A Bayesian Reliability Growth Model for Computer Software

By B. Littlewood and J. L. Verrall

The City University

Summary

A Bayesian reliability growth model is presented which includes special features designed to reproduce special properties of the growth in reliability of an item of computer software (program). The model treats the situation where the program is sufficiently complete to work for continuous time periods between failures, and gives a repair rule for the action of the programmer at such failures. Analysis is based entirely upon the length of the periods of working between repairs and failures, and does not attempt to take account of the internal structure of the program. Methods of inference about the parameters of the model are discussed.

Keywords: RELIABILITY GROWTH; COMPUTER SOFTWARE; RELIABILITY; BAYESIAN RELIABILITY

1. Introduction

During the past few years a vast literature has grown up concerning the reliability of hardware systems. Much of this research has been centred upon systems of electrical and electronic components, such as those comprising the hardware of computing systems, but as far as we know there has been comparatively little effort devoted to the *software* reliability of such systems. To anyone with experience of complex hardware/software systems (e.g. multi-processing computers, process control systems, etc.) it will be apparent that sooner or later someone has got to grasp the nettle of software reliability: this paper is a tentative suggestion for a mathematical model in a rather restricted framework. We would not have the temerity to suggest that all the hardware reliability problems have been solved; clearly, research is going to continue in this field for an indefinite time. What we do believe, however, is that knowledge of the hardware reliability of the aforementioned mixed systems has far outstripped understanding of the software aspects of their reliability.

The research which is reported in the remainder of this paper deals with a model for software reliability *alone:* no attempt is made to model the very important case of the interaction of hardware and software failures. Because of the lack of natural degradation in software, we proceed by constructing a reliability growth model—we hope that the special features of our model are sufficient to provide a first approximation to the more important aspects which are peculiar to the reliability of software. Although these special features of our model are designed to represent software reliability, it may be that workers in some branches of hardware reliability will find our results of value (we have in mind here certain types of burn-in testing).

2. The Reliability Growth Model

Our interest in the problem of reliability of software has been concerned with computer systems in the large power stations of the CEGB. The "data" here are a stream of information (generally cyclically scanned) from something of the order of 10,000 sensing devices, usually via analog/digital converters (ADCs). These input streams represent temperatures, pressures, flow rates in turbines, core temperatures in the nuclear installations, and so on. Typical functions of the computer in such installations are: the display of summaries of the multidimensional input stream in such forms as the operator of the generating unit can easily assimilate (generally on visual display units); providing general logging services; alarm analysis; semi-automatic run-up of turbines, etc. In addition, the computer is often expected to provide data on station efficiency, fuel consumption, etc., and sometimes to provide off-line general programming facilities. The software to support such a complex service is necessarily fairly extensive (in one typical case requiring 17 man-years of programmer effort), and hence will virtually never be error-free.

We have so far used terms such as "reliability" and "error-free" in a way which suggests that their meaning is self-evident: either intuitively, or by analogy with reliability concepts already developed and rigorously defined for use with hardware systems. This is unfortunately not the case: in particular it seems impossible to *define* "error" in an unambiguous way, and here we encounter the first problem in modelling software reliability. It may be, in fact, that situations arise which the original program specification did not envisage: we can therefore think of the model as improving, not merely the program. Because of this and other difficulties we have abandoned any attempt at an analysis in terms of the number of *errors in* the program, and instead have concentrated on treating *failures of* the program. There are some difficulties even in this approach, but generally it is obvious when a program has failed to function satisfactorily—in fact, in many cases complete stoppage occurs. Some cases of controversy might arise (e.g. minor failures going undetected for a time), but we hope that by adopting a subjectivist approach we can overcome these: essentially by *defining* a failure to be what the programmer (or operator) *says* is a failure. This judgment, naturally, will be based upon objective criteria obtained from sources such as design specifications.

By using failures rather than errors in our analysis we do not mean to imply that the concept of an error is entirely without value. On the contrary, when a failure has occurred in a program it will usually be the programmer's job to inspect the program, locate the "errors" and carry out such action as will improve the reliability of the program. But this reliability will be defined in terms of failures of the system: specifically in terms of the time-to-next-failure distributions (or parameters of such distributions). We shall not enquire too closely of the programmer/operator what methods he uses to locate "errors" and eliminate them—we shall merely require that he behaves in such a way as *to try to improve the reliability of the program* whenever a failure occurs. It is in this last feature that we believe our model differs most significantly from existing reliability growth models.

Obviously the internal structure of the program will affect its reliability. The extent of our knowledge of such structure will vary widely in practice, but in general this knowledge should be taken account of in a reliability model. Usually we shall at least know the logical arrangement of certain indivisible blocks (subroutines), perhaps by way of a flow chart or similar logical breakdown. It is common to write and *test* such blocks individually, so a reliability model for such an indivisible block is a prerequisite of a general software reliability model. In this paper we deal only with a model for the reliability of a block like these, leaving the problem of combining knowledge of the logical interconnections and the reliabilities of the different blocks until a later date (it is perhaps worth mentioning that this problem seems difficult: an answer will depend upon the interconnections which transfer control among blocks, and the traversing of such interconnections will probably be data-dependent in many instances). In spite of this restriction to a "black-box" approach, we are confident that the special features of our model make it suitable for representing, at least to a first approximation, the growth in reliability at a certain stage in the creation of a program. In all of what follows we shall refer to "the program": this may be one of the blocks mentioned already, or it may be the complete structure of such blocks. We shall treat it as though it had no known structure—but we must bear in mind that a better model will be obtained when it is learned how to incorporate any structural information we have.

