SOFTWARE RELIABILITY PREDICTION

by

Pee Yuaw CHAN

Thesis Submitted for
the Degree of Doctor of Philosophy
in Statistics and Computer Science

DEPARTMENT OF MATHEMATICS

THE CITY UNIVERSITY

LONDON

December 1986

DAMAGED ORIGINAL

To my wife Christine
who was constantly a victim of
my moods and absent-mindedness
for the past one and a half years,
and had to spend two months
on her own during the preparation
of this thesis

CONTENTS

ACKNOWLEDGEMENTS

This research was done while I was a research assistant at the Centre for Software Reliability, the City University, London. I would like to express my gratitude to:

Professor B. Littlewood for his supervision.

ICL plc for financing the research project.

numerous staff members of the Mathematics Department and the Computer Science Department at The City University, in particular Mr.J.Snell for his involvement and assistance.

Dr.M.G.Cox at the National Physical Laboratory (NPL) for his valuable advice, and the Publications Officer at NPL for supplying me with a number of useful reports.

Professor D.R. Miller and Dr.A. Sofer for the use of their software.

the staff of the Computer Unit at City University.

Ms.G. Palmer for typing the title of the graphs and plots in the thesis.

Mrs.D. Carmen for her efficient and skillful typing of this thesis, and her patience in making numerous changes from the draft version to its present form.

my parents, brother, sisters, and my wife for their understanding and support.

Mr. and Mrs.P. Man for their help back in 1976 which made all this possible.

ABSTRACT

Two methods are proposed to find the maximum likelihood parameter estimates of a number of software reliability models. On the basis of the results from analysing 7 sets of real data, these methods are found to be both efficient and reliable.

The simple approach of adapting software reliability predictions by Keiller and Littlewood (1984) can produce improved predictions, but at the same time, introduces a lot of internal noise into the adapted predictions. This is due to the fact that the adaptor is a joined-up function.

An alternative adaptive procedure, which involves the use of a parametric spline adaptor, can produce at least as good adapted predictions without the predictions being contaminated by internal noise as in the simple approach.

Miller and Sofer (1986a) proposed a method for estimating the failure rate of a program non-parametrically. Here, these non-parametric rates are used to produce reliability predictions and their quality is analysed and compared with the parametric predictions.

KEY TO ABBREVIATIONS

BJM(*)	The (adapted) Bayesian Jelinski and Moranda prediction system
cdf	Cumulative distribution function
dDIF(*)	The (adapted) prediction system with exponential failure time and monotone rate estimates with d difference constraints
DU(*)	The (adapted) Duane prediction system
GO(*)	The (adapted) Goel and Okumoto prediction system
GP	The generalised power law NHPP prediction system
HPP	Homogeneous Poisson process
JM(*)	The (adapted) Jelinski and Moranda prediction system
KL(*)	The (adapted) Keiller and Littlewood prediction system
KS	Kolmogorov and Smirnov
L(*)	The (adapted) Littlewood prediction system
LNHPP(*)	The (adapted) Littlewood NHPP prediction system
LV(*)	The (adapted) Littlewood and Verrall prediction system
ML	Maximum likelihood
MLE	Maximum likelihood estimate
MNA	Modified Newton's algorithm
MO(*)	The (adapted) Musa and Okumoto prediction system
NHPP	Non-homogeneous Poisson process
pdf	Probability density function
PL	Prequential likelihood
PLR	Prequential likelihood ratio
ROCOF	Rate of occurrence of failures
SQAMA	The safeguarded quadratic approximation minimisation algorithm
w.r.t.	With respect to

CHAPTER 1

INTRODUCTION

Complex software systems contain design errors which may or may not manifest themselves by causing failures during execution. A failure occurs when there is a discrepancy between the output of the program and its specification. When this happens, an attempt will usually be made to identify the source of the failure and remove it. This process of debugging, if successful, would lead to an improvement in the reliability of the program. The main objective of this work is to measure the reliability of software systems undergoing debugging. This would be of interest to a software developer who wants to know whether a program is reliable enough for marketing or more development effort is needed before reaching that stage. It is also of interest to a user who wants to know whether the reliability of a program has reached the level required for his particular application. This is of particular importance if the software would be used to control or monitor systems where an operational failure would have disastrous consequences.

Software reliability evaluation methodologies developed up to the present are built on the foundation that the underlying process governing the failure behaviour of a software is random. An account for the randomness in software failures is given by Laprie (1984). While software reliability modellers might disagree on the sources of such randomness, there is general acceptance that the reliability of a program can only be meaningfully represented in terms of probability. The definition which is

commonly used is for the operational reliability of a program, which is the probability of successful execution of the program without failure for a specified length of time in a specified environment.

