A BAYESIAN MODIFICATION TO THE JELINSKI-MORANDA SOFTWARE RELIABILITY GROWTH MODEL

Bev Littlewood
Mathematics Department
The City University
Northampton Square
London EC1V 0HB
England

Ariela Sofer
Department of Operations Research
School of Engineering and Applied Science
George Washington University
Washington DC 20052
USA

Abstract

The Jelinski-Moranda (JM) model for software reliability is examined. We suggest that a major reason for the poor results given by this model is the poor performance of the maximum likelihood method (ML) of parameter estimation. A reparameterisation and Bayesian analysis, involving a slight modelling change, are proposed. It is shown that this new Bayesian-Jelinski-Moranda model (BJM) is mathematically quite tractable, and several metrics of interest to practitioners are obtained. A comparison of the BJM and JM models was carried out using several sets of real software failure data collected by Musa. In all cases the BJM model gave superior reliability predictions.

We discuss ways in which the assumptions underlying both models can be changed in order to represent the debugging process more accurately.

Acknowledgements

This work was begun whilst the first author was visiting the Department of Operations Research, George Washington University, to whose members he would like to express his thanks for their hospitality and many kindnesses over the year of his stay. This part of the work was partially supported by the US Army Research Office under grant number DAAG 29-80-0067. The work was completed on his return to The City University, London, supported by US Army European Research Office under grant number DAERO-79-0038 and National Aeronautics and Space Administration, Langley Research Center, under grant number NAG-1-179.

The second author's work was partially supported by National Aeronautics and Space Administration, Langley Research Center, under grant number NAG-1-179, and partially by US Army Research Office under grant number DAAG-80-0067.

We would like to thank John D. Musa, Bell Labs., for providing us with the results of the ML calculations on his model, and for several illuminating discussions about some of the issues discussed here.

Key words:

Software faults
Software failure rate
Software reliability
Program debugging model
Reliability growth
Jelinski-Moranda model
Musa Model

Running title:

Bayesian Jelinski-Moranda model

1. Introduction

The first software reliability growth models appeared more than ten years ago [1,2], but they seem not to have gained acceptance by practitioners. The reasons for this disappointing performance have not been widely reported in the literature (perhaps as a result of the unfortunate tendency of scientific journals to concentrate on "positive" results). It seems clear, though, that the need for software reliability measurement techniques is not disputed, so perhaps we should look to the poor performance of the models to explain the lack of acceptance.

In this paper we shall examine the Jelinski-Moranda (JM) model [1], possibly the earliest and certainly one of the best-known models. Although our remarks will be addressed to the JM model, it should be borne in mind that other models are similar to, or dependent upon, the JM model. Shooman's work, for example [2,3], seems to have paralleled that of Jelinski and Moranda. The model due to Musa [4] used the JM model as a basis, but introduced many important refinements. These refinements make this model particularly attractive to users, but its validity must ultimately rest upon the validity of its JM foundation. Goel and Okumoto [5] also generalise the JM model. Goel [6] casts the JM model assumptions into a different probabilistic structure. We believe that our remarks about the JM model also concern this work.

The JM model often gives misleading answers when the method of

maximum likelihood (ML) is used to estimate the parameters. We present a Bayesian modification to the model which overcomes a major source of difficulty. In our conclusion we suggest how the model might be further improved by changes to one of the basic underlying assumptions; we hope to report on this new model in a future paper.

2. The JM model

2.1 Model assumptions

The JM model, in common with most early models [1-7], treats the program as a black box with special characteristics supposedly representative of the special properties of software. No account is taken of the internal structure of the program. The only input to the model is the sequence of execution times between successive failures (see [4] for a cogent argument in favour of execution time): $t_1, t_2, \ldots$. The objective is to estimate current and future reliability on the basis of these past inter-failure times. The problem, then, is one of estimating and predicting reliability growth.

Assumptions made in the JM model are:

1. The random variables $T_i (i = 1,2,\ldots)$, representing successive interfailure execution times, are independent, with exponential distributions:

$$pdf(t_i|\lambda_i) = \lambda_i e^{-\lambda_i t_i} \qquad (1)$$

$$(\lambda_i > 0,\ t_i > 0)$$

2. At each failure, a fault is fixed instantaneously, with the result that the failure rate improves. All such improvements are of equal size so that

$$\lambda_i = (N-i+1)\phi \qquad (2)$$

where N : initial number of faults in program

ϕ : change (improvement) in failure rate at each fix.

See Figure 1.

Readers familiar with hardware reliability growth literature will notice that in these assumptions the repair rule is spelled out precisely, in contrast to models based on Duane's empirical postulate [8]. There, continuous reliability growth is allowed via a non-homogeneous Poisson process.

A detailed analysis of these assumptions has been given by one of us elsewhere [9]. Briefly, assumption #1 seems a plausible way of modelling our uncertainty about the nature of the input stream which the program must process. Assumption #2, representing the effect of successive fixes, appears less plausible. A stochastic process would seem to be a better way of representing the sequence $\{\lambda_i\}$ than the deterministic sequence, (2). After all, even in those circumstances where we can guarantee to have carried out a successful fix, we shall be uncertain as to its effect on the failure rate of the program (have we eliminated a large fault or a small one?). This is a theme we shall return to later in the paper.

2.2 Difficulties associated with using the model

There seem to be three main areas of difficulty. They concern the properties of the maximum likelihood estimates of the parameters, and the quality of reliability predictions.

1. $\hat{N}$, the ML estimator of N, is occasionally infinite. Since $\hat{N}$ and $\hat{\phi}$ are obtained by a numerical optimisation of the likelihood function, a user can easily interpret this effect as non-convergence of the optimisation routine [10]. Littlewood and Verrall [11], however, show that in certain circumstances the unique true maximum of the likelihood function will be at $\hat{N} = \infty$, $\hat{\phi} = 0$ (with finite, non-zero $\hat{\lambda} \equiv N\hat{\phi}$). A necessary and sufficient condition for this is shown to be that the least squares regression line of t_i versus i has non-positive slope. The condition is intuitively appealing: it suggests that the JM model, being a reliability growth model, will give nonsensical answers unless the data exhibits reliability growth. It needs to be said, though, that even when we simulate data from the JM model (finite N, non-zero ϕ) there is a non-zero probability that a particular data set will show no growth according to this condition.

In real data sets, this problem does not often arise except at early stages in debugging, i.e. when the sample size is small. This is presumably because most data sets come from programs which are genuinely improving in reliability. We have, however, encountered one set of real-life data, System 5 in Musa's collection

[12], where the effect persists for the first 150 failures. In order to handle situations of this kind, it would be necessary to use a more general model which could estimate reliability growth or reliability decay, for example that described by Littlewood and Verrall (LV) [7,22].

2. A more serious problem is that often $\hat{N}$ is only slightly larger than n, the sample size (number of failures experienced, number of faults fixed). Thus, estimates of N based on increasing amounts of information usually increase with n. This raises doubts about the consistency of the ML estimators, but it is questionable whether such a concept has any meaning in this context: the size of the "sample", n, is bounded above by a parameter, N. Forman and Sinpurwalla [14] have shown that $\hat{N}$ and $\hat{\phi}$ can only be trusted near the end of debugging, i.e. when almost all faults have been removed and the true value of N is only slightly larger than n. This observation, however, is of little practical value since we would never know the end of debugging was near. It is certainly not the case that $\hat{N}$ takes values close to n only near the end of debugging.

Forman & Singpurwalla claim that their empirical stopping rule tells you.

At its most serious, this effect results in $\hat{N} = n$ exactly for a range of values of n. Thus, the ML estimator suggests that the last fault has been removed and the program is perfect even when this is far from being the case. Table 1 shows this in an analysis of Musa's System 3 data [12]. From failure number 25 onwards, successive estimates of N tell us

that the program is perfect; and each time the program reveals its imperfections by failing. A similar effect occurs in Musa's System 40, where $\hat{N} = n$ between $n = 76$ and $n = 99$. In these cases, it is obvious that the model cannot give good reliability predictions: it will estimate the reliability to be 1, the mttf to be infinite, the failure rate zero etc.

3. This bring us to the last, and perhaps most important, problem. In almost every data set we have analysed, the model has produced results which are too optimistic: it seems always to predict the reliability to be greater than it really is. Clearly, this effect will not be independent of the effect described in the previous paragraph: if ML gives poor estimates of the parameters, it seems likely that the resulting estimates of reliability metrics will be poor. On the other hand, if the modelling assumptions are wrong, we shall obtain poor reliability prediction however we make inference about the model parameters.

Our intention in what follows is to improve upon the results which can be obtained by using ML on the JM model. Our Bayesian approach to inference necessitates a slight change to the model itself, but we believe this to be sufficiently minor as to justify calling it a Bayesian Jelinski-Moranda model.

3. The Bayesian Jelinski Moranda (BJM) Model

We begin by reparameterisation to (λ,ϕ) where

$$\lambda \equiv N\phi \tag{3}$$

the initial failure rate of the program. A formal motivation for this parameterisation can be found in [11] and has already been mentioned: even when the likelihood has its maximum at infinity in N, $\hat{\lambda}$ is finite and non-zero. An informal justification comes from inspection of Figure 1. Our data will always concern the earlier stages of debugging, and the statistical problem is one of fitting λ_i as a linear function of i (failure number). Since our data will concern the left of this plot, it seems plausible that we can obtain good estimates of the intercept on the vertical axis: namely, λ. Estimation of N, however, implies estimation of the intercept on the horizontal axis. Such estimation will involve large errors as a result of quite small errors in estimation of ϕ, the slope of the line. Notice that this reasoning explains the observation of Forman and Singpurwalla [14] that estimates of $\hat{N}$ can be trusted only near the end of debugging. Although the argument above is a plausible reason for the poor quality of ML estimates of N, it does not explain why the estimates tend to be too small. We shall discuss this point later.

We shall adopt a Bayesian approach to the inference, with independent Gamma priors for λ and ϕ. Since λ is no longer constrained to be an integer multiple of ϕ, this involves a

slight change to the model. Instead of assumptions 1 and 2 of the JM model, the BJM model assumes:

The successive inter-failure execution times, T_1, T_2, T_3, ... are independent random variables, exponentially distributed with parameters λ, $\lambda-\phi$, $\lambda-2\phi$, ...

The main effect of this is that the repair rule changes at the last failure. Each fix except the last removes an amount ϕ from the failure rate of the program. When the failure rate is less than or equal to ϕ, the next fix makes the program "perfect" (zero failure rate). It seems likely that, except for programs with a very small number of faults, the BJM and JM models will be very similar.

We now let prior pdf of (λ,ϕ) be

$$\text{prior } (\lambda,\phi) \equiv \text{prior } (\lambda) \quad \text{prior } (\phi) \tag{3}$$

where prior (λ) is Gamma $(\lambda;b,c)$ and prior (ϕ) is Gamma $(\phi;f,g)$, i.e.

$$\text{prior } (\lambda) = \frac{c^b \lambda^{b-1} e^{-c\lambda}}{\Gamma(b)} \quad (\lambda > 0) \tag{4}$$

and

$$\text{prior } (\phi) = \frac{g^f \phi^{f-1} e^{-g\phi}}{\Gamma(f)} \quad (\phi > 0) \tag{5}$$

The hyperparameters b, c, f and g are to be chosen by the user according to his prior knowledge and subject to the constraints that all are positive and b is an integer. This last condition is for mathematical tractability alone, but is

probably not unduly constraining. Elicitation of prior knowledge in order to give values to the hyperparameters is not easy. In our own work we have used "ignorance priors" for λ and ϕ: non-informative (improper) uniform distributions obtained by letting $c, g \to 0$ and $f = b = 1$. We shall proceed in the main body of the text to adopt this simplification. The more general results using the full gamma priors are relegated to the appendix.