Our next restriction, this time less important, is to *continuous* time processes. We shall treat a program as a black-box system with failures separated by time intervals which can be regarded as continuous random variables. Continuity of the time variable was a natural choice for the real-time systems in power stations which prompted our investigations, but discrete-time systems (e.g. batch-processing) have reliability problems and we hope to look at these in later work. It could be argued, in fact, that all computing systems work in discrete time: letting the processor cycle time be the basic unit of time, or, in our case, the cycle time of the ADC scanners. However, the basic unit of time in these cases is so small compared with typical times between failures that continuous time methods seem more appropriate.

A typical history of our program is given in Fig. 1. We assume that at time zero the program is run on the computer and works satisfactorily until time t_1, when the

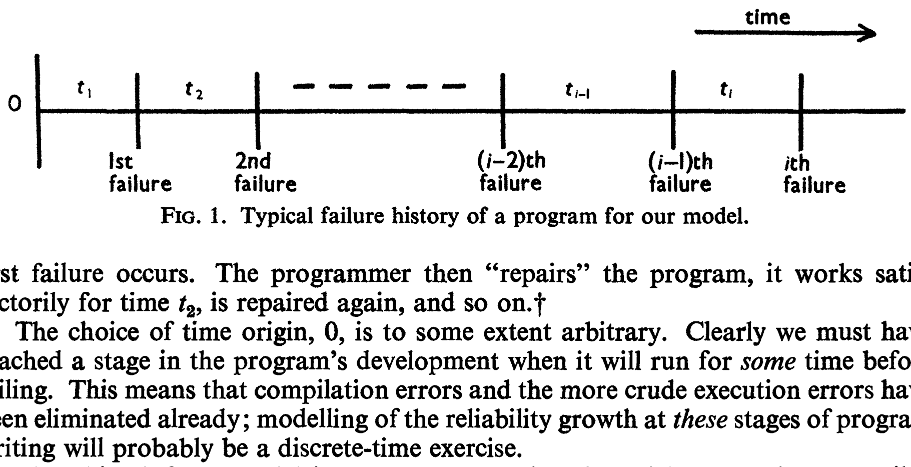

FIG. 1. Typical failure history of a program for our model.

first failure occurs. The programmer then "repairs" the program, it works satisfactorily for time t_2, is repaired again, and so on.†

The choice of time origin, 0, is to some extent arbitrary. Clearly we must have reached a stage in the program's development when it will run for *some* time before failing. This means that compilation errors and the more crude execution errors have been eliminated already; modelling of the reliability growth at *these* stages of program writing will probably be a discrete-time exercise.

The object of the model is to create a repair rule which as nearly as possible reproduces the effect of the programmer's action upon the program. We proceed as follows:

† We shall, as far as possible, use the convention of denoting random variables by capital letters, and the realizations of random variables by lower case letters: thus t_i is a particular observation of the random variable T_i.

Let the random variable T_i, the running time between repair of $(i-1)$th failure and occurrence of ith failure, have a probability density function (p.d.f) denoted by $f\{t, \lambda(i)\}$. We shall assume that the parameter λ is some *failure rate* measure (by this we mean that a program with small λ is better than a program with large λ). The intention of the programmer when he carries out a repair is to make the program better than it was before the failure occurred (i.e. more reliable, or less likely to fail in a given time period). In other words the programmer *intends* by his repair action to diminish λ, making $\lambda(i) < \lambda(i-1)$ for all i. However, he cannot be sure that a particular repair has achieved this intention—in fact programmers recount tales of software which has been made catastrophically *worse* by a "repair". It is here that software reliability differs from most burn-in testing of hardware. In burn-in testing it is usually possible to *know* the characteristics of a replacement component; our programmer, however, usually does not know very much about the characteristics of a rewritten portion of program, nor how it will react with the surrounding software. It is quite possible that he will introduce a *new* source of failures in his repair attempt. We resort to the Bayesian technique of allowing the failure-rate parameter, $\lambda(i)$, to have a distribution. Let the p.d.f. of $\lambda(i)$ be denoted by $g(l, i, \boldsymbol{\alpha})$ where $\boldsymbol{\alpha}$ is a parameter or vector of parameters. We can now easily formulate a mathematical condition to represent the programmer's intention concerning the repair rule. Instead of being *certain* that the failure rate has diminished after his repair action, we argue that it is *probable* that this has happened, i.e.

$$P\{\lambda(i) < l\} \geqslant P\{\lambda(i-1) < l\}, \quad \text{for all } l, i. \tag{1}$$

Letting the distribution function of the parameter $\lambda(i)$ be denoted by

$$\begin{aligned} G(l, i, \boldsymbol{\alpha}) &= P\{\lambda(i) < l\} \\ &= \int_{-\infty}^{l} g(x, i, \boldsymbol{\alpha})\, dx \end{aligned}$$

we have from (1)

$$G(l, i-1, \boldsymbol{\alpha}) \leqslant G(l, i, \boldsymbol{\alpha}). \tag{2}$$

Since λ is a failure rate, $g(l, i, \boldsymbol{\alpha}) = 0$ for $-\infty < l < 0$.

In the model we have assumed that the repair times are zero, for simplicity. It is a fairly easy matter to adjust the results which follow to take into account *constant* repair time, but this does not seem very realistic. We hope to extend the model at a later date by incorporating realistic random variables for the repair times; however, there seems to be no consensus of opinion on how the distributions of these random variables should vary (if at all) with i. Do programmers spend *more* effort in eliminating the later errors, when failures are very infrequent and they have more time at their disposal? Or do they spend *less* effort, on the grounds that infrequent failures are sufficiently tolerable not to warrant expenditure of costly programmer time?

As our model stands, we have the problems of choosing, firstly, a suitable parametric family for the failure times and, secondly, a suitable parametric family with monotonically ordered distribution functions for the failure rates. Neither of these problems presents a great deal of difficulty and we shall give suitable families later.

Let us now assume that we are in the position of having collected data $t_1, t_2, \ldots, t_n$ representing n failures of a program. Our programmer will carry out a repair of the nth failure, using the same criteria as before. We wish to make inference about the distribution of the time to the $(n+1)$th failure, based on our past evidence.