Here we will concentrate on data of the form $t_1,t_2,...,t_i$ of execution times between successive failures of a program undergoing debugging, and methods which utilise this information to predict the unobserved random quantities $T_{i+1},T_{i+2}....$. In particular, we focus our attention on estimating the current reliability, i.e. characteristics of the random variables T_{i+1}; other unobserved quantities can be dealt with in a similar way. Note that this estimation problem is one of prediction because it concerns the unobserved quantity T_{i+1}. In principle, what we need is the probability distribution of the random variable T_{i+1}, and the problem is effectively solved if we can accurately estimate this distribution.

In order to achieve the above objective, a prediction system has to be used. Such a system will allow us to predict the future (T_{i+1}) from the past ($t_1,t_2,....t_i$). It consists of the following:

1. A mathematical model detailing the behaviour of the random variable $\{T_1,T_2,...,T_i\}$ for all i, conditional on some unknown parameters.
2. A statistical inference procedure for estimating these unknown parameters using observed data.
3. A prediction procedure combining (1) and (2) to allow us to predict.

A survey by Dale and Harris (1981) has shown that there are now more than 40 software reliability models in existence. Many such models with a particular choice of (2) and (3) forms a prediction system which could be used for the purpose of reliability measurement.

However, not merely do we want to have a probabilistic profile of the unobserved time to failure, we want one which is close to reality. In order to achieve this, it is essential to have a good model (1) in the prediction system and at the same time, parts (2) and (3) are also of vital importance. Viewed from this angle, the usual discussion of competing software reliability models becomes inadequate. We should, instead, be comparing the relative merits of different prediction systems rather than just the model alone: the success or failure of a prediction system is a result of (1), (2) and (3) jointly.

As far as selecting a model is concerned, it is not possible to decide, a *priori*, the best model in any given context. Although one might argue that some models are more suitable because of their realistic assumptions, this still leaves us those which we cannot reject on the grounds of being unrealistic. Indeed, the knowledge in this aspect of software engineering is so imperfect that it is not possible to identify the best model given all the characteristics of the software concerned. Our approach is to employ many prediction systems simultaneously and to select the best prediction on the basis of the past predictive quality of each individual system on the actual data set under investigation.

A common characteristic shared by the majority of software reliability models is that it is very hard to carry out a full Bayesian analysis on the unknown parameters. The method of maximum likelihood is usually used in part (2). By this method, a set of parameter values has to be determined such that the likelihood function is maximised. In practice, this constitutes the bulk of the numerical work that has to be done in the whole prediction process. The amount of work involved varies from model to model. In a few cases, it is small because the problem is straight-forward and can be solved easily, but in the remaining cases, it is considerable because the maximisation is by no means trivial and can only be done by numerical search. With our multi-prediction systems approach, in particular, the requirement on computing resources can be a problem. In view of this, there is a definite need to develop efficient numerical algorithms in order to save computing time.

Chapter 2 describes two numerical algorithms for unconstrained optimisation. These methods are chosen for their efficiency and well proven success with many practical problems.

Chapter 3 outlines the actual implementation of these algorithms to 7 of the 9 software reliability models included in this study. Difficulties encountered in analysing real data sets make it necessary to refine these algorithms. Full details of the numerical experience in analysing 7 real data sets are given.

Having obtained the parameter estimates, the respective prediction systems can proceed to predict. Our next step is to analyse the predictive quality of each of these prediction systems. In Chapter 4 we present the tools that would be used to evaluate the predictive performance, namely, the u- and y-plot procedures, the prequential likelihood, and the median plot.

An important by-product of our analysis of predictive quality is that the u-plot can be used to recalibrate our future predictions. Keiller and Littlewood (1984) have reported some success, on the basis of the u-plot and y-plot criteria when they adapt future predictions by joining up the u-plot and use it as the calibration curve. However, when we investigate further using prequential likelihood as the criterion of success, the adapted predictions are not always an improvement. The disagreement with the other criteria stems from the fact that the adaptive curve is a joined-up function, and it can be resolved by using a smooth function instead.

The smooth function we have used is a parametric spline. In Chapter 5, the parametric spline is defined in terms of B-splines, and a numerically stable and efficient method for its determination is described. Extensive analyses are also given when we apply this method of adapting to 9 models and 7 sets of data.

