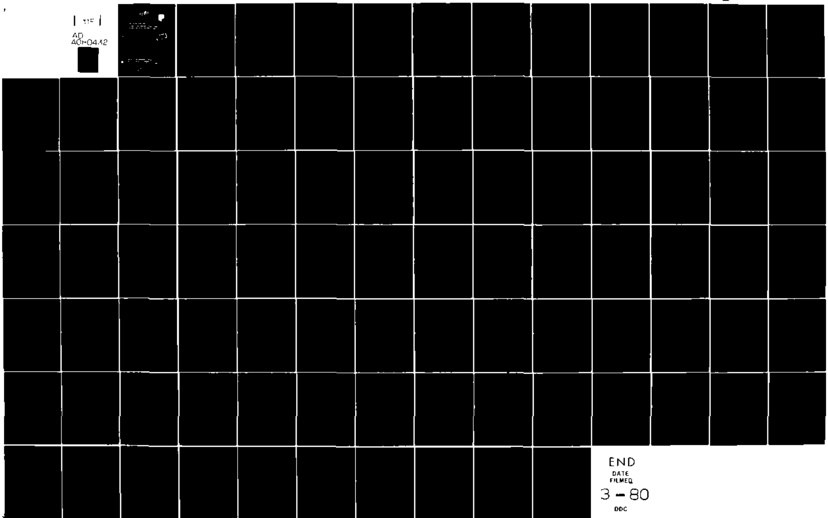
AD-A080 432 UTAH UNIV SALT LAKE CITY DEPT OF ELECTRICAL ENGINEERING F/G 9/2
SOFTWARE RELIABILITY ESTIMATION UNDER CONDITIONS OF INCOMPLETE --ETC(U)
OCT 79 C K RUSHFORTH, F L STAFFANSON F30602-78-C-0025
UNCLASSIFIED UTEC-79-063 RADC-TR-79-230 NL



RADC-TR-79-230
Final Technical Report
October 1979

SOFTWARE RELIABILITY ESTIMATION UNDER CONDITIONS OF INCOMPLETE INFORMATION

University of Utah

C. K. Rushforth
F. L. Staffanson
A. E. Crawford

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-230 has been reviewed and is approved for publication.

APPROVED: Alan N Sukert

ALAN N. SUKERT
Project Engineer

APPROVED: Wendall C Bauman

WENDALL C. BAUMAN, COLONEL, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-79-230	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFTWARE RELIABILITY ESTIMATION UNDER CONDITIONS OF INCOMPLETE INFORMATION		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 Apr 78 — 1 Apr 79
		6. PERFORMING ORG. REPORT NUMBER UTEC-79-063
7. AUTHOR(s) C. K. Rushforth F. L. Staffanson A. E. Crawford		8. CONTRACT OR GRANT NUMBER(s) F30602-78-C-0025
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Utah Electrical Engineering Department Salt Lake City UT 84112		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 25280380
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIS) Griffiss AFB NY 13441		12. REPORT DATE Oct [redacted]79
		13. NUMBER OF PAGES 97
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

12 92

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Alan N. Sukert (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software reliability
Software reliability estimation
Nonlinear parameter estimation
Software mean-time-to-failure

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a computer program for the estimation of the reliability of large software packages. A model is developed which incorporates additional features found in realistic testing environments. The model is analyzed for deterministic data. Performance on simulated random test data is presented. The algorithm estimates the parameters of the model from software error data, and computes therefrom running estimates of the mean-time-to-failure and the number of software errors remaining.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

405 791

TABLE OF CONTENTS

	Page
1.0 Introduction	1.1
1.1 Problem Statement	1.1
1.2 Previous Work	1.1
2.0 Development and Analysis of the Model	2.1
2.1 Assumptions	2.1
2.2 The Model	2.4
2.3 Model Behavior	2.9
2.4 The Data Simulator	2.10
3.0 The Estimation Algorithm	3.1
3.1 The Estimation Problem and Method	3.1
3.2 Description of the Search Algorithm	3.4
3.3 Estimation Results	3.8
4.0 Conclusions and Recommendations	4.1
4.1 RELY I	4.1
4.2 Data Requirements	4.1
4.3 Recommendations	4.2
REFERENCES	5.1
APPENDIX A: MAIN AND SUBROUTINE DESCRIPTIONS	A.1
1. MAIN and Subroutine Diagram	A.1
2. MAIN	A.2
3. Subroutine RNDTA	A.3
4. Subroutine POISS	A.4
5. Subroutine RANDEX	A.5

Page

6. Subroutine KOST A.6

7. Subroutine EIGMIN A.9

8. Subroutine TRIDMX A.10

9. Subroutine EIGVAL A.11

10. Subroutine EIGVEC A.12

11. Subroutine MARQ A.13

APPENDIX B: RELY I GLOSSARY AND INDEX B.1

APPENDIX C: PROCEDURE FOR OPERATION OF RELY I C.1

APPENDIX D: RELY I PROGRAM LISTING AND SAMPLE OUTPUT D.1

LIST OF FIGURES

		Page
1	Number of errors in program as a function of time . . .	2.2
2	Block diagram representation of the proposed model for the error-detection and the error-correction processes.	2.6
3	Simplified model.	2.7
4	Cumulative detected errors $n_d(k)$ for $0 \le \beta \le 0.5$. . . .	2.11
5	Cumulative corrected errors $n_c(k)$ for $0 \le \beta \le 0.5$ . . .	2.12
6	Flow diagram of the estimation algorithm.	3.7
7	Histograms of reliability parameters for Example I. . .	3.13
8	Histograms of reliability parameters for Example II . .	3.14
9	Histograms of reliability parameters for Example III. .	3.15
10	Histograms of reliability parameters for Example IV . .	3.16
11	Tested software in only one version at a time, identical to neighboring versions except for the changes counted in Δ_c at the respective boundaries.	4.3
A.1	RELY I subroutine diagram	A.1

LIST OF TABLES

		Page
1	Estimation results	3.12

EVALUATION

The increased importance of software for embedded avionics systems has led to an increasing desire to insure that avionics software meets very strict reliability and quality goals. However, a significant problem in assuring such goals are met is the inability of Government personnel to accurately predict the reliability of an avionics software development project. This problem has been expressed at several Government and industry sponsored conferences, as well as in documents such as the Joint Logistics Commanders Software Reliability Working Group Report (November 1975) and the Joint Logistics Commanders Software Quality Management Workshop Report (July 1979). As a result, efforts have been initiated to develop and validate mathematical models for predicting the reliability and error content of a software system. However, models developed to date have not adequatley addressed the unique features of avionics software developments.

This effort was initiated in response to the need for developing software reliability prediction models applicable to avionics software developments, and fits into the goals of RADC TPO No. 5, Software Cost Reduction, in the subthrust of Software Quality (Software Modeling). This report summarizes the development of a mathematical model for predicting the reliability and mean-time-to-failure of a software development under the assumptions of incomplete information available on error correction, and discrete versions of the software being developed. The report also describes the modified nonlinear search algorithm developed for finding model parameters and an accompanying computer program for operating the model. The importance of this model development is that the assumptions underlying this model more closely reflect the actual avionics software development process than prior model developments.

The theory and model algorithm developed under this effort will lead to much needed predictive measures for use by software managers of avionics software developments in adequately tracking those developments in terms of reliability and mean-time-to-failure objectives. More importantly, the measures developed under this effort will be applicable to current avionics software developments and thus help to produce the high quality, low cost avionics software needed for today's aircraft.

Alan N. Sukert
ALAN N. SUKERT
Project Engineer

1.0 Introduction

1.1 Problem Statement

As the cost and complexity of computer software continue to increase, there is a growing need for accurate determination of software reliability. Before a software package is put into operation, there is a testing period during which errors are detected and corrected. The problem with which we are concerned is the estimation of certain reliability parameters from the error data generated during the test phase. Specifically, we wish to estimate the number of errors remaining in the software package at any time, and the mean time to failure (MTTF). Accurate determination of these parameters could reduce the cost associated with excessive testing, and could increase the confidence with which the package is used.

In order to estimate software reliability, it is necessary to develop an appropriate model describing the error detection and correction processes, and to develop procedures for estimating the parameters of this model from observed error data. Our intention is to generalize certain models which have previously been used for this purpose in order to depict more accurately an actual testing environment. In addition, we will consider a somewhat different approach to the estimation of the parameters of this generalized model.

1.2 Previous Work

A substantial body of work now exists on the application of statistical modeling and estimation techniques to the determination of software reliability. We make no attempt to describe all this work, but rather restrict ourselves to those efforts which are directly related to our own. For a comprehensive review and bibliography, see [1] or [2].

One of the most widely-used error models was developed by Jelinski and Moranda [3]. A similar model has been considered by Shooman [4] a others. The assumptions about the error-detection and error-correction processes which underlie this model are the following:

(a) The error-detection process is a Poisson process whose detection rate is constant between error detections.

(b) The error-detection rate at the time prior to the detection of the i^{th} error is a function of i; it is denoted by z_i. It is commonly assumed that z_i is proportional to the number of errors in the program at detection time. This can be written as:

$$z_i = \phi \left(N_o - i + 1\right) \tag{1.1}$$

where N_o is the initial number of errors and ϕ is a positive constant. An alternative assumption is that the detection rate forms a geometric progression

$$z_i = \lambda a^i \tag{1.2}$$

with both λ and a being positive constants. It should be noted that the main justification for (1.2) is the improved convergence of the resulting estimator equations [5].

(c) Error detection is followed by an immediate correction. Consequently, upon detection of the i^{th} error, the number of remaining errors drops to $\left(N_o - i\right)$.

(d) The debugging process is perfect and no new errors are generated by the correction process.

These assumptions, although restrictive, were initially adopted by most of the researchers in the field. The estimation of the reliability parameters was based on the above assumptions, and employed the maximum likelihood (ML) criterion to derive the best estimates.

It is realized now that the assumptions given above are quite restrictive and unrealistic in most cases, and steps have been taken to make the model more realistic. The model assumptions have been changed to comply more closely with the real process.

Goel [6] has considered a nonideal debugging process in which the probability of correcting an error is p. Based on this assumption, an analysis of the resulting model is performed. Further generalization is suggested by Shooman [7], who modified both assumptions c and d above concerning the error-correction process. According to the modified model of Shooman, the correction process does not necessarily proceed identically to the detection process, and new errors may be introduced. Denote by $r_d(t)$, $r_c(t)$, and $r_g(t)$ the rates of error detection, correction, and new error generation, respectively. The models suggested by Shooman assume different relationships between these rates. The main models are:

Model 1

$$r_c(t) = \beta \, r_d(t) \tag{1.3}$$

$$r_g(t) = \alpha \, r_c(t) \tag{1.4}$$

and

Model 2

$$r_c(t) = b \, r_d(t) \tag{1.5}$$

$$r_g(t) = a \, n(t) \, r_d(t) \tag{1.6}$$

where n(t) represents the number of errors in the program.

These models and others have been studied by Shooman, and the results are described [7].

Another generalization of the original model concerns the assumption that the corrections are implemented continuously. This is not consistent with actual practice in which a program is replaced by a newer version at discrete times. Between the replacement times, the program undergoing the test is the same and the number of errors in it is constant. A possible solution for this discrepancy is that rediscovery of errors should not be counted. However, this requires the analysis of the source of errors in order to determine whether the error sources are the same, and this is not always practical. A modified model in which this generalization was implemented was discussed by

Tal [5] and by Sukert [2], and estimator equations for use with this model were developed.

These generalizations, along with some additional ones, will be incorporated into a new model. The new model, we believe, more accurately describes an actual testing environment. We will first discuss the behavior of this model as a function of its parameters under the simplifying assumption that the error processes are deterministic rather than random.

After presenting certain results for deterministic processes, we will then show results using simulated random error data. We have developed a least-squares search procedure for estimating the model parameters, and will discuss its convergence behavior. Recommendations are made toward increased utility, and toward closer coupling of the algorithm to information in real test data.

The true test of the usefulness of the model will lie in its ability to describe real software tests. Thus, there remains for subsequent work the application of the model to enough real cases to draw conclusions concerning validity.

One of the difficulties encountered by researchers in the past has been the inadequacy, incompleteness, and ambiguity of available test data. We found some of these same problems with the data available to us during this work. Hence, we include comments regarding data requirements.

2.0 Development and Analysis of the Model

2.1 Assumptions

In order to develop a generalized model to describe the error detection and correction processes, we make the following assumptions:

(a) The error detection process is a Poisson process whose average rate of occurrence is proportional at any time to the number of errors present in the software package. Denoting the number of errors present at time t by $N(t)$ and the average error occurrence rate by $r_d(t)$, we have

$$r_d(t) = \phi N(t) \tag{2.1}$$

where ϕ is a fixed constant of proportionality.

(b) No attempt is made to correct detected errors at the time of detection. Instead, a new and corrected version of the program is provided to the testing group at discrete ("tape replacement") times $t_1, t_2, \ldots, t_j, \ldots$. Thus, the number of errors present in the program at time t, $t_j \leq t < t_{j+1}$, is constant and equal to $N(t_j)$. This is illustrated in Fig. 1.

(c) Of the detected errors reported to the correcting group, some are corrected and some are not. In addition, new errors are generated. Denote the cumulative number of errors corrected to time t *in the program being tested* by $N_c(t)$, and the cumulative number of newly-generated errors by $N_g(t)$. Both N_c and N_g are piecewise constant because of the assumption

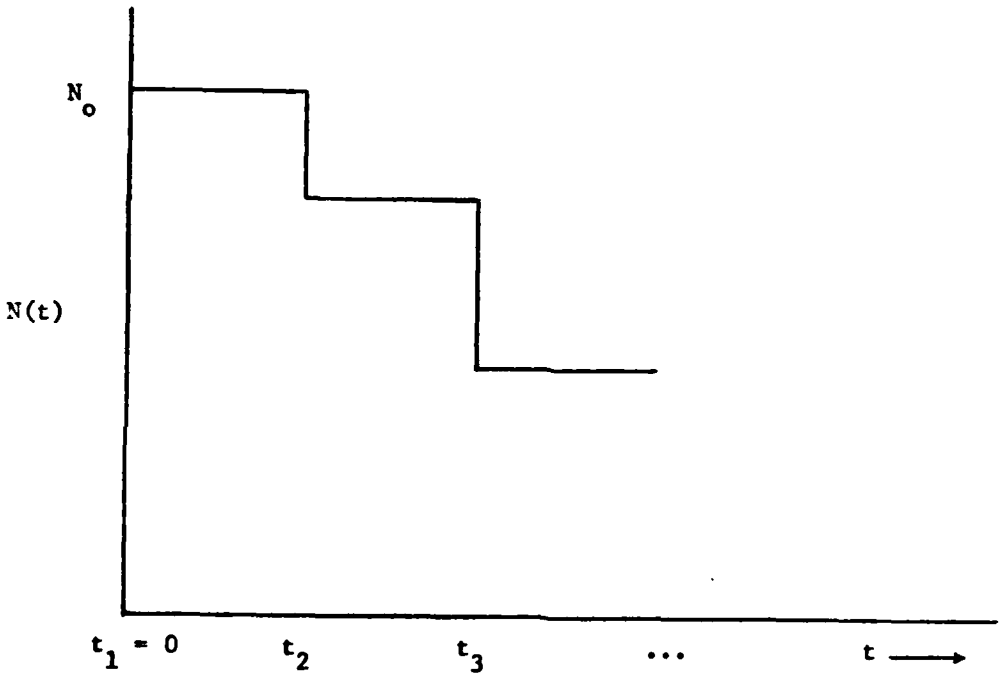


Fig. 1. Number of errors in program as a function of time.

that a new version of the program is provided only at the discrete times t_1, t_2, ... A key feature of our model is that many errors may be detected, corrected, and generated between update times. At any time t during testing, we have

$$N(t) = N_o - N_c(t) + N_g(t) \tag{2.2}$$

where N_o is the initial number of errors in the program.

(d) The error correction rate $r_c(t)$ depends on both the error detection rate $r_d(t)$ and the error backlog $N_b(t)$, where

$$N_b(t) = N_d(t) - N_c(t). \tag{2.3}$$

For simplicity, we assume a linear relationship

$$r_c(t) = \alpha r_d(t) + \beta N_b(t). \tag{2.4}$$

The addition of the second term in (2.4) represents a generalization of the model of Shooman [7].

(e) The rate of generation of new errors is proportional to the error-correction rate:

$$r_g(t) = \gamma r_c(t). \tag{2.5}$$

(f) The error-detection process $N_d(t)$ is precisely known, but the error-correction process $N_c(t)$ is unknown. This appears to be a realistic assumption in view of the way error correction is actually performed. The error generation process is also unknown.

In the above we tacitly equate software "failure" to coding "faults". In effect, we include in α and ϕ the proportionality between the two, and call them both "errors".

2.2 The Model

The model which we develop is actually a deterministic model which relates the expected values of the various random processes involved. The required connection between the observed sample functions of the random processes involved and the deterministic model is established by means of an estimation algorithm which operates on the observed data to estimate model parameters. The deterministic model will be described first, followed by a discussion of the estimation procedure.

Taking expected values of (2.1)-(2.4) yields the equations

$$r_d(t) = \phi n(t), \tag{2.6}$$

$$r_c(t) = \alpha r_d(t) + \beta n_b(t), \tag{2.7}$$

$$n(t) = N_o - n_c(t) + n_g(t), \tag{2.8}$$

$$n_b(t) = n_d(t) - n_c(t), \tag{2.9}$$

$$n_g(t) = \gamma n_c(t), \tag{2.10}$$

where a lower-case n denotes the expected value of the process represented by the corresponding upper-case N.