Inference initially centres upon $\boldsymbol{\alpha}$. Let $p_0(\boldsymbol{\alpha})$ be the prior distribution for $\boldsymbol{\alpha}$, and $p_1(\boldsymbol{\alpha})$ be the posterior distribution for $\boldsymbol{\alpha}$. Then, by Bayes's theorem,

$$\begin{aligned} p_1(\boldsymbol{\alpha}) &= p_1(\boldsymbol{\alpha}\,|\,\text{data}) \\ &\propto p(\text{data}\,|\,\boldsymbol{\alpha}).p_0(\boldsymbol{\alpha}), \end{aligned}$$

where by $p(.)$ we shall mean a joint p.d.f. or likelihood. Now

$$\begin{aligned} p(\text{data}\,|\,\boldsymbol{\alpha}) &= p(T_1 = t_1, T_2 = t_2, \ldots, T_n = t_n\,|\,\boldsymbol{\alpha}) \\ &= \int \ldots \int p\{T_1 = t_1, \ldots, T_n = t_n\,|\,\lambda(1) = l_1, \ldots, \lambda(n) = l_n, \boldsymbol{\alpha}\} \\ &\qquad p\{\lambda(1) = l_1, \ldots, \lambda(n) = l_n\,|\,\boldsymbol{\alpha}\}\,dl_1 \ldots dl_n \\ &= \int \ldots \int \prod_{i=1}^{n} p\{T_i = t_i\,|\,\lambda(i) = l_i\} p\{\lambda(i) = l_i\,|\,\boldsymbol{\alpha}\}\,dl_i \\ &= \prod_{i=1}^{n} \int p\{T_i = t_i\,|\,\lambda(i) = l_i\} p\{\lambda(i) = l_i\,|\,\boldsymbol{\alpha}\}\,dl_i \\ &= \prod_{i=1}^{n} \int f(t_i, l)\, g(l, i, \boldsymbol{\alpha})\,dl. \end{aligned} \tag{3}$$

Hence

$$p_1(\boldsymbol{\alpha}) \propto \left\{\prod_{i=1}^{n} \int f(t_i, l)\, g(l, i, \boldsymbol{\alpha})\,dl\right\}.\,p_0(\boldsymbol{\alpha}). \tag{4}$$

Having obtained $p_1(\boldsymbol{\alpha})$, we could proceed by using

$$\int g(l, n+1, \boldsymbol{\alpha})\, p_1(\boldsymbol{\alpha})\,d\boldsymbol{\alpha} \tag{5}$$

as p.d.f. of $\lambda(n+1)$, if interest centred upon failure rate itself. Probably of more interest would be the p.d.f. of T_{n+1}

$$\begin{aligned} &= \int f(t_{n+1}, l)\, p\{\lambda(n+1) = l\}\,dl \\ &= \int f(t_{n+1}, l) \left\{\int g(l, n+1, \boldsymbol{\alpha})\, p_1(\boldsymbol{\alpha})\,d\boldsymbol{\alpha}\right\} dl. \end{aligned} \tag{6}$$

Although these integrals seem analytically rather daunting for general $f(.)$ and $g(.)$ functions, we show that for our choice of parametric families it is possible to make considerable analytical headway. When the $f(.)$ and $g(.)$ families are so chosen that analytic answers are not available, it should be possible to use numerical integration techniques, but this would make the model less easily usable.

3. The Model with Suggested Choice of Distributional Families

The choice of parametric families satisfying our requirements which combines realism with mathematical tractability is as follows:

$$\left.\begin{aligned} f(t,\lambda) &= \lambda\exp(-\lambda t), \quad t>0, \\ &= 0, \quad t<0, \end{aligned}\right\}, \quad \lambda>0, \tag{7}$$

$$\left.\begin{aligned} g(l,i,\alpha) &= \frac{\psi(i)\{\psi(i)\,l\}^{\alpha-1}\exp\{-\psi(i)\,l\}}{\Gamma(\alpha)}, \quad l>0, \\ &= 0, \quad l<0, \end{aligned}\right\} \tag{8}$$

where $\psi(i)$ is a monotonically increasing function of i.

The exponential distribution of (7) for time between failures is a natural choice. If we regard a failure as being a data/software interaction (a program may work perfectly with data from a subset of the sample space, but fail when a data point comes from a particular region of the sample space) it is natural to consider failures as constituting a random process (Poisson process), since the data stream is itself most simply regarded as a random process. This distribution already has an honourable history of use in hardware reliability, so its familiarity is an extra justification for its choice.

The choice of a family of Gamma distributions for the failure rates is largely justifiable by its flexibility (having two parameters), correct range $(0,\infty)$ and mathematical tractibility. Of the two parameters in this family, $\psi(i)$ and α, we have chosen to concentrate our reliability growth in the former. Since $\psi(i)$ is a scaling factor, it is easy to see that a monotonically increasing $\psi(i)$ guarantees the ordering of the distribution functions in i (see Section 2, (2)). The programmer's *intention*, but not certainly, of improving the program is thus represented by this function $\psi(i)$. We shall initially assume $\psi(i)$ to be *completely specified*, but will consider methods of estimation later in the paper.

Assumption of a fixed $\psi(i)$ in this way represents an assumption of a kind of time stationarity in the behaviour of the programmer. His behaviour will vary as the program develops, i.e. as the failures occur and i increases, and it may even differ from program to program (for a more complex program it would appear reasonable to give a programmer a more slowly increasing $\psi(i)$, for example). But we assert that given a program, a programmer and a stage in the development of the program, the action of the programmer upon the program at this stage is fixed. Saying that $\psi(i)$ is fixed for a given program/programmer configuration and given i does not, of course, tell us at what value it is fixed. This problem of estimating $\psi(i)$ does not seem easy. As we have mentioned, some estimation methods are considered later.