One of the strongest criticisms of software reliability models is that they are highly parameterised. Miller and Sofer (1986a) proposed a

non-parametric approach to estimate the failure rate of a program. In Chapter 6 we shall analyse the quality of the predictions based on rates estimated by this non-parametric method and exponential failure time distribution. These predictions are adapted using our parametric spline adaptor and we compare all the results including those in Chapter 5. An alternative non-parametric rate estimation procedure is given in Appendix 2.

The final Chapter is devoted to discussions and future research possibilities, thus concluding this thesis.

CHAPTER 2

TWO METHODS FOR UNCONSTRAINED MINIMISATION

2.1. INTRODUCTION

In this Chapter we shall describe two numerical optimisation methods for the efficient determination of the maximum likelihood estimate (MLE) of the unknown parameters in the software reliability models included in this study. Both methods are for unconstrained optimisation. The first is for optimising functions in one variable. This method, which is due to Gill and Murray (1974), uses the value of the objective function only. The second is for functions in more than one variable and is also due to Gill and Murray (1972a, 1972b); extensive results on solving many test functions and an Algol implementation can be found in the cited publications. Unlike the univariate optimisation method, the latter requires both the gradient and the value of the objective function. This is because the multivariate problems are more difficult, at least in the cases considered here. With the extra gradient information the problem can be solved much more efficiently. The implementation details of these techniques are in Chapter 3.

2.2. THE NEED OF AN EFFICIENT ALGORITHM FOR PARAMETER ESTIMATION

Although all the three components in a prediction system are important to the prediction process as a whole, the attention so far being given to various aspects concerning the related problem in the statistical inference might give the impression that it is usually simple and straight-forward. However, in our experience, this is far from being the case.

We believe that a user of any software reliability models should adopt a Bayesian inference procedure. This involves updating the prior distribution, via the likelihood of the data, to arrive at the posterior distribution of the parameters. This posterior distribution is then used to generate a predictive distribution. A good account of this approach in the context of conventional statistical problems is given in the book by Aitchison and Dunsmore (1975).

In practice, it is often not possible to carry out a full Bayesian analysis with most models and the method of maximum likelihood (ML) is commonly used instead. When the maximum of the likelihood function cannot be found analytically, as is usually the case, numerical optimisation technique has to be employed. The sequential nature in which the inference has to be repeated means that the requirement on computer time can be substantial, if an inefficient numerical algorithm is being used. This would be compounded if one were to use many prediction systems simultaneously in order to select the most appropriate one.

With the advancement of micro-computer technology, it is now possible for a personal computer to carry out mathematical calculations with accuracy comparable to that of a mainframe computer. Therefore, a step towards alleviating the computer resource problem is to implement the analysis program on a personal computer. This is obviously much more affordable even if the personal computer has to be dedicated entirely for this sole purpose. However, it does mean that we become even more dependent on the availability of a fast algorithm. For example, if one

were to carry out a simulation study on the performance of several prediction systems, the wastage in using an inefficient method will preclude the analysis of more replicates.

Henceforth, we shall only address the problem of minimisation. The equivalent maximisation problem merely involves a change of sign in the objective function. f(x) will be used to denote the objective evaluated at x. The variable x is either a scalar or a vector depending on whether the problem is univariate or multi-variate. In the latter, g(x) is used to denote the vector of partial derivatives of the objective and G(x) denotes the Hessian matrix.

2.3. THE UNIVARIATE MINIMISATION METHOD

This is a hybrid technique which combines two univariate minimisation methods, successive quadratic approximation and function-comparison, in such a way that it is has the speed of the former and the reliability of the latter. Like most univariate minimisation methods, it utilises the concept of an interval of uncertainty, i.e. an interval [a,b], which while the minimum is known to lie within it, we are uncertain as to where exactly it is. The value of such an interval derives from the fact that any estimate lying within it will not be more than the length of the interval away from the true minimum. If the function is unimodal this minimum is global otherwise it can also be a local minimum.

Given an initial interval of uncertainty, different methods adopt different strategies to progressively reduce the size of this interval until it is sufficiently small. To illustrate how this is done iteratively, we give an example where only function values are involved. Let [a,b] be the current interval of uncertainty, x is the best point yet, w the previous x value and v is the highest point of the three. The labelling of w and v has no significance here but it would be necessary to do so in the method we propose. Figure 2.1. shows how they are configurated.