We shall assume, then, that

$$\text{prior } (\lambda,\phi) \equiv 1, \qquad (\lambda, \phi > 0) \tag{6}$$

and that we have observed $t_1, t_2, \ldots, t_n$. We have

$$\begin{aligned} \text{posterior } (\lambda,\phi) &\equiv p(\lambda, \phi | t_1, \ldots t_n) \\ &= C.p(t_1, \ldots, t_n|\lambda,\phi) \quad \text{prior } (\lambda,\phi), \end{aligned} \tag{7}$$

by Bayes theorem, where

$$C^{-1} = \iint p(t_1, \ldots t_n|\lambda,\phi) \quad \text{prior } (\lambda,\phi) d\lambda d\phi \tag{8}$$

and the likelihood function is

$$\begin{aligned} p(t_1, \ldots, t_n|\lambda,\phi) &= \lambda(\lambda-\phi) \ldots(\lambda-[n-1]\phi) \\ &\quad \exp\{-\lambda t_1 - (\lambda-\phi)t_2 - \ldots -(\lambda-[n-1]\phi)t_n\} \\ &\quad \text{if } \lambda \geq (n-1)\phi \\ &= 0 \quad \text{otherwise.} \end{aligned} \tag{9}$$

If we define

$$\prod_{i=1}^{n} (x+i) = \sum_{i=0}^{n} a_{i,n}\, x^{n-i} \tag{10}$$

a little analysis shows that the posterior distribution is

$$p(\lambda, \phi | t_1, \ldots, t_n) =$$

$$\frac{\sum_{i=0}^{n-1} a_{i,n-1} (-1)^i \phi^i \lambda^{n-i} e^{-\lambda \sum_1^n t_j} e^{-\phi \sum_1^n (j-1) t_j}}{\sum_{i=0}^{n-1} a_{i,n-1} \frac{i!\,(n-i)!}{\left[\sum_1^n (n-j) t_j\right]^{i+1} \left[\sum_1^n t_j\right]^{n-i+1}}} \qquad (11)$$

for $\lambda > (n-1)\phi$ and zero otherwise.

This expression is much more tractable than might appear at first glance. In the next section we shall obtain analytic expressions for many of the reliability metrics which are of practical interest. It is surprising, in fact, that the computational difficulties associated with the BJM model are considerably less than those associated with the numerical optimisations required by ML estimation in the JM model.

The coefficients defined in (10) are closely related to Stirling numbers of the first kind [15], and are most easily computed from the relation

$$a_{i,n} = n\, a_{i-1,n-1} + a_{i,n-1}\,, \quad i \geq 1, \qquad (12)$$

noting that $a_{0,1} = 1$, $a_{1,1} = 1$, $a_{0,n} = 1 \;\forall\; n$.

4. Using the BJM model

In this section we show how the BJM model can be used to calculate metrics of practical value. We begin with measures of current reliability which will be used in the next section to compare the performance of this model with the JM model.

4.1 Current reliability

Having observed n failures, and carried out n fixes, the simplest question we can ask is: how reliable is the program now? The various ways in which this question can be answered all involve statements about the random variable T_{n+1}, the time of failure-free execution until the next failure of the program. Consider the reliability function

$$R_{n+1}(t|\lambda,\phi) \equiv P(T_{n+1} > t|\lambda,\phi) \tag{13}$$

$$= e^{-(\lambda-n\phi)t} \quad \text{if} \quad \lambda > n\phi \tag{14}$$

$$1 \quad \text{if} \quad (n-1)\phi < \lambda < n\phi$$

remembering that T_n is not observed if $\lambda < (n-1)\phi$.

In our comparison between JM and BJM we shall use the posterior mean of this:

$$R_{n+1}(t|t_1, \ldots, t_n) = P(T_{n+1} > t|t_1,\ldots,t_n) \tag{15}$$

$$= \int\int R_{n+1}(t|\lambda,\phi)p(\lambda,\phi|t_1,\ldots,t_n)d\lambda d\phi \tag{16}$$

This can be interpreted as the reliability function calculated from the posterior distribution of T_{n+1}.

Substituting for (14) and (11) into (16) and simplifying considerably, we obtain

$$R_{n+1}(t|t_1,\ldots,t_n) = C\left[\sum_{i=0}^{n}\frac{a_{i,n}(n-i)!\ i!}{(t+\Sigma t_j)^{n-i+1}(\Sigma(n-j+1)t_j)^{i+1}}\right.$$

$$+\sum_{i=0}^{n-1}\frac{a_{i,n-1}\,i!(n-i)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j)t_j)^{i+1}}$$

$$\left.-\sum_{i=0}^{n}\frac{a_{i,n}\,i!(n-i)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j+1)t_j)^{i+1}}\right] \qquad (17)$$

where C^{-1} is

$$\sum_{i=0}^{n-1}\frac{a_{i,n-1}\,i!(n-i)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j)t_j)^{i+1}} \qquad (18)$$

By differentiating (17) it can be shown that the posterior distribution of T_{n+1} is a mixture of Pareto distributions.

The posterior probability that the program is now "perfect", i.e. that the last error has been removed, is

$$P_0 = R_{n+1}(\infty|t_1,\ldots,t_n) = P((n-1)\phi < \lambda < n\phi|t_1,\ldots,t_n)$$

$$= 1 - \frac{\displaystyle\sum_{i=0}^{n}\frac{a_{i,n}\,i!(n-1)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j+1)t_j)^{i+1}}}{\displaystyle\sum_{i=0}^{n-1}\frac{a_{i,n-1}\,i!(n-i)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j)t_j)^{i+1}}} \qquad (19)$$

We shall see, in the case of Musa's System 3 and System 40 data (see earlier comments), that this expression gives much more plausible answers than the JM model.

Some care has to be taken in calculating these expressions (17-19) because the coefficients $a_{i,n}$ can be extremely large. However, very little machine time is required. The authors have a Fortran program which is available to readers on request.

An alternative to using the posterior mean of the conditional reliability function, (13), is to set ourselves a reliability target and then ask how strong is our posterior belief that this target has been achieved. If we let the target reliability be a pair of numbers (t,r) such that

$$P(T > t) > r \quad , \tag{20}$$

it can be seen that our posterior probability that this has now been achieved is

$$\begin{aligned} & P\{R_{n+1}(t|\lambda,\phi) > r|t_1,\ldots,t_n\} \\ & = P\{R_{n+1}(t|\lambda,\phi) = 1 \text{ or } r < R_{n+1}(t|\lambda,\phi) < 1|t_1,\ldots,t_n\} \\ & = P_o + 1 - P\{R_{n+1}(t|\lambda,\phi) \le r|t_1,\ldots,t_n\} \end{aligned} \tag{21}$$

where

$$\begin{aligned} & P\{R_{n+1}(t|\lambda,\phi) \le r|t_1,\ldots,t_n\} \\ & = P\{\lambda - n\phi \ge -\frac{\log r}{t} \,|\, t_1,\ldots t_n\} \\ & = C \sum_{i=o}^{n} \frac{a_{in}\, i!}{(\Sigma(n-j+1)t_j)^{i+1}} \int_{x=-\frac{\log r}{t}}^{\infty} x^{n-i} e^{-x\Sigma t_j}\, dx \\ & = C \sum_{i=o}^{n} \frac{a_{in}\, i!(n-1)!}{(\Sigma(n-j+1)t_j)^{i+1}\left(-\frac{\Sigma t_j \log r}{t}\right)^{n-i+1}} \sum_{k=o}^{n-i} \frac{e^{\frac{\Sigma t_j \log r}{t}}\left(-\frac{\Sigma t_j \log r}{t}\right)^{k}}{k!} \end{aligned} \tag{22}$$

4.2 Current failure rate

The current failure rate of the program, when the nth fault has been fixed, is given by

$$\Lambda = \lambda - n\phi \quad \text{if} \quad \lambda - n\phi > 0 \qquad (23)$$
$$0 \qquad \text{otherwise}.$$

We can find the posterior pdf of Λ conditional on the program not being perfect as follows:

$$p(\Lambda \mid \Lambda > 0,\ t_1, t_2 .. t_n)$$

$$= \frac{p(\Lambda,\ \Lambda > 0 | t_1, \ldots t_n)}{P(\Lambda > 0 | t_1, .. t_n)} \qquad (24)$$

The denominator has already been evaluated, (19). We can find the numerator by transforming (11) from (λ, ϕ) to (Λ, ϕ) and integrating out ϕ. Then (24) becomes

$$B \sum_{i=0}^{n} a_{i,n} \frac{i!}{[\Sigma(n-j+1)t_j]^{i+1}} \Lambda^{n-i} e^{-\Lambda \Sigma t_j} \quad (\Lambda > 0) \qquad (25)$$

where B is a normalising constant. This is a mixture of Gamma $(\Lambda; n-i+1, \Sigma t_j)$ densities. If our reliability target has been formulated in terms of a target failure rate, ℓ say, (19), (23), (25) can be used to obtain the probability that the target has been achieved, $P(\Lambda < \ell)$. This calculation involves evaluation of incomplete gamma integrals for which tables are available.

A simpler procedure is to calculate the posterior expected value of Λ. This can be interpreted as the failure rate obtained

at the origin of the posterior distribution of T_{n+1}

$$\equiv \lim_{\delta \to 0} \frac{P(0 < T_{n+1} < \delta | t_1, .. t_n)}{\delta} \tag{26}$$

$$= \frac{-d}{dt} R_{n+1}(t | t_1, \ldots t_n) \Big|_{t=0}$$

This is the "current posterior failure rate", and is given by

$$E(\Lambda | t_1, .. t_n)$$

$$= E(\Lambda | \Lambda > 0, t_1, .. t_n) P(\Lambda > 0 | t_1, \ldots, t_n) \tag{27}$$

where $P(\Lambda > 0 | t_1, .. t_n)$

$$= \frac{\sum_{i=0}^{n} \frac{a_{i,n}\, i!\, (n-i)!}{(\Sigma t_j)^{n-i+1} (\Sigma(n-j+1)t_j)^{i+1}}}{\sum_{i=0}^{n-1} \frac{a_{i,n-1}\, i!(n-i)!}{(\Sigma t_j)^{n-i+1} (\Sigma(n-j+1)t_j)^{i+1}}} \tag{28}$$

from (19), and $E(\Lambda | \Lambda > 0, t_1, .. t_n)$

$$= \frac{\sum_{i=0}^{n} \frac{a_{i,n}\, i!\, (n-i)!}{(\Sigma t_j)^{n-i+1} (\Sigma(n-j+1)t_j)^{i+1}} \cdot \frac{n-i+1}{\Sigma t_j}}{\sum_{i=0}^{n} \frac{a_{i,n}\, i!\, (n-i)!}{(\Sigma t_j)^{n-i+1} (\Sigma(n-j+1)t_j)^{i+1}}} \tag{29}$$

4.3 Mean time to failure

Since there is always a non-zero probability that the last fix removed the last error, the distributions for T_{n+1}, T_{n+2}, ... conditional on $t_1, .. t_n$ are improper and their expectations do not exist. It is, however, possible to find the posterior distribution of T_{n+k} conditional on $T_{n+k} < \infty$ (it is a mixture of Paretos). These distributions, together with the probabilities $P(T_{n+k} < \infty)$, are useful, and we consider them next.

