COMPLETELY MONOTONE REGRESSION ESTIMATES OF SOFTWARE FAILURE RATES

by

Douglas R. Miller
Department of Operations Research
The George Washington University
Washington, DC 20052

and

Ariela Sofer
Department of Systems Engineering
George Mason University
Fairfax, Virginia 22030

GWU/IMSE/Serial T-497/85
January 18, 1985

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

Research Supported
by
National Aeronautics and Space Administration
Grant NAG-1-179

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

COMPLETELY MONOTONE REGRESSION ESTIMATES OF SOFTWARE FAILURE RATES

by

Douglas R. Miller
Department of Operations Research
The George Washington University
Washington, DC 20052

and

Ariela Sofer
Department of Systems Engineering
George Mason University
Fairfax, Virginia 22030

Abstract

GWU/IMSE/Serial T-497/85
January 18, 1985

A new method for estimating the present failure rate of a program is presented. A crude nonparametric estimate of the failure rate function is obtained from past failure times. This estimate is then smoothed by fitting a completely monotonic function, which is the solution of a quadratic programming problem. The value of the smoothed function at present time is used as the estimate of present failure rate. A Monte Carlo study gives an indication of how well this method works.

Research Supported
by
National Aeronautics and Space Administration
Grant NAG-1-179

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

COMPLETELY MONOTONE REGRESSION ESTIMATES OF SOFTWARE FAILURE RATES

by

Douglas R. Miller
Department of Operations Research
The George Washington University
Washington, DC 20052

and

Ariela Sofer
Department of Systems Engineering
George Mason University
Fairfax, Virginia 22030

GWU/IMSE/Serial T-497/85
January 18, 1985

Introduction

Suppose a program contains some bugs each of which eventually manifests itself as a failure of the program to execute correctly. If each bug is removed when it causes a failure, failures should occur at a decreasing rate and the program exhibit reliability growth. Suppose successive failures occur at times

$$0 < t_1 < t_2 < t_3 \ \ldots < t_n \tag{1}$$

and the number of failures observed in $[0,t]$ is denoted as $n(t)$, $0 \leq t$; Figure 1 shows an example of such data, $\{n(t) , 0 \leq t \leq t_{50}\}$. Important

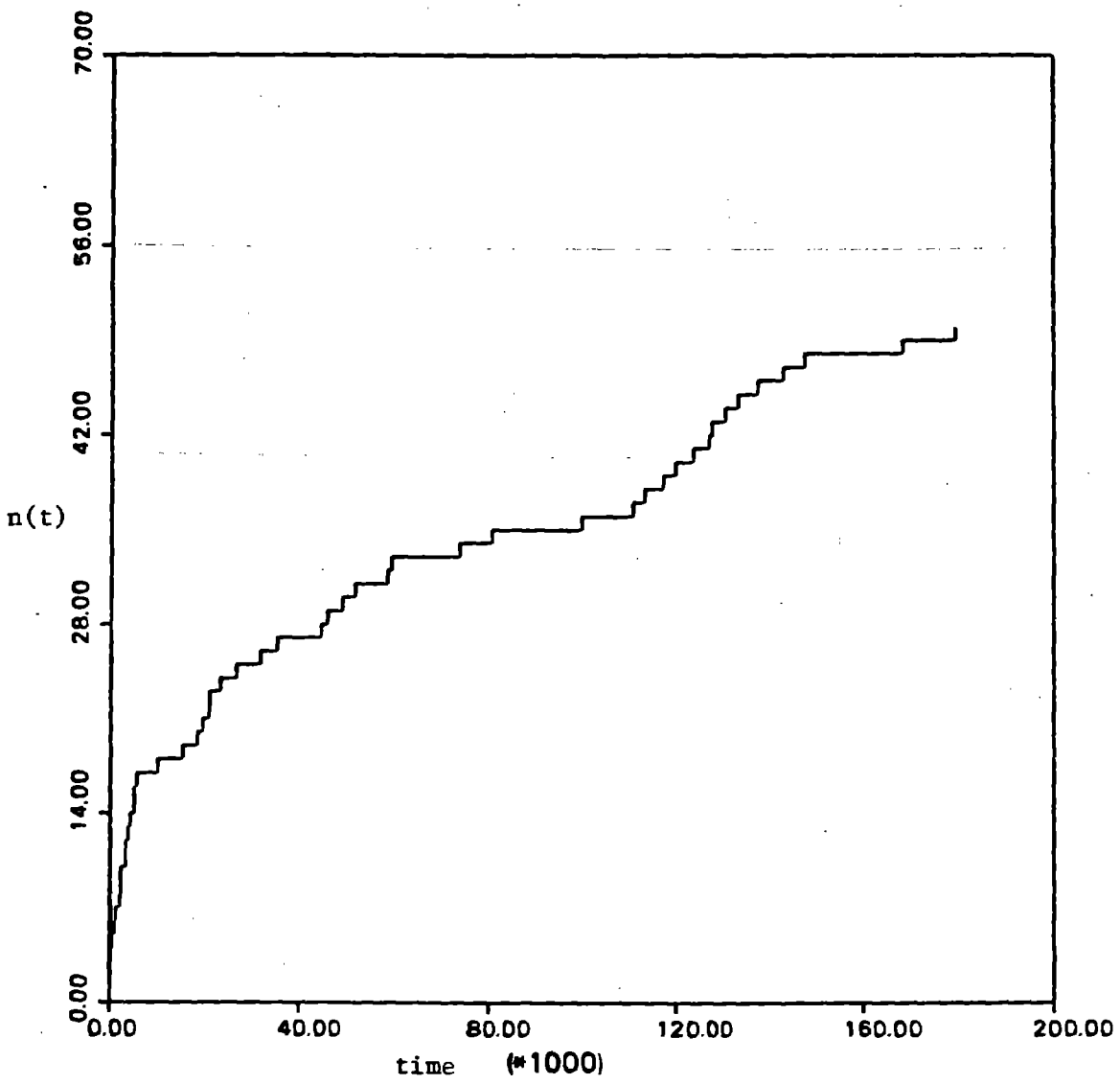

Figure 1. Observed number of failures as a function of time; n(t) = number of failures in [0, t] .

problems in the area of software reliability focus on data as in (1) or Figure 1: we would like to make various statistical inferences from the data. Assuming a time-homogeneous usage environment, what can be said about future failures? How many failures are expected over some finite future horizon? What is the distribution of time until the next failure? What is the present failure rate of the program? (If bugs causing future failures are not removed at those failure times, the present failure rate should remain constant into the future.) Many papers in the software reliability literature address these and related issues. The purpose of this paper is to address the problem of estimating the present failure rate of a program. Our method is new; it consists of smoothing a raw estimate of the failure rate with a completely monotone function.