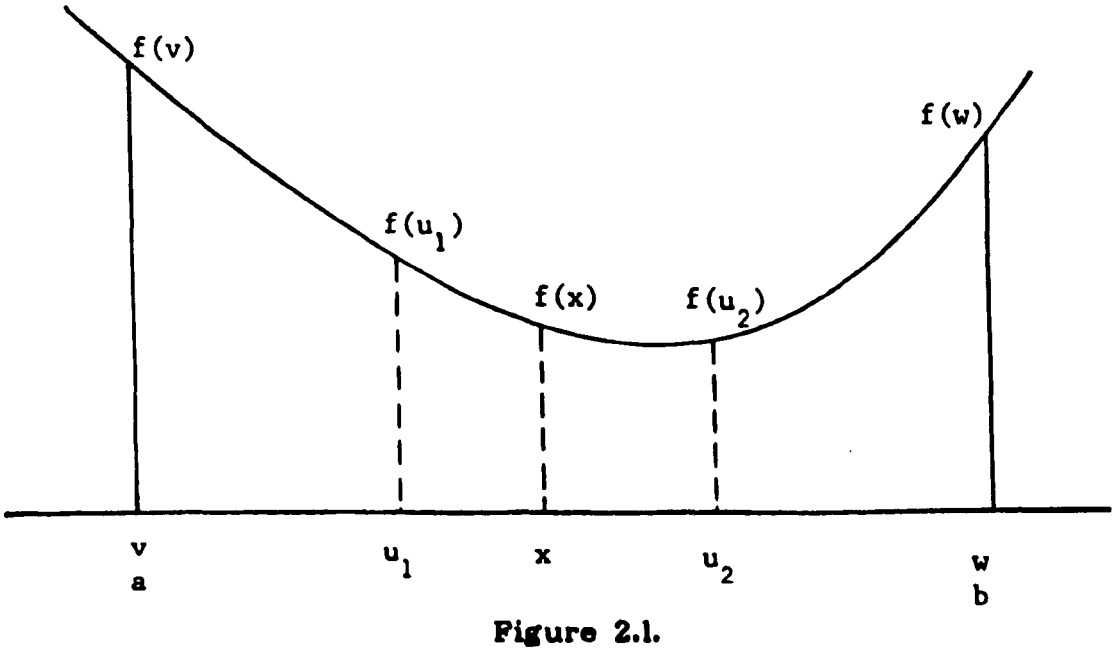


Figure 2.1.

A univariate minimisation algorithm will then predict a point u∈[a.b] at which the objective will be evaluated. How u is determined will depend on the method being used. If this is u_1 and $f(u_1)$ is higher than f(x), then the lower bound, a, is moved to point u_1. If this point is u_2 and $f(u_2)$ is lower than f(x), then the lower bound, a, is moved to x and x to u_2. In either case, w and v will be re-arranged accordingly before entering the

next iteration, and the length of the interval is reduced while the minimum remains bracketed.

Popular function-comparison methods like the Fibonacci search and the Golden Section search (see Jacoby et al, 1972, or most introductory textbooks on numerical optimisation for further details of these methods) are reliable, but because they do not take into consideration the quantity by which the objective changes within the interval, they are inefficient. To overcome this, methods based on successive polynomial approximation were developed. This class of methods involves repeated fitting of polynomials to approximate the function and to use the minimum of the fitted curve to predict the minimum of the objective. In the case where only function values are available, a quadratic function is used. When the gradient is also available, a cubic function is used instead.

The stationary point of a quadratic passing through the points (x,f(x)), (w,f(w)) and (v,f(v)) is given by x + p/q where:

$$p = \pm [(w-x)^2(f(v)-f(x)) - (v-x)^2(f(w)-f(x))] \qquad (2.3.1a)$$

$$q = \mp 2[(v-x)(f(w)-f(x)) - (w-x)(f(v)-f(x))] \qquad (2.3.1b)$$

The equation of the slope of a cubic passing through the points (x,f(x)), (w,f(w)) with derivatives f'(x) and f'(w) respectively, is given by:

$$f'(u) = f'(x) - \frac{2u}{(w-x)}(f'(x)+\eta) + \frac{u^2}{(w-x)^2}(f'(x)+f'(w)+2\eta),$$

where

$$\eta = 3\left[\frac{f(x) - f(w)}{w-x}\right] + f'(x) + f'(w)$$

The root of $f'(u) = 0$ corresponding to a minimum of the fitted cubic can be expressed as $x + p/q$, where:

$$p = \pm (w - x)\,[f'(x) - \gamma - \eta] \qquad (2.3.2a)$$

$$q = \mp[f'(w) - f'(x) + 2\gamma] \qquad (2.3.2b)$$

with

$$\gamma = \text{sign}(w - x)[\eta^2 - f'(x)f'(w)]^{\frac{1}{2}} \qquad (2.3.2c)$$

These formulae are commonly available in the literature on numerical optimisation.