It follows from the relationship between $r_d(t)$ and $n_d(t)$ that

$$n_d(t) = \int_o^t r_d(t)du. \tag{2.11}$$

Similarly,

$$n_c(t) = \int_o^t r_c(u)du. \tag{2.12}$$

The model represented by the above equations can be viewed as a linear system with sampling and feedback as shown in Fig. 2. Our problem is now one of system identification: Given $N_d(t)$, estimate the parameters of the system shown in Fig. 2. Revisions to the software are applied at time instants t_k, between which times n(t) remains constant. The system therefore is treated as a discrete-time system. We employ the usual notation k in place of the argument t_k.

The four system equations (2.6–2.9) can be reduced to two:

$$r_d(k) = \phi \left[N_o - (1 - \gamma)\ n_c(k)\right] \tag{2.13}$$

$$r_c(k) = \alpha\phi \left[N_o - (1 - \gamma)\ n_c(k)\right] + \beta \left[n_d(k) - n_c(k)\right] \tag{2.14}$$

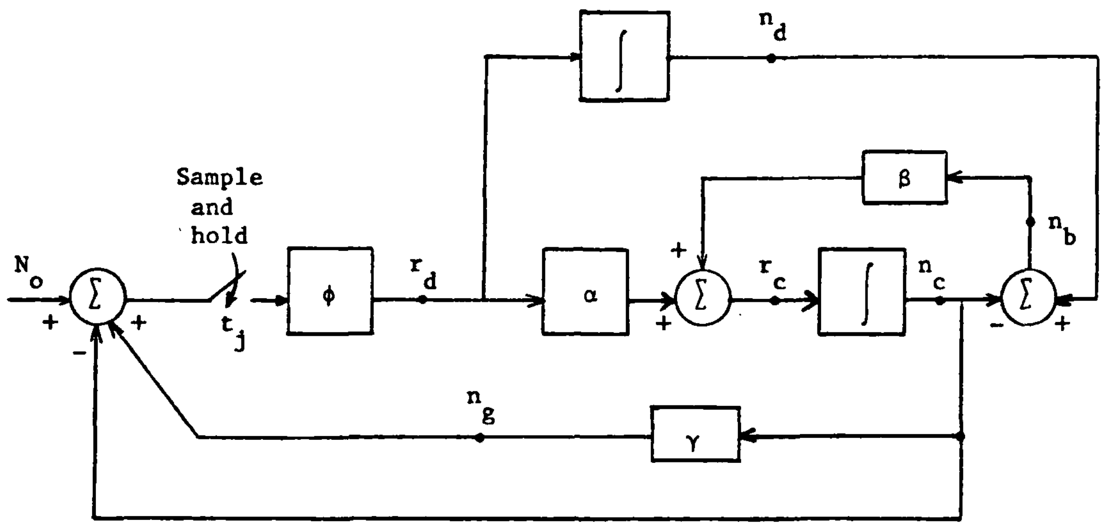


Fig. 2. Block diagram representation of the proposed model for the error-detection and the error-correction processes.

and the number of parameters reduced to four:

$$r_d(k) = \phi_a \left[N_a - n_c(k)\right] \tag{2.15}$$

$$r_c(k) = \alpha\phi_a \left[N_a - n_c(k)\right] + \beta \left[n_d(k) - n_c(k)\right] \tag{2.16}$$

where

$$\phi_a = (1 - \gamma)\phi, \qquad N_a = N_o/(1 - \gamma). \tag{2.17}$$

The application of Laplace transform techniques and some algebraic manipulation similarly lead to the equivalent block diagram shown in

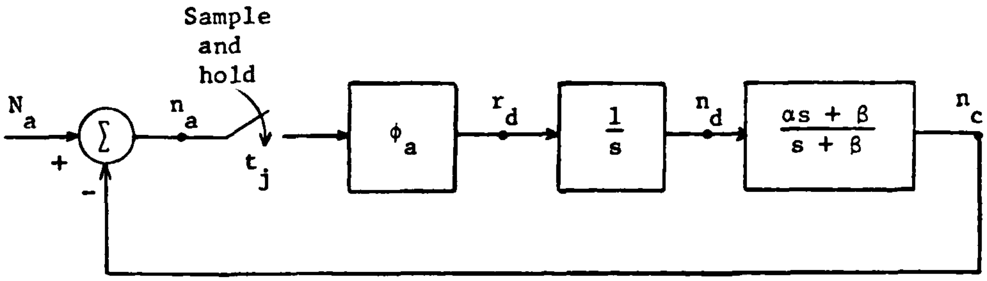


Fig. 3. Simplified model.

Fig. 3. The identification problem reduces to the estimation of the four parameters N_a, ϕ_a, α, and β. Note that N_a is the sum of the initial errors N_o and all errors which are subsequently generated during the correction process. Note further that the ultimately sought reliability factor, mean-time-to-failure, is:

$$\mathrm{MTTF} \equiv \frac{1}{r_d(k)} = \frac{1}{\phi_a n_a(k)} = \frac{1}{\phi_a\left[N_a - n_c(k)\right]}. \tag{2.18}$$

Defining the discrete state

It is noted that the dynamics of the system can be studied using the even simpler nondimensionalized three-parameter system, using (ϕT), (βT), and $\left(n/N_a\right)$, with unit step input.

$$\vec{n}(k) = \begin{pmatrix} n_d(k) \\ \\ n_c(k) \end{pmatrix} \tag{2.19}$$

and using the usual approximation, which in our case is exact,

$$\vec{r}(k) = \frac{\vec{n}(k+1) - \vec{n}(k)}{T(k)}, \quad T(k) = t(k+1) - t(k) \tag{2.20}$$

the model becomes

$$\vec{n}(k+1) = L\vec{n}(k) + N_a\phi_a B \tag{2.21}$$

where

$$L = \begin{pmatrix} 1 & -\phi_a T(k) \\ \\ \beta T(k) & 1 - T(k)\left[\beta + \alpha\phi_a\right] \end{pmatrix}, \quad B = \begin{pmatrix} T(k) \\ \\ \alpha T(k) \end{pmatrix} \tag{2.22}$$

When tape replacement occurs at uniform time intervals, T is constant over k and the system is seen to be stationary, and the equations can be solved immediately by successive evaluation:

$$\vec{n}(1) = N_a\phi_a B, \quad \vec{n}(0) = 0$$

$$\vec{n}(2) = N_a\phi_a(L + 1)B$$

$$\vec{n}(3) = N_a\phi_a\left(L^2 + L + 1\right)B$$

$$\vdots$$

$$\vec{n}(k) = N_a\phi_a \sum_{j=0}^{k-1} L^j B$$

and applying the familiar procedure for the geometric sum,

$$Ln(k) - n(k) = N_a \phi_a \left(L^k B - B \right)$$

which gives for the state at the k^{th} tape replacement time,

$$\vec{n}(k) = N_a \phi_a \; (L - I)^{-1} \left(L^k - I \right) B \tag{2.23}$$

The increment $\vec{\delta}(k) \equiv \vec{n}(k) - \vec{n}(k-1)$ at the k^{th} tape replacement time is given by:

$$\vec{\delta}(k) = N_a \phi_a \; (L - 1)^{-1} \left(L^k - L^{k-1} \right) B$$

$$= N_a \phi_a \; L^{k-1} \; B \tag{2.24}$$

2.3 Model Behavior

Note from the discrete state equations above that the parameter N_a is simply a scale factor on the state $\vec{n}$. Recall also that the initial slope of $n_d(k)$ is $N_a \phi_a$, and that of $n_c(k)$ is $\alpha N_a \phi_a$, regardless of the value of β. Furthermore, for $\beta = 0$, $n_d(k)$ and $n_c(k)$ maintain the constant ratio $n_c(k)/n_d(k) = \alpha \leq 1$, and, of course, coincide as $\alpha \to 1$.

The effect of $\beta > 0$ is to increase the error correction rate, and therefore increase $n_c(k)$, especially for the larger differences $n_d(k) - n_c(k)$ (backlog) which tend to occur later in the test program. The resulting decrease in remaining errors $N_a - n_c(k)$ causes the detected error curve $n_d(k)$ to be bent downward. Thus the effect of β is to draw the two curves together. Figures 4 and 5 display this effect for $0 \leq \beta \leq 0.5$. The "bending" of the curves due to β, together with the effects of the discrete nature of the model, are expected to occur in real data.

2.4 The Data Simulator

RELY I contains a data simulator for the purposes of study and experimentation. The simulator is an optional source of input data to the estimator (see Appendix C). The simulator reads from input cards the nominal parameter values, α, β, ϕ_a, N_a, the time interval T, the number of test intervals K, and an input initial random number (RRR), and computes the associated software test history $\Delta_d(k)$. The random number RRR is changed by the investigator when he wishes a different sample of the random data set $\Delta_d(k)$ (see Appendix A, RNDTA, RANDEX).

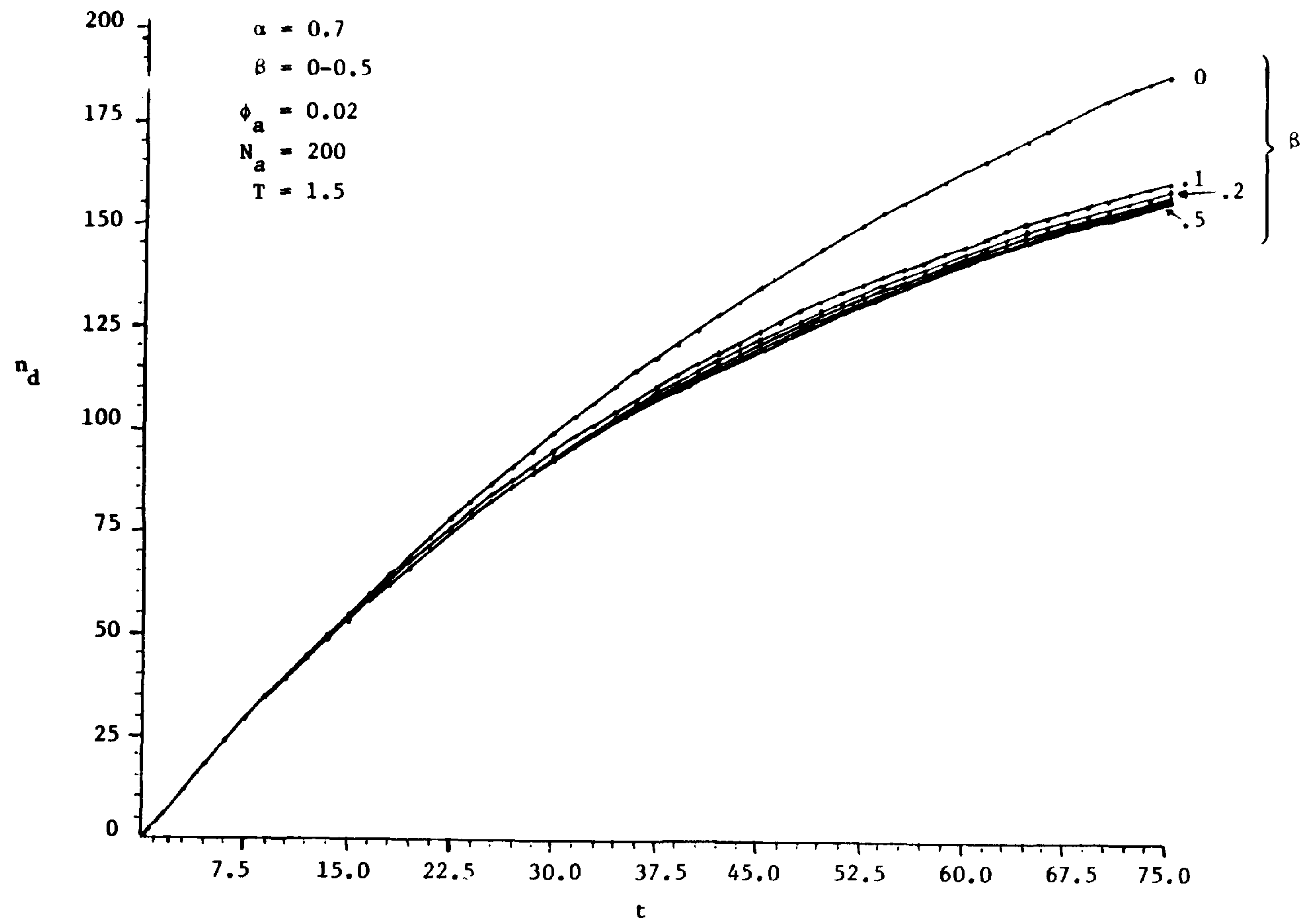


Fig. 4. Cumulative detected errors $n_d(k)$ for $0 \leq \beta \leq 0.5$.

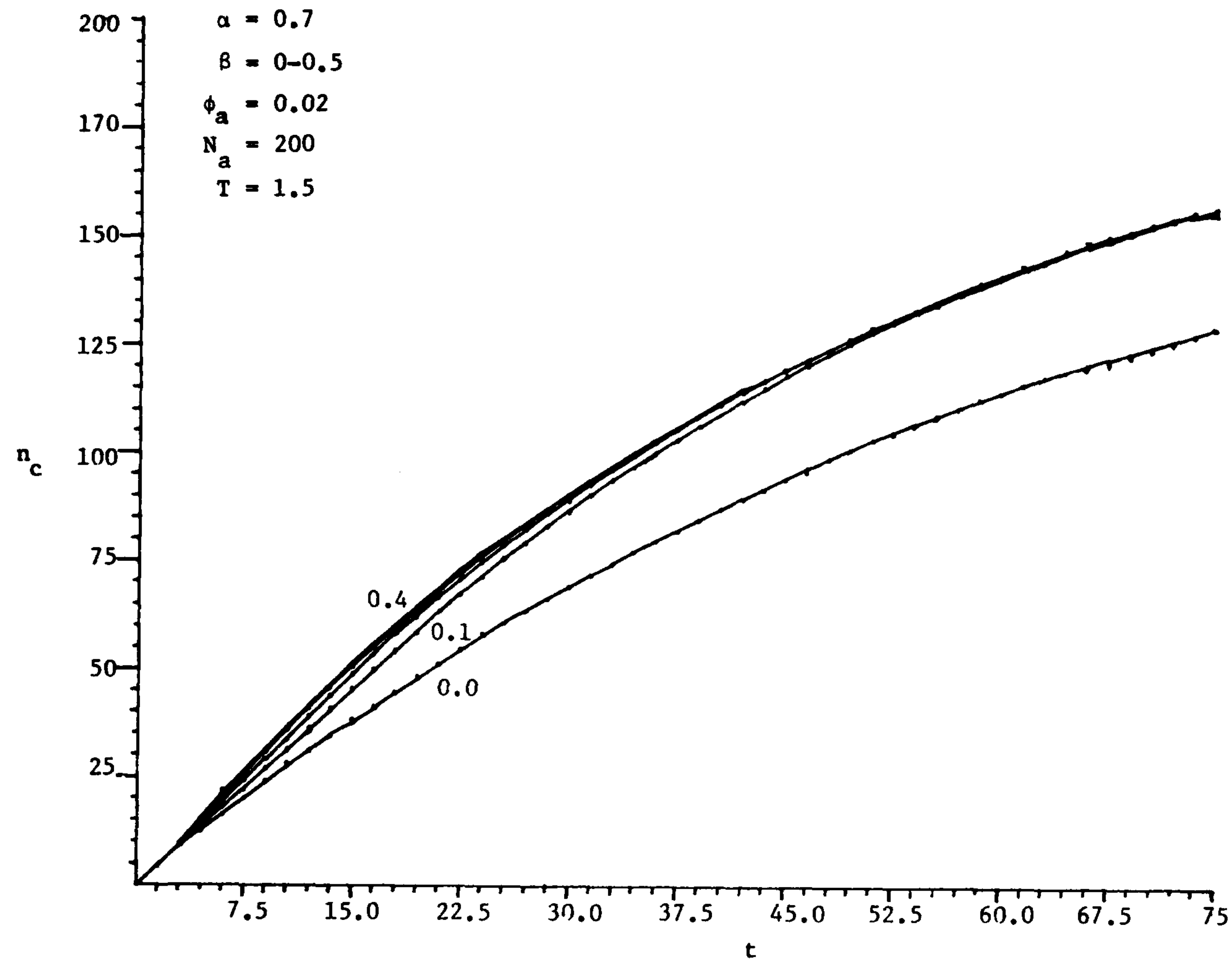


Fig. 5. Cumulative corrected errors $n_c(k)$ for $0 \le \beta \le 0.5$.

3.0 The Estimation Algorithm

3.1 The Estimation Problem and Method

Having described the model, we turn to the parameter estimation algorithm which estimates the values of the model parameters corresponding to a given set of real test data. The resulting parameter estimates provide the reliability information sought regarding the tested software package.

Though the model is linear in the sense that the equations are linear in the state $\vec{n}$, the model equations are nevertheless nonlinear in the parameters $\vec{\theta}$ (i.e., in α, β, ϕ_a, N_a). Determining the parameter values corresponding to a given set of real test data $\vec{\Delta}(k)$ is then a nonlinear estimation problem.

Nonlinear parameter estimation methods, in general, are iterative procedures in which the estimate is approached from some initial guess for the parameter values, in steps which successively decrease a cost functional J. Since our purpose is to determine the parameter values $\vec{\theta}$ for which the solution $\delta_d(k)$ of the model equations approximates the measured function (or sequence) $\Delta_d(k)$, we choose the cost functional J to be the sum of the squares of the residuals, $\delta_d(k) - \Delta_d(k)$, viz.,

$$J = \sum_{k=1}^{K} \left[\delta_d(k) - \Delta_d(k)\right]^2 \qquad (3.1)$$

Minimizing this cost functional, then, minimizes the difference between the observed function $\Delta_d(k)$ and its expected value $\delta_d(k)$ in the least squares sense.