4.4 Future reliability, number of faults remaining

Let the random variable K_n denote the number of faults remaining immediately after the removal of the n th fault

$$P(K_n = k | t_1, \ldots t_n)$$

$$= P(K_n \geq k | t_1, \ldots t_n) - P(K_n \geq k+1 | t_1, \ldots t_n)$$

$$= P(\lambda \geq (n+k-1)\phi | t_1, \ldots t_n)$$

$$- P(\lambda \geq (n+k)\phi | t_1, \ldots t_n) \qquad (30)$$

Now

$$P(K_n \geq k+1 | t_1, \ldots t_n)$$

$$= C \int_{\phi=0}^{\infty} \int_{\lambda=(n+k)\phi}^{\infty} p(\lambda, \phi | t_1, \ldots, t_n) d\lambda d\phi$$

$$= C \sum_{i=0}^{n} \frac{b_{in}^{k} \, (n-i)! \, i!}{(\Sigma t_j)^{n-i+1} (\Sigma (n+k-j+1) t_j)^{i+1}} \qquad (31)$$

after some analysis. Here C^{-1} is given by (11) and the coefficients b_{in}^{k}, related to the a_{in} coefficients, are given by

$$\sum_{i=0}^{n} b_{in}^{k} \, \phi^{i} \, x^{n-i} = \prod_{j=1}^{n} (x + (k+j)\phi) \qquad (32)$$

It can be shown that

$$b_{in}^{k} = \sum_{m=0}^{i} a_{mn} \binom{n-m}{i-m} k^{i-m} \qquad (33)$$

Finally, substituting into (30), we get

$$P(K_n = k | t_1, \ldots t_n)$$

$$= C \sum_{i=0}^{n} \frac{i!(n-i)!}{(\Sigma t_j)^{n-i+1}} \sum_{m=0}^{i} \frac{a_{mn}(n-r_1)!}{(n-i)!(i-m)!} \left[\frac{(k-i)^{i-m}}{(\Sigma(n+k-j)t_j)^{i+1}} \right.$$

$$\left. - \frac{k^{i-m}}{(\Sigma(n+k-j+1)t_j)^{i+1}} \right] \quad (34)$$

for $k = 1,2,\ldots$ and

$$P(K_n = 0 | t_1, \ldots t_n)$$

$$= 1 - C \sum_{i=0}^{n} \frac{a_{in}\, i!\,(n-i)!}{(\Sigma t_j)^{n-i+1}(\Sigma(n-j+1)t_j)^{i+1}} \quad (35)$$

which agrees with (19).

Expressions such as these can give us useful upper bounds on the number of failures which can occur during the lifetime of the program, assuming a fault-fixing strategy is adopted. They thus give upper bounds on lifetime maintainance costs. However, since the times between failures have distributions which are mixtures of Pareto distributions, the time needed to uncover the last fault may be very much larger than a realistic program lifetime. In such cases we shall obtain pessimistic estimates of maintainance costs by using (34), (35).

Consider now the random variable T_{n+k}. We shall observe this random variable only if $K_n \geq k$. Then

$$P(T_{n+k} > t, K_n \geq k | t_1, \ldots, t_n)$$

$$= \int_{\phi=0}^{\infty} \int_{\lambda=(n+k-1)\phi}^{\infty} P(T_{n+k} > t | \lambda, \phi) p(\lambda, \phi | t_1, \ldots t_n) d\lambda\, d\phi$$

$$= C \sum_{i=o}^{n} \frac{(n-i)!\ i!}{(t+\Sigma t_j)^{n-i+1}(\Sigma(n+k-j)t_j)^{i+1}} \sum_{m=1}^{i} \frac{a_{mn}(n-m)!(k-1)^{i-m}}{(i-m)!(n-i)!}$$

after some analysis. (36)

The above expression suggests how we might answer the important question : how much debugging is still needed before the program will have achieved its target reliability? Consider a target reliability expressed as a pair (t,r) such that

$$P(T > t) > r \tag{37}$$

The target will be achieved in less than k failures if

$$P([T_{n+k} > t \text{ and } K_n \geq k] \ \underline{\text{or}} \ K_n < k | t_1, .. t_n) > r, \tag{38}$$

that is

$$P(T_{n+k} > t \text{ and } K_n \geq k | t_1, .. t_n) + P(K_n < k | t_1, ..., t_n) > r \tag{39}$$

The procedure then, is to calculate the L.H.S. of (39) for $k = 1,2,...,$ until the first value of k for which the condition is satisfied. This is then an estimate of how many more fixes have to be carried out to achieve the reliability target.

5. Comparison of BJM and JM analyses of software reliability data.

We shall concentrate in this section on analyses of some software reaiability data sets published by Musa [12]. It is a source of amazement to us that there is so little good quality software reliability data available in the open literature. There are two reasons for this. Often data is collected in a manner which renders it unsuitable for modelling purposes. More commonly, wher suitable data does exist, it is guarded jealously by the producer organisation in the belief that its publication would cause loss of confidence in their software products. We think this belief is mistaken : reputations are more likely to suffer from the suspicions which these secretive actions engender.

Musa is to be congratulated on publishing seventeen sets of data which were collected under carefully controlled conditions. These data sets seem to be the only ones of reasonable quality which are reauily available. Even this data, representing the successive execution times between failures $(t_1, t_2, \ldots t_n, \ldots)$, occasionally gives rise to disquiet. Simple tests of trend show that only a few of the programs are exhibiting reliability growth [13]. In what follows, we have concentrated on these (with tne exception of System 40, which is discussed later). There is some evidence that the successive times are correlated. Of course, this does not necessarily imply criticism of the data collection, but successions of small observations might suggest "poor fixes". We have not attempted to eliminate any of these effects in what follows, so as not to be open to charges of massaging the data.

Our procedure for examining model performance is discussed in detail in [16] and has also been used by other workers in the reliability growth field [17, 18]. Briefly, we shall compare model predictions with actual observations with the intention of asking the same question as might be asked by a potential user of two rival models. In what follows we shall concentrate on the ability of the models to estimate current reliability; see [16] for a discussion of the problem of examining the quality of longer-term predictions of reliability.

Assume that $(i-1)$ failures have been observed, so our data set is $t_1, t_2, \ldots t_{i-1}$, and we are interested in the current reliability. This is a statement about the random variable T_i. Consider the predictor cdf of T_i, say $F_i(.)$. For the JM model this is

$$\hat{F}_i(t) = F_i(t\,;\hat{N},\hat{\phi}) = 1 - e^{-(\hat{N}-i+1)\hat{\phi}t} \tag{40}$$

Where $\hat{N}$ and $\hat{\phi}$ are ML estimates of N,ϕ based on $t_1,\ldots t_{i-1}$. For the BJM model we use

$$\hat{F}_i(t) = 1 - R_i(t|t_1,\ldots t_{i-1}) \tag{41}$$

which is obtainable from (17) and (18). All statements about the current time to failure random variable involve $\hat{F}_i(.)$, so it seems plausible to base our examination of the quality of the model upon this. If $\hat{F}_i(.)$ were the "true" distribution of T_i, then

$$U_i = \hat{F}_i(T_i) \tag{42}$$

would be uniformly distributed on $(0,1)$, and be at least asymptotically

independent for different i.

We shall consider their realisations

$$u_i = \hat{F}_i(t_i), \tag{43}$$

where t_i is the realisation of T_i, i.e. it is the actual observed time between the (i-1)th and i th failures. These numbers thus form the basis of a comparison of our predictions (based on $t_1, \ldots t_{i-1}$) and our actual t_i. Our first tool will be the quantile-quantile (Q-Q) plot: i.e. a plot of the ordered set of m u_i's against i/m. The closeness of this to the line of unit slope is an indication of the closeness of the u's to uniformity, and so an indication of the quality of prediction of the model. We shall refer to this as Procedure 1.

Braun and Paine [17] suggest that the plot of u_i (not reordered) versus i should also be examined : it should look "patternless" if the model is performing well. Presumably the intention is to attempt to discover how well the model is capturing the trend. We have found these plots quite difficult to interpret, and have instead used the following informal procedure. If the u_i given by (43) really were realisations of independent, identically distributed (iid) uniform random variables then

$$x_i = -\log(1-u_i) \tag{44}$$

would be realisations of iid unit exponential random variables. Thus a process with interevent times given by these x_i's would be a realisation of a simple Poisson process. It is well known [19] that if we take the time to the (m+1)th event in such a process to be unity, the times of occurrence of the m events are independently uniformly distributed over (0,1). In our case,

if the model is performing well, this statement will be approximately true. Since the procedure is well known, in the exact case, to be sensitive to trend [19], we might expect to be able to detect when a model is not capturing the trend (reliability growth) adequately. We proceed by plotting the empirical cumulant distribution function of the numbers

$$y_i = \sum_1^i x_j \Big/ \sum_1^m x_j \qquad (45)$$

Again, good performance is indicated by closeness of this to the line of unit slope through the origin. We shall refer to this as Procedure 2.

It is perhaps worth stating explicitly than in neither of these two informal procedures is it our intention to carry out a goodness-of-fit test. On the contrary, in the context of this paper it is our contention that the JM and BJM models are virtually identical : thus if one performs notably better than the other the reason will presumably be that the inference procedures are performing differently. Our two informal procedures are designed to emulate the behaviour of an actual user of a model, who is interested primarily in whether he can trust the model predictions. The general problem of examining the quality of model predictions, as opposed to testing goodness-of-fit of models, is an interesting one which has received relatively little attention.

Table 1 shows the data and calculations on both models for Musa's System 3. As has been stated earlier, the JM model performs badly by giving $N = n$ for sample sizes 25 through 38. The BJM model gives probabilities for such perfection, P_0, which although appreciable, differ considerably from unity. Notice

that the range of sample sizes for which BJM gives P_0 significantly different from zero, sizes 25 through 38, is the same as produced $\hat{N} = n$ for JM. The Q-Q plots of the two models (Procedure 1) are shown in Plot 1, where it is obvious that BJM gives considerable improvement over JM. The poor behaviour of the JM model is almost entirely accounted for by the cluster of values at zero corresponding to $i = 25,..38$. It is surprising that the worst results from the model come from the end of the data series: it might be expected that with larger samples the estimation procedure would perform better. If this program is close to being bug-free, the results cast further doubt on the practical usefulness of the observation of Forman and Singpurwalla [14], that ML estimates of N can be trusted at the end of debugging. It is certainly not the case that a zero value for $\hat{N}-n$ gives high confidence that the program is now perfect.

Plot 2 shows the result of applying Procedure 2 to this data with the two models. The cluster of zero observations at the end of testing for JM cause the poor performance, as might be expected. However, the plot is reasonably linear, albeit with wrong slope: this suggests that the trend is being captured fairly well in the earlier stages of testing. Although the BJM model is considerably better, the concavity of the plot again suggests that the trend is not being captured completely. In fact the BJM model is also giving optimistic answers for $n = 25$ onwards, although not nearly so optimistic as JM.