Proceeding by substitution of these $f(.)$, $g(.)$ functions into the results of Section 2 we have

$$\begin{aligned} \int f(t_i,l)\,g(l,i,\alpha)\,dl &= \int_0^\infty \frac{l\exp(-lt_i)\,\psi(i)\{\psi(i)\,l\}^{\alpha-1}\exp\{-\psi(i)\,l\}}{\Gamma(\alpha)}\,dl \\ &= \alpha\left(\frac{\psi(i)}{t_i+\psi(i)}\right)^{\alpha}\cdot\frac{1}{t_i+\psi(i)} \\ &\propto \alpha k_i^{\alpha}, \end{aligned}$$

where $k_i = \{\psi(i)\}/\{\psi(i)+t_i\}$. Hence the posterior distribution of α is

$$p_1(\alpha) \propto \alpha^n \left(\prod_{i=1}^{n} k_i\right)^{\alpha} \cdot p_0(\alpha). \tag{9}$$

Assuming a *uniform prior distribution* we have

$$p_1(\alpha) \propto \alpha^n \left(\prod_{i=1}^{n} k_i\right)^{\alpha}.$$

This is of *Gamma form*, hence

$$p_1(\alpha) = \frac{\gamma(\gamma\alpha)^{(n+1)-1}\exp(-\gamma\alpha)}{\Gamma(n+1)} \tag{10}$$

where

$$\gamma = \log \prod_{i=1}^{n}\left(\frac{t_i+\psi(i)}{\psi(i)}\right)$$

$$= \sum_{i=1}^{n} \log\left(\frac{t_i+\psi(i)}{\psi(i)}\right).$$

The distribution of failure rate given by (5) does not seem tractable analytically (although its percentage points could be obtained numerically in any particular case), so we proceed directly to obtain the p.d.f. of T_{n+1}. We require, changing the order of integration in (6):

$$\int f(t_{n+1}, l) g(l, n+1, \alpha)\, \mathrm{d}l$$

$$= \int_0^\infty \frac{l\exp(-lt_{n+1})\,\psi(n+1)\{\psi(n+1)l\}^{\alpha-1}\exp\{-\psi(n+1)l\}}{\Gamma(\alpha)}\,dl$$

$$= \alpha\left(\frac{\psi(n+1)}{\psi(n+1)+t_{n+1}}\right)^{\alpha}\frac{1}{t_{n+1}+\psi(n+1)}. \tag{11}$$

Hence, the p.d.f. of T_{n+1} is

$$\int_0^\infty \frac{\gamma(\gamma\alpha)^{(n+1)-1}\exp(-\gamma\alpha)}{\Gamma(n+1)}\,\alpha\left(\frac{\psi(n+1)}{t_{n+1}+\psi(n+1)}\right)^{\alpha}\frac{1}{t_{n+1}+\psi(n+1)}\,d\alpha$$

which reduces to

$$\frac{(n+1)\gamma^{n+1}}{t_{n+1}+\psi(n+1)}\left[\!\left[\left[\left\{\gamma+\log\left(\frac{t_{n+1}+\psi(n+1)}{\psi(n+1)}\right)\right\}^{n+2}\right]\right]\!\right]^{-1}. \tag{12}$$

We can obtain percentage points of this distribution analytically, since its distribution function is easily obtainable. Let $F(t_{n+1})$ denote this distribution function, i.e.

$$F(t_{n+1}) = P(T_{n+1} < t_{n+1})$$

$$= \int_0^{t_{n+1}} \frac{(n+1)\gamma^{n+1}}{\{t+\psi(n+1)\}}\left[\!\left[\left[\gamma+\log\left\{\frac{t+\psi(n+1)}{\psi(n+1)}\right\}\right]^{n+2}\right]\!\right]^{-1} dt$$

$$= 1-\left[\gamma\Big/\left\{\gamma+\log\left(\frac{t_{n+1}+\psi(n+1)}{\psi(n+1)}\right)\right\}\right]^{n+1}. \tag{13}$$

If we denote by $y_\alpha^{(n+1)}$ an upper $100\alpha\%$ confidence bound of this distribution, i.e.

$$P[T_{n+1} < y_\alpha^{(n+1)}] = \alpha$$

we have

$$y_\alpha^{(n+1)} = \psi(n+1)\left(\prod_{i=1}^{n} k_i\right)^{1-(1/1-\alpha)^{1/(n+1)}} - \psi(n+1). \tag{14}$$

Given a history of the process in the form $t_1, t_2, \ldots, t_n$ and knowledge of $\psi(i)$, this bound is easily calculable.

It should be emphasized that we are here dealing with the distribution of T_{n+1}, the time to $(n+1)$th failure *measured from the instant at which the nth repair is effected.* A more general approach would consider the time-to-next-failure measured from an arbitrary time-point. We propose to obtain this distribution now.

A typical history of such a process is given in Fig. 2. This differs from the previous analysis in the observation t_n, which is the time which has elapsed between occurrence

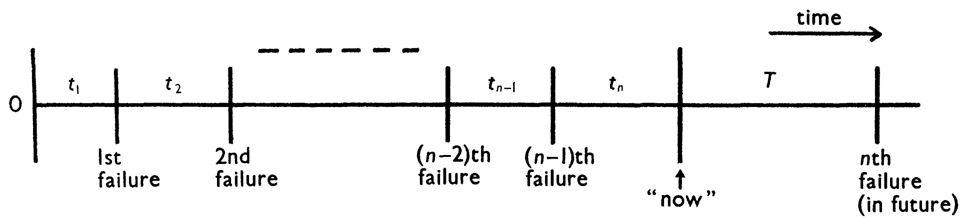

FIG. 2. Model when time-to-next-failure is measured from an arbitrary origin ("now" in Figure).

