., $hour^{-1}$ or sec^{-1}. However, the user must be consistent.) Second, if the transition is fast then the mean and standard deviation (i.e., square root of the variance) of the recovery time distribution must be given. Since recovery rate is roughly inversely proportional to recovery time, the rate is fast when the mean of the recovery time distribution is small. The syntax for such a transition is

```
15,207 = <0.01,0.003>;
```

This defines a fast transition from state 15 to state 207 with a recovery time mean and standard deviation of 0.01 and 0.003, respectively.

The two types of commands described above are the only essential ingredients in the SURE language. However, to increase the flexibility of the SURE program a

few additions were made to the command language. These include:

(1) Constant Definitions,
(2) Expressions,
(3) Variable Definition,
(4) Read From Disk,
(5) Show,
(6) List Options,
(7) Graphics Display Interface,
(8) Miscellaneous Commands.

These additional commands are discussed in the next section.

SURE Command Syntax

Lexical Details. - The state numbers must be integers between 1 and the "MAXSTATE" implementation limit, usually 10000. (This limit can be changed by redefining a Pascal constant and recompiling the SURE program.) The transition rates, means, and standard deviations are floating point numbers. The Pascal REAL syntax is used for these numbers. Thus, all the following would be legal input to SURE:

0.001,
12.34,
1.2E-4.

The number must begin with a digit. Thus .001 is illegal.

The semicolon is used for command termination. Therefore, more than one command may be entered on a line. If commands are entered from a terminal (as opposed to by the READ command described below), then the carriage return is interpreted as a semicolon. Thus, interactive commands do not have to be terminated by an explicit semicolon unless more than one command is entered on

the line.

In interactive mode the SURE system will prompt the user for input by a number followed by a ?:

```
1?
```

The number is a count of the commands entered into the system thus far plus the current one. If there is an error then the command is ignored and the count is not incremented.

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;
23,25 = LAMBDA;
25,26 = <RECOVER,0.001>;
```

Variable Definition. - In order to facilitate "parametric analyses" a variable may be defined. A range is given for this variable. The SURE system will compute the system reliability as a function of this variable. If the system is run in graphics mode (to be described later) then a plot of this function will be made. The following command 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.

Expressions. - When specifying rates or means and standard deviations in a command, a linear combination of constants and a variable may be used:

```
ALPHA = 2E-4;
LAMBDA = 1E-6 TO 1E-4;
5,6 = 7*ALPHA+12*LAMBDA;
5,9 = <3.0*ALPHA*ALPHA + ALPHA*LAMBDA, 0.001*ALPHA*LAMBDA>;
```

The only operations currently supported are addition(+), subtraction(-), and multiplication(*). The only restriction on the use of these operations is that a variable cannot be multiplied by itself (i.e., only linear functions of the variable are allowed; this restriction does not apply to constants). Currently, there has been no need for higher powers of a variable parameter. If higher powers of the variable are needed, this could be added to the system. Arbitrary constant expressions may be used.

Read from Disk. - A sequence of commands may be read from a disk file. The following interactive command reads SURE commands from a disk file named SIFT.MOD:

```
READ SIFT.MOD.
```

If no file name extent is given, the default extent, MOD, is assumed. This feature may be used in conjunction with interactive command input.

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 command:

```
SHOW 654,193;
```

List Options. - The amount of information output by the program is controlled by this command. Four list modes are available:

LIST = 0; -- No output is sent to the terminal.

LIST = 1; -- Only the total system upper and lower bounds are listed. This is the default.

LIST = 2; -- Every path in the model is listed. The probability bounds for each death state in the model is reported along with the totals.

LIST = 3; -- Details about each step along a path is given along with all of the information displayed by option 2.

Graphics Display Interface. - If the appropriate graphics hardware/software is available, then SURE generates graphical displays of the reliability models and plots the upper and lower bounds of the total system probability of failure as a function of a single variable. The user indicates by "wand" input where each state of the model should be displayed. The user must issue the MEGA command,

MEGA;

prior to the transition commands to cause the system to prompt for the state locations. 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 of the transitions and labels them. The user may retain the state location information on disk by the SAVEMEGA command. For example, the current state location information is written to file SIFT.MEG by the following command:

SAVEMEGA SIFT.MEG.

State location information may be retrieved from a disk file by the GETMEGA

command. If state location has been stored on disk file FTMP.MEG from a prior SURE session then the following command will retrieve this information:

```
GETMEGA FTMP.MEG.
```

If the location information is on a file with the same VMS file name (except the extent) as the command file which describes the model then the following is an abbreviation for the commands GETMEGA TRIPLEX.MEG; READ TRIPLEX.MOD:

```
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.