Reliability Growth Models and Complete Monotonicity

One approach to the inference problems arising from the observed failure times in (1) is postulation of probability models for the failure times. Such models include Jelinsky-Moranda [8], Goel-Okumoto [7], Duane-Crow [4,3], Littlewood [9], and Musa-Okumoto [13]. These are all parametric models. The general approach consists of selecting a particular model by goodness-of-fit, past quality-of-prediction, or some other criterion and then, using that model (with estimated parameters), predict the future or give estimates of the current failure rate. It is interesting to note that all the above models have a common property: complete monotonicity of failure rate function.

Let $N(t)$ equal the (random) number of failures observed in $[0,t]$ and let $M(t) = EN(t)$ be the expected number. The intensity function of the point process $\{N(t),\ 0 \le t\}$ is $m(t) = \frac{d}{dt} M(t)$, $0 \le t$; $m(\cdot)$ is also sometimes referred to as the failure rate of the process. A function $m(\cdot)$ is completely monotone if and only if it possesses derivatives of all orders and

$$(-1)^n \frac{d^n m(t)}{dt^n} \ge 0 , \quad 0 \le t , \quad n \ge 0 , \tag{2}$$

see Feller [5, p. 439]. It is simple to verify that all of the above mentioned software reliability growth models satisfy (2). Futhermore if the failure times are modeled as order statistics of independent but not necessarily identically distributed Exponential random variables, (2) is also satisfied; see Miller [11]. Finally the class of completely monotone functions is identical to the totality of intensity functions for the family of doubly stochastic Exponential order statistic processes; see [11] again. Thus, the set of completely monotone intensity functions seems to be a natural basis for a nonparametric approach to estimating the failure rate. Our approach will be to find a completely monotone rate function which, in some sense, best fits the failure data in (1). Various specific formulations of the problem are possible; we present two closely related formulations here.

Problem Statement - First Formulation

Consider failure data as in (1). A raw estimate of the failure rate function is

$$\hat{r}(t) = \frac{1}{t_{i+1} - t_i} \quad , \quad t_i \leq t < t_{i+1} \quad , \quad i=1,\ldots,n \; . \tag{3}$$

(This is a rather crude and naive estimate, but it does have some nice properties such as being totally nonparametric and it has appeared in the reliability literature, e.g. [12].) The above failure rate estimate is shown in Figure 2 for the data from Figure 1. (A very naive estimate of $r(t_n)$ would be $\hat{r}(t_n^-)$.) Our goal is to find the "closest" completely monotone function $r^*(t)$, $0 \leq t \leq t_n$, to $\hat{r}(t)$, $0 \leq t \leq t_n$; we shall use the criterion of "least-squares." It appears necessary to formulate the problem in discrete time. We shall use the failure times of (1): let $\hat{r}_i = \hat{r}(t_i^-)$ and $r_i^* = r^*(t_i^-)$. The least squares distance for the discretized problem is

$$D(\hat{r} \; , \; r^*) = \sum_{i=1}^{n} (\hat{r}_i - r_i^*)^2 \; (t_i - t_{i-1}) \tag{4}$$