of the last (i.e. $(n-1)$th) failure and "now", the time origin of our time-to-next-failure random variable, T. Observations $t_1, t_2, \ldots, t_{n-1}$ are times between failures (or, more strictly, between instantaneous repair of one failure and occurrence of the next) as before. Now, in the general model we have

$$p(\text{data}\,|\,\boldsymbol{\alpha}) = \prod_{i=1}^{n-1} \int f(t_i, l)\, g(l, i, \boldsymbol{\alpha})\, dl$$

$$\times \int P\{\text{no failure in } (0, t_n)\,|\,l\}\, g(l, n, \boldsymbol{\alpha})\, dl. \tag{15}$$

Using the same $f(.)$, $g(.)$ as in (7) and (8) we obtain

$$P\{\text{no failure in } (0, t_n)\,|\,l\} = \exp(-lt_n)$$

and so

$$p(\text{data}\,|\,\alpha) = \prod_{i=1}^{n-1} \alpha k_i^\alpha \left[\int \frac{\exp(-lt_n)\,\psi(n)\{\psi(n)l\}^{\alpha-1}\exp(-\psi(n)l)}{\Gamma(\alpha)}\, dl\right] \tag{16}$$

$$\propto \prod_{i=1}^{n-1} \alpha k_i^\alpha \left(\frac{\psi(n)}{t_n + \psi(n)}\right)^\alpha$$

where $k_i = \{\psi(i)\}/\{t_i + \psi(i)\}$ as before

$$= \alpha^{n-1}\left(\prod_{i=1}^{n} k_i\right)^\alpha. \tag{17}$$

Assuming a uniform prior distribution for α we obtain the posterior distribution

$$p_1(\alpha) \propto \alpha^{n-1} k^{\alpha},$$

where

$$k = \prod_{i=1}^{n} k_i.$$

This is again of Gamma form, cf. (10), hence

$$p_1(\alpha) = \frac{\gamma(\gamma\alpha)^{n-1}\exp(-\gamma\alpha)}{\Gamma(n)}, \tag{18}$$

where

$$\gamma = \sum_{i=1}^{n} \log\left\{\frac{\psi(i)+t_i}{\psi(i)}\right\}.$$

To get the p.d.f. of T, the time-to-next-failure (see Fig. 2), we require

$$\int f(t,l)\,g(l,n,\alpha)\,\mathrm{d}l = \alpha\left(\frac{\psi(n)}{\psi(n)+t}\right)^{\alpha}\frac{1}{\psi(n)+t}.$$

Then the p.d.f. of T is

$$\int_0^{\infty}\frac{\gamma(\gamma\alpha)^{n-1}\exp(-\gamma\alpha)}{\Gamma(n)}\alpha\left(\frac{\psi(n)}{\psi(n)+t}\right)^{\alpha}\frac{1}{\psi(n)+t}\,\mathrm{d}\alpha$$

$$= \frac{n\gamma^n}{t+\psi(n)}\left[\left\{\gamma+\log\left(\frac{t+\psi(n)}{\psi(n)}\right)\right\}^{n+1}\right]^{-1}. \tag{19}$$

Again it is easy to obtain the distribution function analytically, and consequently the percentage points of the distribution. In particular an upper 100α% confidence bound is

$$\psi(n)\left(\prod_{i=1}^{n} k_i\right)^{1-(1/1-\alpha)^{1/n}} - \psi(n). \tag{20}$$

Although these distributions and bounds for time-to-next-failure are, we believe, of potential value in practical situations, a measure which produces greater theoretical insight into the model is *instantaneous failure rate estimate* $\lambda(\mathbf{t}) = \lambda(t_1, t_2, \ldots, t_n)$. We define this, for the instant "now" in Fig. 2, as follows:

P(a failure occurs in a time interval of length δt beginning at "now"

$$|\, t_1, t_2, \ldots, t_n) = \lambda(\mathbf{t})\,.\,\delta t. \tag{21}$$

If we denote the p.d.f. of T in (19) by $f(t)$ it follows that

$$\begin{aligned}\lambda(\mathbf{t}) &= f(0)\\ &= \frac{n}{\psi(n)}\left(\log\prod_{i=1}^{n}\frac{\psi(i)+t_i}{\psi(i)}\right)^{-1}.\end{aligned} \tag{22}$$

We must emphasize that $\lambda(\mathbf{t})$ is a function of the entire failure history

$$\mathbf{t} = (t_i, t_2, \ldots, t_n):$$

it is *not* a parameter of an exponential distribution, but an *instantaneous* failure rate estimate. In fact we are interested in the way $\lambda(\mathbf{t})$ changes as $\mathbf{t}$, the history, develops, since decreasing $\lambda(\mathbf{t})$ means increasing reliability and vice versa. Consider first the change in $\lambda(\mathbf{t})$ during one of the periods of working between failures: this means that n remains fixed, as do $t_1, t_2, \ldots, t_{n-1}$, but t_n increases. It follows that $\lambda(\mathbf{t})$ decreases monotonically since the second term in (22) does so. This accords with intuition: we would tend to believe a system more reliable the longer the time of perfect working which has elapsed since the last failure. The other kind of change occurs at a failure point. Clearly $\lambda(\mathbf{t})$ is not defined at such a time instant, but we can obtain values of $\lambda(\mathbf{t})$ at an instant $\mathbf{t} = (t_1, t_2, \ldots, t_n)$ immediately preceding a failure, and at

$$(t_1, t_2, \ldots, t_n, t_{n+1}),$$

where $t_{n+1} = 0$, immediately following the instantaneous repair. The second term in (22) remains constant in this change, and the first term is transformed from $n/\{\psi(n)\}$ into $(n+1)/\{\psi(n+1)\}$. This means that the repair attempt decreases the instantaneous failure rate (improves reliability) if

$$\frac{n+1}{\psi(n+1)} < \frac{n}{\psi(n)},$$

i.e.

$$\frac{\psi(n+1)}{n+1} > \frac{\psi(n)}{n}. \tag{23}$$

This occurs if $\psi(i)$ increases more rapidly with i than a linear function of i. We would suggest that this happens in practice.