Of the numerous methods described in the literature, both direct search methods (Fletcher [9]) and gradient methods (Bard [8]), the gradient methods are generally preferred when the computation of the derivatives of J is not prohibitive. Gradient methods, in principle, step from one point $\vec{\theta}_i$ in parameter space to the next $\vec{\theta}_{i+1}$ according to

$$\vec{\theta}_{i+1} = \vec{\theta}_i - \tau_i R_i \vec{g}_i \tag{3.2}$$

where $\vec{g}_i$ is the gradient of J evaluated at $\vec{\theta}_i$, R_i is some matrix which operates on the gradient to define the i^{th} step direction $R_i\vec{g}_i$, and τ_i is a scalar which determines the step size. The methods differ in what each employs for R_i, i.e., in the step direction each takes relative to the gradient. The method of steepest descent, for example, uses the identity matrix for R_i, so that the step direction is opposite to that of the gradient. This is "the steepest way down" locally but tends to be less efficient and therefore less desirable than methods which use second order information about the surface $J(\vec{\theta})$.

The Newton-Raphson method uses for R_i the inverse of the Hessian, the matrix of the second partial derivatives,

$$H_{mn} = \frac{\partial^2 J}{\partial\theta_m \partial\theta_n}, \tag{3.3}$$

of the cost functional.

Notice that the Taylor series expansion of J to second order terms,

$$J = J_i + \vec{g}_i^T \left(\vec{\theta} - \vec{\theta}_i\right) + \frac{1}{2} \left(\vec{\theta} - \vec{\theta}_i\right)^T H_i \left(\vec{\theta} - \vec{\theta}_i\right)$$

has an extremum,

$$\frac{\partial J}{\partial \vec{\theta}_i} = g_i + H_i \left(\vec{\theta} - \vec{\theta}_i\right) = 0,$$

at

$$\vec{\theta} = \vec{\theta}_i - H_i^{-1} g_i \qquad (H_i \text{ nonsingular})$$

so if $R_i = H_i^{-1}$, $\rho_i = 1$, and J is quadratic, then $\vec{\theta}_{i+1}$ coincides with the extremum. The Newton-Raphson method in this case converges in a single iteration. This method is quite efficient even for nonquadratic J, but only if H_i is positive definite. This latter condition is the principal weakness of the method. The Marquardt method meets this weakness by guaranteeing positive definiteness in R_i by adding to H_i (or to some convenient approximation of H_i) a variable amount of a positive definite matrix C_i^2:

$$R_i = \left(H_i + \lambda_i C_i^2\right)^{-1} \tag{3.4}$$

and suggests C_i^2 be a matrix of the diagonal elements of H_i, viz.,

$$C_{i,ss}^2 = \left|H_{i,ss}\right|. \tag{3.5}$$

For sufficiently large λ_i, R_i then is positive definite, even when H_i is not. The Marquardt method behaves as the Newton-Raphson for small

λ_i, but where larger λ_i is necessary it steps nevertheless in some acceptable (downward) direction. A step is said to be acceptable if it decreases J. If λ_i is large and H_i has low condition number (eigenvalues of near-equal magnitude), the method approximates that of steepest descent. The Marquardt method varies from step to step according to λ_i, between the behavior of the Newton-Raphson method and that of steepest descent.

3.2 Description of the Search Algorithm

The program, RELY I, uses the above Marquardt R_i, i.e.,

$$\theta_{i+1} = \theta_i - \tau_i \left(H_i + \lambda_i C_i^2 \right)^{-1} g_i \qquad (3.6)$$

and selects τ_i or λ_i from step to step according to the procedure described below. Essentially the program progresses in one or the other of two modes. In mode A, λ_i is fixed while the largest τ_i $(0.0001 < \tau_i \leq 1)$ is sought which results in an acceptable step size. If the sought τ_i is found, the program continues in mode A preferring smaller and smaller values of λ (more nearly Newton-Raphson). If at any point insufficient progress is being made in mode A, the routine moves to mode B, in which τ_i is initially fixed, and λ_i is successively increased until an acceptable step direction is reached. In mode B, when a sufficiently large λ_i is reached, then the program steps in that direction until J begins to increase, or until, for large J, J has decreased more than 10 percent, at which point the routine returns to mode A. In short, when progress is slow in mode A, the program resorts to mode B to move to a different "locality". "Progress" in mode B is deliberately restricted for large J due to experience which indicates that mode B for large J tends to settle into local minima. The program terminates when J becomes less than a predetermined value (ERR), or upon a time limit for machine computation.

More specifically, the estimator proceeds as follows: Given initial guess $\vec{\theta}_i$ and $\lambda_i = 1$, $i = 0$:

<u>Mode A</u>

1. Compute cost J_i and step direction $R_i g_i$.
2. If J_i < ERR terminate, otherwise determine an acceptable step size in the following way:
   a. Compute a τ_i such that twice the associated step causes $\beta = 5$; i.e.,

   $$\tau_i = \left(5. - \beta_i\right) / \left[2 \left| R_i g_i \right|\right]$$

   b. If such a step causes $\phi_a > 0.2$, choose instead

   $$\tau_i = \left(0.2 - \phi_{ai}\right) / \left[2 \left| R_i g_i \right|\right]$$

   c. If the resulting $\tau_i > 1$, set $\tau_i = 1$.
   d. If the resulting $\tau_i < .0001$, jump to mode B.
   e. Limit $0 \leq \beta \leq 10$. Compute J_{i+1}.
   f. If $J_{i+1} \geq J_i$, jump to item 4 below.
3. Accept θ_{i+1} and reduce λ; i.e.,
   a. Set $\theta_i = \theta_{i+1}$, $\lambda_i = \lambda_i/10$.

b. If J has decreased less than 1 percent in more than five iterations (reductions of λ in item 3.a) since passing through mode B, jump to mode B. Otherwise continue in mode A (jump to item 1 above).

4. Reduce τ_i by a factor of 10. If the resulting $J_{i+1} < J_i$ jump to item 3 above, otherwise repeat item 4 above up to five times (according to counter INDEX). If J does not decrease with five reductions of τ_i, jump to mode B. If $J_{i+1} <$ ERR, terminate.

Mode B

5. Fix $\tau = 0.1$, set $\lambda = 0.01$.
6. Increase λ by a factor of 10, increment the count ICLAM, determine the corresponding step direction $\left(H_i + \lambda_i C_i^2\right)^{-1} g_i$, parameter set θ_{i+1}, and J_{i+1}. If $J_{i+1} \geq J_i$, repeat item 6.
7. Accept θ_{i+1} (i.e., set $\theta_i = \theta_{i+1}$) and set $J_\lambda = J_{i+1}$.
8. Increase τ_i by a factor of 5^ℓ, ℓ = ICLAM.
9. Try $\vec{\theta}_{i+1} = \vec{\theta}_i - \tau_i\left(H_i + \lambda_i C_i^2\right)^{-1} \vec{g}_i$, if $J_{i+1} > J_i$ or if $J_i >$ 50 and $J_{i+1}/J_\lambda < 0.9$, accept $\vec{\theta}_i$ and return to item 1 above, otherwise accept $\theta_{i+1} \rightarrow \theta_i$ and repeat item 8.

The algorithm is depicted in the flow diagram of Fig. 6.

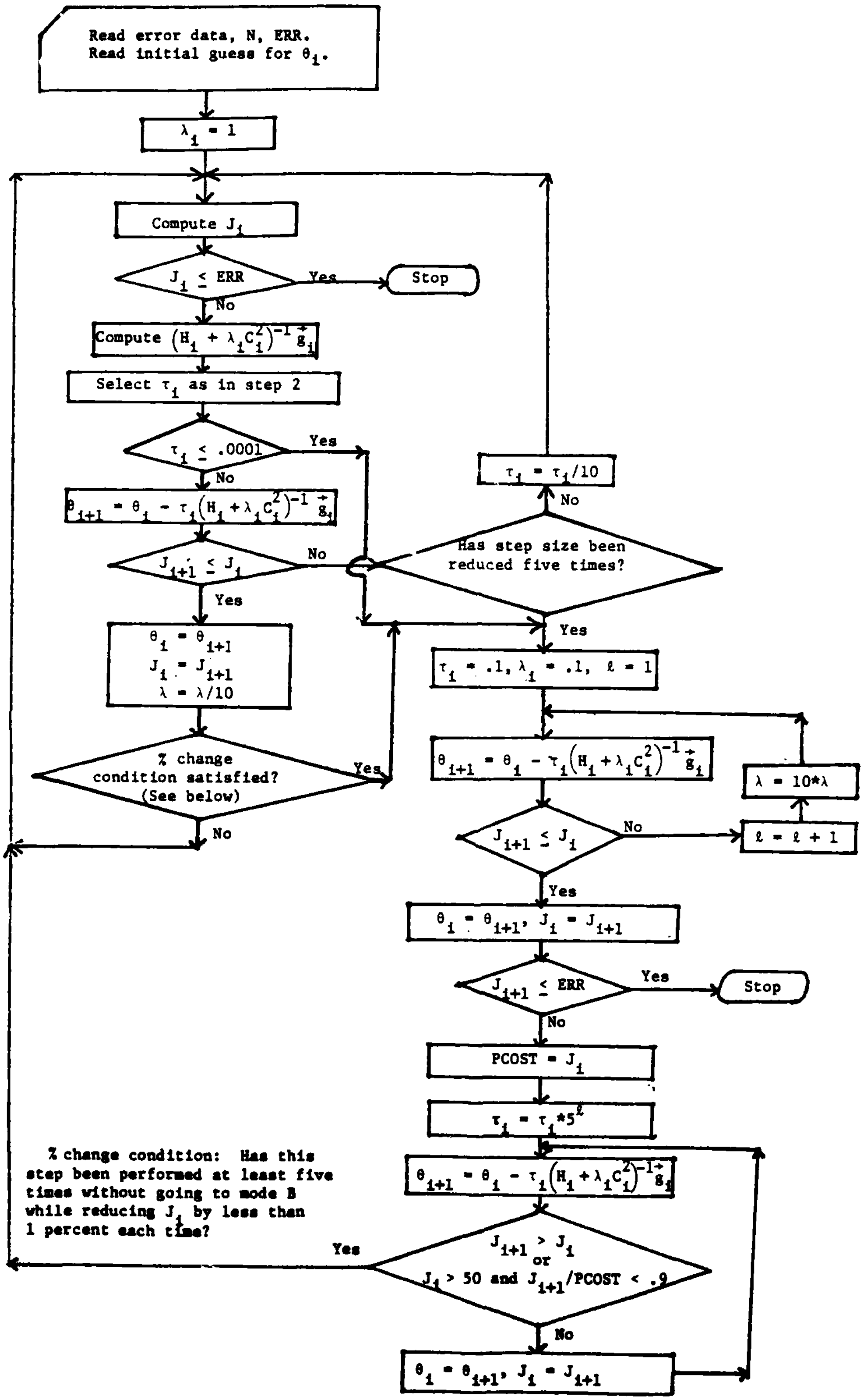


Fig. 6. Flow diagram of the estimation algorithm.

Experience during development of RELY I proved α to be insufficiently independent of the other parameters to warrant a fourth degree of freedom in the search process. The computer program therefore was modified to accept a priori the estimate of α, and to search in three dimensions for the values of β, ϕ_a, and N_a. From these the estimated number of remaining errors,

$$n_a = n_a(k) = N_a - n_c(K), \quad k = 1, 2, 3, \ldots K$$

and mean time to failure

$$MTTF = \frac{1}{\phi_a n_a}$$

are computed. The latter two computed quantities, the sought software reliability factors, were found to be essentially insensitive to reasonable a priori estimates of α. Results below include cases of correct and incorrect fixed α.

3.3 Estimation Results

Results are tabulated and displayed in histograms below for several simulated random data examples. Examples I and II differ in the selection of K and T to vary the number of errors, N_R, remaining in the software. Example I uses test interval length T = 1.5 and 60 intervals which leaves about 30 remaining of the initial 200 software errors. Example II uses longer test intervals, T = 8, and fewer intervals, K = 20, to leave about 5 errors remaining. Example III corresponds to a larger software system having a considerably larger number of initial errors ($N_a = 1000$), smaller error detection rate ($\phi_a = 0.01$), but a better correction rate ($\alpha = 0.8$), and the longer test intervals ($T = 8.0$). Finally, the fourth example demonstrates the insensitivity to the fixed value of α. Example IV essentially is Example I with α fixed at 0.5 instead of the "true" value, 0.7.

Before examining the results, we anticipate the nature of the distributions by analytically determining the mean and variance for the simple single interval ($K = 1$) case. Let the observed number of detected errors N_d be Poisson with mean and variance ρN_a, where $\rho = \phi_a T$. We minimize the squared error J,

$$J = \left(\rho N_a - N_d\right)^2$$

$$\frac{\partial J}{\partial N_a} = 2\rho \left(\rho N_a - N_d\right) = 0$$

to obtain an estimate

$$\hat{N}_a = \frac{N_d}{\rho}$$

which has mean

$$E\left\{\hat{N}_a\right\} = \frac{E\left\{N_d\right\}}{\rho} = N_a$$

and variance

$$E\left\{\left(\hat{N}_a - N_a\right)^2\right\} = E\left\{\hat{N}_a^2\right\} - 2\,E\left\{\hat{N}_a N_a\right\} + E\left\{N_a^2\right\}$$

$$= \frac{E\left\{N_d^2\right\}}{\rho^2} - 2N_a E\left\{\hat{N}_a\right\} + N_a^2$$

$$= \frac{\left(\text{var}\left(N_d\right) + \bar{N}_d^2\right)}{\rho^2} - N_a^2$$

$$= \frac{\rho N_a + \rho^2 N_a^2 - \rho^2 N_a^2}{\rho^2}$$

$$= \frac{N_a}{\rho}$$

The number of remaining errors,

$$N_R = N_a - N_d$$

is estimated

$$\hat{N}_R = \hat{N}_a - N_d = N_d\left(\frac{1 - \rho}{\rho}\right)$$

with unbiased mean

$$E\left\{\hat{N}_R\right\} = E\left\{\hat{N}_a - N_d\right\} = N_a - N_d$$

and with root mean squared difference from its true value,

$$E\left\{\left[N_d\left(\frac{1-\rho}{\rho}\right) - \left(N_a - N_d\right)\right]^2\right\}^{1/2} = E\left\{\left(N_a - \frac{N_d}{\rho}\right)^2\right\}^{1/2}$$

$$= \left[N_a^2 - \frac{2N_a}{\rho} E\left\{N_d\right\} + \frac{E\left\{N_d^2\right\}}{\rho^2}\right]^{1/2} = \left[N_a^2 - 2N_a^2 + \frac{\rho^2 N_a^2 + \rho N_a}{\rho^2}\right]^{1/2} = \sqrt{\frac{N_a}{\rho}}$$

Notice that the value of the latter quantity corresponding to:

a. Example I: Let $N_R = N_a - \rho N_a$, or ($\rho = 1 - 31/200 = .845$) is

$$= \sqrt{\frac{200}{.845}} = 15$$

b. Example II: ($\rho = 1 - 5/200 = 0.975$) is

$$= \sqrt{\frac{200}{.975}} = 14$$

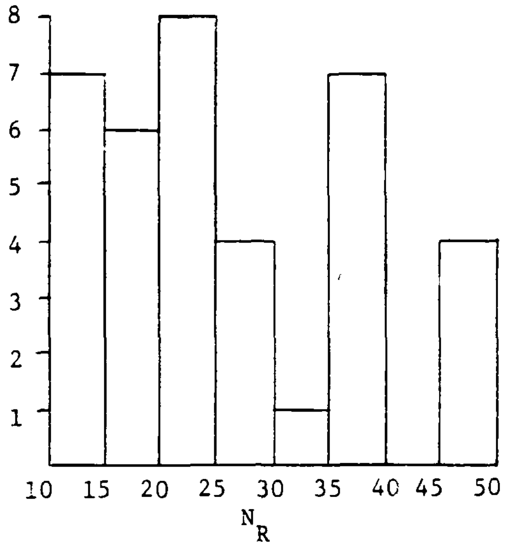
c. Example III: ($\rho = 1 - 185/1000 = 0.815$) is

$$\sqrt{\frac{N_a}{\rho}} = \sqrt{\frac{1000}{.815}} = 35$$

One would expect these values to approximate the standard deviations σ (N_R) for the respective multiple-interval cases (though perhaps with less validity when N_R/N_a is small). The $\sigma(N_R)$ indicated below for the four examples then are of the magnitude to be expected. Table 1 lists numerical information from the four examples. Examination of results from the four simulated examples indicates that the estimator produces reasonable estimates of the reliability parameters N_R and MTTF.

Table 1. Estimation results.