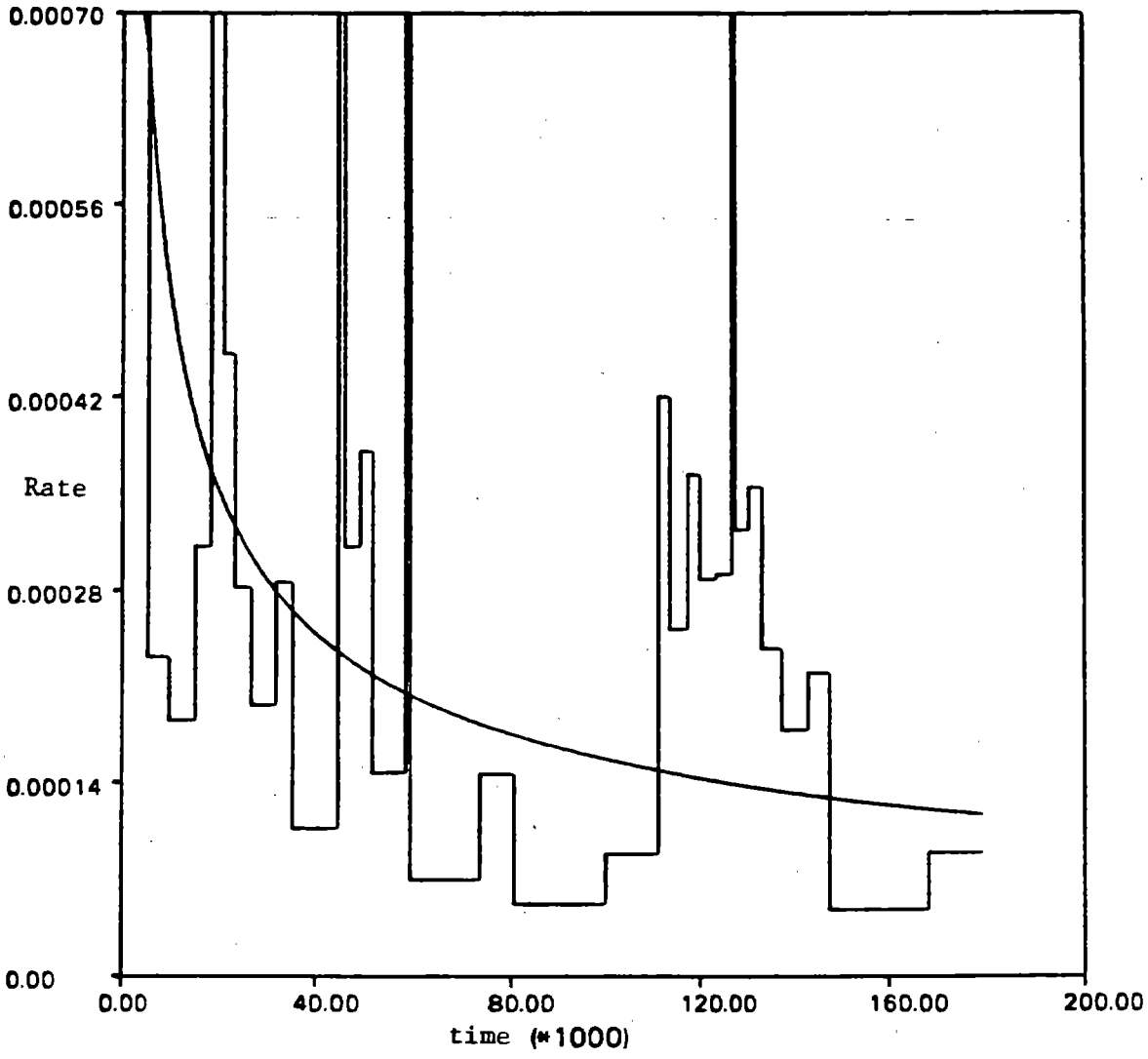

Figure 2. Raw estimated failure rate for data in Figure 1 superimposed on true rate $r(t) = 1/\sqrt{400t}$

There is some problem in carrying the complete monotonicity of $r^*(\cdot)$ over to the sequence $\{r_i^*, i=1,\ldots,n\}$ because of the unequal spacing between discrete time points. Thus we shall temporarily consider only the first two derivatives, which lead to

$$\Delta r_i^* = \frac{r_i^* - r_{i-1}^*}{t_i - t_{i-1}} \leq 0 \qquad i=2,\ldots,n \qquad (5)$$

$$\Delta^2 r_i^* = \frac{\Delta r_i^* - \Delta r_{i-1}^*}{t_i - t_{i-1}} \geq 0 \qquad i=3,\ldots,n \qquad (6)$$

We now have two optimization problems:

I. For given $\{\hat{r}_i, i=1,2,\ldots n\}$ find $\{r_i^*, i=1,2,\ldots n\}$ which minimizes $D(\hat{r}, r^*)$ subject to the constraints $r_n^* \geq 0$, $r_i^* - r_{i-1}^* \leq 0$, $i=2,3,\ldots,n$.

II. For given $\{\hat{r}_i, i=1,2,\ldots,n\}$ find $\{r_i^*, i=1,2,\ldots,n\}$ which minimizes $D(\hat{r}, r^*)$ subject to the constraints $r_n^* \geq 0$, $r_i^* - r_{i-1}^* \leq 0$, $i=2,3,\ldots,n$, and $(r_i^* - r_{i-1}^*)/(t_i - t_{i-1}) \geq (r_{i-1}^* - r_{i-2}^*)/(t_{i-1} - t_{i-2})$, $i=3,4,\ldots,n$.

Note that the above statements both relax the constraint of complete monotonicity considerably: Problem I requires only monotonicity and Problem II requires only convexity and monotonicity.

The above problems are both quadratic programs. We have encoded an algorithm to solve them which is described in the Appendix. The solutions of Problems I and II for the data of Figure 1 were computed and are presented in Figures 3 and 4, respectively. The data of Figure 1 is actually Monte Carlo simulated data from a nonhomogeneous Poisson process with intensity $r(t) = 1/\sqrt{400t}$, $0 \leq t$, a Duane-Crow model. This "true" failure rate function is shown on Figures 2, 3, and 4. In this example