The instantaneous failure rate of (22), therefore, partitions into two parts, each of which shows a tendency to decrease: the first only at repairs, and only for suitably rapidly increasing $\psi(.)$ functions; the second only during periods of perfect working between failures. A plot of $\lambda(\mathbf{t})$ against (real) time is shown in Fig. 3 for a typical portion of the history of a program.

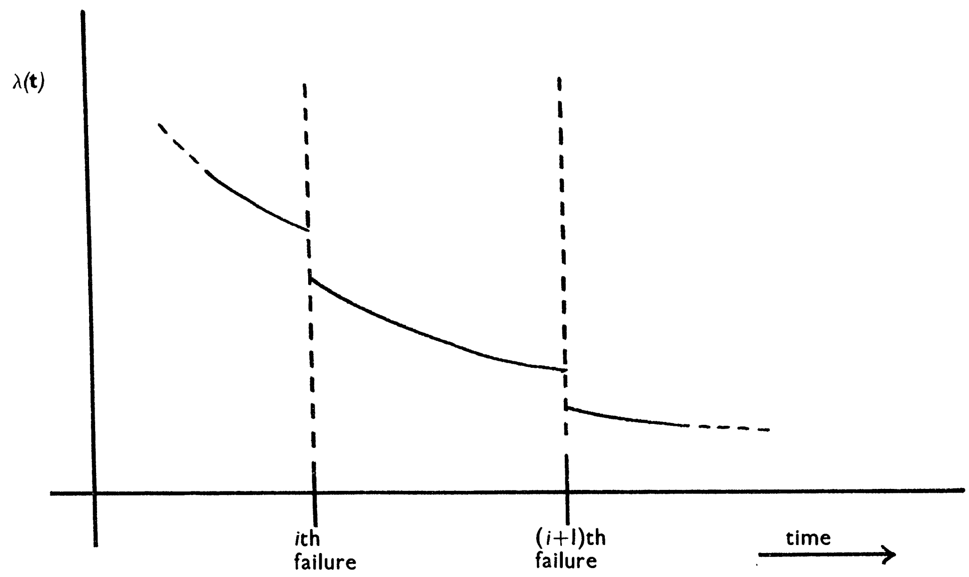

FIG. 3. Portion of a plot of instantaneous failure rate estimate, $\lambda(\mathbf{t})$, against time.

We conclude this section with a note about the growth parameter $\psi(i)$. We have tended to imply that a strict increase must take place in this function at each repair: this may be unrealistic since in some cases which we have encountered the programmer *takes no action at all*, merely restarting the program (implying that the failure is a highly data-dependent one, and that the program will work when restarted at a different point in the data stream). In such cases it would be reasonable to keep the $\psi(.)$ function constant, i.e. $\psi(i) = \psi(i+1)$ if no repair is attempted at the ith failure. As long as we know at which failures the programmer behaves in this way, i.e. $\psi(i)$ is specified for all i, we can use (12), (13), (14), etc. as before. Difficulties arise if we try to project into the future, however, since future $\psi(i)$ will not be defined. In what follows we shall assume that $\psi(i)$ is completely known.

It is also perhaps worth mentioning at this stage that the "repair" at the ith failure is not necessarily *of* the ith failure. Usually a programmer's action will be directed towards eliminating the cause of the failure which has just occurred, but this need not be so. We are thinking here of cases where it is convenient to restart the program immediately a failure has happened, leaving the correction of the softwave until a leisurely perusal has taken place. All that our model demands is that when a programmer intervenes he does so with the intention of improvement, and that such interventions take place only when the program has failed of its own accord. We hope to remove this final condition in later work.

4. Problems Concerning the Growth Function ψ (i)

The model we have presented so far in this paper has assumed knowledge of the function $\psi(i)$ which governs the discrete reliability change at repair attempt i. We had hoped that this function would represent an invariant property of a particular program/programmer configuration—perhaps varying from programmer to programmer and even from program to program (for a fixed programmer), but otherwise constant. This assumption has been questioned, it being suggested that it would be more realistic to regard one of our histories as being composed of succeeding intervals consisting of different *patterns of usage*. It is envisaged that the times at which one pattern of usage succeeds another would be known, hence our model with a constant $\psi(i)$ function would be applicable for the homogeneous behaviour between two such times. We are faced, then, with the problem of how to *choose* a $\psi(i)$ function which is applicable to a particular pattern of usage. Of course, even with our naive assumption of complete invariance of the $\psi(i)$ function we face this problem to some degree: we have to choose $\psi(i)$ at the outset in some way, even though it will be a once-and-for-all exercise. It is worth mentioning, too, that even if the functional form of $\psi(.)$ is known, we still have the problem of what values to begin with, since our choice of time origin, 0, is arbitrary.

4.1. *Full Bayesian Analysis*

If we reject the assumption that $\psi(i)$ is known completely the most obvious mathematically tractable course of action is to make the assumption that $\psi(i)$ is a member of a known parametric family, $\psi(\boldsymbol{\beta}, i)$, with unknown parameter(s) $\boldsymbol{\beta}$.