Table 2, Plots 3 and 4, show the results of analyses of System 40 [12] data. The JM model, using ML estimation, gives $\hat{N} = n$ for $n = 76$ through 99. This accounts for the large

deviation at the origin of Plot 3, and the extreme departure from the line of unit slope in Plot 4. BJM gives a better Plot 3, but is still quite optimistic in its predictions, and the Plot 4 behaviour is little better than JM. A closer examination of the data set causes some disquiet, and may explain the poor results. Simple trend tests ([19], p.47) were carried out on the whole data set, and the first and last halves separately. These show significant growth overall, but not significant growth in either half. The lack of growth in the first half of the data is revealed in the many infinite estimates of N in the JM model, see [11]. There seems, therefore, to be evidence that this program exhibited a quite sudden, perhaps discrete, improvement in reliability half way through the collected data. This would explain the overall reliability growth, but the absence of growth in each half. Musa, however, does not recall any conditions in testing which would have produced such an effect. Of course, it is unreasonable to expect any reliability growth model to perform well in a case like this : all models assume some homogeneity of growth behaviour. This data set exhibits some of the pitfalls we have to beware of when analysing software failure data. In many cases we shall know when a discrete change of behaviour has occurred (change in testing procedure, integration of more code): we cannot normally expect models to perform well over such a discontinuity of behaviour.

Examination of the JM model Plot 4 reveals an extremely "jagged" behaviour even for the first 20 or so plotted points : evidence of larger variability in the x_i's than would be suggested if the model were performing well. This is clear from the raw data, where there appear to be more very large and very small observations than would

be plausible under the exponential assumptions of JM (and BJM). The BJM model handles this variability a little better than JM (see, for example, the distances between steps 2 and 3 in Plot 4 for the two models, corresponding to the large 54th observation): perhaps because of the long-tailed nature of the mixed Pareto posterior time-to-next-failure distributions. Our suspicion is that the data is contaminated in some way: it may be, for example, that the small observations represent imperfect fixes and ought to be excluded. Unfortunately, any filtering of the data has to be carried out by the data-collector at the time of collection, using criteria which are based on an analysis of the actual circumstances of the failures. It does not seem possible to base a data rejection procedure solely upon the data itself. Accordingly, our analyses were performed on the data as published, and we merely record our reservations.

Tables 1 and 2 are revealing about the general untrustworthiness of estimates of N in JM. Advocates of the JM model have argued that knowledge of N, or more precisely $N - n$ (the number of remaining faults) is of great practical interest and can be provided by use of this model. One of us has suggested elewhere [20] that reliability itself is the only metric of interest. Tables 1 and 2, which are fairly typical of the analyses we have seen of real data sets, show estimates of the number of remaining faults fluctuating wildly between infinity and zero. Our inability to obtain good estimates seems to us to render purely academic any discussion of their utility. If only the quality of reliability prediction is the issue, then models which treat failure rate directly [7, 22] can be considered on an equal footing with fault-counting models.

Plots 5 and 6 relate to System 1 data [12]. Although this data set does not produce zero or infinite estimates of the number of remaining faults, the estimates of N are consistently optimistic, being only slightly larger than the sample size. Reliability estimation (Plot 5) for JM is here better than in the two previous examples, although again optimistic. The BJM model is slightly better, but also gives optimistic results.

Plot 6 is quite interesting. In the first place, there is very little difference between BJM and JM on this plot, suggesting that each model captures the trend with similar accuracy. Since BJM is slightly better on Plot 5, this might suggest that the shape of the distributions of time to failure is represented better by the mixed Paretos than by exponentials. Of more interest, though, is the shape of Plot 6. Until about observation 90, the plot is reasonably linear (if we ignore the first four extremely small observations on this plot). Both models seem to be performing well between sample sizes 34 and 90, and only start to give very optimistic reliability predictions from 90 onwards. This may again suggest that some discrete change of behaviour has occurred. Musa, who collected this data set personally, is not aware of any such change, so the apparent effect may be spurious.

It is possible that, in cases such as these, better results would be obtained by not using all the data for the later predictions. We might choose to base each calculation only on the last 50 observations, say, in order to make the model fairly responsive to discrete changes in behaviour. It would be very difficult to justify a particular choice of "lag", though, and our own feeling is that greater care should be taken to ensure

homogeneity of behaviour during data collection.

Plots 7 and 8 concern Musa's System 2 data [12]. These plots, like the previous ones, are typical of the results we have found on other data sets: BJM is noticeably better than JM, but still gives optimistic answers. In all the data sets we have analysed, BJM is better than JM. However, the degree of improvement obtained by using BJM varies considerably; it is greatest when JM gives $\hat{N} = n$ for a substantial range of n. In all cases the JM model errs on the side of optimism, as does the BJM model but less markedly. We shall discuss this issue in more detail in the next section.

6. Conclusions and discussion of possible future work.

In our work so far, we have not found any data sets for which JM performs better than BJM, and in several cases BJM is much better. We therefore suggest that any practitioner who is tempted to use the JM model should instead use the BJM model. Since all important metrics for the BJM model are available in closed form, use of this model brings an important bonus of computational simplicity.

The BJM model has certain conceptual advantages over the JM model. Perhaps most important, it allows calculation of the probability that the program is currently fault-free. It also gives estimates in closed form of the remaining number of fixes to be carried out to achieve target reliability, as well as the number of faults remaining in the program. These metrics could be of great value in estimating the extra development effort needed, as well as providing information about maintenance costs.

We believe, then, that there are considerable potential advantages to be gained in using the BJM rather than JM model. Accordingly we recommend the new model to users, who can be confident that they will at least obtain answers which are no worse than would have been obtained by the old model.

Having said that, we think it is important that users of any software reliability model do not simply assume that the metrics are trustworthy. We suggest that whenever a data set is analysed, the quality of the metrics on that data set should be examined. This can easily be performed using our techniques or other informal methods. This kind of analysis can never give assurance

that the correct model has been applied, but it will usually be able to detect the use of a grossly unsuitable model and/or inference procedure.

So far we have compared the two models and shown that BJM never seems to be worse than JM. We now consider the fact that BJM still seems to give optimistic predictions in most cases. The degree of this optimism varies considerably from one program to another, which reinforces our suggestion that in each application it is advisable to investigate the quality of the predictions. It is interesting that the model always seems to deviate in the same direction: towards being too optimistic. We believe this is a consequence of the basic assumption underlying both models, that all faults contribute the same amount to the overall failure rate of the program. In fact, it seems much more plausible that a program starts life containing faults of different sizes, i.e. faults which contribute different amounts to the program failure rate. Since both models assume faults are uncovered randomly, this would imply that the times to discovery of different faults are differently (not identically) exponentially distributed. This scenario seems to accord with experience: some program faults seem "difficult to find" in the sense that, if they were left in the program, they would manifest themselves in program failures very infrequently. Others seem to be associated with high occurrence rates. This effect is modelled via a Bayesian argument in a recent paper [21] by one of us.

If the rates associated with the pool of faults initially in the program really are different, and if fault discovery (failure occurrence) occurs randomly (as both JM and BJM assume), then it

might be expected that early fixes cause greater improvement in the failure rate of the program than later fixes. This "law of diminishing returns" of debugging would be represented by a failure rate, as a function of i (failure number), which is shown in Fig 2. Using the JM or BJM model could then be seen as somehow "best fitting" a linear function to this non-linear graph. Fig 2 shows how such an operation might be expected to give optimistic estimates of the current failure rate. Confirmation of this hypothesis comes from an examination of the behaviour of the ML estimate of $\lambda \equiv N\phi$, the initial failure rate, in the JM model. This is the intercept on the vertical axis on Fig 2. If our assertion were correct, we would expect that, as the sample size increased, the "best-fitting" straight line on Fig 2 would have decreasing slope $\hat{\phi}$, and decreasing intercept $\hat{\lambda}$. This is easily seen to be true for System 3 (and System40, despite our reservations about this data) by considering how $\hat{N} \times \hat{\phi}$ changes with n. It is also true for the other two data sets considered here, and all other Musa data sets we have analysed. We are not aware of any other explanation for such a consistent effect, and it does not seem to have been noticed by other authors.

We hope to report shortly on some recent work using the new model [21], with non-linear failure rate function. Preliminary results show that it seems to perform notably better than JM or BJM. Our hypothesis that early fixes cause greater improvements than later ones is supported by some recent empirical studies of Nagel and Skrivan [24]. This interesting work suggests that the differences in size of different faults may be surprisingly large.

References

[1] Z. Jelinski and P.B. Moranda, "Software reliability research", in Statistical Computer Performance Evaluation (W. Freiberger, ed.). New York: Academic Press, 1972, pp. 465-484.

[2] M. Shooman, "Operational testing and software reliability during program development", in Record, 1973 IEEE Symp. Computer Software Reliability, New York, NY, 1973, April 30 - May 2, pp. 51-57.

[3] M. Shooman, "Probabilistic models for software reliability and prediction", in Statistical Computer Performance Evaluation (W. Freiberger, ed.). New York: Academic Press, 1972, pp. 485-502.

[4] J.D. Musa, "A theory of software reliability and its application", IEEE Trans. Software Engineering, Vol SE-1, 1975 Sept, pp. 312-327.

[5] A.K. Goel and K. Okumoto, "Bayesian software prediction models, Vol 1: An imperfect debugging model for reliability and other quantitatine measures of software systems", RADC-TR-78-155, Rome Air Development Center, NY, 1978.

[6] A.K. Goel, "Software error detection model with applications", J. Systems and Software, 1, 1980, pp. 243-249.

[7] B. Littlewood and J.L. Verral, "A Bayesian reliability growth model for computer software", J. Royal Statist. Soc., C(Applied Statistics), 22, 1973, pp. 332-346.

[8] L.H. Crow, "Confidence interval procedures for reliability growth analysis", Tech. Report No.197, US Army Material Systems Analysis Activity, Aberdeen, Md., 1977.

[9] B. Littlewood, "How to measure software reliabilty growth and how not to", IEEE Trans. Reliability, Vol R-28, 1979 June, pp. 103-110.

[10] E.H. Forman, "Statistical models and methods for measuring software reliability", D.Sc. dissertation, SEAS, George Washington U., Washington DC, 1974.

[11] B. Littlewood and J.L. Verrall, "On the likelihood function of a debugging model for computer software reliability", IEEE Trans. Reliability, Vol R-30, 1981 June, pp. 145-148.

[12] J.D. Musa, "Software reliability data", report available from Data and Analysis Center for Software, Rome Air Development Center, Rome, NY.

[13] P.A. Keiller, "Comparison of two software reliability models", Project for professional degree, submitted to School of Engineering and Applied Science, George Washington University, Washington, DC, 1982.

[14] E.H. Forman and N.D. Singpurwalla, "An empirical stopping rule for debugging and testing computer software", J Amer. Statist. Assoc., Vol 72, 1977 Dec, pp. 750-757.

[15] National Bureau of Standards, Handbook of Mathematical Functions. Washington, DC, 1964.

[16] A. Iannino, J.D. Musa, K. Okumoto and B. Littlewood, "Criteria for software reliability model comparisons", draft available from first author: Bell Labs., Whippany, NJ07981 (1981).