The SCAN and ZOOM commands may be used to peruse the model. The "wand" button is used to end the ZOOM and SCAN commands.

Miscellaneous Commands. - The following commands are also valid:

RUN;	-- initiates the computation. This command is issued after the model description is fully entered
RUN OUTFILE;	-- initiates the computation as above, but, the output is written to file OUTFILE.
EXIT;	-- causes program termination without computation.
TIME = 100;	-- sets the mission time to 100. The default TIME is 10.
POINTS = 100;	-- indicates that 100 points should be calculated/plotted over the range of the variable.

ECHO = 0; -- turns off the echo when reading a disk file. The default value of ECHO is 1 which enables echoing. (See example 3 in the appendix.)

HELP; -- generates a brief description of each SURE command.

Typical Program Usage

The SURE program was designed for interactive use. The following method of use is recommended:

(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 information from this file.

(3) Then use the miscellaneous commands to change the list option or other defaults as desired.

(4) Enter the RUN command to initiate the computation.

Several interactive sessions are given in the appendix.

LIMITATIONS OF THE PROGRAM

The SURE program is applicable to a large class of semi-Markov models. However, the following list of restrictions should be observed:

(1) Fault arrival transitions must be exponentially distributed.

(2) No circuits may exist in the model, i.e., the system must be a pure death process.

(3) Recovery transitions may be arbitrarily distributed, but, only 1 recovery type transition can leave any particular state.

(4) Only one start state is allowed in a model (i.e., only one state with no transitions to it).

(5) The algebraic approximations to E(T) and E(T-Δ) currently used in the program lead to widely separated bounds when the mission time, T, becomes too large, i.e., when $[T/(n+1)] \Sigma (\lambda_i+\alpha_i) > 1$. A more elaborate calculation of E(T) and E(T-Δ) will eventually be incorporated into SURE.

(6) The lower bound defined by the theorem is close to the upper bound as long as the standard deviations of the recovery transitions are the same order of magnitude as the means or smaller. As the standard deviation of these transitions become larger the bounds separate.

ERROR MESSAGES

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

CIRCUIT FOUND WHILE TRAVERSING THE FOLLOWING PATH - A circuit has been found in the model.

COMMA EXPECTED - syntax error, a comma is needed.

CONSTANT EXPECTED - syntax error, a constant is expected.

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

E(T) APPROXIMATION IS INACCURATE - the entered mission time is so large as to make the upper and lower bounds very far apart.

FILE NAME TOO LONG - file names must be 80 or less characters.

FILE NAME EXPECTED - syntax error, the file name is missing.

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 STATEMENT - the command word is unknown by the system.

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

INTEGER EXPECTED - syntax error, an integer is expected.

NUMBER TOO LONG - only 15 digits/characters allowed per number.

ONLY 1 VARIABLE ALLOWED - only 1 variable can be defined per model.

REAL EXPECTED - a floating point number is expected here.

SEMICOLON EXPECTED - syntax error, a semicolon is needed.

STATE OUT OF RANGE - The state number is negative or greater than the maximum state limit (default = 10000, set at SURE compilation time).

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

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.)

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.

WARNING: MORE THAN ONE STARTSTATE - The model entered by the user has more than one start state (i.e., a state with no transitions to it).

= EXPECTED - syntax error, the = operator is needed.

> EXPECTED - syntax error, the closing bracket > is missing.

*** ID CHANGED TO X - The value of the identifier (Constant) is being changed.

*** ID CHANGED TO X TO Y - The value of the identifier (Variable) is being changed.

*** MORE THAN 1 RECOVERY FROM STATE X - the indicated state has more than 1 recovery type transition leaving it.

***** TRANSITION X -> Y ALREADY ENTERED - The user is attempting to re-enter the same transition again.

CONCLUSIONS

The SURE program is a flexible, user-friendly interactive design/validation tool. The program provides a rapid computational capability for a wide class of semi-Markov models useful in describing the permanent fault behavior of fault-tolerant computer systems. The major deficiency of the program is the inability to deal with transient or intermittent behavior of such systems. Currently, the program provides useful bounds only when the mission time is relatively short, e.g., on the order of 10 to 1000 hours for a system with component failure rates of 10^{-4}/hour. However, this deficiency soon will be remedied by the use of more powerful numerical routines.