Item	Example: I	II	III	IV
<u>"True" Values</u>				
α	.700	.700	.800	.700
β	.100	.100	.100	.100
ϕ_a	.020	.020	.010	.020
N_a	200	200	1000	200
N_R	31.0	4.93	185	30.6
MTTF	1.61	10.1	0.538	1.64
<u>Time Interval</u>				
T	1.5	8.0	8.0	1.5
<u>Number of Intervals</u>				
K	60.	20.0	20.0	60.0
<u>A Priori α</u>				
α'	.700	.700	.800	.500
<u>Initial Guess</u>				
β	.300	.300	0	0
ϕ_a	.010	.010	.02	.010
N_a	300	300	1500	300
<u>Estimated Values</u>				
$\overline{N_R}$	26.3	5.03	195	29.7
$\sigma\ (N_R)$	11.2	3.07	40.4	10.8
$\overline{MTTF}$	2.19	12.6	.543	1.75
σ(MTTF)	.65	7.43	.093	.33



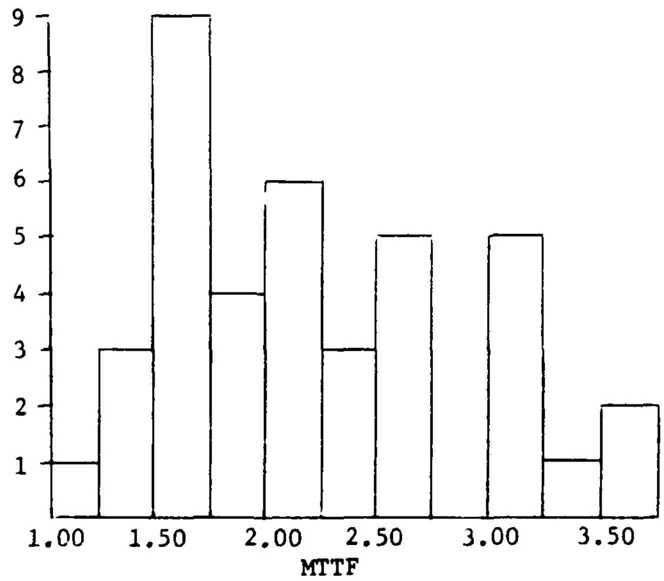
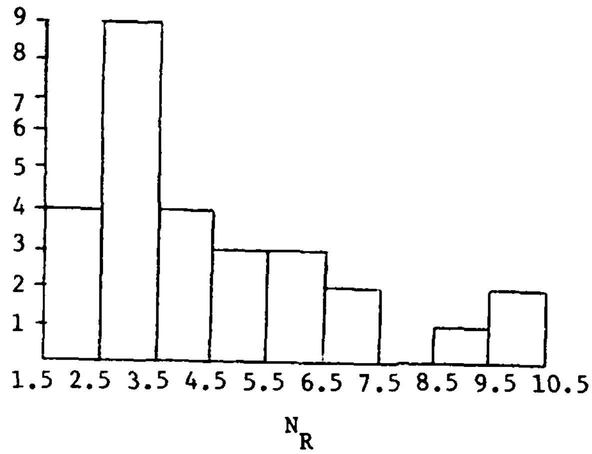


Fig. 7. Histograms of reliability parameters for Example I.



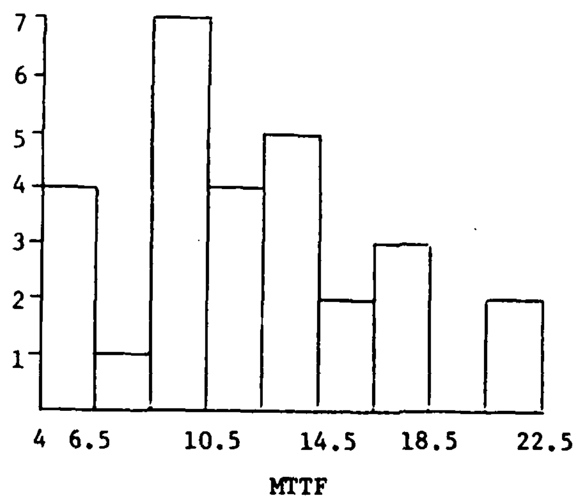
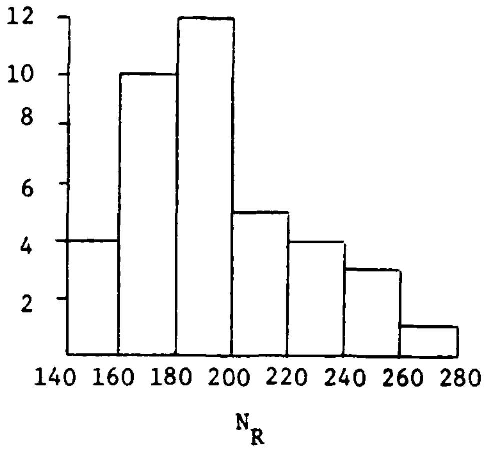


Fig. 8. Histograms of reliability parameters for Example II.



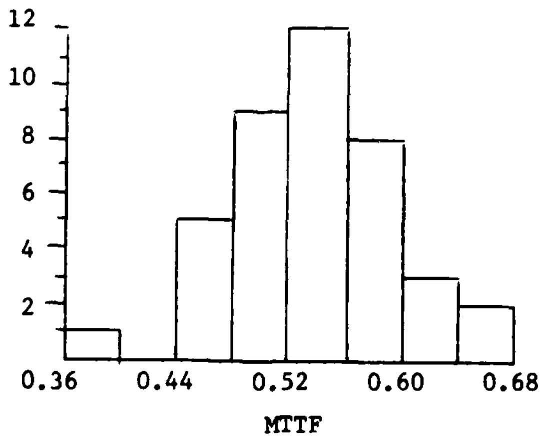


Fig. 9. Histograms of reliability parameters for Example III.

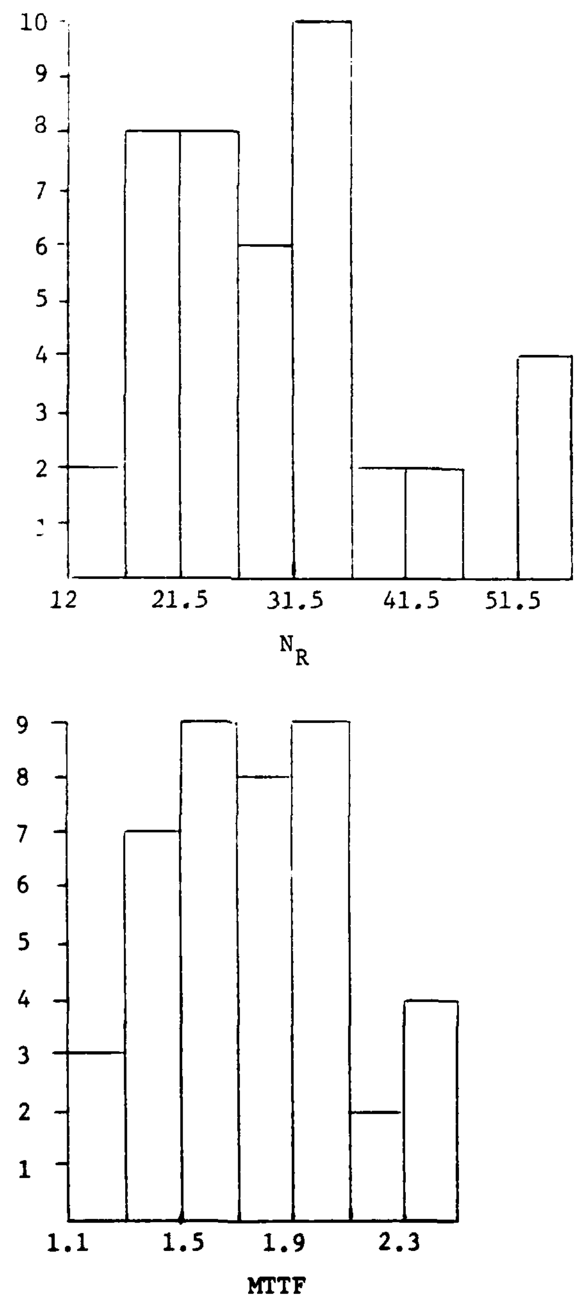


Fig. 10. Histograms of reliability parameters for Example IV.

4.0 Conclusions and Recommendations

4.1 RELY I

We have developed and displayed a model which we believe more accurately describes an actual testing environment of a large software package. This new generalized model has been incorporated in an estimation algorithm for the purpose of discerning reliability of the software from its test data. The first version of the algorithm RELY I described here converges in a given region of interest of the model parameters. RELY I is applicable to software test cases where tape replacement (software revision) occurs at uniform intervals of time, and where sufficiently reliable information is available concerning the number of errors detected during each of the successive intervals.

4.2 Data Requirements

Data required for RELY I are simple, viz., the time interval T between software revisions (tape replacements), and the sequence $\Delta_d(k)$ during each of the K successive versions of the software, where k = 1, 2, ..., K. Secondly, the data must be from a process of the type upon which the assumptions of the model were based, viz., the testing of large-scale software packages such as that in the F-16 control system.

There must be an identifiable single continuous line of software package identity throughout the test process. The package passes successively through a sequence of versions k, k = 1, 2, ..., K. At any given time during test, the software is in only one of the versions, "all" of which software version is being tested.[1] Each version, k, is identical to the preceding version, k - 1, and succeeding version, k + 1, except for the software corrections "counted" in $\Delta_c(k)$ and $\Delta_c(k + 1)$, respectively. Figure 11 indicates the time relationship of the several sequential quantities. The requirement is that $\Delta_d(k)$ be precisely known for each version k, where all versions are identified and satisfy this single and continuous identity as described. This requirement is violated if a major untested version is suddenly introduced midstream, or if an alternate part of the software package simultaneously being tested suddenly is adopted. The generated error feature can accommodate a minor amount of this kind of violation, but generated errors are modeled as occurring as a constant proportion of the correction rate.

Errors are usually classified into certain arbitrary categories, ranging from those obviously to be counted, to those of doubtful pertinence (obviously "repeated" errors, errors associated purely with erroneous test conduct, etc.). Suffice it here to suggest that the criterion for counting a given error or not will be related to its likelihood of occurrence, and its interpretation as a "failure", under operational conditions.

4.3 Recommendations

Recommendations toward improved interfacing with information in a real test process (thus taking greater advantage of inherent features

---

[1] That is, all the parts of the software package are being exercised in a manner representing that for which the reliability factors, e.g., MTTF, are to be applied later.

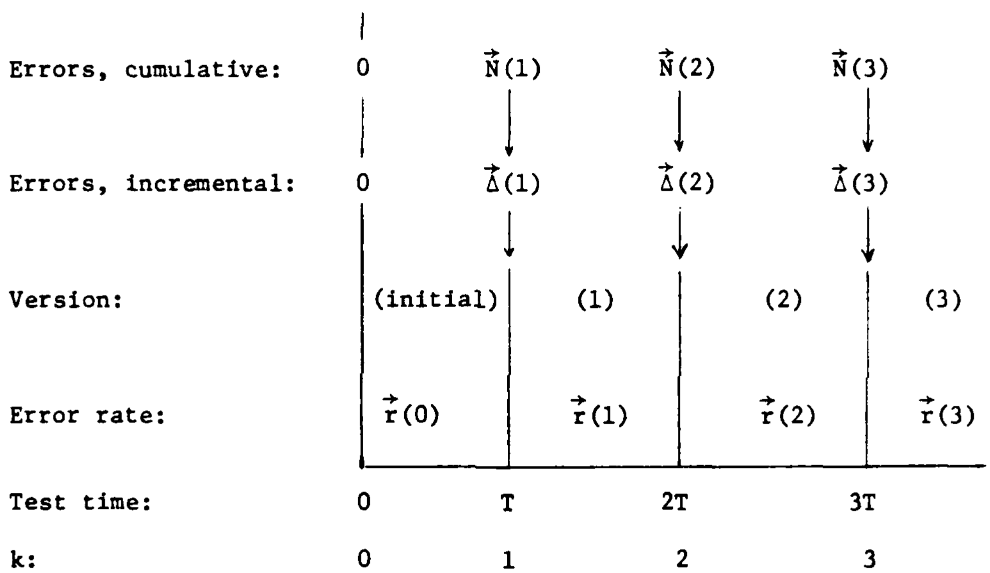


Fig. 11. Tested software in only one version at a time, identical to neighboring versions except for the changes counted in Δ_c at the respective boundaries.

of the underlying new model) include the following further work:

1. Revise the KOST subroutine to accommodate nonuniform test intervals $T(k)$, by using a finite difference technique for the solution of $\vec{\delta}(k, \vec{\theta})$. The increased utility is expected to far outweigh the lesser analytic tractability of the resulting system and the possible increase in required computation time.

2. Apply the algorithm to real data. Available data should be gathered, studied, and adapted, by interpretation and transformations, to the requirements of RELY. Residual functions over a variety of cases will indicate how well the model represents the real test process. Experience will lead to further recommendations concerning data requirements, and to possible improvements in RELY such as provisions for using information in real data concerning error correction and error generation.

   For example, the quantity $n_a(k)$,

$$n_a(k) = N_a - n_c(k) = \frac{N_o - n_c(k) + n_g(k)}{1 - \gamma} = \frac{n(k)}{1 - \gamma}$$

   is the augmented number of errors remaining in the tested software. That is, $n_a(k)$ is the number of errors which would be detected henceforth if the testing process were to continue indefinitely, including those generated after time k. The number of errors remaining in the software, exluding those yet to be generated in the correction process, is

$$n(k) = (1 - \gamma)\ n_a(k)$$

The parameter γ is assumed not observable in the present implementation of the model. If, however, among the detected errors, generated errors are distinguishable from original errors, then the additional quantity $n_{gd}(k)$, the number of generated errors detected, is available. The model state is easily augmented to include $n_{gd}(k)$. The model remains unchanged but, to the extent that the additional information is available in test data, the model parameter γ becomes observable.

Experience in the development of RELY I suggests further investigation of the nature of the $J(\vec{\theta})$ surface. Such investigation should include also the surface associated with the alternative cost functional using cumulative functions $N(k)$, rather than the incremental number of errors $\Delta(k)$. Convergence properties in certain regions of parameter space may be significantly improved using $N(k)$ rather than their derivatives $\Delta(k)$. Indeed, parallel computation using each, respectively, may prove both feasible and advantageous. Another gradient type parameter estimation method, such as the Fletcher-Powell deflected gradient method, may also prove more efficient with the alternative functional.

REFERENCES

1. G. J. Schick and R. W. Wolverton, "An Analysis of Competing Software Reliability Models", *IEEE Transactions on Software Engineering*, March 1978, pp. 104-120, A030437.

2. A. N. Sukert, "A Software Reliability Modeling Study", Rome Air Development Center, Report RADC-TR-76-247, August 1976.

3. Z. Jelinski and P. B. Moranda, "Software Reliability Research", *Statistical Computer Performance Evaluation*, edited by Walter Freiberger, Academic Press, London, England, 1972, p. 464 ff.

4. M. L. Shooman, "Probabilistic Models for Software Reliability Prediction", *Record of 1973 IEEE Symposium on Computer Software Reliability*, New York, May 1973.

5. J. Tal, "Development and Evaluation of Software Reliability Estimators", University of Utah Report UTEC SR 77-013, December 31, 1976.

6. A. L. Goel, "Summary of Technical Progress on Bayesian Software Prediction Models", Rome Air Development Center, Report RADC-TR-77-112, March 1977, A039022.

7. M. L. Shooman and S. Natarajan, "Effect of Manpower Deployment and Bug Generation on Software Error Models", Rome Air Development Center, Report RADC-TR-76-400, January 1977, A036106.

8. Y. Bard, "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems", *SIAM Journal of Numerical Analysis*, Vol. 7, March 1970, pp. 157-186.

9. R. Fletcher, "Function Minimization without Evaluating Derivatives -- A Review", *Computer Journal*, Vol. 8, pp. 33-41.

APPENDIX A

MAIN AND SUBROUTINE DESCRIPTIONS

1. MAIN and Subroutine Diagram

The subroutines of RELY I are indicated in Fig. A.1. Internal

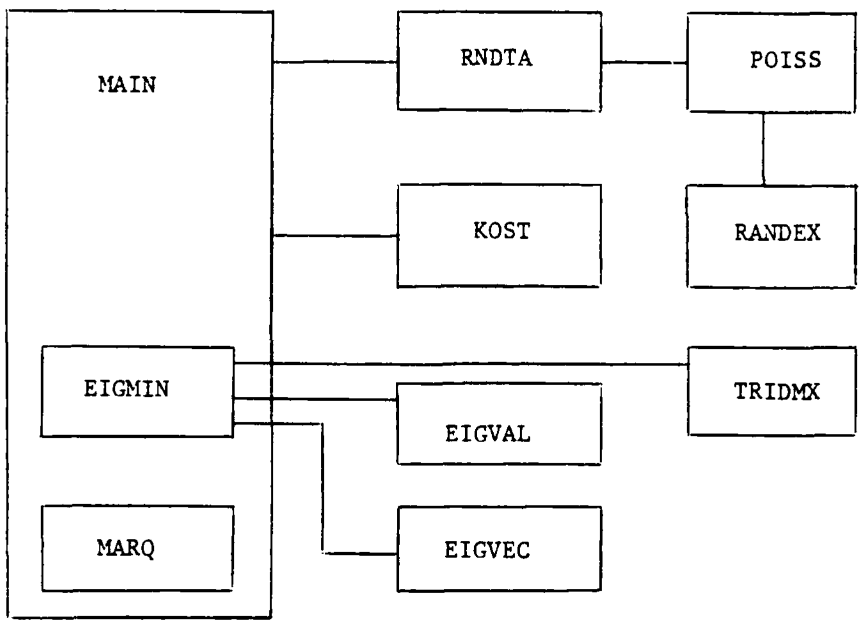


Fig. A.1. RELY I subroutine diagram.

subroutines EIGMIN and MARQ are shown, as well as the UNIVAC MATH-PACK library subroutines RANDEX, TRIDMX, EIGVAL, and EIGVEC. A brief description of MAIN and its subroutines follows.

2. MAIN

MAIN reads input data, executes the estimation algorithm (see Sec. 3.2), and prints output. It also computes simulated random data $\Delta_d(k)$ under the ISIM = 1 option (see Appendix C).