[17] H. Braun and J.M. Paine, "A comparative study of models for reliability growth", Tech. Report No. 126, Series 2, Department of Statistics, Princeton University, July 1977.

[18] H. Braun and N. Schenker, "New models for reliability growth", Tech. Report No. 174, Department of Statistics, Princeton University, October 1980.

[19] D.R. Cox and P.A.W. Lewis, Statistical Analysis of Series of Events, Methuen, London 1966.

[20] B. Littlewood, "What makes a reliable program: few bugs or a small failure rate?" Proc. 1980 National Computer Conference, AFIPS Press, Arlington, VA, 1980, pp. 707-713.

[21] B. Littlewood, "Stochastic reliability growth: a model for fault-removal in computer programs and hardware designs", IEEE Trans. Reliability, Vol R-30, 4 Oct. 1981, pp. 313-320.

[22] B. Littlewood, "Theories of software reliability: how good are they and how can they be improved", IEEE Trans. Software Engineering, SE-6, 1980, pp. 489-500.

[23] M.H. De Groot, Optimal Statistical Decisions. New York: McGraw-Hill, 1970.

[24] P.M. Nagel and J.A. Skrivan, "Software reliability: repetitive run experimentation and modelling", BCS-40399, Boeing Computer Services Company, Seattle, Washington, December 1981.

Appendix

The posterior distribution of (λ,ϕ) with proper independent gamma priors is

$$p(\lambda,\phi|t_1,\ldots,t_n) =$$

$$C^*\lambda(\lambda-\phi)..(\lambda-[n-1]\phi)\exp\{-\lambda t_1-\ldots-(\lambda-[n-1]\phi)t_n\}\lambda^{b-1}e^{-c\lambda}\phi^{f-1}e^{-g\phi} \quad (A1)$$

for $\lambda \geq (n-1)\phi$ and zero otherwise, where C^* is a normalising constant.

If we transform to (x,ϕ) where $x = \lambda-(n-1)\phi$, we get

$$p(x,\phi|t_1,\ldots t_n) \propto$$

$$\prod_{i=0}^{n-1}(x+i\phi)\ [x+(n-1)\phi]^{b-1}.\ e^{-x\Sigma t_j}\ e^{-\phi\Sigma(n-j)t_j}$$

$$e^{-c(x+(n-1)\phi)}\ \phi^{f-1}\ e^{-g\phi}$$

$$= \sum_{i=0}^{n-1}\sum_{j=0}^{b-1} a_{i,n-1}\binom{b-1}{j}(n-1)^{b-j-1}$$

$$.\ x^{n+j-i}\ e^{-x[c+\Sigma t_j]}\ \phi^{b+f+i-j-2}\ e^{-\phi[g+c(n-1)+\Sigma(n-j)t_j]} \quad (A2)$$

Which is a finite mixture of distributions of the form

$$\Gamma(x;\ n+j-i+1,\ c+\Sigma t_j).\Gamma(\phi;\ b+f+i-j-1,\ g+c(n-1)+\Sigma(n-j)t_j) \quad (A3)$$

Denote by F the class of distributions which are finite mixtures like (A3): i.e. finite mixtures of $\Gamma(x;\alpha,\beta).\Gamma(\phi;\gamma,\delta)$ where α is integer. Since x can be thought of as "current failure rate" (i.e. the failure rate of the program after the n th failure but before the n th fix), the above result can be generalised and given the following interpretation. If we choose our prior for current failure rate and ϕ from F, then under the

BJM model our posterior for new current failure rate and ϕ will also be a member of F. This idea is similar to the concept of "closed under sampling" [23], and we can think of the family F as being in some way a natural conjugate to the BJM model. The prior we have used is clearly a member of F, and the above observation gives some support to our choice; it can also be used to support the particular form of "ignorance prior" used in the body of the paper.

We want

$$R_{n+1}(t|t_1 \ldots t_n) = P(T_{n+1} > t|t_1, \ldots, t_n)$$

$$= \int\int_{\lambda \geq (n-1)\phi} R_{n+1}(t|\lambda,\phi)p(\lambda,\phi|t_1, \ldots, t_n)d\lambda d\phi \qquad (A4)$$

where

$$R_{n+1}(t|\lambda,\phi) = e^{-(\lambda-n\phi)t} \quad \text{if} \quad \lambda > n\phi$$
$$= 1 \quad \text{if} \quad (n-1)\phi < \lambda < n\phi \qquad (A5)$$

So $R_{n+1}(t|t_1 \ldots t_n)$

$$= \int_{\phi=0}^{\infty} \int_{\lambda=n\phi}^{\infty} e^{-(\lambda-n\phi)t} p(\lambda,\phi|t_1, \ldots, t_n)d\lambda d\phi$$

$$+ \int_{\phi=0}^{\infty} \int_{\lambda=(n-1)\phi}^{\lambda=n\phi} p(\lambda,\phi|t_1 \ldots t_n)d\lambda d\phi \qquad (A6)$$

The first integral in (A6) is

$$C^* \int_{\phi=0}^{\infty} \phi^{f-1}e^{-g\phi} \int_{\lambda=n\phi}^{\infty} e^{-(\lambda-n\phi)t} \lambda(\lambda-\phi)\ldots(\lambda-[n-1]\phi)$$

$$\exp\{-\lambda t_1 - \ldots - (\lambda-[n-1]\phi)t_n\}.\lambda^{b-1}e^{-c\lambda} \, d\lambda d\phi$$

Putting $x = \lambda - n\phi$ this becomes

$$C^* \int_{\phi=0}^{\infty} \phi^{f-1} e^{-g\phi} \int_{x=0}^{\infty} \prod_{i=1}^{n} (x+i\phi) e^{-(t+\Sigma t_j)x} e^{-\Sigma(n-j+1)t_j\phi} (x+n\phi)^{b-1} e^{-cx} e^{-cn\phi} \, dx d\phi$$

$$= C^* \int_{\phi=0}^{\infty} \phi^{f-1} e^{-g\phi} \int_{x=0}^{\infty} \sum_{i=0}^{n} \sum_{j=0}^{b-1} a_{in} x^{n-i} \phi^i \binom{b-1}{j} x^j (n\phi)^{b-j-1} e^{-x(t+\Sigma t_j)} e^{-\phi\Sigma(n-j+1)t_j} e^{-cx} e^{-cn\phi} \, dx d\phi$$

(Notice that it is at this stage that we need b to be an integer)

$$= C^* \sum_{i=0}^{n} \sum_{j=0}^{b-1} a_{in} \binom{b-1}{j} n^{b-j-1} \int_{\phi=0}^{\infty} \phi^{b+f+i-j-2} e^{-\phi(g+nc+\Sigma(n-j+1)t_j)} \int_{x=0}^{\infty} x^{n-i+j} e^{-x(c+t+\Sigma t_j)} \, dx \, d\phi$$

$$= C^* \sum_{i=0}^{n} \sum_{j=0}^{b-1} a_{in} \binom{b-1}{j} n^{b-j-1} \frac{\Gamma(n-i+j+1)\Gamma(b+f+i-j-1)}{(c+t+\Sigma t_j)^{n-i+j+1} (g+nc+\Sigma(n-j+1)t_j)^{b+f+i-j-1}} \tag{A7}$$

The second integral in (A6) can be expressed as a difference of two integrals, of which one is

$$C^* \int_{\phi=0}^{\infty} \int_{\lambda=n\phi}^{\infty} \lambda(\lambda-\phi)\ldots(\lambda-[n-1]\phi)\exp\{-\lambda t_1 - \ldots - (\lambda-[n-1]\phi)t_n\} \lambda^{b-1} e^{-c\lambda} \phi^{f-1} e^{-g\phi} \, d\lambda d\phi$$

Which is simply (A7) with $t = 0$:

$$C^* \sum_{i=0}^{n} \sum_{j=0}^{b-1} a_{in} \binom{b-1}{j} n^{b-j-1} \frac{\Gamma(n-i+j+1)\Gamma(b+f+i-j-1)}{(c+\Sigma t_j)^{n-i+j+1} (g+nc+\Sigma(n-j+1)t_j)^{b+f+i-j-1}} \tag{A8}$$

The other integral needed for the second term of (A6) is

$$C^* \int_{\phi=0}^{\infty} \int_{\lambda=(n-1)\phi}^{\infty} \lambda(\lambda-\phi)\ldots(\lambda-[n-1]\phi)\exp\{-\lambda t_1 -\ldots-(\lambda-[n-1]\phi)t_n\}$$

$$\cdot \lambda^{b-1} e^{-c\lambda} \phi^{f-1} e^{-g\phi} d\lambda d\phi$$

Put $\lambda-[n-1]\phi = x$ and we have

$$C^* \int_{\phi=0}^{\infty} \int_{x=0}^{\infty} \prod_{i=0}^{n-1} (x+i\phi).(x+[n-1]\phi)^{b-1} e^{-x\Sigma t_j}$$

$$e^{-\phi\Sigma(n-j)t_j} e^{-c(x+[n-1]\phi)} \phi^{f-1} e^{-g\phi} d\lambda d\phi$$

$$= C^* \int_{\phi=0}^{\infty} \int_{x=0}^{\infty} \sum_{i=0}^{n-1} \sum_{j=0}^{b-1} a_{i,n-1} x^{n-i} \phi^i \binom{b-1}{j} x^j$$

$$e^{-\phi\Sigma(n-j)t_j([n-1]\phi)^{b-j-1} e^{-x\Sigma t_j} e^{-c(x+[n-1]\phi)} \phi^{f-1} e^{-g\phi}} d\lambda\, d\phi$$

$$= C^* \sum_{i=0}^{n-1} \sum_{j=0}^{b-1} a_{i,n-1} \binom{b-1}{j} (n-1)^{b-j-1}$$

$$\int_{\phi=0}^{\infty} \phi^{b+f+i-j-2} e^{-\phi(g+c(n-1)+\Sigma(n-j)t_j)}$$

$$\int_{x=0}^{\infty} x^{n+j-i} e^{-x(c+\Sigma t_j)} dx\, d\phi$$

$$= C^* \sum_{i=0}^{n-1} \sum_{j=0}^{b-1} a_{i,n-1} \binom{b-1}{j} (n-1)^{b-j-1} \frac{\Gamma(n-i+j+1)\Gamma(b+f+i-j-1)}{(c+\Sigma t_j)^{n-i+j+1}(g+(n-1)c+\Sigma(n-j+1)t_j)^{b+f+i-j-1}}$$

Finally, we have from (A6) (A9)

$$R_{n+1}(t|t_1,\ldots t_n) = (A5) + (A7) - (A6)$$

and

$$C^{*-1} = R_{n+1}(0|t_1,\ldots t_n)$$

$$= (A9).$$

It is easy to see that we obtain the improper uniform prior result, (17), when we let $f = b = 1$ and $c, g \to 0$.