APPENDIX

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

Example 1. - This session illustrates direct interactive input and the type of error message given by SURE:

```
$ sure
  1? lambda = 1e-5;
  2? 1,2 = 6*lambda;
  2? 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 = 10;
  9? run

 STARTSTATE = 1
 5 STATES IN GRAPH
 MISSION TIME = 10.0000

 PATH # 1:    5    4    2    1
 PATH # 2:    3    2    1

 2 PATHS IN GRAPH
```

DEATHSTATE	PATH	LOWERBOUND	UPPERBOUND
5	1	5.98620E-08	6.00000E-08
3	2	2.96614E-12	3.00000E-12
TOTAL		5.98650E-08	6.00030E-08

190 MILLISECS CPU TIME UTILIZED

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 and TRIADP1.MEG was created by the SAVEMEGA command in a previous session.

```
$ sure
  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;

 13? run
```

```
STARTSTATE = 1
8 STATES IN GRAPH
MISSION TIME =    6.0000
3 PATHS IN GRAPH
```

LAMBDA	LOWERBOUND	UPPERBOUND
1.00000E-06	6.15621E-15	1.00441E-14
2.78256E-06	5.20068E-14	8.22407E-14
7.74264E-06	4.96200E-13	7.33127E-13
2.15444E-05	5.85662E-12	7.75250E-12
5.99484E-05	8.87328E-10	1.04754E-10
1.66810E-04	1.62064E-09	1.77475E-09
4.64159E-04	3.25940E-08	3.45029E-08
1.29155E-03	6.80160E-07	7.14440E-07
3.59382E-03	1.42382E-05	1.51684E-05
1.00000E-02	2.88873E-04	3.25060E-04

```
290 MILLISECS CPU TIME UTILIZED
```

Figure 3 illustrates the model displayed on the output graphics device. The plot of figure 4 was generated from this run.