With a good starting point, this method can be very efficient, especially when it is near the minimum and the objective is well approximated by a quadratic or cubic. However, if the starting value is not sufficiently close to the minimum, it can be unreliable. The common situation causing difficulty is when one of the function values used for the curve fitting is very large compared to the rest. On applying (2.3.1) or (2.3.2), the predicted minimum will tend to be very close to the small value. Since one point has to be discarded at each iteration, if one insists on using sets of points which bracket the minimum for the curve fitting, i.e. interpolation

only, the large value will be retained for a while before it is being discarded, thus slowing down the search considerably. Indeed, a natural alternative is to discard the high point since it is likely to be the least useful in future approximations. Unfortunately, the new points which result will not necessarily bracket the minimum and in this situation the predicted minimum using (2.3.1) or (2.3.2) cannot be trusted, because it is being extrapolated.

The disadvantages of these methods when used alone can be effectively eliminated by combining them together. We will focus our attention on repeated quadratic approximation. The basic strategy here is to retain the lowest function values obtained so far for fitting the quadratic, and when we are in an extrapolating position, a bound is set up to safeguard the reliability of the predicted minimum.

Figure 2.2. is a typical situation in practice with [a,b] being the current interval of uncertainty. The minimum point predicted by the quadratic through x, w and v will be obviously unreliable because it is

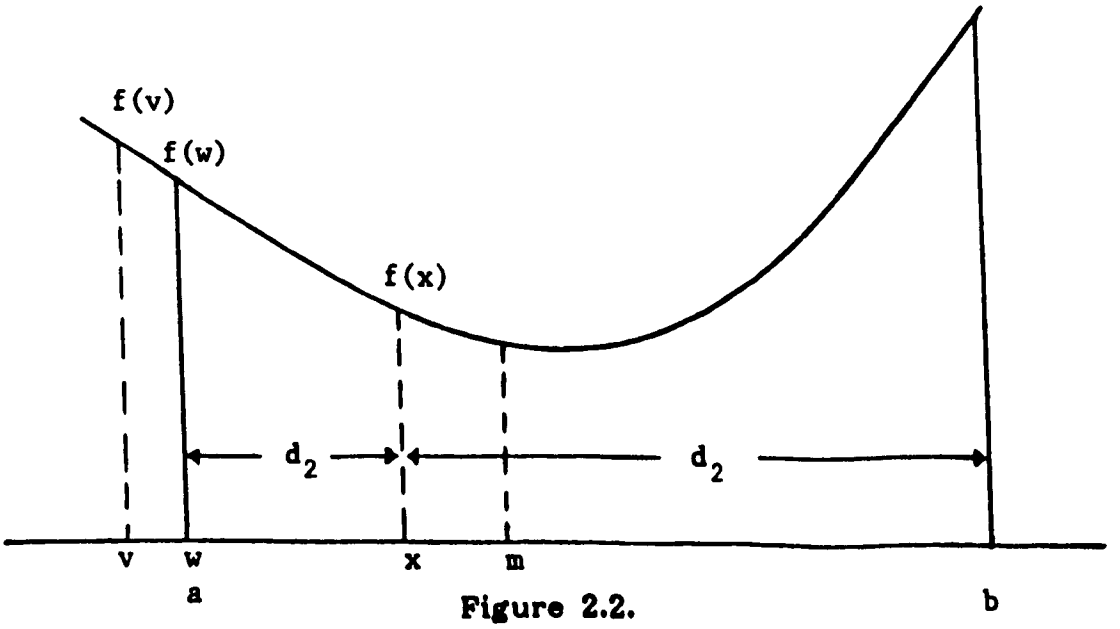


Figure 2.2.

extrapolating. A possible safeguard is to build an artificial bound m within the interval [a,b] such that the predicted point cannot exceed m. The artificial bound being used here is defined as follows:

$$m = \begin{cases} x + \beta(a - x) & \text{if } w > x \\ x + \beta(b - x) & \text{if } w < x \end{cases} \qquad (2.3.3a)$$

where

$$\beta = \begin{cases} \frac{1}{2}(-d_1/d_2)^{\frac{1}{2}} & \text{if } |d_1| < |d_2| \\ {}^{5}/_{11}(0.1 - d_2/d_1) & \text{if } |d_1| \geq |d_2| \end{cases} \qquad (2.3.3b)$$

and

$$\left.\begin{aligned} d_1 &= a - x \\ d_2 &= b - x \end{aligned}\right\} \quad \text{if } w < x$$

or (2.3.3c)

$$\left.\begin{aligned} d_1 &= b - x \\ d_2 &= a - x \end{aligned}\right\} \quad \text{if } w > x$$