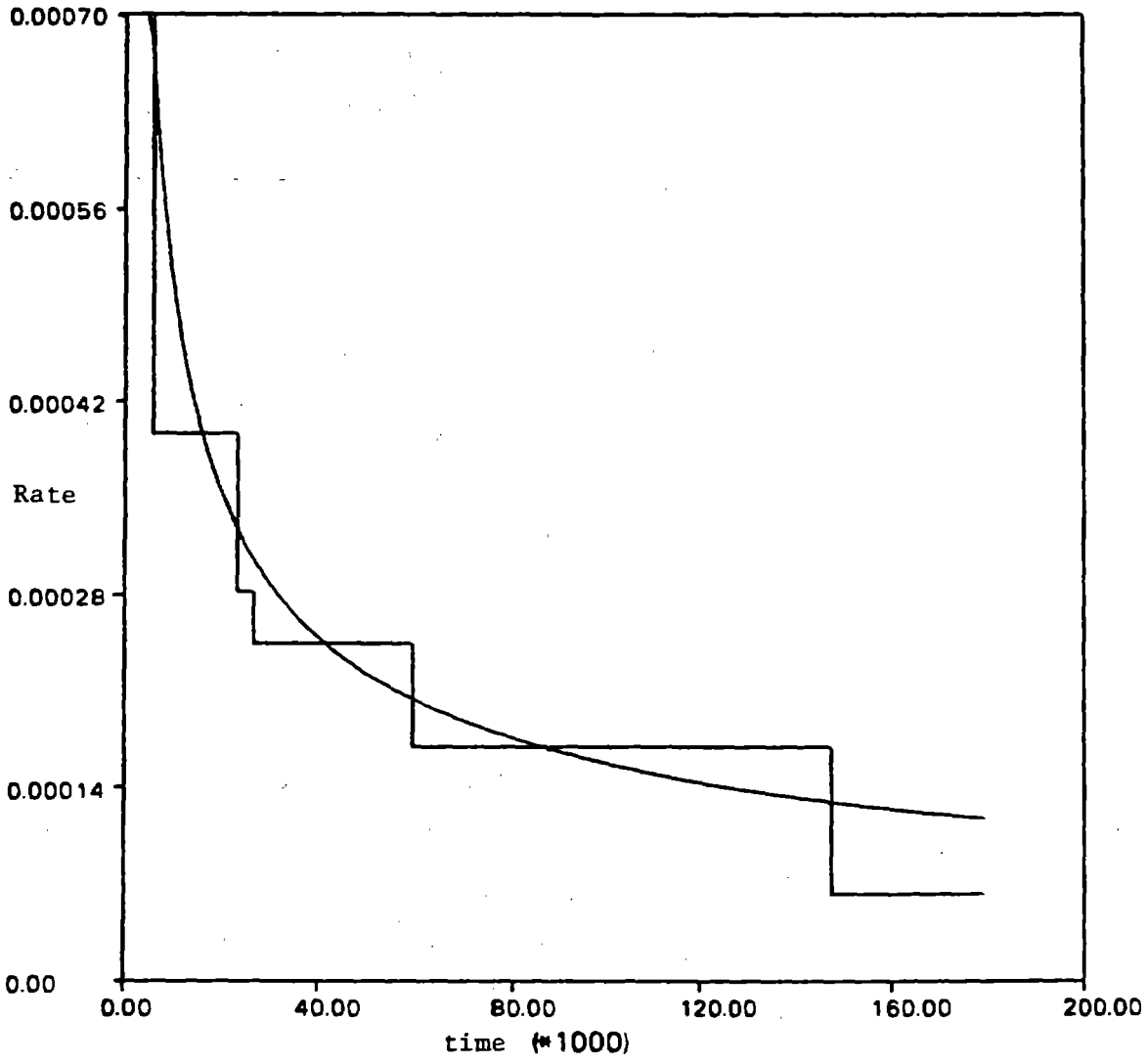

Figure 3. Monotone estimate of failure rate for data in Figure 1 superimposed on true rate $r(t) = 1/\sqrt{400t}$

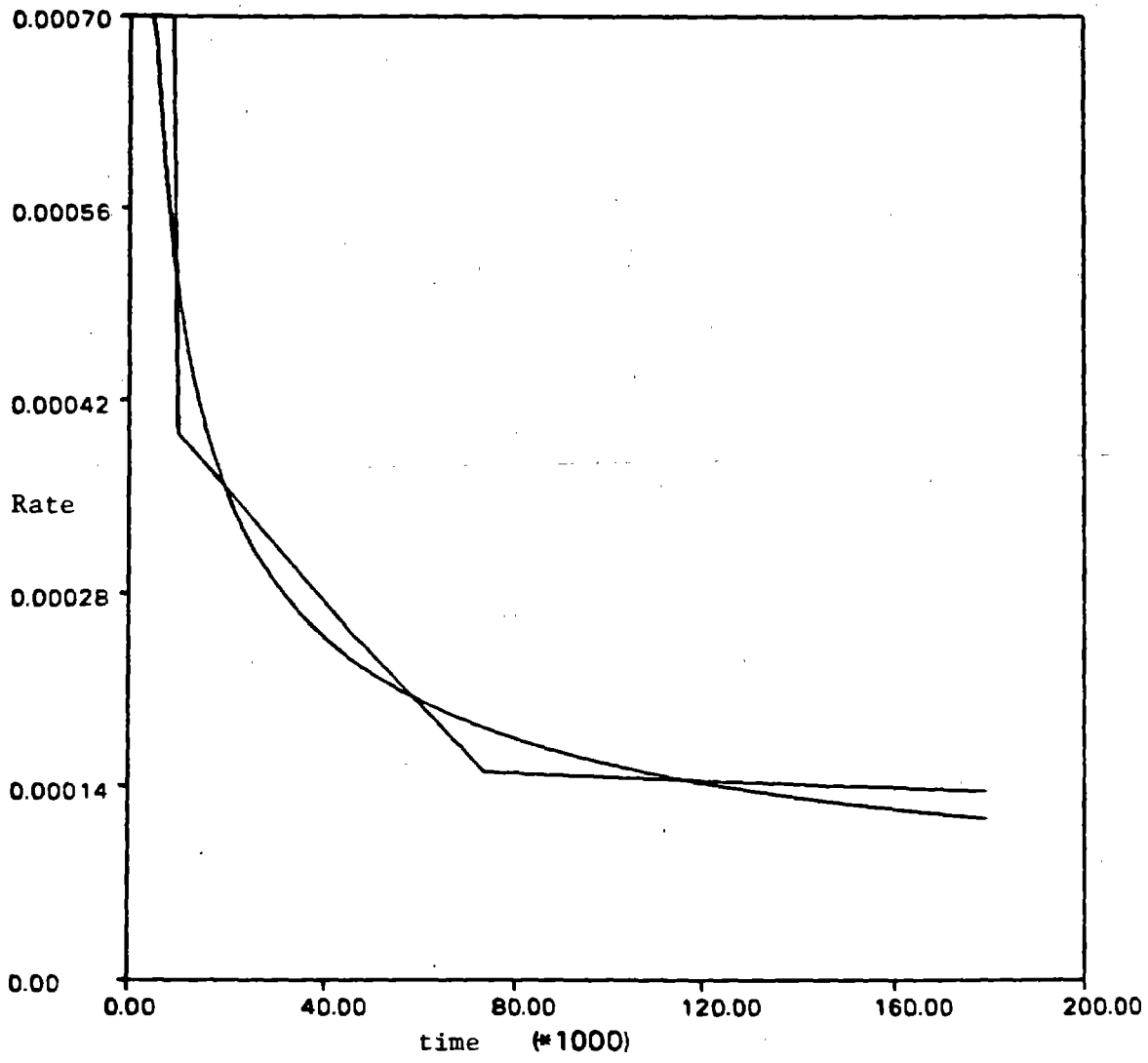

Figure 4. Convex, monotone estimate of failure rate for data in Figure 1 superimposed on true rate $r(t) = 1/\sqrt{400t}$

present time is $t_{50} = 179{,}367$. The true rate is $r(t_{50}) = 1.18 \times 10^{-4}$. The rate estimated from the monotone function (Figure 3) is 0.63×10^{-4}. The rate estimated from the convex function (Figure 4) is 1.38×10^{-4}. These results seem promising.