We proceed as in the previous sections with an expanded parameter space (the β's having the same status as α's previously). Thus, in a notation which is an obvious extension of that used before:

$$\begin{aligned} f(t, \lambda) &= \lambda \exp(-\lambda t), \quad t>0, \\ &= 0, \quad t<0 \end{aligned}$$

and

$$\left.\begin{aligned} g(l,i,\alpha,\boldsymbol{\beta}) &= \frac{\psi(\boldsymbol{\beta},i)\{\psi(\boldsymbol{\beta},i)\,l\}^{\alpha-1}\exp\{-\psi(\boldsymbol{\beta},i)\,l\}}{\Gamma(\alpha)}, \quad l>0 \\ &= 0, \quad l<0 \end{aligned}\right\} \tag{24}$$

Choosing, say, a uniform joint prior distribution for $(\alpha, \boldsymbol{\beta})$ we can find the joint posterior distribution $p_1(\alpha, \boldsymbol{\beta})$. The posterior distribution of $\boldsymbol{\beta}$ is obtained from this by integrating out α. Unfortunately this distribution appears to be completely intractable: the most it seems possible to salvage is a numerical procedure for computing an estimate of $\boldsymbol{\beta}$, say $\hat{\boldsymbol{\beta}}$ (e.g. mean, median, mode of $p_1(\boldsymbol{\beta})$). We could then use $\psi(\hat{\boldsymbol{\beta}}, i)$ and proceed as in previous sections.

The full Bayesian analysis, however, would find the distribution of T_{n+1} (cf. (6)), namely

$$\int f(t,l)\left\{\int\int g(l,n+1,\alpha,\boldsymbol{\beta})\,p_1(\alpha,\boldsymbol{\beta})\,\mathrm{d}\alpha\,\mathrm{d}\boldsymbol{\beta}\right\}\mathrm{d}l. \tag{25}$$

The inner integrals are obviously completely intractable, and the work involved in a numerical approach will be considerably greater than that required to find $\hat{\boldsymbol{\beta}}$ as suggested.

It would seem, therefore, that the most we can salvage from this is a method for computing $\hat{\boldsymbol{\beta}}$, albeit at the price of considerable computing effort. Such effort may be acceptable if we can assume a constant pattern of usage (hence ψ), in which case it will be a once-and-for-all exercise. However, if we are in the situation of requiring constant updating of the $\psi(.)$ family, as in the case of different patterns of usage, such a heavy computing load would seem prohibitive.

4.2. *A Method based on Probability Integral Transforms*

Apart from the computing difficulties of the previous method, it suffers from the (slight) disadvantage of assuming a parametric family: it would not enable us to choose between two $\psi(.)$ functions from different families. The following method overcomes this problem by providing a *computationally simple* non-parametric approach to choosing among a class of given functions. The method relies on the fact, which is rather surprising, that we can obtain at each stage a confidence bound for the next time to failure, conditional upon the previously observed times between failures.

Observe times between failures $t_1, t_2, \ldots, t_n$, which are realizations of the random variables $T_1, T_2, \ldots, T_n$. At the ith stage of this process we can find from (12) the p.d.f. of T_i, say $f(t_i)$, which we shall write $f(t_i \mid t_{i-1}, t_{i-2}, \ldots, t_2, t_1)$ for clarity, since this p.d.f. depends on the history preceding the present stage. For each i, x, we can calculate an upper $100x\%$ confidence bound from (14).

Denote this by $t_i(x)$, i.e.

$$P\{T_i < t_i(x)\} = x. \tag{26}$$

Then corresponding to our data $t_1, t_2, \ldots, t_n$ we have, for each x in $(0, 1)$, a sequence $t_1(x), t_2(x), \ldots, t_n(x)$ of $100x\%$ bounds.

Define a sequence $\{Y_i(x)\}$ of random variables by

$$\begin{aligned} Y_i(x) &= 1, \quad \text{if } t_i < t_i(x), \\ &= 0, \quad \text{otherwise} \end{aligned} \tag{27}$$

and let

$$Y(x) = n^{-1} \sum_i Y_i(x)$$

$$= \text{proportion of } t_i\text{'s in data which satisfy } t_i < t_i(x).$$

Clearly

$$E\{Y(x)\} = x \tag{28}$$

from (26) and (27).

Our procedure, therefore, is to plot $Y(x)$ against x. The result will be a step function, as shown in Fig. 4, which, if our choice of $\psi(.)$ function is correct, will

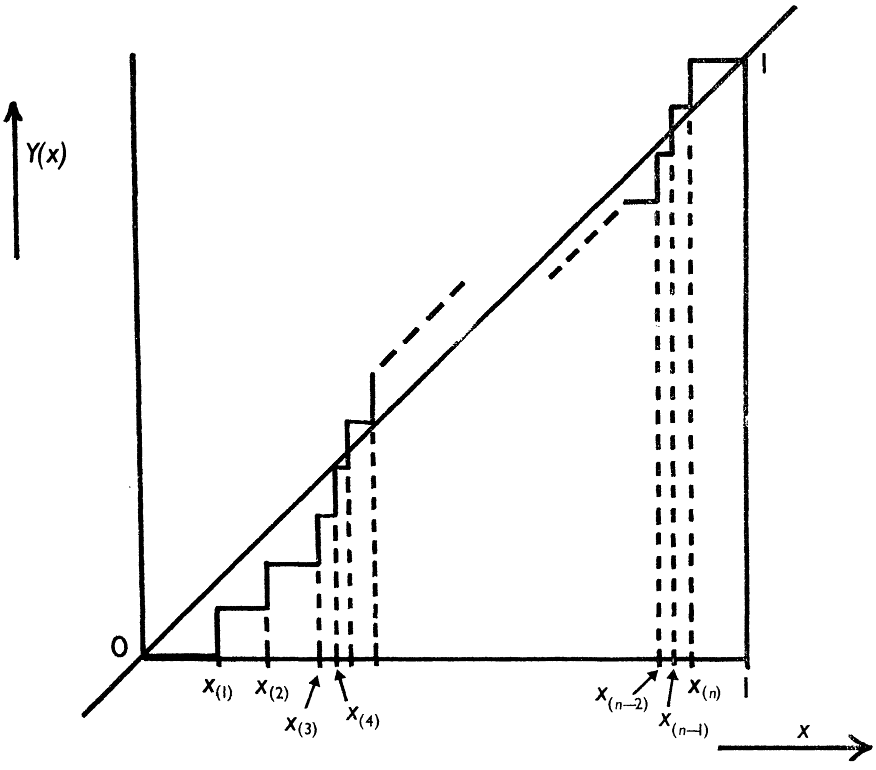

FIG. 4. Typical $Y(x)$ step function.

satisfy (28) and hence fluctuate about a mean line which is the line of unit slope through the origin. In other words the "closeness" of the step function to this line is a measure of the "correctness" of the $\psi(.)$ function.