3. Subroutine RNDTA (A, B, P, NA, T, V, RR, JJJ)

From α, β, ϕ_a, N_a, T, the input initial random numbers and L (corresponding to RNDTA variables A, B, P, NA, T, RR, JJJ, respectively, and corresponding to the MAIN variables ALPH, BETA, PHI, NA, TD, RRR, NN, respectively), RNDTA computes the sequence $\Delta_d(k)$, k = 1, 2, ... L (the RNDTA variable V, and MAIN variable S). The resulting sequence $\Delta_d(k)$ is used as simulated data, the (incremental) number of errors detected successively in each software test interval. The initial random number RRR is passed through POISS to RANDEX.

RNDTA, at each interval k, integrates the system equations:

$$n_d(k) = N_d(k-1) + \phi_a T \left(N_a - n_c\,(k-1)\right)$$

$$n_c(k) = n_c(k-1) + \alpha\phi_a T \left(N_a - n_c\,(k-1)\right) + \beta T\left[N_d(k-1) - n_c(k-1)\right]$$

$$N_d(0) = n_c(0) = 0$$

using the cumulative (random) number of errors detected $N_d(k-1)$, to obtain $n_c(k)$ for use in POISS. Subroutine POISS generates the random integer $\Delta_d(k)$ according to the mean detection rate $r_d(k) = \phi_a\left[N_a - n_c(k)\right]$.

4. Subroutine POISS (DD, ZZ, PP, TT, RRRR, KKK)

From $n_a \left(=N_a - n_c\right)$, ϕ_a, T, the input initial random number RRR, and k (POISS variables DD, PP, TT, RRRR, and KKK, respectively, corresponding to RNDTA variables D, RP, RT, RQ, and K), POISS computes the value $\Delta_d(k)$, according to

$$\sum_{i=1}^{m} C(i) < T, \quad \Delta_d(k) = m$$

The random sequence C(i), i = 1, 2, ... 100, with exponential distribution function $1 - e^{r_d C}$, $r_d = \phi\left(N_a - n_c\right)$, is generated by RANDEX. The starting random number required by RANDEX in C(1) is the input initial random number RRR for k = 1, and is the preceding random number $R\left(2^{25}\right)$ for k > 1, where R is the value C(100) previously computed for the $(k - 1)^{th}$ pass.

5. Subroutine RANDEX (C, 100, U)

Reference: UNIVAC Large Scale System MATH-PACK, Programmer's Reference, UP-7542, Rev. 1.

RANDEX produces a set of 100 pseudo-random numbers C with exponential distribution

$$1 - C^{-UC}$$

by operating on a uniformly distributed variate X, according to the inverse transform method

$$C = \frac{-\ln(1 - X)}{U} .$$

RANDEX uses two other UNIVAC MATH-PACK subroutines RANDU and RANDN. RANDU generates X, $0 \leq X < 1$, for which computation it calls RANDN for random integers $0 \leq I < 2^{35}$. RANDEX requires an initial value, $0 \leq C(1) < 2^{35}$, different integer parts of which produce different output sequences.

6. Subroutine KOST (AA, BB, PP, NNA, SS, NJJ, TT, NMN, DT, H, GJ, DND, COST, DNC, RMTTF, ZNC, RERR)

Given values of the parameters $\vec{\theta}$, time interval T, number of intervals K, test data $\Delta_d(k)$, KOST computes the incremental error sequences (see Sec. 2.2)

$$\delta_d(k) = N_a\phi_a(1 \quad 0)L^{k-1} B$$

$$\delta_c(k) = N_a\phi_a(0 \quad 1)L^{k-1} B$$

where $(1 \quad 0)$ and $(0 \quad 1)$ are the transposes of the vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$, respectively, and

$$L = \begin{pmatrix}1 & -\phi_a T \\ \beta T & 1 - T\left(\beta + \alpha\phi_a\right)\end{pmatrix}, \qquad B = \begin{pmatrix}T \\ \alpha T\end{pmatrix}$$

KOST further computes the cost scalar (see Sec. 3.1)

$$J = \sum_{k=1}^{K} \left[\delta_d(k) - \Delta_d(k)\right]^2$$

the gradient vector components

$$\frac{\partial J}{\partial \theta_m} = 2 \sum_{k=1}^{K} \delta_d(k) \frac{\partial \delta_d(k)}{\partial \theta_m}$$

and the Hessian matrix elements

$$\frac{\partial^2 J}{\partial \theta_n \partial \theta_m} = 2 \sum_{k=1}^{K} \left[\delta_d(k) \frac{\partial^2 \delta_d(k)}{\partial \theta_n \partial \theta_m} + \frac{\partial \delta_d(k)}{\partial \theta_n} \frac{\partial \delta_d(k)}{\partial \theta_m}\right]$$

KOST also computes the associated estimate of total errors corrected

$$n_c = \sum_{k=1}^{K} \delta_c(k)$$

the number of errors remaining

$$n_R = N_a - n_c$$

and the mean time to failure

$$MTTF = \frac{1}{\phi_a n_R}$$

The first derivatives above are given by:

$$\frac{\partial \delta_d(k)}{\partial \theta_m} = N_a \phi_a (1 \quad 0) L^{k-2} \left[(k-1) \frac{\partial L}{\partial \theta_m} B + L \frac{\partial B}{\partial \theta_m} \right]$$

$$+ \delta_d(k) \left[\frac{1}{\phi_a} \frac{\partial \phi_a}{\partial \theta_m} + \frac{1}{N_a} \frac{\partial N_a}{\partial \theta_m} \right]$$

where

$$\frac{\partial \theta_i}{\partial \theta_m} = \begin{cases} 1 & i = m \\ 0 & i \neq m \end{cases}$$

and

$$\frac{\partial L}{\partial \alpha} = \begin{pmatrix} 0 & 0 \\ 0 & -T\phi_a \end{pmatrix} \qquad \frac{\partial B}{\partial \alpha} = \begin{pmatrix} 0 \\ T \end{pmatrix}$$

$$\frac{\partial L}{\partial \beta} = \begin{pmatrix} 0 & 0 \\ T & -T \end{pmatrix} \qquad \frac{\partial B}{\partial \beta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\frac{\partial L}{\partial \phi_a} = \begin{pmatrix} 0 & -T \\ 0 & -\alpha T \end{pmatrix} \qquad \frac{\partial B}{\partial \phi_a} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\frac{\partial L}{\partial N_a} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad \frac{\partial B}{\partial N_a} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The second derivatives are:

$$\frac{\partial^2 \delta_d(k)}{\partial\theta_n \partial\theta_m} = N_a \phi_a (1 \quad 0) \left\{ (k-2) L^{k-3} \frac{\partial L}{\partial \theta_n} \left[(k-1) \frac{\partial L}{\partial \theta_m} B + L \frac{\partial B}{\partial \theta_m} \right] \right.$$

$$+ L^{k-2} \left[(k-1) \left(\frac{\partial^2 L}{\partial\theta_n \partial\theta_m} B + \frac{\partial L}{\partial \theta_m} \frac{\partial B}{\partial \theta_n} \right) + \frac{\partial L}{\partial \theta_n} \frac{\partial B}{\partial \theta_m} \right.$$

$$\left. \left. + L \frac{\partial^2 B}{\partial\theta_n \partial\theta_m} \right] \right\} + \frac{\partial}{\partial \theta_n} \left\{ \delta_d(k) \left[\frac{1}{\phi_a} \frac{\partial \phi_a}{\partial \theta_m} + \frac{1}{N_a} \frac{\partial N_a}{\partial \theta_m} \right] \right\}$$

7. Subroutine EIGMIN (HH, CORR, GGJ, DDAM, EH, EV)

Given the Hessian HH(4, 4) and the gradient GGJ(4) of the cost functional $J(\vec{\theta})$, and the current Marquardt parameter (λ) DDAM, EIGMIN computes the eigenvalues EV(4) and eigenvectors EGV(4, 4), and the outer product EH(4, 4, 4), for J. EIGMIN then computes λ_i such that the Marquardt matrix $\left(H_i + \lambda_i C_i^2\right)$ is positive definite, and the corresponding parameter correction factors CORR(4) = $\left(H_i + \lambda_i C_i^2\right)^{-1} \vec{g}_i$.

8. <u>Subroutine TRIDMX (N, NM, A, D, B)</u>

Reference: UNIVAC Large-Scale System MATH-PACK, Programmer's Reference, UP-7542, Rev. 1, Sec. 9, p. 1.

TRIDMX transforms a real symmetric matrix, B(4, 4), to tri-diagonal form using Householder's method, where D(4) are the resulting diagonal elements and B(4) are the off-diagonal elements. Input integers N and NM are equal to the order, 4, of B.

9. <u>Subroutine EIGVAL (LP, E, A, B, W, F)</u>

Reference: UNIVAC Large-Scale Systems MATH-PACK, Programmer's Reference, UP-7542, Rev. 1, Sec. 9, p. 8.

EIGVAL evaluates the eigenvalues of a symmetric tridiagonal matrix, using Sturm sequences. A(4) are the diagonal elements and B(4) are the off-diagonal elements of the matrix. The eigenvalues E(4) are stored in descending order of absolute value.

10. <u>Subroutine EIGVEC (LP, NM, R, A, B, E, V, P, Q)</u>

Reference: UNIVAC Large-Scale Systems MATH-PACK, Programmer's Reference, UP-7542, Rev. 1, Sec. 9, p. 15.

EIGVEC evaluates the eigenvectors of a real symmetric tridiagonal matrix using Wilkinson's method. A(4) are the diagonal elements and B(4) are the off-diagonal elements of the matrix. E(4) are the eigenvalues, and V(4, 4) are the eigenvectors.

11. <u>Subroutine MARQ (EEH, EEV, DDLAM, CCORR, GGGJ, HHH)</u>

Given the outer products EEH(4, 4, 4) of the eigenvectors of the Hessian, the eigenvalues EEV(4), the gradient GGGJ(4), and Marquardt parameter (λ_i) DDLAM, MARQ computes the step CCORR(4), $\left(H_i + \lambda_i C_i^2\right)^{-1} g_i$, in parameter space.

APPENDIX B

RELY I GLOSSARY AND INDEX

(Library subroutines RANDEX, RANDU, RANDN, TRIDMX, EIGVAL, EIGVEC are not included here. See UNIVAC MATH-PACK references given in Appendix A for detailed information.)

MAIN (including internal subroutines EIGMIN and MARQ)

Variable	Description	Line Number
ALPH		18, 22, 25, 30
	Value for α in simulation mode	
B(4, 4)		162, 167, 170, 172
	Normalized Hessian matrix in EIGMIN	
BETA		18, 22, 25, 30
	Value for β in simulation mode	
CNC		24, 33, 34
	Cumulative values for n_c in simulation mode	
CND		23, 32, 34
	Cumulative values for n_d in simulation mode	
CORR(4)		3, 48, 57, 58, 61-63, 72, 78, 86-88, 95, 114, 115-117, 127, 137, 160, 163, 201
	Vector of corrections to parameters	
(MARQ: CCORR)		205, 207, 209, 229
COST		25, 30, 43, 46, 49, 50, 56, 65, 67, 69, 70, 73, 76, 91, 93, 94, 100, 120, 122, 124-127, 129, 130, 135, 145, 147
	Most recently computed value for the mean-squared error	
COSTI		46, 130, 145
	Cost for currently accepted parameter values used in mode B to determine whether the new estimate for the parameters reduces the cost	
DDD		194, 195
	Denominator used to normalize the matrix $(H + \lambda I)^{-1}$ in EIGMIN	

Variable	Description	Line Number
DDDD	Denominator used to normalize the matrix $(H + \lambda I)^{-1}$ in MARQ	223, 224
DIA(4)	Diagonal entries of the tridiagonalized Hessian matrix used in EIGMIN to compute eigenvalues	162, 170-172
DLAM	Value for λ	45, 48, 72, 74, 78, 95, 103, 111, 114, 128, 137
(in EIGMIN:	DDAM	160, 188, 205, 217)
(in MARQ:	DDLAM	182)
DND(300)	Incremental values in n_d	4, 25, 32, 43, 67, 76, 91, 120, 135
EGV(4, 4)	Eigenvectors for the Hessian matrix	162, 172, 176
EH(4, 4, 4)	Outer products of the eigenvectors for the Hessian matrix used to compute $(H + \lambda I)^{-1}$	3, 48, 72, 78, 95, 114, 128, 137, 160, 162, 176, 188
(in MARQ:	EEH	205, 207, 217)
ENC(300)	Incremental values for n_c	4, 25, 33, 43, 67, 76, 91, 120, 135
ERR	Value for termination criterion	5, 7, 56, 93, 126
EV(4)	Eigenvalues for the Hessian matrix	3, 48, 72, 78, 95, 114, 128, 137, 160, 162, 172, 181, 182, 185, 188
(in MARQ:	EEV	205, 207, 217)
GJ(4)	Gradient vector for the cost functional	3, 25, 43, 48, 67, 72, 76, 91, 95, 114, 120, 127, 135, 137
(in EIGMIN:	GGJ	160, 163, 201)
(in MARQ:	GGGJ	205, 207, 229)
H(4, 4)	Hessian matrix for the cost functional	3, 25, 43, 48, 67, 72, 76, 78, 91, 95, 114, 120, 127, 135, 137
(in EIGMIN:	HH	160, 162, 167, 194)
(in MARQ:	HHH	205, 207, 223)

Variable	Description	Line Number
I	Index for various loops	
ICLAM	Index which counts the number of times that λ is increased in mode B	104, 113, 144
IFLAG	Index that counts the number of times that an iteration of mode A reduces the cost by less than 1 percent	47, 70, 71, 106
INDEX	Index that counts the number of times that the step size has been reduced	51, 80, 81
ISIM	Indicates whether the run is a simulation (ISIM = 1) or an estimation with real data	15, 17
J	Index for various loops	
JDEX	JDEX = 1 indicates the first time that changing λ has been successful in a given iteration of mode B	105, 143, 144
KK	Index used for DO loop in EIGMIN for computing outer products of eigenvectors	
KL	Index used for DO loop in EIGMIN and MARQ for computing $(H + \lambda I)^{-1}$	184, 185, 188, 214, 217
LMBEX	LMBEX = 1 indicates that λ was changed in mode B	112, 122, 123
NA	Value for N_a in simulation mode	2, 18, 22, 25, 30
NJ	Number of test intervals	10, 25, 43, 67, 76, 91, 120, 135
NN	Number of tape versions	5, 7, 10, 11, 22, 25, 31, 43, 67, 76, 91, 120, 135

Variable	Description	Line Number
OFDI(4)	Off-diagonal entries in tridiagonalized Hessian matrix	162, 170-172
PCOST	Current minimum value for the cost functional	50, 69, 70, 73, 94, 124, 127, 129
PHI	Value of ϕ_a in simulation mode	18, 22, 25, 30
R(4, 4)	Matrix $(H + \lambda I)^{-1}$ computed in EIGMIN	163, 166, 188, 195, 201
RR(4, 4)	Matrix $(H + \lambda I)^{-1}$ computed in MARQ	207, 211, 217, 224, 229
REFF	Estimated number of errors remaining at the end of test period	26, 30, 44, 49, 68, 77, 92, 100, 121, 122, 125, 136, 147
RMTTF	Estimated mean time to failure	25, 30, 43, 49, 67, 76, 91, 100, 120, 122, 125, 135, 147
RRR	Randomization value in simulation mode	18, 19, 22
S(300)	Error data	3, 12, 22, 25, 38, 43, 67, 76, 91, 120, 135
T(300)	Tape version replacement times	3, 13, 25, 43, 67, 76, 91, 120, 135
TAU	Step size	57-63, 79, 86-88, 102, 115-117, 144
TD	Length of each test interval	5, 7, 13, 22, 25, 43, 67, 76, 91, 120, 135
TEMP1(4)		163, 171, 172
TEMP2(4)	Vectors which are used temporarily in the computation of the eigenvalues and eigenvectors of Hessian matrix	163, 171, 172

Variable	Description	Line Number
ZA	Value for α in simulation and estimation mode	40, 43, 49, 52, 67, 82, 91, 96, 100, 107, 120, 122, 125, 131, 135, 139, 147
ZB	Value for β in simulation and estimation mode	40, 43, 49, 53, 57, 61, 64, 66-67, 83, 86, 89-91, 97, 100, 108, 115, 118-120, 122, 125, 132, 135, 140, 147
ZN	Value for N_a in simulation and estimation mode	40, 43, 49, 55, 63, 67, 85, 88, 91, 99, 100, 110, 117, 120, 122, 125, 134, 135, 142, 147
ZNC	Estimated value for n_c at the end of the test period	26, 30, 44, 49, 68, 77, 92, 100, 121, 122, 125, 136, 147
ZP	Value for ϕ_a in simulation and estimation mode	40, 43, 49, 54, 58, 62, 67, 84, 87, 91, 98, 100, 109, 116, 120, 122, 125, 133, 135, 141, 147
ZZA	Currently accepted value for α	52, 76, 82, 96, 107, 137, 139
ZZB	Currently accepted value for β	53, 76, 83, 97, 108, 115, 132, 140
ZZN	Currently accepted value for N_a	55, 76, 85, 99, 110, 117, 134, 142
ZZP	Currently accepted value for ϕ_a	54, 76, 84, 98, 109, 116, 133, 141

Subroutine RNDTA

Variable	Description	Line Number
A (dbl)	Correction rate parameter α	1, 7, 19
B (dbl)	Correction rate parameter β	1, 8, 19
D	Estimated number of remaining errors n_a	4, 21, 18

Variable	Description	Line Number
E (dbl)	D	17, 18
JJJ	Number L of intervals (software versions)	1, 15
K	Index corresponding to k^{th} interval	15, 21, 27
NA (dbl)	Initial number N_a (γ-augmented) of software errors	1, 3, 10, 16, 17, 19
P (dbl)	Detection rate parameter ϕ_a	1, 9, 16, 19
RA	A	4, 7
RB	B	4, 8
RNA	NA	4, 10
RP	P	4, 9, 21
RQ	RR	5, 6, 21
RR	Storage place for MAIN input initial random number for RANDN. Contains first exponential random number from RANDEX upon return.	1, 6
RT	T	4, 11, 21
RZ	Poisson random $\Delta_d(k)$ returned by POISS	4, 21, 22, 27
T (dbl)	Time interval T	1, 11, 16, 19
V(300) (dbl)	RZ, Poisson random sequence $\Delta_d(k)$ returned by RNDTA	1, 12, 27

Variable	Description	Line Number
X(2) (dbl)	Temporary memory for current cumulative random $\vec{N}$	12-14, 16, 17, 19, 20, 22, 25
Y(2) (dbl)	Temporary memory for current cumulative estimate $\vec{M}$	12, 16, 19, 20

Subroutine POISS

Variable	Description	Line Number
C(100)	Set of uniformly distributed random numbers generaged by RANDEX to be used by POISS as the sequence of times between successive detected errors	2, 4, 6, 8, 11
DD	Number of remaining errors n_a	1, 7
K	Counter of successive detected errors	10, 11, 13
KKK	Number L of test intervals	1, 3
PP	Parameter value ϕ_a	1, 7
Q	Cumulative time $\Sigma_i C_i$ during the test interval, accumulated until it exceeds T	9, 11, 12
RRRR	Initial random number for starting RANDN. Its value for k = 1 is input by MAIN. Subsequent values are set by POISS, RRRR = 2^{25} C(100).	1, 4
TT	Test interval T	1, 12
U	Mean frequency $\phi_a n_a$ of the error detection r_d	7, 8
ZZ	Poisson random number of detections $\Delta_d(k)$ generated by POISS for the k^{th} interval. It is Δ_d such that: $\sum_{i=1}^{\Delta_d} C_i \leq T, \quad \Delta_d \leq 100$	1, 13, 16

APPENDIX C

PROCEDURE FOR OPERATION OF RELY I

The program listing (FORTRAN V, Appendix D) accompanying this report consists of MAIN with internal subroutines EIGMIN and MARQ, and external double-precision subroutines RNDTA, POISS, KOST, TRIDMX, EIGVAL, and EIGVEC. The subroutines which call single-precision library functions have the necessary coding for converting between double- and single-precision variables. Certain double-precision library functions are used, viz., DABS, DEXP, and DSQRT.