Note that Problem I above is the well-known "isotone-regression" problem described by Barlow, Bartholomew, Bremner and Brunk [1] and addressed in the reliability growth context by Campbell and Ott [2] and Nagel, Schloz, and Skrivan [14]. If the last interfailure time happens to come from the right tail of the interfailure time distribution, $\hat{r}_n = \hat{r}(t_n^-)$ will underestimate $r(t_n)$ and the monotone constraint on r^* will have no effect, leading to a negative bias ; imposing the additional constraint of convexity does have an effect as can be seen in Figures 2, 3, and 4. In most software reliability applications a positively biased estimate of failure rate is safer than a negatively biased estimate, thus the convexity constraint seems to be desirable and the generalization of isotone regression to completely monotone regression worth pursuing.

Problem Statement - Second Formulation

The above formulation only considered first and second difference constraints. It is more straightforward to deal with higher differences if we formulate the problem in terms of equally-spaced discrete time-points. Let $[0, t_n]$ be divided into k intervals of equal length, $\Delta s = t_n/k$; and define k time points $s_i = i\Delta s$, $i=1,2,\ldots,k$. A smoothed version of the raw failure rate $\hat{r}(\cdot)$ is

$$\tilde{r}(t) = \int_{s_{i-1}}^{s_i} \hat{r}(s)\, ds/\Delta s \ , \quad s_{i-1} \leq t < s_i \ . \tag{7}$$

Let $\tilde{r}_i = \tilde{r}(s_i^-)$, $i=1,2,\ldots, k$. We use $\{\tilde{r}_i$, $i=1,\ldots,k\}$ as the data

point in the least-squares regression problem. The optimal solution is denoted as $\{r_i^*, i=1,2,\ldots,k\}$ as before; we constrain this solution to satisfy

$$(-1)^m \Delta^m r_i^* \geq 0 , \quad m-1 \leq i \leq k , \; 0 \leq m \leq d , \tag{8}$$

where d corresponds to the maximum difference considered. The least squares distance is

$$D(\tilde{r}, r^*) = \sum_{i=1}^{k} (\tilde{r}_i - r_i^*)^2 . \tag{9}$$

We have a family of quadratic programming problems (for $d = 1,2,3,\ldots$): Minimize $D(\tilde{r}, r^*)$ subject to constraints in (8).

We note that there are many possible formulation of the problem of finding the "best" completely monotone rate function. The two we have given seem to be closest to the standard formulations of isotone regression problems and therefore are logical initial approaches to this general problem. We plan to consider several other possible formulations in the future.

A Monte Carlo Study of Performance

We have chosen to do a more thorough investigation of the above second formulation. We wish to determine how accurately the present program failure rate (i.e. the failure rate at the end of the observed data) can be estimated by using the value of a least-squares completely monotone (up to d differences or derivatives) regression curve at that point. This is a complex inference procedure which is not amenable to analytic evaluation, therefore we use Monte Carlo simulation to estimate the average relative error and standard deviation of the estimator.

For our initial set of simulated failure data we chose the Musa-Okumoto [13] Logarithmic Nonhomogeneous Poisson process. This model seems to describe software failures well and for different parameter values it covers a range from no growth to extreme growth.

The mean function $M(\cdot)$ and the parameter values considered are shown in Table I and the corresponding mean functions are graphed in Figure 5; the values of β were chosen so that the ith curve goes through $(50-5i,\ 20+2i)$, $i=0,1,\ldots,9$. The amount of growth can be seen from the curves or by comparing $M(50)$ and $M(100) - M(50)$ in Table I. Each data point generated consists of 40 observations from one of these M-O processes.

The results of the Monte Carlo study appear in Table II. We considered 7 different models: β_0 through β_6 . For each of the first five we generated 1000 data points, i.e., sample paths of 40 observations each; for the last two we generated 400 data points. For each data point we discretized the time interval into 40 equal subintervals and found the least-squares regression lines whose first d differences satisfied the completely monotone property, $d = 1, 2, 3, 4, 5,$ and 6. We then computed the relative error

$$e_d = (r^*_{40} - r(t_{40}))/r(t_{40}) \tag{10}$$

for each of the 6 values of d . We then computed the average and standard deviation of the relative errors over all 1000 (or 400) replicates for each value of d and β . For example, in Table II, for β_0 the monotone $(d=1)$ least-squares estimate underestimated the true rate by 26.7 percent on the average and the relative error had a standard deviation of 24.0 percent.

The numbers in Table II merit some discussion. The first column $(d=1)$ corresponds to the traditional isotone regression. We see a significant negative bias in the cases of small and moderate reliability growth. (The positive bias for β_5 and β_6 may reflect an inappropriateness of the mean as a measure of location in these cases more than anything else.) Note also that the variability, i.e., standard deviation, of the estimator goes from an acceptable to an unacceptable level as the growth becomes more extreme. Using higher differences

Table I

Cases of Musa-Okumoto Logarithmic Poisson Model

$$M(t) = 40 \frac{\log (\beta t + 1)}{\log (100\beta + 1)}$$

β	M(50)	M(100) - M(50)
$\beta_0 = .100 \times 10^{-4}$	20.00	20.00
$\beta_1 = .124 \times 10^{-1}$	23.93	16.07
$\beta_2 = .429 \times 10^{-1}$	27.51	12.49
$\beta_3 = .131$	30.56	9.44
$\beta_4 = .461$	33.02	6.98
$\beta_5 = .243 \times 10$	34.99	5.01
$\beta_6 = .311 \times 10^{2}$	36.55	3.45
$\beta_7 = .311 \times 10^{4}$	37.81	2.19
$\beta_8 = .100 \times 10^{9}$	38.80	1.20
$\beta_9 = .104 \times 10^{25}$	39.54	.46