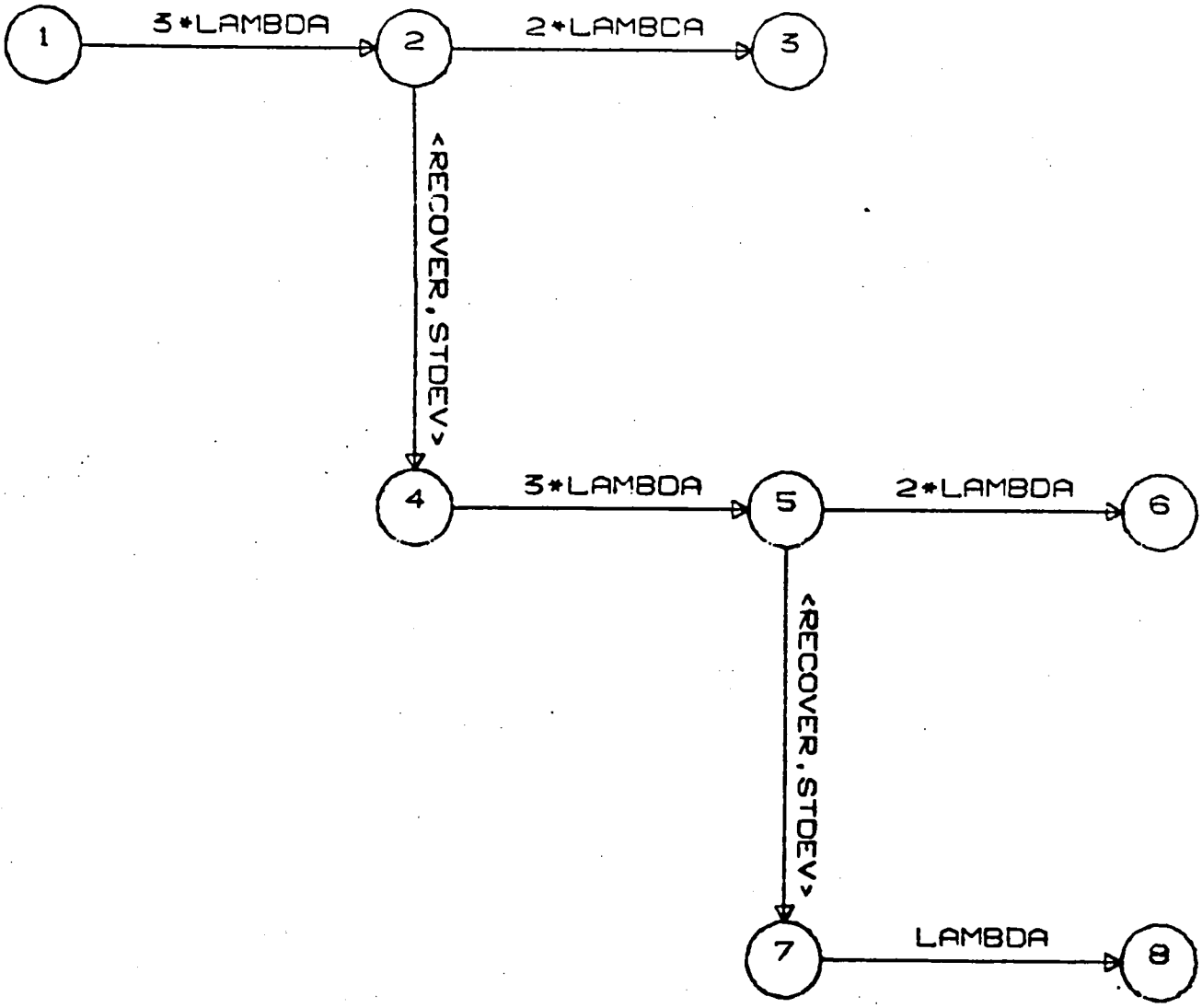


Figure 3. - Markov Model from example 2.