The input deck depends on whether data are to be simulated or are to be read from input cards.

INPUT DECK (to simulate data and estimate)

Card 1: TD, NN, ERR, [F5.2, 2X, I4, 2X, F8.6]

Card 2: ISIM [I2] (must be unity)

Card 3: ALPH, BETA, PHI, NA [4(G14.6, 2X)]

Card 4: RRR [16X, G14.1] (input initial random number)

Card 5: ZA, ZB, ZP, ZN [4(G14.6, 2X)]

INPUT DECK (to estimate using punched card data)

Card 1: TD, NN, ERR, [F5.2, 2X, I4, 2X, F8.6]

Card 2: ISIM [I2] (must be zero)

Card 3: ZA, ZB, ZP, ZN [4(G14.6, 2X)]

Card 4-(K+3): S(i), i = 1, 2, ... K [G10.1]

The integer part of any real number RRR, $0 \leq \text{RRR} < 2^{35}$, determines a unique repeatable random sequence $\Delta_d(k)$ from the simulator.

Initial guesses $\vec{\theta}_o$ for the model parameters are recommended as follows:

$$0 \leq ZA < 1. \quad \text{(typically 0.8)}$$

$$0 \leq ZB < 1. \quad \text{(typically 0.0)}$$

$$0 \leq ZP < 0.2$$

$$0 \leq ZN$$

To produce different simulated random data, the operator must change the input initial random number RRR, $0 < RRR < 2^{35}$.

Though $J < ERR$ is the internal stopping criterion, experience in random cases proved maximum pages of printed output (say, 10) to be as practical a stopping criterion as any.

The program prints out the current accepted parameter estimates $\vec{\theta}_i = \alpha, \beta, \phi, N_a$, together with running estimates of the reliability parameters N_R and MTTF, and selected auxiliary quantities, at each step i. However, the label "CHANGING LAMBDA" indicates only tentative parameter values produced in mode B (see Sec. 3.2). Therefore, these tentative values must not be taken as values which minimize the cost functional J. The final estimates of the reliability parameters are those associated with the last accepted iteration i.

APPENDIX D

RELY I PROGRAM LISTING AND SAMPLE OUTPUT

```
RELY*RELY(1).MAIN
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 NA
      DIMENSION T(300),S(300),H(4,4),GJ(4),CORR(4),EH(4,4,4),EV(4)
      DIMENSION DND(300),ENC(300)
      READ(5,887)TD,NN,ERR
  887 FORMAT(F5.2,2X,I4,2X,F8.6)
      WRITE(6,886)TD,NN,ERR
  886 FORMAT(5X,'INTERVAL LENGTH =',2X,F5.2,2X,'NUMBER OF INTERVALS =',
     12X,I4,2X,'TERMINATION CRITERION =',F8.6)
      NU=NN
      DO 14 J=1,NN
      S(J)=0
      T(J)=(J-1)*TD
   14 CONTINUE
      READ(5,150)ISIM
  150 FORMAT(I2)
      IF(ISIM.NE.1)GO TO 10
      READ(5,888)ALPH,BETA,PHI,NA
      READ(5,889)RRR
      WRITE(6,889)RRR
  889 FORMAT(16X,G14.4)
      CALL RNDTA(ALPH,BETA,PHI,NA,TD,S,RRR,NN)
      CND=0
      CNC=0
      CALL KOST(ALPH,BETA,PHI,NA,S,NU,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      WRITE(6,169)
  169 FORMAT(37X,'ALPHA',5X,' BETA ',4X,' PHI ',5X,' NA  ',5X,
     1' COST',3X,'EST.NC',4X,'   MTTF    ',4X,'NA-NC')
      WRITE(6,170)ALPH,BETA,PHI,NA,COST,ZNC,RMTTF,RERR
      DO 89 I=1,NN
      CND=CND+DND(I)
      CNC=CNC+ENC(I)
      WRITE(6,890)CND,CNC
  890 FORMAT(10X,'ND=',2X,F10.3,2X,'NC=',2X,F10.3)
   89 CONTINUE
      GO TO 11
   10 READ(5,151)(S(I),I=1,NN)
  151 FORMAT(G10.1)
   11 READ(5,888)ZA,ZB,ZP,ZN
  888 FORMAT(G14.6,2X,G14.6,2X,G14.6,2X,G14.6)
      WRITE(6,169)
      CALL KOST(ZA,ZB,ZP,ZN,S,NU,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      DLAM=1
      COST1=COST
      IFLAG=0
      CALL EIGMIN(H,CORR,GJ,DLAM,EH,EV)
   72 WRITE(6,171)ZA,ZB,ZP,ZN,COST,ZNC,RMTTF,RERR
   75 PCOST=COST
      INDEX=0
      ZZA=ZA
      ZZB=ZB
      ZZP=ZP
      ZZN=ZN
      IF(COST.LT.ERR)GO TO 73
```

```
      TAU=(5-ZB)/(2*DABS(CORR(2)))
      IF(TAU.GT.(.2-ZP)/(2*DABS(CORR(3))))TAU=(.2-ZP)/(2*DABS(CORR(3)))
      IF(TAU.GT.1)TAU=1
      IF(TAU.LT..0001)GO TO 74
      ZB=ZB-CORR(2)*TAU
      ZP=ZP-CORR(3)*TAU
      ZN=ZN-CORR(4)*TAU
      IF(ZB.LT.0)ZB=0.
      IF(COST.GT.10)DLAM=1.
      IF(ZB.GT.10.)ZB=10.
      CALL KOST(ZA,ZB,ZP,ZN,S,NJ,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      IF(COST.GE.PCOST)GO TO 76
      IF(COST/PCOST.GT..99)IFLAG=IFLAG+1
      IF(IFLAG.GT.4) GO TO 74
      CALL EIGMIN(H,CORR,GJ,DLAM,EH,EV)
      PCOST=COST
      DLAM=DLAM/10
      GO TO 72
   76 CALL KOST(ZZA,ZZB,ZZP,ZZN,S,NJ,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      CALL EIGMIN(H,CORR,GJ,DLAM,EH,EV)
   71 TAU=TAU/10
      INDEX=INDEX+1
      IF(INDEX.GT.5)GO TO 74
      ZA=ZZA
      ZB=ZZB
      ZP=ZZP
      ZN=ZZN
      ZB=ZB-CORR(2)*TAU
      ZP=ZP-CORR(3)*TAU
      ZN=ZN-CORR(4)*TAU
      IF(ZB.LT.0)ZB=0.
      IF(ZB.GT.10.)ZB=10.
      CALL KOST(ZA,ZB,ZP,ZN,S,NJ,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      IF(COST.LT.ERR)GO TO 73
      IF (COST.GE.PCOST)GO TO 71
      CALL EIGMIN(H,CORR,GJ,DLAM,EH,EV)
      ZZA=ZA
      ZZB=ZB
      ZZP=ZP
      ZZN=ZN
      WRITE(6,172)ZA,ZB,ZP,ZN,COST,ZNC,RMTTF,RERR
      GO TO 75
   74 TAU=.1
      DLAM=.01
      ICLAM=0
      JDEX=0
      IFLAG=0
      ZA=ZZA
      ZB=ZZB
      ZP=ZZP
      ZN=ZZN
   77 DLAM=10*DLAM
      LMBEX=1
      ICLAM=ICLAM+1
```

```
      CALL MARQ(EH,EV,DLAM,CORR,GJ,H)
   78 ZB=ZZB-CORR(2)*TAU
      ZP=ZZP-CORR(3)*TAU
      ZN=ZZN-CORR(4)*TAU
      IF(ZB.GT.10)ZB=10.
      IF(ZB.LT.0)ZB=0.
      CALL KOST(ZA,ZB,ZP,ZN,S,NJ,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      IF(LMBEX.EQ.1)WRITE(6,190)ZA,ZB,ZP,ZN,COST,ZNC,RMTTF,RERR
      LMBEX=0
      IF(COST.GT.PCOST)GO TO 77
      WRITE(6,191)ZA,ZB,ZP,ZN,COST,ZNC,RMTTF,RERR
      IF(COST.LT.ERR)GO TO 73
      IF(COST/PCOST.LT..9.AND.PCOST.GT.50)CALL EIGMIN(H,CORR,GJ,
     1DLAM,EH,EV)
      IF(COST/PCOST.LT..9.AND.PCOST.GT.50)GO TO 72
      IF(COST1.GT.COST)GO TO 91
      ZA=ZZA
      ZB=ZZB
      ZP=ZZP
      ZN=ZZN
      CALL KOST(ZA,ZB,ZP,ZN,S,NJ,T,NN,TD,H,GJ,DND,COST,ENC,RMTTF,
     1ZNC,RERR)
      CALL EIGMIN(H,CORR,GJ,DLAM,EH,EV)
      GO TO 72
   91 ZZA=ZA
       ZZB=ZB
       ZZP=ZP
       ZZN=ZN
      JDEX=JDEX+1
      IF(JDEX.EQ.1)TAU=TAU*5**ILLAM
      COST1=COST
       GO TO 78
   73 WRITE(6,173)ZA,ZB,ZP,ZN,COST,ZNC,RMTTF,RERR
  170 FORMAT(5X,'SIMULATION VALUES ARE     ',2X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
  171 FORMAT(5X,'NEW PARAMETER VALUES',8X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
  172 FORMAT(5X,'AFTER REDUCING STEP SIZE',4X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
  173 FORMAT(9X,'FINAL VALUES ARE',2X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
  190 FORMAT(5X,'CHANGING LAMBDA            ',2X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
  191 FORMAT(5X,'STEEPEST DESCENT VALUES   ',2X,3(F8.5,2X),
     13(F8.2,2X),F10.6,2X,F8.2)
      SUBROUTINE EIGMIN(HH,CORR,GGJ,DDAM,EH,EV)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION HH(4,4),B(4,4),EV(4),EGV(4,4),EH(4,4,4),DIA(4),OFDI(4),
     1TEMP1(4),TEMP2(4),R(4,4),CORR(4),GGJ(4)
      DO 700 I=1,4
      DO 701 J=1,4
      R(I,J)=0
      B(I,J)=HH(I,J)/DSQRT(DABS(HH(I,I)*HH(J,J)))
  701 CONTINUE
  700 CONTINUE
      CALL TRIDMX(4,4,B,DIA,OFDI)
```

```
      CALL EIGVAL(4,EV,DIA,OFDI,TEMP1,TEMP2)
      CALL EIGVEC(4,4,6,DIA,OFDI,EV,EGV,TEMP1,TEMP2)
      DO 708 I=1,4
      DO 709 J=1,4
      DO 710 KK=1,4
      EH(I,J,KK)=EGV(J,I)*EGV(KK,I)
  710 CONTINUE
  709 CONTINUE
  708 CONTINUE
      DO 713 I=1,4
      IF(EV(I).GT.0)GO TO 713
      IF(DDAM.LT.-EV(I))DDAM=DABS(EV(I))+.1
  713 CONTINUE
      DO 702 KL=1,4
      IF(DABS(EV(KL)).LT..0001)EV(KL)=1
      DO 703 I=1,4
      DO 704 J=1,4
      R(I,J)=R(I,J)+EH(KL,I,J)/(EV(KL)+DDAM)
  704 CONTINUE
  703 CONTINUE
  702 CONTINUE
      DO 705 I=1,4
      DO 706 J=1,4
      DDD=DSQRT(DABS(HH(I,I)*HH(J,J)))
      R(I,J)=R(I,J)/DDD
  706 CONTINUE
  705 CONTINUE
      DO 711 I=1,4
      CORR(I)=0
      DO 712 J=1,4
      CORR(I)=CORR(I)+R(I,J)*GGJ(J)
  712 CONTINUE
  711 CONTINUE
      RETURN
      SUBROUTINE MARQ(EEH,EEV,DDLAM,CCORR,GGGJ,HHH)
      IMPLICIT REAL*8(A-H,O-Z)
       DIMENSION EEH(4,4,4),EEV(4),CCORR(4),GGGJ(4),RR(4,4),HHH(4,4)
      DO 714 I=1,4
      CCORR(I)=0
      DO 725 J=1,4
      RR(I,J)=0
  725 CONTINUE
  714 CONTINUE
      DO 716 KL=1,4
      DO 717 I=1,4
      DO 718 J=1,4
      RR(I,J)=RR(I,J)+EEH(KL,I,J)/(EEV(KL)+DDLAM)
  718 CONTINUE
  717 CONTINUE
  716 CONTINUE
      DO 719 I=1,4
      DO 720 J=1,4
      DDDD=DSQRT(DABS(HHH(I,I)*HHH(J,J)))
      RR(I,J)=RR(I,J)/DDDD
  720 CONTINUE
  719 CONTINUE
      DO 721 I=1,4
```

```
      DO 722 J=1,4
      CCORR(I)=CCORR(I)+RR(I,J)*GGGJ(J)
  722 CONTINUE
  721 CONTINUE
      RETURN
      END
```

```
<**>

@PRT,S RELY(1).ROST
```