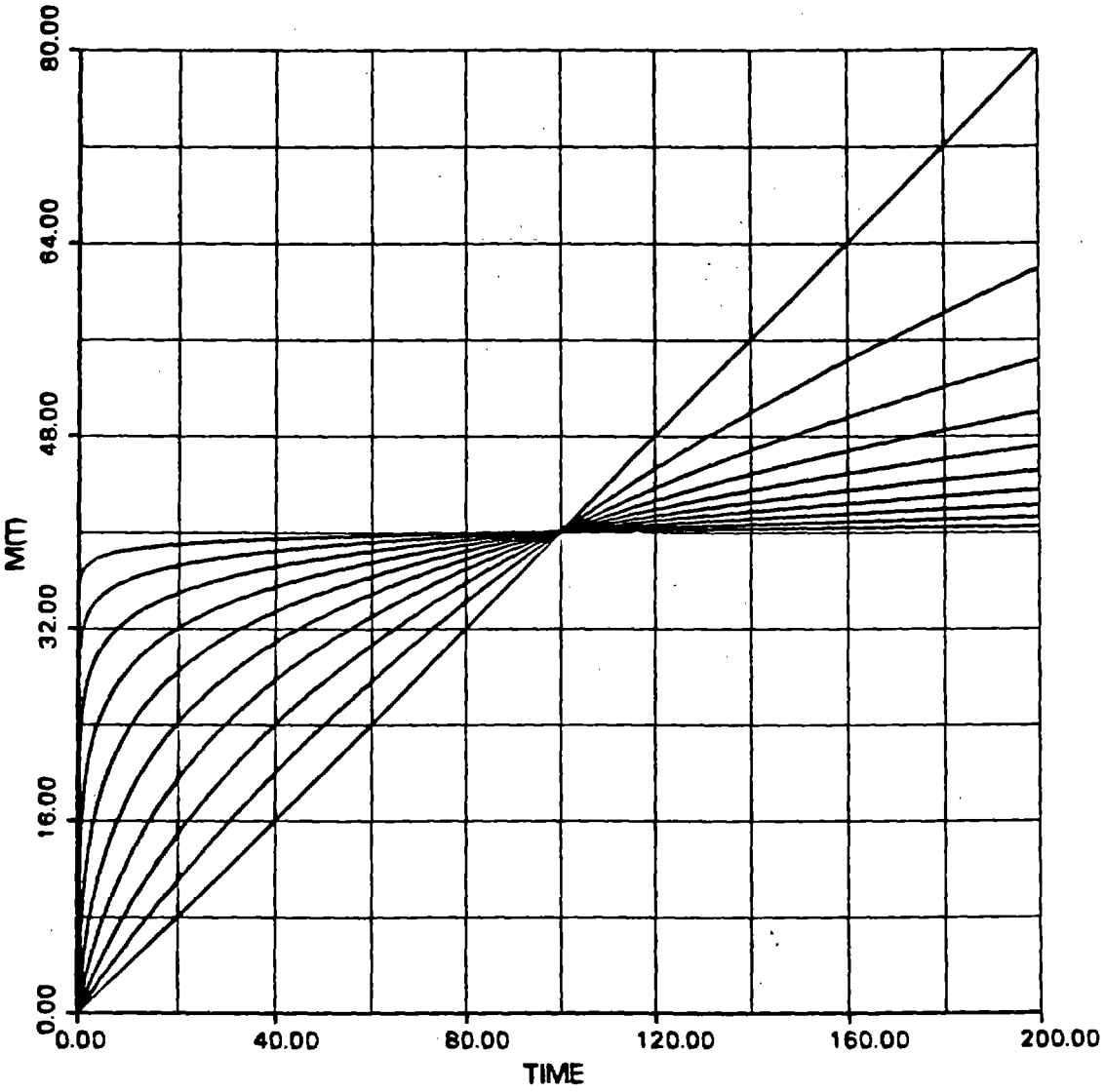

Figure 5. Mean functions for Musa-Okumoto Logarithmic Poisson process. Cases 0 through 9 of Table I.

Table II

Performance of completely monotone least squares estimators of failure rate at 40th failure of M-0 model: Average relative error and (in parentheses) standard deviation of relative error

β	d=1	d=2	d=3	d=4	d=5	d=6
β_0	-.267 (.240)	-.051 (.199)	-.054 (.205)	-.055 (.206)	-.055 (.206)	-.065 (.203)
β_1	-.184 (.307)	.093 (.290)	.059 (.326)	.055 (.328)	.054 (.328)	.061 (.315)
β_2	-.126 (.366)	.149 (.395)	.082 (.442)	.071 (.441)	.069 (.439)	.076 (.434)
β_3	-.067 (.397)	.186 (.462)	.106 (.518)	.092 (.517)	.086 (.514)	.089 (.509)
β_4	-.008 (.439)	.227 (.519)	.150 (.577)	.133 (.579)	.129 (.575)	.131 (.570)
β_5	.071 (.476)	.277 (.578)	.209 (.633)	.187 (.639)	.180 (.640)	.179 (.632)
β_6	.141 (.531)	.347 (.654)	.258 (.723)	.233 (.730)	.222 (.728)	.219 (.723)

The average and standard deviations are estimated from 1000 independent replicates for β_0, β_1, β_2, β_3 and β_4 and from 400 independent replicates for β_5 and β_6

(d > 1) we see an improvement in the bias and a small increase in the variability for small and moderate growth situations. A positive bias is conservative when estimating the failure rate and therefore is preferred to a negative bias. Thus from Table II it seems to follow that estimators based on higher differences are preferred to the isotone estimators for small and moderate growth. For cases of more extreme growth it simply may be impossible to get good nonparametric estimates.