The probability that the program is now "perfect", i.e. that the last error has been removed, is

$$P_0 \equiv R_{n+1}(\infty | t_1, \ldots t_n) = (A9) - (A8)$$

$$= 1 - \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{b-1} a_{i,n-1} \binom{b-1}{j} \frac{(n-1)^{b-j-1} \Gamma(n-i+j+1)\Gamma(b+f+i-j-1)}{(c+\Sigma t_j)^{n-i+j+1}(g+(n-1)c+\Sigma(n-j+1)t_j)^{b+f+i-j-1}}}{\sum_{i=0}^{n-1} \sum_{j=0}^{b-1} a_{i,n} \binom{b-1}{j} \frac{n^{b-j-1} \Gamma(n-i+j+1)\Gamma(b+f+i-j-1)}{(c+\Sigma t_j)^{n-i+j+1}(g+nc+\Sigma(n-j+1)t_j)^{b+f+i-j-1}}} \qquad (A10)$$

These expressions do not present insuperable computational difficulties. The main problem is one of eliciting the prior information in the form of the numbers b(integer), c, f and g. There are various ways in which, in principle, "your" prior beliefs about, say, λ could be elicited within the gamma distribution framework. You could be asked to give your best guess of the mean and variance of your beliefs about λ. Such an approach does not seem to represent how we "naturally" think about uncertainty. An alternative approach would be to ask "you" to fix two percentiles of λ, say the 25% and 75% points. From either of these approaches it is a simple matter to calculate "your" b and c. A more satisfactory approach might involve a certain amount of feedback, with "you" being able to see the consequences of your choices of b and c and modify them. This problem is one which is central to Bayesian inference, and it is not appropriate to dwell on it at length here.

Captions

Table 1 : Analysis of Musa System 3 data. Here n represents number of observation, and sample size in ML calculations and Bayesian analysis; t_n is n th inter-failure time measured in seconds. $\hat{N},\hat{\phi}$ are ML estimates of N,ϕ in JM model based on n observations. $\hat{F}_{n+1}(t_{n+1})$ are the probability integral transforms of t_{n+1} using the predictor distributions based on $t_1, \ldots t_n$ (see (40) for JM and (41) for BJM). P_0 represents the probability that the program is perfect, i.e. the last fault has been removed, for the BJM model.

Table 2: Same structure as Table 1, for Musa System 40 data.

Figure 1: Failure rate versus failure number for JM model: $\lambda_i = (N - i + 1)\phi$.

Figure 2 : Dots represent failure rate versus failure number as we suggest it ought to be: i.e. early fixes have greater effect than later ones. Crosses represent "best fitting" JM linear failure rate function.

Plot 1 Procedure 1 for Musa System 3 data, sample sizes here range from 18 through 37; JM model is represented by crosses, BJM by dots.

Plot 2 Procedure 2 for Musa System 3 data, same sample size range as in Plot 1. Again JM represented by crosses, BJM by dots. For clarity the actual step-function sample cdf's are shown.

Plot 3 As Plot 1, for Musa System 40 data; sample sizes 51 through 100.

Plot 4 As Plot 2, same data as for Plot 3.

Plot 5 As Plot 1, for Musa System 1 data; sample sizes 30 through 129.

Plot 6 As Plot 2, same data as for Plot 5. Here BJM and JM are extremely close and only JM is shown. The line with smallest slope shows the closeness to linearity of points 34 through 90 (see text).

Plot 7 As Plot 1, for Musa System 2 data; sample sizes 14 through 53.

Plot 8 As Plot 2, same data as for Plot 7.

Table 1

		JM model			BJM model	
n	t_n	$\hat{N}$	$\hat{\phi}$	$\hat{F}_{n+1}(t_{n+1})$	$\hat{F}_{n+1}(t_{n+1})$	P_o
1	115					
2	0	∞	0.		.504	.250
3	83	∞	0.		.585	.246
4	178	5	.0034		.431	.342
5	194	6	.0026		.326	.314
6	136	8	.0017		.728	.200
7	1077	7	.0018	0.	.007	.688
8	15	8	.0014	0.	.023	.289
9	15	12	.00070	.175	.204	.107
10	92	19	.00038	.156	.147	.049
11	50	55	.00011	.297	.237	.020
12	71	∞	0.	.097	.863	.009
13	606	22	.00033	.970	.868	.030
14	1189	15	.00059	.023	.050	.139
15	40	18	.00043	.634	.649	.058
16	788	18	.00043	.172	.243	.086
17	222	21	.00033	.089	.107	.048
18	72	25	.00025	.655	.634	.021
19	615	25	.00025	.638	.575	.025
20	589	25	.00025	.018	.023	.025
21	15	31	.00018	.501	.497	.010
22	390	33	.00016	.965	.904	.008
23	1863	26	.00024	.617	.636	.045
24	1337	26	.00024	.885	.804	.070
25	4508	25	.00026	0.	.146	.318
26	834	26	.00024	0.	.437	.238
27	3400	27	.00021	0.	.001	.338
28	6	28	.00019	0.	.498	.201
29	4561	29	.00017	0.	.269	.329
30	3186	30	.00015	0.	.441	.345
31	10571	31	.00013	0.	.017	.623
32	563	32	.00012	0.	.120	.465
33	2770	33	.00011	0.	.038	.400
34	652	34	.000097	0.	.305	.281
35	5593	35	.000088	0.	.390	.317
36	11696	36	.000079	0.	.163	.479
37	6724	37	.000071	0.	.069	.475
38	2546	38	.000065	0.		.381

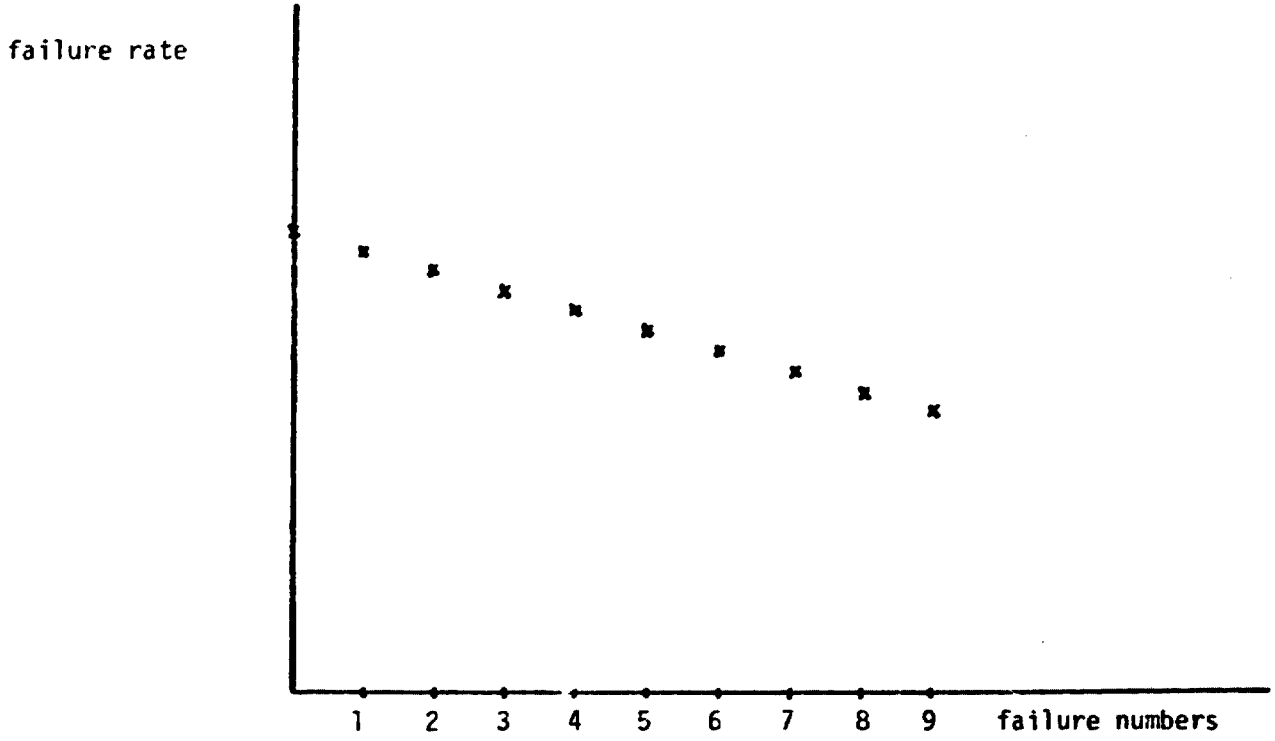
Table 2

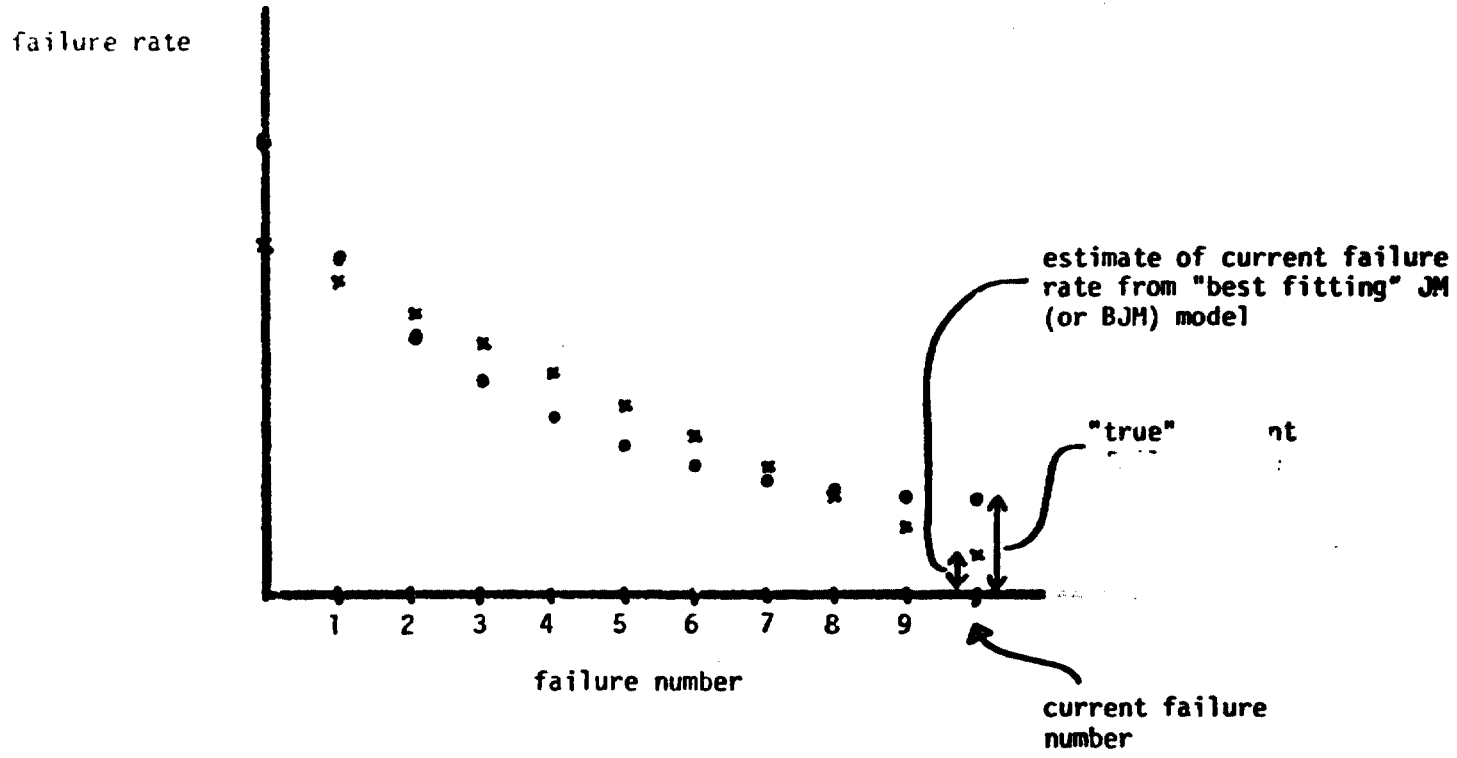
		JM model			BJM model	
n	t_n	$\hat{N}$	$\hat{\phi}$	$\hat{F}_{n+1}(t_{n+1})$	$\hat{F}_{n+1}(t_{n+1})$	P_o
1	320					
2	14390	2	.13E-03		.002	.996
3	9000	3	.77E-04		.106	.554
4	2880	31	.51E-05		.348	.213
5	5700	∞	0.		.706	.128
6	21800	7	.29E-04		.537	.246
7	26800	7	.30E-04		.612	.307
8	113540	8	.19E-04		.143	.767
9	112137	9	.12E-04		.001	.751
10	660	10	.96E-05		.021	.370
11	2700	12	.66E-05		.300	.154
12	28793	14	.51E-05		.035	.095
13	2173	20	.29E-05		.146	.035
14	7263	31	.17E-05		.244	.015
15	10865	57	.84E-06		.117	.007
16	4230	∞	0.		.243	.003
17	8460	∞	0.		.405	.001
18	14805	∞	0.		.353	0.
19	11844	∞	0.		.189	0.
20	5361	∞	0.		.238	0.
21	6553	∞	0.		.248	0.
22	6499	∞	0.		.134	0.
23	3124	∞	0.		.898	0.
24	51323	∞	0.		.511	0.
25	17010	∞	0.		.079	0.
26	1890	∞	0.		.220	0.
27	5400	∞	0.		.934	0.
28	62312	∞	0.		.631	0.
29	24826	∞	0.		.647	0.
30	26335	∞	0.		.015	0.
31	363	∞	0.		.446	0.
32	13989	∞	0.		.476	0.
33	15058	∞	0.		.746	0.
34	32377	∞	0.		.813	0.
35	41632	∞	0.		.154	0.
36	4160	∞	0.		.954	0.
37	82040	132	.41E-06		.368	0.
38	13189	170	.31E-06		.117	0.
39	3426	357	.14E-06		.200	0.
40	5833	9496	.51E-08		.025	0.
41	640	∞	0.		.027	0.
42	640	∞	0.		.119	0.
43	2880	∞	0.		.005	0.
44	110	∞	0.		.642	0.
45	22080	∞	0.		.932	0.
46	60654	∞	0.		.882	0.
47	52163	845	.60E-07		.395	0.
48	12546	6312	.78E-08		.032	0.
49	784	∞	0.		.353	0.
50	10193	∞	0.		.290	0.
51	7841	∞	0.	.800	.745	0.
52	31365	∞	0.	.709	.648	0.