RELY*RELY(1).KOST

```
      SUBROUTINE KOST(AA,BB,PP,NNA,SS,NJJ,TT,NMN,DT,H,GJ,DND,
     1COST,DNC,RMTTF,ZNC,RERR)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 NNA
      DIMENSION TT(300),SS(300),GJ(4),H(4,4)
      DIMENSION DDND(4),DND(300),DNC(300),ZMTTF(300)
      DIMENSION D1(4,4)
      COST=0
      ZLL11=1
      ZLL22=1
      ZLL12=0
      ZLL21=0
      ZND=0
      ZNC=0
      DO 13 I=1,4
      DDND(I)=0
      GJ(I)=0
      DO 12 J=1,4
      H(I,J)=0
      D1(I,J)=0
   12 CONTINUE
   13 CONTINUE
      A1=-AA*PP*DT
      A2=(1-AA)*DT*BB+AA-PP*DT*AA**2
      ZL=1-DT*(BB+AA*PP)
       A7=PP*DT
       A8=(PP*DT)**2
       A9=AA*A8
       A10=A8*AA**2
       A11=AA*A7
       A12=PP*(AA*DT)**2
       D1=BB*DT+AA-AA*DT*(BB+AA*PP)
      DDND(3)=NNA*DT
      DDND(4)=PP*DT
      DO 10 K=1,NMN
      DND(K)=PP*NNA*DT*(ZLL11+AA*ZLL12)
      DNC(K)=PP*NNA*DT*(ZLL21+AA*ZLL22)
      ZND=ZND+DND(K)
      ZNC=ZNC+DNC(K)
      ZMTTF(K)=1/(PP*(NNA-ZNC))
      IF(K.EQ.NMN)RMTTF=ZMTTF(K)
      IF(K.EQ.NMN)RERR=NNA-ZNC
      IF(K.EQ.NMN)ZZNA=NNA-ZNC
      COST=COST+(SS(K)-DND(K))**2
      DO 41 I=1,4
      GJ(I)=-2*(SS(K)-DND(K))*DDND(I)+GJ(I)
      DO 42 L=1,4
      H(I,L)=2*(DDND(I)*DDND(L)-(SS(K)-DND(K))*D1(I,L))+H(I,L)
   42 CONTINUE
   41 CONTINUE
      ZLK11=ZLL11+ZLL12*BB*DT
      ZLK12=ZLL11*(-PP*DT)+ZLL12*ZL
      ZLK21=ZLL21+ZLL22*BB*DT
      ZLK22=ZLL21*(-PP*DT)+ZLL22*ZL
      DDND(1)=PP*NNA*DT*(K*A1*ZLL12+ZLK12)
      DDND(2)=PP*NNA*DT*(K*A2*ZLL12)
```

```
      DCND(3)=NNA*DT*(ZLK11+AA*ZLK12)+PP*NNA*DT*K*(-AA)*DT*
     1(ZLL11+AA*ZLL12)
      DCND(4)=PP*DT*(ZLK11+AA*ZLK12)
      IF(K.NE.1)GO TO 30
      D1(1,3)=-2*PP*NNA*DT
      D1(1,4)=-DT*PP**2
      D1(3,3)=-2*AA*NNA*DT
      D1(3,4)=DT-2*AA*PP*DT
      D1(3,1)=D1(1,3)
      D1(4,1)=D1(1,4)
      D1(4,3)=D1(3,4)
      GO TO 31
   30 D1(1,1)=PP*NNA*DT*K*((K-1)*ZM12*A9-2*ZLL12*A7)
      D1(1,2)=PP*NNA*DT*K*((K-1)*ZM12*A12-DT*ZLL12)
      D1(1,3)=NNA*DT*(K*ZLL12*(-A11)+ZLK12)+PP*NNA*DT*K*((K-1)*
     1(ZM11*A11*DT+ZM12*A10)-2*ZLL12*AA*DT-ZLL11*DT)
      D1(1,4)=PP*DT*(K*ZLL12*(-A11)+ZLK12)
      D1(2,2)=PP*NNA*DT*K*(K-1)*ZM12*B1*(-DT)
      D1(2,3)=NNA*DT*K*ZLL12*B1+PP*NNA*DT*K*(K-1)*(ZM11*(-DT)*
     1B1-ZM12*AA*DT*B1)
      D1(2,4)=PP*DT*K*ZLL12*B1
      D1(3,3)=2*NNA*DT*K*(-AA)*DT*(ZLL11+AA*ZLL12)+PP*NNA*
     1(AA**2)*(DT**3)*K*(K-1)*(ZM11+AA*ZM12)
      D1(3,4)=DT*(ZLK11+ZLK12*AA)+PP*DT*K*(-AA)*DT*(ZLL11+AA*ZLL12)
      D1(4,4)=0
      D1(2,1)=D1(1,2)
      D1(3,1)=D1(1,3)
      D1(4,1)=D1(1,4)
      D1(3,2)=D1(2,3)
      D1(4,2)=D1(2,4)
      D1(4,3)=D1(3,4)
   31 ZM11=ZLL11
      ZM12=ZLL12
      ZM21=ZLL21
      ZM22=ZLL22
      ZLL11=ZLK11
      ZLL12=ZLK12
      ZLL21=ZLK21
      ZLL22=ZLK22
   10 CONTINUE
      RETURN
      END

<**>

@PRT,S RELY(1).SUB2
```

```
RELY*RELY(1).SUB2
      SUBROUTINE RNDTA(A,B,P,NA,T,V,RR,JJJ)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 NA
      REAL RA,RB,RP,RNA,RT,D,RZ
      REAL RQ
      RQ=SNGL(RR)
      RA=SNGL(A)
      RB=SNGL(B)
      RP=SNGL(P)
      RNA=SNGL(NA)
      RT=SNGL(T)
      WRITE(6,67)
      DIMENSION X(2),V(300),Y(2)
      X(1)=0.
      X(2)=0.
       DO 100 K=1,JJJ
      Y(1)=X(1)+P*T*(NA-X(2))
      E=NA-X(2)
      D=SNGL(E)
      Y(2)=B*T*X(1)+(1-T*(B+A*P))*X(2)+A*P*T*NA
      X(2)=Y(2)
      CALL POISS(D,RZ,RP,RT,RQ,K)
      X(1)=X(1)+DBLE(RZ)
      WRITE(6,68)X(1),X(2)
   68 FORMAT(10X,'RD',2X,G14.6,2X,'RC',2X,G14.6)
      V(K)=DBLE(RZ)
  100 CONTINUE
   67 FORMAT(5X,'RANDOM VALUES FOR ND AND NC GENFRATED FOR SIMULATION')
      RETURN
      END

<**>

@PRT,S RELY(1).SUB3
```

```
RELY*RELY(1).SUB3
          SUBROUTINE POISS(DD,ZZ,PP,TT,RRRR,KKK)
          DIMENSION C(100)
          IF  (KKK.GT.1)GO TO 555
          C(1)=RRRR
          GO TO 600
      555 C(1)=C(100)*2**25
      600 U=DD*PP
          CALL RANDEX(C,100,U)
          Q=0.
          DO 100 K=1,100
          Q=Q+C(K)
          IF(Q.LT.TT)GO TO 100
          ZZ=FLOAT(K-1)
          GO TO 200
      100 CONTINUE
          ZZ=100
      200 RETURN
          END

<**>
```

```
HELY*GAUSSHEET(1).SUBO
      SUBROUTINE TRIDMX (N,NM,A,D,B)                                   TRID0010
      IMPLICIT REAL*8(A-H,O-Z)
C             ------------------------------------------------------
C                 TRIDIAGONALIZATION OF REAL SYMMETRIC MATRIX.
C             ------------------------------------------------------
      DIMENSION A(NM,NM),D(NM),B(NM)                                   TRID0020
C             ------------------------------------------------------
C                 SAVE ORIGINAL DIAGON/LS IN ARRAY D.
C             ------------------------------------------------------
      DO 10 I=1,N                                                      TRID0040
   10 D(I)=A(I,I)                                                      TRID0050
C             ------------------------------------------------------
C                 FOR N=2 RETURN WITHOUT COMPUTING.
C             ------------------------------------------------------
      IF(N-2)60,55,15                                                  TRID0050
C             ------------------------------------------------------
C             ------------------------------------------------------
   15 DO 46 K=3,N                                                      TRID0060
      KK=K-1                                                           TRID0070
C             ------------------------------------------------------
C                 SUM CONTAINS THE SUM OF THE SQUARE ELEMENTS
C                 OF A COLUMN, EXCEPT THE FIRST K-2 ELEMENTS.
C             ------------------------------------------------------
      SUM=A(K-1,K-2)*A(K-1,K-2)                                        TRID0080
      DO 20 J=K,N                                                      TRID0090
   20 SUM=SUM+A(J,K-2)*A(J,K-2)                                        TRID0100
C             ------------------------------------------------------
C                 THE B ARRAY CONTAINS THE BETA VALUES.
C                 (I.E., BETA=SUM**(1/2))
C             ------------------------------------------------------
      B(K-2)=DSIGN(DSQRT(SUM),-A(K-1,K-2))
C             ------------------------------------------------------
C                 IF BETA=0 NO TRANSFORMATION IS INITIATED.
      IF(B(K-2)) 24,46,24                                              TRID0120
C             ------------------------------------------------------
C                 THE COMPONENTS OF THE COLUMN VECTOR W ARE
C                 STORED IN THE POSITIONS OF THE ANNIHILATED
C                 ELEMENTS OF A (I.E., LOWER HALF OF A)
C             ------------------------------------------------------
   24 A(K-1,K-2)=DSQRT(0.5*DABS(A(K-1,K-2)/B(K-2))+0.5)
      DENOM=-2.*A(K-1,K-2)*B(K-2)                                      TRID0140
      DO 30 I=K,N                                                      TRID0160
   30 A(I,K-2)=A(I,K-2)/DENOM                                          TRID0170
      SCAL=0.                                                          TRID0175
   37 DO 36 J=KK,N                                                     TRID0180
C             ------------------------------------------------------
C                 THE BETAS ARE FORMED ONE BY ONE.
C                 WHEN R OF THEM HAVE BEEN FORMED.
C                 P AND Q CONTAIN ONLY (N-R) ELEMENTS.
C                 THE E ARRAY IS UED TO STORE SUCCESSIVE
C                 P,S AND Q,S.
C             ------------------------------------------------------
      B(J)=0.                                                          TRID0190
      IF(J.EQ.KK)GO TO 350                                             TRID0191
      DO 349 L=KK,JJ                                                   TRID0192
  349 B(J)=B(J)+A(J,L)*A(L,K-2)                                        TRID0193
```

```
 350  DO 35 L=J,N                                                    TRID0200
 35   B(J)=B(J)+A(L,J)*A(L,K-2)                                      TRID0210
 C              ----------------------------  --------------------------
 C                   SCAL=W(TRANSPOSE)*P
      JJ=J                                                           TRID0215
 36   SCAL=SCAL+B(J)*A(J,K-2)                                        TRID0220
      DO 40 J=KK,N                                                   TRID0230
 40   B(J)=B(J)-SCAL*A(J,K-2)                                        TRID0240
 C              ------------------------------------------------------
 C                   TRANSFORM ALL ELEMENTS OF A
 C                   EXCEPT PIVOTAL ROW AND COLUMN.
 C              ------------------------------------------------------
      DO 45 J=KK,N                                                   TRID0250
      DO 45 L=J,N                                                    TRID0260
 45   A(L,J)=A(L,J)-2.*(A(L,K-2)*B(J)+A(J,K-2)*P(L))                 TRID0270
  46  CONTINUE                                                       TRID0275
 C              ------------------------------------------------------
 C                   RESTORE ORIGINAL DIAGONALS OF A. STORE DIAGOANL
 C                   OF TRANSFORMED MATRIX IN ARRAY D.
 C              ------------------------------------------------------
      DO 50 I=1,N                                                    TRID0280
      T=A(I,I)                                                       TRID0290
      A(I,I)=D(I)                                                    TRID0300
      J=N-I                                                          TRID0301
      D(J+1)=D(J)                                                    TRID0302
 50   D(I)=T                                                         TRID0310
  55  B(N)=A(N,N-1)                                                  TRID0320
  60  B(1)=0.0                                                       TRID0330
      RETURN                                                         TRID0340
      END
```

<**>

@PRT,S GAUSSNEW1.SOL7

```
RELY*GAUSSN.E*T(1).SUB7
            SUBROUTINE EIGVAL(LP,E,A,B,W,F)                                   EVAL
            IMPLICIT REAL*8(A-H,O-Z)
      C                 -------------------------------------------------
      C                 LP IS THE SIZE OF ARRAY A.                        EVAL
      C                 E IS A VECTOR OF LP ELEMENTS WHICH WILL HOLD      EVAL
      C                 THE EIGENVALUES IN DESCENDING ABSOLUTE ORDER.     EVAL
      C                 A IS A VECTOR OF LP ELEMENTS GIVING THE DIAGONAL  EVAL
      C                 ELEMENTS OF THE TRIDIAGONAL MATRIX.               EVAL
      C                 B IS A VECTOR OF LP ELEMENTS, THE LAST LP - 1     EVAL
      C                 W IS A VECTOR OF LP ELEMENTS USED FOR TEMPORARY   EVAL
      C                 STORAGE.                                          EVAL
      C                 F IS A VECTOR OF LP ELEMENTS USED FOR TEMPORARY   EVAL
      C                 STORAGE.                                          EVAL
      C                 -------------------------------------------------
            DIMENSION E(LP),A(LP),B(LP),W(LP)                                 EVAL
            DIMENSION F(LP)
      C                 -------------------------------------------------
      C                 FIND ABSOLUTE BOUND FOR THE EIGENVALUES           EVAL
      C                 -------------------------------------------------
            AM=DABS(A(1))
            BM=0.                                                             EVAL
            DO 1 I=2,LP                                                       EVAL
            AM=DMAX1(AM,DABS(A(I)))
        1   BM=DMAX1(BM,DABS(B(I)))
            BD=AM+BM+BM                                                       EVAL
            DO 6 I=1,LP                                                       EVAL
      C                 -------------------------------------------------
      C                 THIS LOOP FORCES THE EIGENVALUES TO LIE BETWEEN   EVAL
      C                 PLUS AND MINUS ONE. THE E AND W VECTORS ARE       EVAL
      C                 RESPECTIVELY LOW AND HIGH ESTIMATES TO ALL        EVAL
      C                 THE EIGENVALUES.                                  EVAL
      C                 -------------------------------------------------
            A(I)=A(I)/BD                                                      EVAL
            B(I)=B(I)/BD                                                      EVAL
            E(I)=-1.0                                                         EVAL
      6     W(I)=1.0                                                          EVAL
            DO 50 K=1,LP                                                      EVAL
      C                 -------------------------------------------------
      C                 FIND THE K-TH EIGENVALUE. ALSO LOW AND HIGH       EVAL
      C                 ESTIMATES FOR THE K+1-ST TO LP-TH EIGENVALUES     EVAL
      C                 ARE IMPROVED.  THE EIGENVALUES ARE FOUND IN       EVAL
      C                 ASCENDING ORDER.                                  EVAL
      C                 THE K-TH EIGENVALUE IS CONSIDERED FOUND IF THE    EVAL
      C                 THE HIGH AND LOW PLACES AGREE TO SEVEN DECIMAL    EVAL
      C                 PLACES.                                           EVAL
      C                 -------------------------------------------------
        8   IF((W(K)-E(K))/DMAX1(DABS(W(K))+DABS(E(K)),1.E-29)-5.E-8)50,50,10
       10   X=(W(K)+E(K))*0.5                                                 EVAL
      C                 -------------------------------------------------
      C                 X IS A GUESS FOR THE K-TH EIGENVALUE. COMPUTE     EVAL
      C                 NUMBER OF EIGENVALUES EQUAL OR EXCEEDING X BY     EVAL
      C                 USING STURM SEQUENCE (ORTEGA'S METHOD).           EVAL
      C                 -------------------------------------------------
            S2=1.0                                                            EVAL
            F(1)=A(1)-X                                                       EVAL
            IF(F(1)) 102,104,104                                              EVAL
```

```
      102  S1=-1.0                                                                    EVAL
           N=0                                                                        EVAL
           GO TO 105                                                                  EVAL
      104  S1=1.0                                                                     EVAL
           N=1                                                                        EVAL
      105  DO 120 I=2,LP                                                              EVAL
           IF(B(I)) 106,113,106                                                       EVAL
      106  IF(B(I-1)) 107,114,107                                                     EVAL
      107  IF(DABS(F(I-1))+DABS(F(I-2))-1.E-15)111,112,112
C                          ------------------------------------------------------
C                          IF THE PREVIOUS TWO TERMS OF THE STURM SEQUENCE         EVAL
C                          WERE VERY SMALL, THEY ARE FORCED TO BE CLOSER           EVAL
C                          TO ONE IN MAGNITUDE TO AVOID UNDERFLOW PROBLEMS.        EVAL
C                          ------------------------------------------------------
      111  F(I-1)=F(I-1)*1.E15                                                        EVAL
           F(I-2)=F(I-2)*1.E15                                                        EVAL
      112  F(I)=(A(I)-X)*F(I-1)-B(I)*B(I)*F(I-2)                                      EVAL
           GO TO 115                                                                  EVAL
      113  F(I)=(A(I)-X)*S1                                                           EVAL
           GO TO 115                                                                  EVAL
      114  F(I)=(A(I)-X)*F(I-1)-DSIGN(B(I)*B(I),S2)
      115  S2=S1                                                                      EVAL
           IF(F(I))116,117,116                                                        EVAL
      116  S1=DSIGN(S1,F(I))
           IF(S1+S2)117,120,117                                                       EVAL
      117  N=N+1                                                                      EVAL
      120  CONTINUE                                                                   EVAL
C                          ------------------------------------------------------
C                          NOW LET N BE THE NUMBER OF                              EVAL
C                          EIGENVALUES SMALLER THAN X.                             EVAL
C                          ------------------------------------------------------
           N=LP-N                                                                     EVAL
           IF(N.LT.K) GO TO 20                                                        EVAL
C                          ------------------------------------------------------
C                          X BECOMES AN UPPER BOUND FOR THE                        EVAL
C                          K-TH TO N-TH EIGENVALUES.                               EVAL
C                          ------------------------------------------------------
      12   DO 15 J=K,N                                                                EVAL
      15   W(J)=X                                                                     EVAL
      20   N=N+1                                                                      EVAL
C                          ------------------------------------------------------
C                          IF ALL THE EIGENVALUES ARE SMALLER THAN X,              EVAL
C                          TEST WHETHER WE HAVE CONVERGED TO THE                   EVAL
C                          K-TH EIGENVALUE.                                        EVAL
C                          ------------------------------------------------------
           IF(LP.LT.N) GO TO 8                                                        EVAL
      24   DO 26 J=N,LP                                                               EVAL
C                          ------------------------------------------------------
C                          IF X IS LARGER THAN PREVIOUS LOWER                      EVAL
C                          BOUND, INCREASE THE LOWER BOUND.                        EVAL
C                          ------------------------------------------------------
           IF(X - E(J))8,8,26
      26   E(J)=X                                                                     EVAL
           GO TO 8                                                                    EVAL
      50   CONTINUE                                                                   EVAL
C                          ------------------------------------------------------
C                          RESTORE INPUT AND SCALE EIGENVALUES.                    EVAL
```

```
C          --------------------------------------------------
      DO 60 I=1,LP                                                      EVAL
      A(I)=A(I)*BD                                                      EVAL
      B(I)=B(I)*BD                                                      EVAL
   60 W(I)=(W(I)+E(I))*BD*0.5                                           EVAL
C-----------------------------------------------------------------------
C                    SORT EIGENVALUES IN ABSOLUTE DESCENDING ORDER      EVAL
C-----------------------------------------------------------------------
      J=LP                                                              EVAL
      K=1                                                               EVAL
      DO 80 I=1,LP                                                      EVAL
      IF(DABS(W(K))-DABS(W(J)))63,63,65
   63 E(I)=W(J)                                                         EVAL
      J=J-1                                                             EVAL
      GO TO 80                                                          EVAL
   65 E(I)=W(K)                                                         EVAL
      K=K+1                                                             EVAL
   80 CONTINUE                                                          EVAL
      RETURN                                                            EVAL
      END
```

```
<**>

@PRT,S GAUSSNEWT.SUBB
```

```
RELY*GAUSSNEWT(1).SUBB
                 SUBROUTINE EIGVEC(LP,NM,R,A,B,E,V,P,Q)                         EVEC
                 IMPLICIT REAL*8(A-H,O-Z)
      C                    ---------------------------------------------------
      C                    R IS THE GIVEN MATRIX.                               EVEC
      C                    LP IS DIMENSION OF THE MATRIX R.                     EVEC
      C                    NM IS THE MAXIMUM DIMENSION OF R AND V.              EVEC
      C                    A ARE THE DIAGONAL ELEMENTS OF THE TRIDIAGONALIZED R.EVEC
      C                    B ARE THE OFF-DIAGONAL ELEMENTS OF TRIDIAGONALIZED R.EVEC
      C                    E ARE THE EIGENVECTORS OF R.                         EVEC
      C                    V WILL HOLD THE EIGENVECTORS,STORED COLUMNWISE.      EVEC
      C                    P AND Q ARE VECTORS FOR TEMPORARY STORAGE TO         EVEC
      C                    HOLD COEFFICIENTS OF THE LINEAR EQUATIONS WHICH      EVEC
      C                    DETERMINE THE EIGENVECTORS.                          EVEC
      C                    ---------------------------------------------------
                 DIMENSION R(NM,NM),A(LP),B(LP),E(LP),V(NM,NM),P(LP),Q(LP)      EVEC
                 LP1=LP-1                                                       EVEC
                 DO 50 IX=1,LP                                                  EVEC
                 X=A(1)-E(IX)                                                   EVEC
                 Y=B(2)                                                         EVEC
      C                    ---------------------------------------------------
      C                    GENERATE COEFFICIENTS TO COMPUTE THE IX-TH           EVEC
      C                    EIGENVECTOR, FIRST PICK BEST PIVOT ELEMENT.          EVEC
      C                    ---------------------------------------------------
                 DO 10 I=1,LP1                                                  EVEC
                 IF(DABS(X)-DABS(B(I+1)))4,6,8
      4          P(I)=B(I+1)                                                    EVEC
                 Q(I)=A(I+1)-E(IX)                                              EVEC
                 V(I,IX)=B(I+2)
                 Z=-X/P(I)                                                      EVEC
                 X=Z*Q(I)+Y                                                     EVEC
                 IF(LP1.NE.I) Y=Z*V(I,IX)                                       EVEC
                 GO TO 10                                                       EVEC
      6          IF(X) 8,7,8                                                    EVEC
      7          X=1.0E-10                                                      EVEC
      8          P(I)=X                                                         EVEC
                 Q(I)=Y                                                         EVEC
                 V(I,IX)=0.0
                 X=A(I+1)-(B(I+1)/X*Y+E(IX))                                    EVEC
                 Y=B(I+2)                                                       EVEC
      10         CONTINUE                                                       EVEC
      C                    ---------------------------------------------------
      C                    NOW SOLVE THE ABOVE EQUATIONS FOR THE EIGENVECTOR.   EVEC
      C                    TEST LAST PIVOT ELEMENT.                             EVEC
      C                    ---------------------------------------------------
      20         IF(X)21,20,21                                                  EVEC
      21         V(LP,IX)=1.0/X                                                 EVEC
      C                    ---------------------------------------------------
      C                    CONTINUE WITH THE BACK SOLUTION.                     EVEC
      C                    ---------------------------------------------------
      22         I=LP1                                                          EVEC
                 V(I,IX)=(1.-Q(I)*V(LP,IX))/P(I)                                EVEC
                 X=V(LP,IX)**2+V(I,IX)**2                                       EVEC
      25         I=I-1                                                          EVEC
                 IF(I)26,30,26                                                  EVEC
      26         V(I,IX)=(1.-(Q(I)*V(I+1,IX)+V(I,IX)*V(I+2,IX)))/P(I)           EVEC
                 X=X+V(I,IX)**2                                                 EVEC
```

```
              GO TO 25
      28      V(LP,IX)=1.0E10                                             EVEC
              GO TO 22                                                    EVEC
        30    X=DSQRT(X)
              DO 31 I=1,LP                                                EVEC
      31      V(I,IX)=V(I,IX)/X                                           EVEC
      C                  ------------------------------------------------
      C                  TRANSFORM EIGENVECTOR FOR THE TRIDIAGONAL        EVEC
      C                  MATRIX TO AN EIGENVECTOR OF THE ORIGINAL MATRIX. EVEC
      C                  ------------------------------------------------
      C                  ------------------------------------------------
              IF(LP.EQ.2) GO TO 50                                        EVEC
              DO 42 KK=2,LP1                                              EVEC
              K = LP - KK + 1
              Y=0.0                                                       EVEC
              DO 35 I=K,LP                                                EVEC
      35      Y=Y+V(I,IX)*R(I,K-1)                                        EVEC
              DO 40 I=K,LP                                                EVEC
        40    V(I,IX)=V(I,IX)-2.0*Y*R(I,K-1)
      42      CONTINUE                                                    EVEC
      50      CONTINUE                                                    EVEC
              RETURN                                                      EVEC
              END

<**>

@XQT RECT(1).ABS
     INTERVAL LENGTH =   1.50  NUMBER OF INTERVALS =    60  TERMINATION CRITERION = .100000
```