In order to get some feeling for the significance of the variability of the above estimators we consider the problem of estimating the failure rate in a known time-homogeneous environment, i.e. no growth. Let $X_i, \ldots X_n$ be i.i.d. Exponential random variables with mean 1, then

$$\hat{\lambda} = n / \sum_{i=1}^{n} X_i$$

is an estimator of the failure rate, and because $\lambda = 1$, $\hat{\lambda} - 1$ is an estimator of the relative error. It can be shown that

$$E(\hat{\lambda}) = \frac{n}{n-1}$$

$$E(\hat{\lambda}^2) = \frac{n^2}{(n-1)(n-2)}$$

$$Var(\hat{\lambda}) = \frac{n^2}{(n-1)^2(n-2)}$$

For $n=40$, we get $Var(\hat{\lambda}) = .0277$ and the standard deviation equals .166. Comparing this to the standard deviations in Table II for β_0 we see a surprisingly small difference; so we do not seem to lose much precision by using the nonparametric approach which makes no assumptions about time homogeneity.

Finally, we note that the mean and standard deviation of the estimators do not give a complete picture of the performance. The complete distribution of relative errors is more revealing. Figure 6 shows the empirical distributions of the 1000 relative errors observed

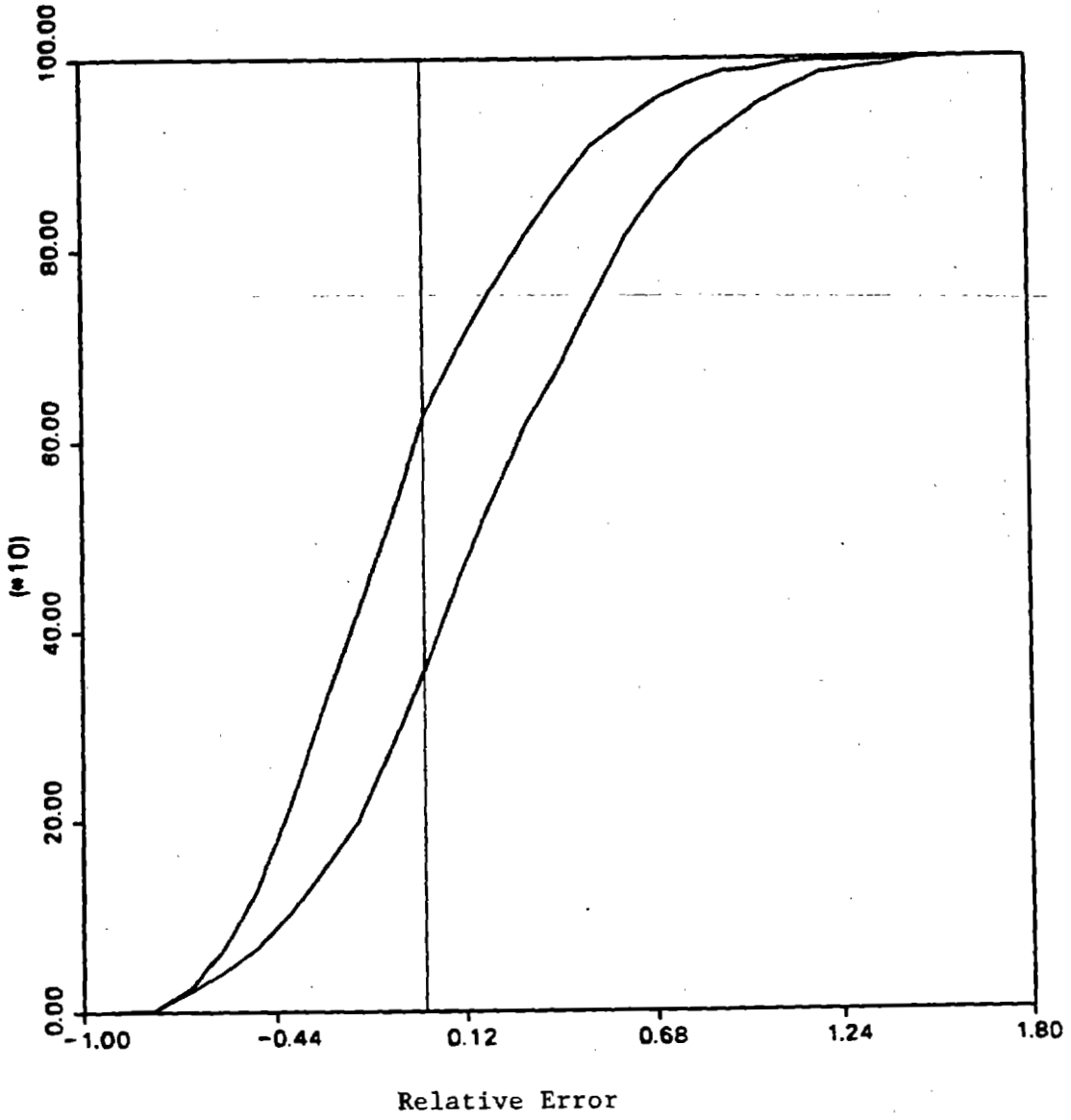

Figure 6. Cumulative distributions of relative errors for estimates of failure rate at 40th failure of M-O model with $\beta = \beta_3 = .131$ based on d=1 and d=2. (The distribution for d=1 is to the left.)

for model β_3 using d=1 and d=2 . From these plots we are tempted to prefer the convex (d=2) estimator to the monotone (d=1) one even though the absolute average error is three times larger. One reason for this choice is that the monotone estimator underestimates 62.6 percent of the time while the convex estimator underestimates 35.9 percent of the time.

Conclusions and Future Work

We have presented a new nonparametric method for estimating the current failure rate of a program, i.e. the failure rate at the end of a sequence of observed failures. A limited Monte Carlo study shows that the method works well for data sets with small or moderate growth but not for data sets with extreme growth.