Table 2 continued

53	24313	∞	0.	.999	1.	0.
54	298890	78	.86E-06	.026	.028	0.
55	1280	85	.76E-06	.397	.407	0.
56	22099	89	.72E-06	.364	.372	0.
57	19150	94	.67E-06	.062	.065	0.
58	2611	104	.58E-06	.649	.643	0.
59	39170	104	.58E-06	.767	.754	0.
60	55794	99	.62E-06	.644	.641	0.
61	42632	99	.62E-06	.998	.992	0.
62	267600	76	.92E-06	.676	.677	0.
63	87074	76	.92E-06	.833	.811	0.
64	149606	74	.97E-06	.130	.140	0.001
65	14400	76	.92E-06	.296	.317	0.
66	34560	78	.88E-06	.342	.362	0.
67	39600	80	.84E-06	.974	.950	0.
68	334395	75	.95E-06	.859	.916	0.003
69	296015	73	.10E-05	.511	.542	0.013
70	177355	74	.97E-06	.566	.552	0.018
71	214622	74	.98E-06	.367	.403	0.027
72	156400	75	.94E-06	.375	.398	0.030
73	166800	76	.91E-06	.028	.033	0.033
74	10800	77	.88E-06	.506	.553	0.021
75	267000	78	.85E-06	.995	.911	0.033
76	2098833	76	.91E-06	0.	.164	0.471
77	614080	77	.88E-06	0.	.002	0.528
78	7680	78	.81E-06	0.	.376	0.435
79	2629667	79	.75E-06	0.	.094	0.804
80	2948700	80	.68E-06	0.	.002	0.937
81	187200	81	.63E-06	0.	0.	0.897
82	18000	82	.58E-06	0.	.006	0.834
83	178200	83	.54E-06	0.	.023	0.772
84	487800	84	.50E-06	0.	.035	0.734
85	639200	85	.47E-06	0.	.021	0.708
86	334560	86	.44E-06	0.	.092	0.653
87	1468800	87	.41E-06	0.	.005	0.700
88	86720	88	.39E-06	0.	.016	0.620
89	199200	89	.37E-06	0.	.022	0.551
90	215200	90	.35E-06	0.	.011	0.485
91	86400	91	.33E-06	0.	.015	0.409
92	88640	92	.32E-06	0.	.249	0.340
93	1814400	93	.30E-06	0.	.001	0.431
94	4160	94	.29E-06	0.	.001	0.354
95	3200	95	.28E-06	0.	.042	0.284
96	199200	96	.27E-06	0.	.082	0.237
97	356160	97	.26E-06	0.	.123	0.207
98	518400	98	.25E-06	0.	.089	0.190
99	345600	99	.24E-06	0.	.009	0.164
100	31360	101	.22E-06	.056	.086	0.125
101	265600	102	.21E-06		.361	0.103

Fig





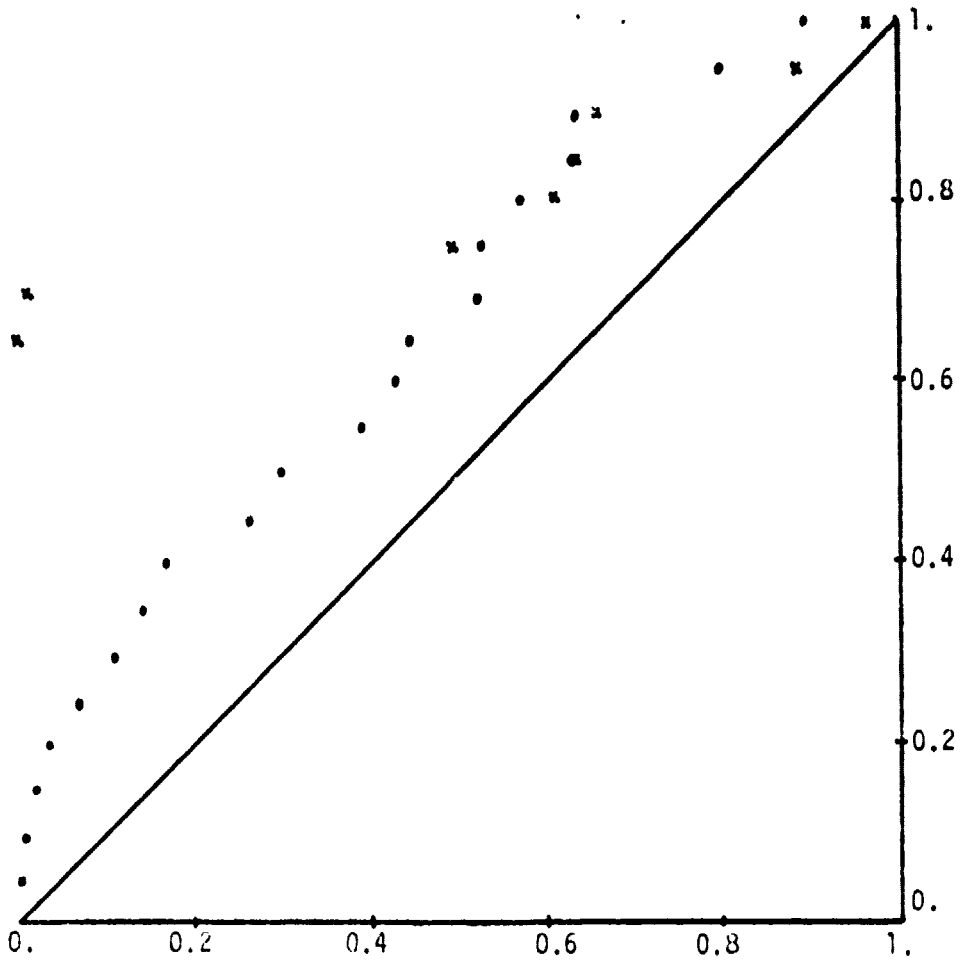
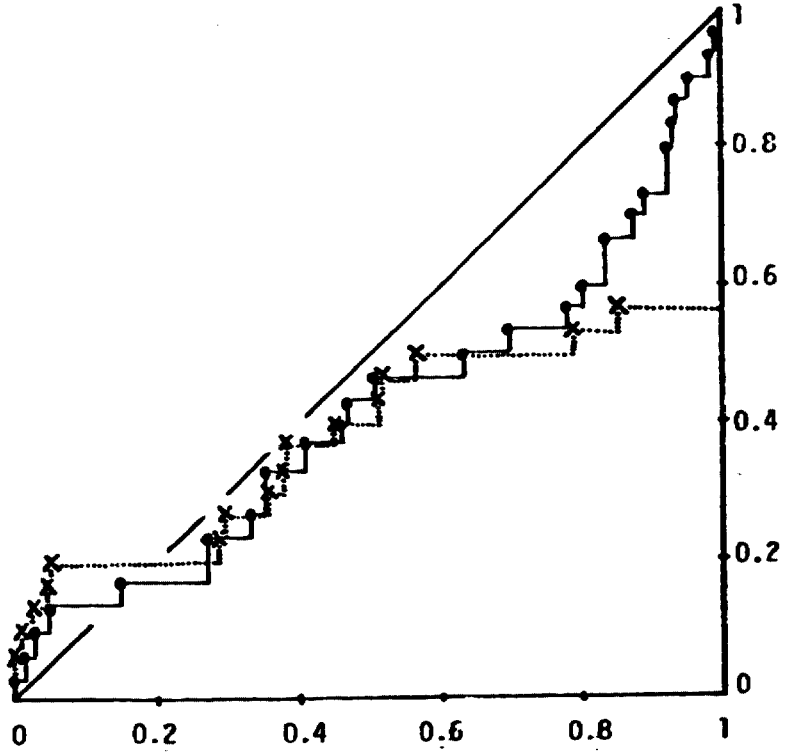
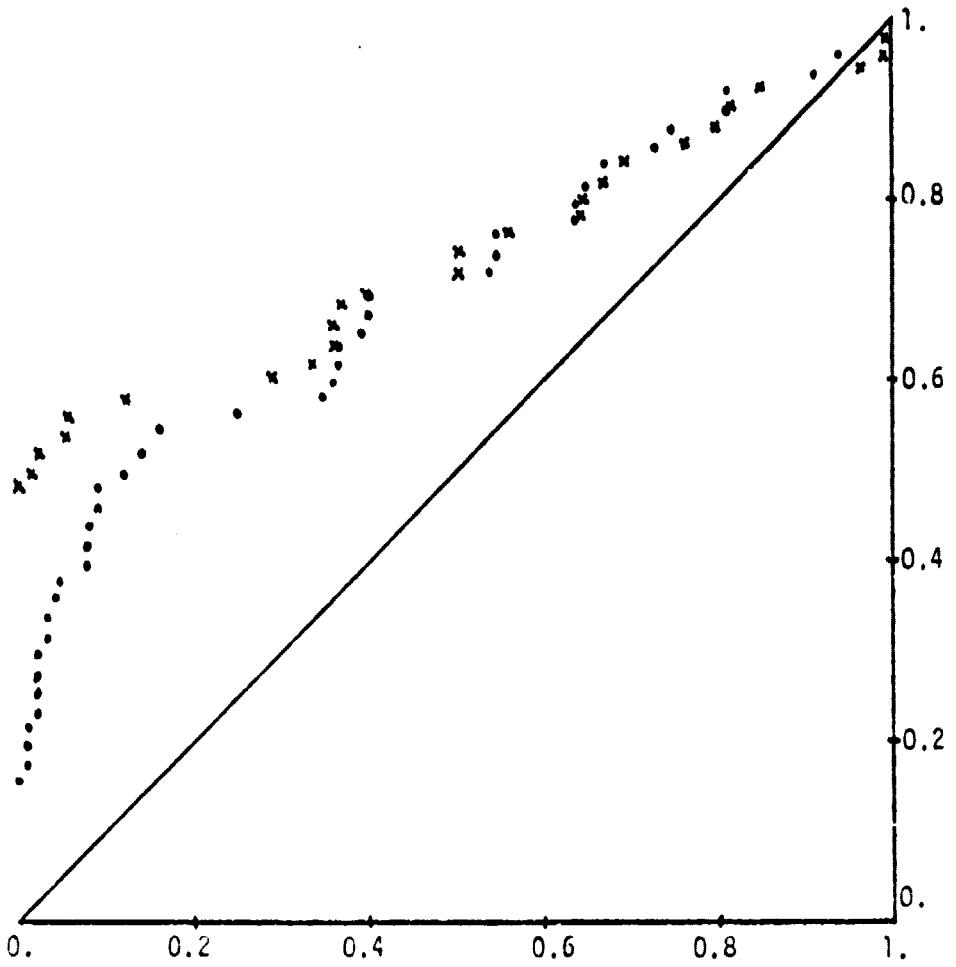
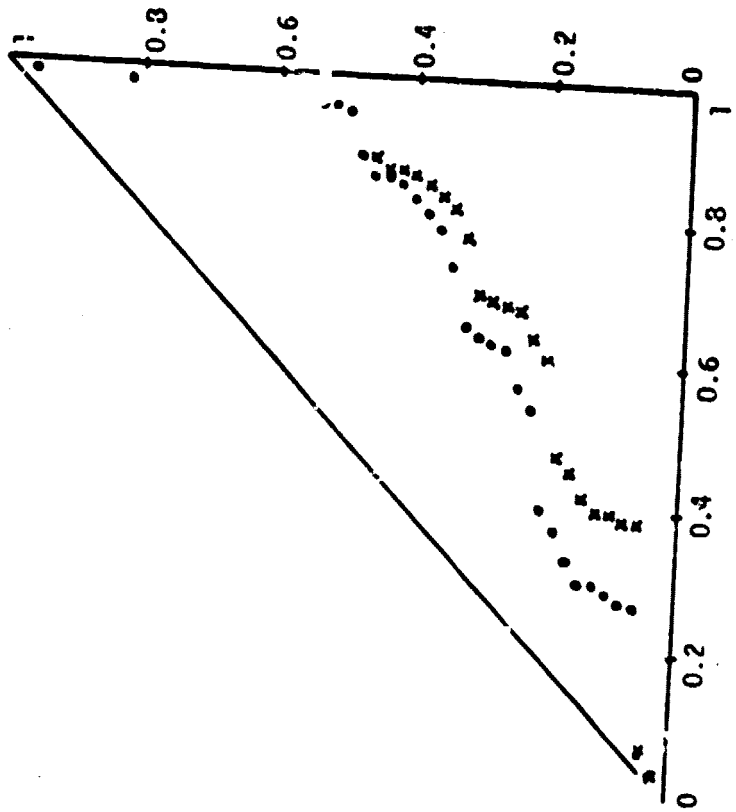