DMCG MINIMIZATION WITH ALPHA FIXED

@ADT KELLY(1).RDS
INTERVAL LENGTH = 1.50 NUMBER OF INTERVALS = 60 TERMINATION CRITERION = .100000
.2317+006
RANDOM VALUES FOR RD AND RC GENERATED FOR SIMULATION

RD	.700000+001	RC	.420000+001
RD	.140000+002	RC	.873160+001
RD	.170000+002	RC	.135387+002
RD	.220000+002	RC	.179736+002
RD	.250000+002	RC	.224001+002
RD	.330000+002	RC	.265197+002
RD	.350000+002	RC	.311348+002
RD	.400000+002	RC	.352607+002
RD	.460000+002	RC	.394312+002
RD	.540000+002	RC	.437884+002
RD	.600000+002	RC	.486006+002
RD	.640000+002	RC	.534899+002
RD	.700000+002	RC	.581431+002
RD	.770000+002	RC	.629007+002
RD	.820000+002	RC	.678946+002
RD	.830000+002	RC	.727847+002
RD	.870000+002	RC	.769885+002
RD	.940000+002	RC	.810735+002
RD	.960000+002	RC	.855099+002
RD	.101000+003	RC	.894877+002
RD	.101000+003	RC	.935353+002
RD	.103000+003	RC	.968908+002
RD	.106000+003	RC	.999724+002
RD	.107000+003	RC	.102977+003
RD	.112000+003	RC	.105616+003
RD	.115000+003	RC	.108557+003
RD	.117000+003	RC	.111444+003
RD	.118000+003	RC	.114137+003
RD	.122000+003	RC	.116520+003
RD	.124000+003	RC	.119095+003
RD	.127000+003	RC	.121530+003
RD	.128000+003	RC	.123996+003
RD	.133000+003	RC	.126194+003
RD	.134000+003	RC	.128765+003
RD	.134000+003	RC	.131046+003
RD	.136000+003	RC	.132937+003
RD	.139000+003	RC	.134805+003
RD	.139000+003	RC	.136805+003
RD	.140000+003	RC	.138460+003
RD	.143000+003	RC	.139985+003
RD	.143000+003	RC	.141696+003
RD	.144000+003	RC	.143116+003
RD	.146000+003	RC	.144445+003
RD	.149000+003	RC	.145845+003
RD	.149000+003	RC	.147454+003
RD	.151000+003	RC	.148796+003
RD	.154000+003	RC	.150197+003
RD	.154000+003	RC	.151615+003

```
MINIMIZATION WITH ALPHA FIXED                                                                   DATE 06/11/79     PAGE    3
          ND      .156000+003  NC      .153155+003
          ND      .158000+003  NC      .154564+003
          ND      .158000+003  NC      .156035+003
          ND      .159000+003  NC      .157252+003
          ND      .162000+003  NC      .158412+003
          ND      .165000+003  NC      .159625+003
          ND      .166000+003  NC      .161445+003
          ND      .166000+003  NC      .162937+003
          ND      .166000+003  NC      .164174+003
          ND      .168000+003  NC      .165201+003
          ND      .168000+003  NC      .166351+003
          ND      .170000+003  NC      .167305+003
                                            ALPHA      BETA       PHI        NA       COST    EST.NC      MTTF      NA-NC
      SIMULATION VALUES ARE                .70000    .10000    .02000    200.00    160.14    169.45   1.636535    30.55
          ND=        6.000  NC=            4.200
          ND=       11.874  NC=            8.582
          ND=       17.617  NC=           13.095
          ND=       23.224  NC=           17.699
          ND=       28.693  NC=           22.356
          ND=       34.022  NC=           27.037
          ND=       39.211  NC=           31.717
          ND=       44.259  NC=           36.375
          ND=       49.168  NC=           40.994
          ND=       53.938  NC=           45.559
          ND=       58.572  NC=           50.059
          ND=       63.070  NC=           54.485
          ND=       67.435  NC=           58.828
          ND=       71.670  NC=           63.084
          ND=       75.778  NC=           67.247
          ND=       79.761  NC=           71.315
          ND=       83.621  NC=           75.284
          ND=       87.363  NC=           79.154
          ND=       90.988  NC=           82.923
          ND=       94.500  NC=           86.591
          ND=       97.903  NC=           90.159
          ND=      101.198  NC=           93.627
          ND=      104.389  NC=           96.997
          ND=      107.479  NC=          100.269
          ND=      110.471  NC=          103.444
          ND=      113.368  NC=          106.526
          ND=      116.172  NC=          109.515
          ND=      118.886  NC=          112.414
          ND=      121.514  NC=          115.224
          ND=      124.057  NC=          117.946
          ND=      126.519  NC=          120.587
          ND=      128.901  NC=          123.145
          ND=      131.207  NC=          125.622
          ND=      133.438  NC=          128.022
          ND=      135.598  NC=          130.346
          ND=      137.687  NC=          132.596
          ND=      139.709  NC=          134.775
          ND=      141.666  NC=          136.885
          ND=      143.559  NC=          138.928
          ND=      145.392  NC=          140.905
          ND=      147.164  NC=          142.819
          ND=      148.880  NC=          144.672
          ND=      150.540  NC=          146.465
```

ND=	152.146	NC=	148.200
ND=	153.700	NC=	149.880
ND=	155.203	NC=	151.505
ND=	156.658	NC=	153.078
ND=	158.066	NC=	154.601
ND=	159.428	NC=	156.074
ND=	160.746	NC=	157.499
ND=	162.021	NC=	158.879
ND=	163.254	NC=	160.214
ND=	164.448	NC=	161.505
ND=	165.603	NC=	162.755
ND=	166.720	NC=	163.964
ND=	167.801	NC=	165.134
ND=	168.847	NC=	166.267
ND=	169.859	NC=	167.362
ND=	170.838	NC=	168.422
ND=	171.786	NC=	169.446

	ALPHA	BETA	PHI	NA	COST	EST.NC	MTTF	NA-NC
NEW PARAMETER VALUES	.50000	.00000	.01000	300.00	270.22	109.04	.523658	190.96
NEW PARAMETER VALUES	.50000	.00260	.01018	281.09	247.67	113.79	.587004	167.31
NEW PARAMETER VALUES	.50000	.00577	.01063	268.74	234.29	122.45	.643107	146.30
NEW PARAMETER VALUES	.50000	.00931	.01119	259.56	223.30	131.80	.699277	127.76
NEW PARAMETER VALUES	.50000	.01311	.01181	252.04	213.48	140.45	.758780	111.59
NEW PARAMETER VALUES	.50000	.01711	.01245	245.57	204.67	147.87	.822415	97.69
NEW PARAMETER VALUES	.50000	.02127	.01308	239.88	196.88	153.95	.889636	85.93
NEW PARAMETER VALUES	.50000	.02557	.01370	234.86	190.15	158.76	.959138	76.11
NEW PARAMETER VALUES	.50000	.02997	.01429	230.42	184.43	162.44	1.029219	67.98
NEW PARAMETER VALUES	.50000	.03448	.01485	226.50	179.65	165.18	1.098074	61.31
NEW PARAMETER VALUES	.50000	.03907	.01538	223.04	175.70	167.17	1.164046	55.87
NEW PARAMETER VALUES	.50000	.04373	.01586	219.99	172.48	168.55	1.225810	51.44
NEW PARAMETER VALUES	.50000	.04844	.01630	217.30	169.87	169.48	1.282470	47.82
NEW PARAMETER VALUES	.50000	.05321	.01671	214.94	167.77	170.06	1.333565	44.88
NEW PARAMETER VALUES	.50000	.05801	.01708	212.85	166.08	170.39	1.379016	42.46
NEW PARAMETER VALUES	.50000	.06284	.01741	211.02	164.74	170.54	1.419030	40.48
NEW PARAMETER VALUES	.50000	.06768	.01771	209.40	163.67	170.56	1.454002	38.84
NEW PARAMETER VALUES	.50000	.07252	.01798	207.97	162.82	170.50	1.484426	37.47
NEW PARAMETER VALUES	.50000	.07733	.01822	206.71	162.14	170.38	1.510828	36.33
CHANGING LAMBDA	.50000	.07894	.01840	205.23	161.76	169.84	1.535925	35.39
STEEPEST DESCENT VALUES	.50000	.07894	.01840	205.23	161.76	169.84	1.535925	35.39
STEEPEST DESCENT VALUES	.50000	.08695	.01927	197.81	160.38	166.73	1.669540	31.08
STEEPEST DESCENT VALUES	.50000	.09497	.02015	190.39	159.94	163.11	1.818564	27.29
STEEPEST DESCENT VALUES	.50000	.10299	.02103	182.97	160.46	159.03	1.985748	23.95
NEW PARAMETER VALUES	.50000	.09497	.02015	190.39	159.94	163.11	1.818564	27.29
NEW PARAMETER VALUES	.50000	.10230	.02033	191.34	159.70	164.27	1.817116	27.07
NEW PARAMETER VALUES	.50000	.10391	.02044	191.89	159.59	165.02	1.821219	26.86
NEW PARAMETER VALUES	.50000	.10537	.02050	192.17	159.56	165.41	1.822824	26.76
NEW PARAMETER VALUES	.50000	.10672	.02053	192.31	159.54	165.60	1.823137	26.71
CHANGING LAMBDA	.50000	.10725	.02051	192.27	159.56	165.48	1.820099	26.79
CHANGING LAMBDA	.50000	.10684	.02054	192.32	159.54	165.61	1.823145	26.71
STEEPEST DESCENT VALUES	.50000	.10684	.02054	192.32	159.54	165.61	1.823145	26.71
STEEPEST DESCENT VALUES	.50000	.11006	.02061	192.62	159.53	166.01	1.823420	26.61
STEEPEST DESCENT VALUES	.50000	.11328	.02060	192.91	159.53	166.41	1.823836	26.51
NEW PARAMETER VALUES	.50000	.11006	.02061	192.62	159.53	166.01	1.823420	26.61
CHANGING LAMBDA	.50000	.11057	.02058	192.57	159.54	165.87	1.819853	26.70
CHANGING LAMBDA	.50000	.11017	.02061	192.62	159.53	166.01	1.823302	26.61
CHANGING LAMBDA	.50000	.11007	.02061	192.62	159.53	166.01	1.823438	26.61
STEEPEST DESCENT VALUES	.50000	.11007	.02061	192.62	159.53	166.01	1.823438	26.61

STEEPEST DESCENT VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11119	.02066	192.91	159.52	166.38	1.824744	26.52
STEEPEST DESCENT VALUES	.50000	.11119	.02066	192.91	159.52	166.38	1.824744	26.52
NEW PARAMETER VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11150	.02064	192.96	159.53	166.36	1.821065	26.61
CHANGING LAMBDA	.50000	.11111	.02067	192.99	159.52	166.50	1.825411	26.50
CHANGING LAMBDA	.50000	.11101	.02068	193.01	159.52	166.52	1.825662	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825683	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
STEEPEST DESCENT VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
NEW PARAMETER VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11150	.02064	192.96	159.53	166.36	1.821065	26.61
CHANGING LAMBDA	.50000	.11111	.02067	192.99	159.52	166.50	1.825411	26.50
CHANGING LAMBDA	.50000	.11101	.02068	193.01	159.52	166.52	1.825662	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825683	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
STEEPEST DESCENT VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
NEW PARAMETER VALUES	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11150	.02064	192.96	159.53	166.36	1.821065	26.61
CHANGING LAMBDA	.50000	.11111	.02067	192.99	159.52	166.50	1.825411	26.50
CHANGING LAMBDA	.50000	.11101	.02068	193.01	159.52	166.52	1.825662	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825683	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49
CHANGING LAMBDA	.50000	.11100	.02068	193.01	159.52	166.52	1.825685	26.49

The above sample output listing is given here only to illustrate the format as programmed in RELY I at the time of delivery. The values shown are of no significance relative to the content of the report (though they are those of a sample from Example IV, Sec. 3.3), nor are they expected to be repeatable exactly with implementation of RELY I at a different computer facility. The sample is offered as an aid to users, showing format and exemplary behavior in a given random case.