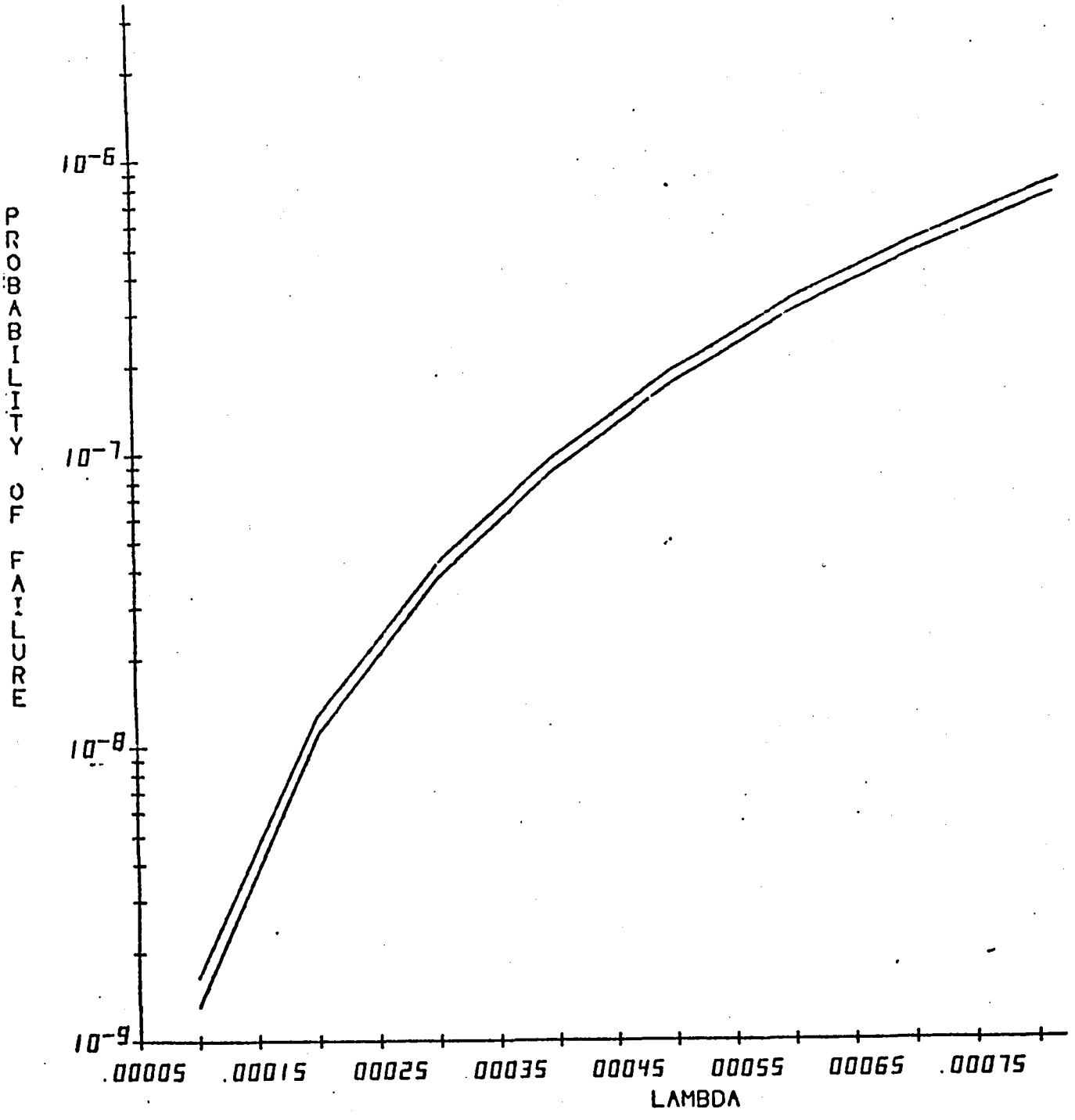


Figure 4. – SURE output from example 2.

Example 3. - The following interactive session illustrates the use of the ECHO constant. This is used when the model description file is large and it is desired that it not be displayed on the terminal as it is read by the SURE program.

```
$sure

 1? getmega ftmp.meg
 2? echo = 0;
 3? read ftmp2.mod;

26? points = 1;
*** POINTS    CHANGED TO 1.00000E+00
27? run

STARTSTATE = 1
20 STATES IN GRAPH
MISSION TIME = 10.0000
7 PATHS IN GRAPH

    LAMBDA        LOWERBOUND     UPPERBOUND
  -----------    -----------    -----------
  1.00000E-04    4.58240E-10    5.02254E-10

170 MILLISECS CPU TIME UTILIZED
```

Example 4. - This interactive session illustrates how to use SURE to obtain system unreliability as a function of mission time.

```
$ sure
```

```
1? read ftmp9

2? LAMBDA = 5E-4;
3? STDEV = 3.6E-4;
4? RECOVER = 2.7E-4;
5? TIME = 0.1 TO 1000;
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? POINTS = 5;

26? run

STARTSTATE = 1
20 STATES IN GRAPH
MISSION TIME =  0.0000
7 PATHS IN GRAPH
```

TIME	LOWERBOUND	UPPERBOUND
1.00000E-01	9.69006E-11	1.21527E-10
1.00000E+00	1.14124E-09	1.21774E-09
1.00000E+01	1.16587E-08	1.24261E-08
1.00000E+02	1.27131E-07	1.59925E-07
1.00000E+03	0.00000E+00	8.13719E-02

E(T) APPROXIMATION IS INACCURATE

260 MILLISECS CPU TIME UTILIZED

REFERENCES

(1) White, Allan L.: Upper and Lower Bounds for Semi-Markov Reliability Models of Reconfigurable Systems. NASA CR-172340, 1984.

(2) 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.

(3) Goldberg, Jack, et. al.: Development and Analysis of the Software Implemented Fault Tolerance (SIFT) Computer. NASA CR-172146, 1984.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM- 86261		

4. Title and Subtitle	5. Report Date
THE SEMI-MARKOV UNRELIABILITY RANGE EVALUATOR PROGRAM	July 1984
	6. Performing Organization Code: 505-34-13-30

7. Author(s)	8. Performing Organization Report No.
Ricky W. Butler	
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Langley Research Center Hampton, Virginia 23665	

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546	Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The SURE program is a new design/validation tool for ultrareliable computer system architectures. The system uses simple algebraic formulas to compute accurate upper and lower bounds for the death state probabilities of a large class of semi-Markov models. The mathematical formulas used in the program were derived from a recent mathematical theorem proven by Allan White under contract to NASA Langley Research Center. This mathematical theorem is discussed along with the user interface to the SURE program.

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

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	32	A03

* For sale by the National Technical Information Service, Springfield, Virginia 22161