Fig 2



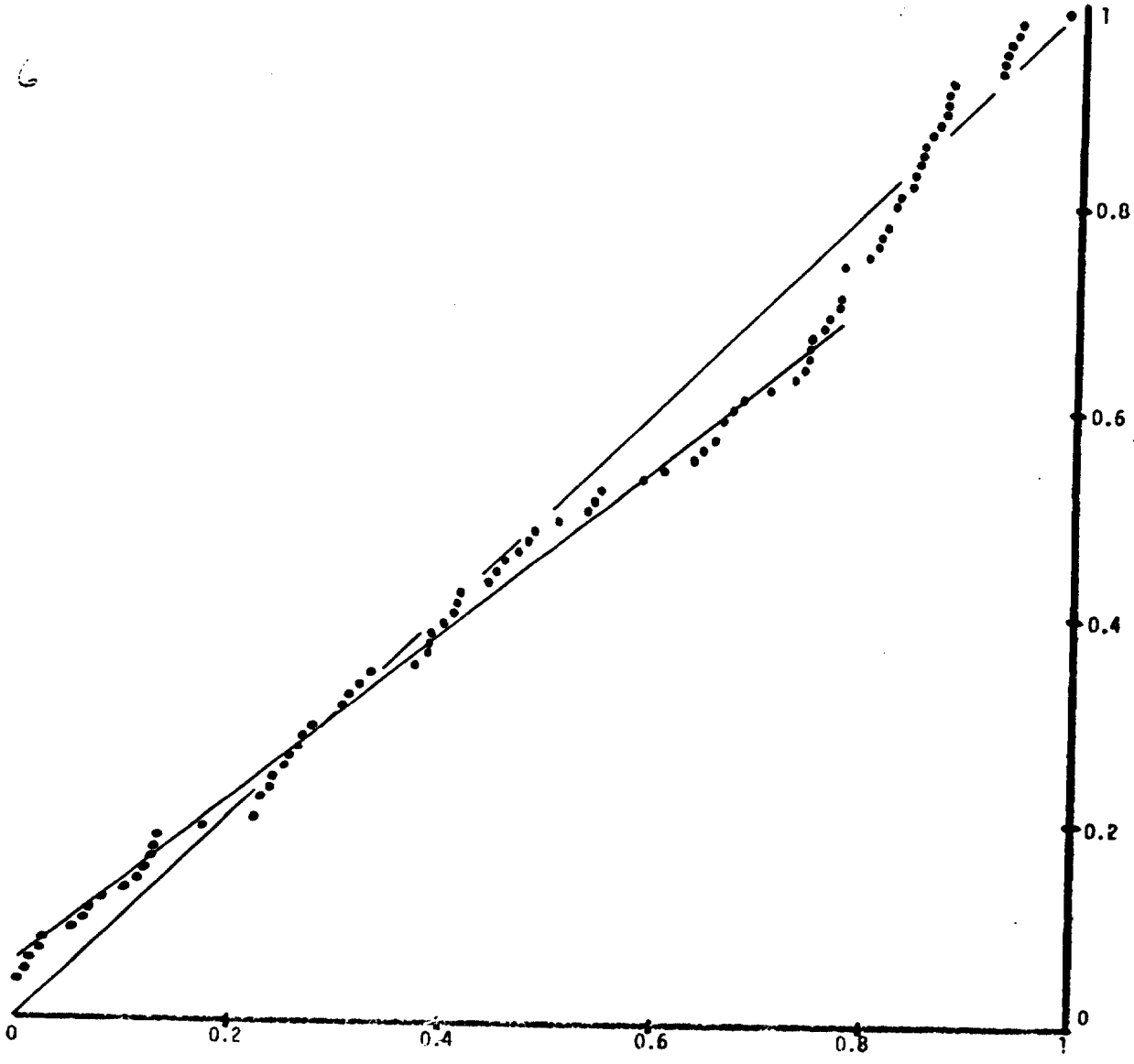


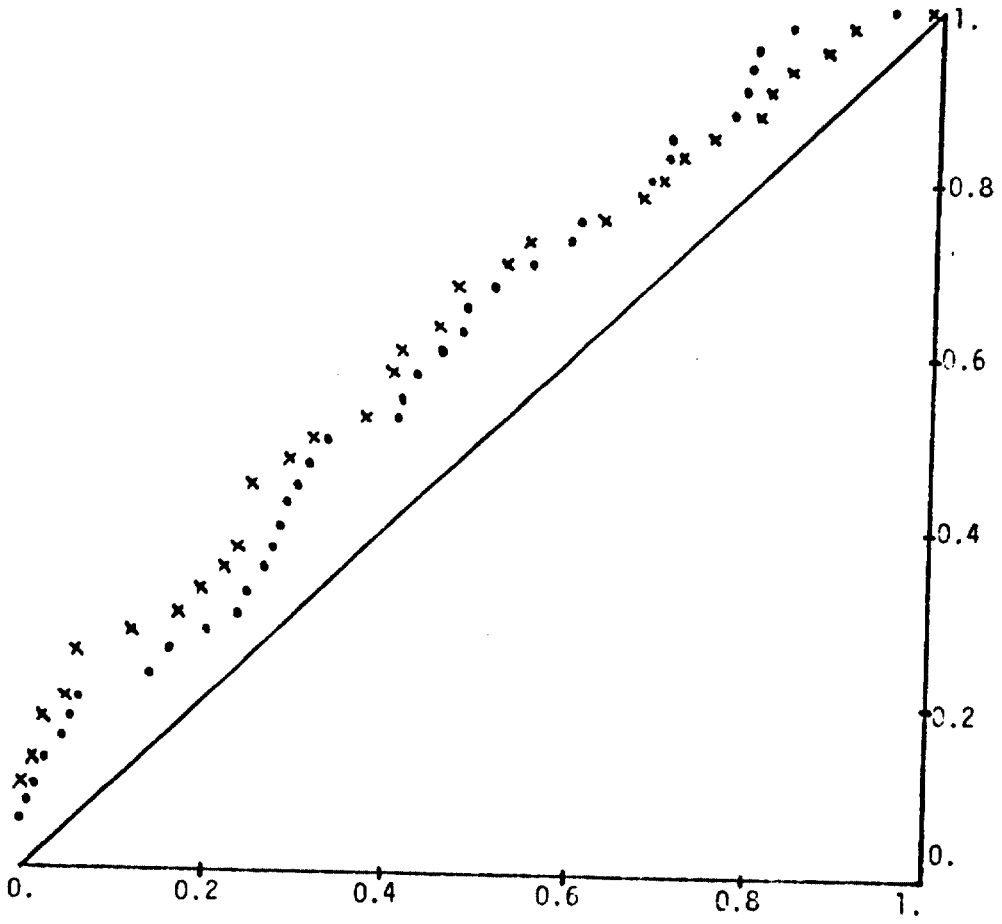


Plot 5

ORIGINAL PAGE 13
OF POOR QUALITY

Plot 6





Plot 9