This initial study shows that the method has potential. We are pursuing several paths which should improve the method and allow it to achieve its full potential. We are looking at formulations in terms of the mean function instead of the failure rate function. Constraining the solution to be completely monotonic into the future may improve the estimate of present failure rate; this extension may lead to predictions of future reliability growth. Finally, it may be possible to develop a more efficient and more stable numerical algorithm.

APPENDIX

The Optimization Algorithm

The optimization problems of the two formulations presented are both linearly constrained quadratic programming problems (in the variables r_i) . A computer program was written for solving these problems, using a Newton type variable reduction algorithm . (Such algorithms are described in detail in McCormick [10] and in Gill et al. [6]. The program uses a Cholesky decomposition (see Gill et al [6]) to factorize the projected Hessian matrix. It should be noted that the system of equations (8) is equivalent to the reduced system of equations

$$\begin{aligned} (-1)^d \Delta^d r_i^* &\geq 0 \qquad d+1 \leq i \leq k \\ (-1)^i \Delta^i r_k^* &\geq 0 \qquad 0 \leq i \leq d-1 \end{aligned} \tag{11}$$

This reduction to a system of non-redundant equations (prior to solution), is necessary, since all algorithms for constrained optimization assume linear independence of the constraint gradients.

For smaller values of d , the optimization program proved quite efficient. As examples, the running time for 900 problems of 40 variables each, with $\beta = 0.1$ and $d=3$ was under 3 CPU minutes on an IBM 4341, while 600 problems of 40 variables with $\beta = 0.00001$ and values of d varying from 1 to 6 took just under 1 CPU minute. However, for larger values of the maximal order of constraint differences d , numerical problems arise. As the value of d increases, the constraint coefficient matrix becomes increasingly ill conditioned. As a result, a solution which satisfies the system of equations (11) up to an acceptable tolerance (say, 10^{-7}) does not necessarily satisfy system (8) to an acceptable tolerance, and hence is unacceptable. Another problem encountered for larger d is that the program fails to complete the Cholesky decomposition of the projected Hessian, even though this matrix is known to be positive definite.

Various steps can yet be taken to improve the efficiency of the program. Various measures to restore feasibility have been introduced, and these could still be improved. In addition, an orthogonal factorization, rather than a Cholesky decomposition could be used. This would tend to increase the numerical stability at the expense of increased running time.

REFERENCES

[1] BARLOW, R. E., D. J. BARTHOLOMEW, J. M. BREMNER and H. D. BRUNK (1972). Statistical Inference Under Order Restrictions, Wiley, New York.

[2] CAMPBELL, G. and K. O. OTT (1979). Statistical evaluation of major human errors during the development of new technological system. Nuclear Science and Engineering, 71, 267-279.

[3] CROW, L. H. (1974). Reliability analyses for complex, reparable systems. Reliability and Biometry (F. Proshan and R. J. Serfling, eds.) SIAM, Philadelphia, 379-410.

[4] DUANE, J. T. (1964). Learning curve approach to reliability monitoring. IEEE Trans. Aerospace 2, 563-566.

[5] FELLER, W. (1971). An introduction to probability theory and its applications. Vol. II, 2nd ed. Wiley, New York.

[6] GILL, P. E., W. MURRAY, and M. H. WRIGHT (1981). Practical Optimization, Academic Press, New York.

[7] GOEL, A. K. and K. OKUMOTO (1979). Time-dependent error detection rate model for software reliability and other performance measures. IEEE Trans. Rel. R-28, 206-211.

[8] JELINSKY, Z. and P. MORANDA (1972). Software reliability research. Statistical Computer Performance Evaluation, (W. Ferberger, ed.) Academic Press, New York, pp. 465-484.

[9] LITTLEWOOD, B. (1981). Software reliability growth: a model for fault-removal in computer-programs and hardware-design. IEEE Trans. Rel. R-30, 313-320.

[10] MCCORMICK, G. P. (1983). Nonlinear programming: theory, algorithms and applications. Wiley, New York.

[11] MILLER, D. R. (1984). Exponential order statistics models of software reliability growth. Technical Paper Serial T-496, Institute for Management Science and Engineering, George Washington University.

[12] MILLER, D. R. and N. D. SINGPURWALLA (1980). Failure rate estimation using random smoothing. Sankhya, 42, 217-228.

[13] MUSA, J. D. and K. OKUMOTO (1984). A logarithmic poisson execution time model for software reliability measurement. Proc. Seventh International Conference on Software Engineering. IEEE, New York, 230-238.

[14] NAGEL, P. M. F. W. SCHOLZ and J. A. SKRIVAN (1984). Software reliability: additional investigations into modeling with replicated experiments. CR-172378, NASA.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GWU/IMSE/Serial T-497/85	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPLETELY MONOTONE REGRESSION ESTIMATES OF SOFTWARE FAILURE RATES		5. TYPE OF REPORT & PERIOD COVERED SCIENTIFIC 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) DOUGLAS R. MILLER ARIELA SOFER		8. CONTRACT OR GRANT NUMBER(s) GRANT NAG-1-179
9. PERFORMING ORGANIZATION NAME AND ADDRESS GEORGE WASHINGTON UNIVERSITY INSTITUTE FOR MANAGEMENT SCIENCE AND ENGINEERING WASHINGTON, DC 20052		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS NATIONAL AERONAUTICS AND SPACE ADMINISTRATION		12. REPORT DATE 18 January 1985 13. NUMBER OF PAGES 21
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC SALE AND RELEASE; DISTRIBUTION UNLIMITED.

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

18. SUPPLEMENTARY NOTES

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

SOFTWARE RELIABILITY
FAILURE RATE ESTIMATION
QUADRATIC PROGRAMMING

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

A new method for estimating the present failure rate of a program is presented. A crude nonparameter estimate of the failure rate function is obtained from past failure times. This estimate is then smoothed by fitting a completely monotonic function, which is the solution of a quadratic programming problem. The value of the smoothed function at present time is used as the estimate of present failure rate. A Monte Carlo study gives an indication of how well this method works.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)