It can easily be seen that each step in $Y(x)$ is of magnitude $1/n$, and that they occur at points x_i given by the solutions of

$$t_i = t_i(x), \quad i = 1, 2, \ldots, n. \tag{29}$$

In Fig. 4 these x_i are indicated in ascending order by $x_{(i)}$. It can be shown easily that requesting closeness of $Y(x)$ to the straight line is equivalent to specifying that x_i are distributed uniformly on $(0,1)$. That is $x_1, x_2, \ldots, x_n$ can be regarded as a random sample from a uniform distribution on $(0,1)$, and $Y(x)$ as a *sample distribution function* from this distribution. Tests such as Kolmogorov–Smirnov, or W^2 can be used to measure goodness-of-fit of this sample distribution function (see *Kendall and Stuart*, 1961, pp. 450–457), since the x_i are easily computable, being given by†

$$x_i = 1 - \left\{\left(\log \prod_{m=1}^{i-1} k_m\right) \Big/ \left(\log \prod_{m=1}^{i} k_m\right)\right\}^i. \tag{30}$$

The main consequence of the computational simplicity of this method is that it provides us with an attractive alternative to the numerical integration procedures of Section 4.1, when we make the same parametric assumptions as are made there. We recall that the method described in Section 4.1 assumed $\psi(i)$ to be a member of a family $\psi(\boldsymbol{\beta}, i)$. For any $\boldsymbol{\beta}$ in the allowable parameter space we can compute a Kolmogorov–Smirnov, or W^2, or any other suitable statistic of goodness-of-fit. It is an easy matter to search the allowable $\boldsymbol{\beta}$-space and find that value, $\tilde{\boldsymbol{\beta}}$ say, which optimizes the goodness-of-fit statistic (or even produce a $100\alpha\%$ confidence region of the β-space).

Numerical results illustrating this technique on simulated **t** data are presented in the Appendix.

Acknowledgement

The authors wish to express their gratitude to the CERL, Central Electricity Generating Board, U.K., for their suggestion of this problem, financial support and continuing encouragement.

Reference

Kendall, M. G. and Stuart, A. (1961). *The Advanced Theory of Statistics*, Vol. II. London: Griffin.

Appendix

Method of Section 4.2 *applied to Simulated Data*

The 80 observations of times between repair and failure given in Table 1 were generated using the model of (7) and (8) with

$$\alpha = 2{\cdot}0$$

$$\psi(i) = \exp(2{\cdot}0 + 0{\cdot}2i).$$

The method of Section 4.2 proceeds by calculating a set of goodness-of-fit statistics using $\psi(i) = \exp(\beta_0 + \beta_1 i)$ for different (β_0, β_1). We used the nW^2 statistic (see Kendall and Stuart, 1961), so our estimate $(\hat{\beta}_0, \hat{\beta}_1)$ is the point in the $\boldsymbol{\beta}$ space which minimises nW^2. We found

$$(\hat{\beta}_0, \hat{\beta}_1) = (1{\cdot}5, 0{\cdot}20640).$$

The minimum value of the Kolmogorov–Smirnov statistic (see Kendall and Stuart, 1961), on the other hand, occurred at

$$(\hat{\beta}_0, \hat{\beta}_1) = (2{\cdot}3, 0{\cdot}20085).$$

† Since x_1 is indeterminate we ignore it and in practice base our test upon the remaining $(n-1)$ x's. Such a small reduction of the sample size will not result in an appreciable loss of information, since we shall generally be dealing with fairly large n values.

TABLE 1

Eighty simulated times to failure, rounded to three significant figures (read left to right, last four rows × 1,000)

3·04	0·315	1·39	3·04	28·4	6·22	18·6	7·03	68·5	15·0
24·2	608·0	30·3	21·3	118·0	116·0	51·0	98·0	141·0	36·5
73·6	35·3	14·0	1,360·0	114·0	4,440·0	6,580·0	408·0	3,070·0	6,230·0
11·2	829·0	1,370·0	1,350·0	2,290·0	1,520·0	4,400·0	15,200·0	12,900·0	6,790·0
13·7	302·0	13·6	4·18	5·20	59·9	374·0	8·17	58·2	875·0
82·8	56·9	420·0	166·0	60·3	811·0	446·0	2,070·0	175·0	1,820·0
1,699·0	381·0	1,730·0	654·0	5,070·0	3,120·0	6,330·0	668·0	66·1	198·0
1,790·0	3,410·0	1,820·0	60·9	7,310·0	6,090·0	22,700·0	7,590·0	23,700·0	7,490·0

These results are good for the "growth parameter", β_1, being close to the true value of 0·2. For the "initializing parameter", β_0, the method does not produce estimates very close to the true value of 2·0. This probably suggests the relative unimportance of this parameter, since the effect of β_0 will quickly be swamped by the growth in $\exp(\beta_1 i)$.

It is encouraging that the Kolmogorov–Smirnov statistic produces such good results (better than nW^2 in this case) as it is relatively easy to compute.

As can immediately be seen from our table of t_i's, the growth in reliability with this $\psi(.)$ function is very rapid. We would therefore like to emphasize that the object of this exercise was not to produce particularly realistic data but to check the efficiency of our suggested inferential procedure.