An account supporting the use of the above as an artificial bound in the hybrid method can be found in the original paper (Gill and Murray, 1974). This technique which combines (2.3.1) and (2.3.3) was termed the

safeguarded quadratic univariate minimisation method. From now on a step defined by (2.3.3) will be referred to as a comparison step because one can devise a univariate function-comparison minimisation algorithm using (2.3.3) only.

The tolerance for the purpose of terminating the search is a function of the relative error ϵ and the absolute error τ. This function is defined as:

$$tol(x) = \epsilon|x| + \tau$$

and the algorithm is said to have converged to the minimum if $\max(x-a,b-x) < 2tol(x)$. The two scalars ϵ and τ will depend on the accuracy of the computer being used. A suitable choice is to set ϵ and τ to $2^{-t/2}$ when the computation is carried out on a computer with t-bit wordlength.

With practical problems, we are often ignorant even about the approximate location of the minimum. If we were to specify an interval of uncertainty through guessing, this would most certainly lead to an overstatement of the initial interval so that we can be sure that it brackets the minimum. In our program, we set the initial step-size to be $0.1|x|+100tol(x)$, where x is the starting value. This step is then taken. If the new point is higher than the starting point then all subsequent steps will be taken in the opposite direction, otherwise we will continue to take positive steps. The size of each subsequent step is 4 times the previous step. The lower (if positive step) or upper (if negative step) bound of the half-opened interval is updated every iteration. This

continues until a higher point is located in which case we have bracketed the minimum and the safeguarded quadratic minimisation algorithm is used to shrink this interval until it satisfies the termination condition.

Because of rounding error in the computation of f(x), spurious modes can be introduced even if they are not present in the function. This can be dealt with by prohibiting the evaluation of the objective at points which are less than some small distance apart. In algorithmic terms, this means we have to make sure that the predicted point is at least a distance of tol(x) away from x, a and b. This is the reason why in definition (2.3.3) the situation of w = x is excluded. However, there is a need to modify (2.3.3c) slightly. Since the point predicted by a comparison step is always in d_2, therefore, if $d_2 < tol(x)$ we must interchange d_1 and d_2, and the larger half of the interval can then be reduced rapidly.

If we merely keep the predicted point a distance of tol(x) from points x, w and v, this could lead to many steps of size tol(x) being taken. As a precaution against this, a comparison step is taken whenever $|e| < tol(x)$, where e is the step taken in the last-but-one iteration. A comparison step is also taken if $|p/q| < \frac{1}{2}|e|$, this is to ensure that the interval will at least be halved in every two iterations. To avoid comparison steps in succession, which is possible if e is small and the last step was a comparison step, e is set to be $\max(|d_1|, |d_2|)$ whenever a comparison step is taken. All these modifications can be found in the original paper. Most of which were first suggested by Brent (1973) in his steplength algorithm which combines polynomial approximation with Golden Section.

2.4. THE MULTI-VARIATE MINIMISATION METHOD

This is the modified Newton's method by Gill and Murray (1972a, 1972b). In the k^{th} iteration, this method performs the following three main steps:

1) Determine a descent direction vector, $p(k)$.
2) Find a scalar $\alpha^{(k)}$, known as the steplength, such that $f(x^{(k)} + \alpha^{(k)}p^{(k)}) < f(x^{(k)})$.
3) Perform the descent, i.e. set $x^{(k+1)} = x^{(k)} + \alpha^{(k)}p^{(k)}$.

The superscript in brackets is used to denote the iteration number. The basic difference between this method and the classical Newton's method lies merely in (1).

According to the classical Newton's method, the descent direction $p^{(k)}$ is obtained by solving:

$$G(x^{(k)})p^{(k)} = -g(x^{(k)}) \qquad (2.4.1)$$

where $g(x^{(k)})$ is the vector of partial derivatives of f and $G(x^{(k)})$ is the Hessian matrix both at point $x^{(k)}$. If the Hessian matrix $G(x^{(k)})$ is positive definite, i.e. all the eigenvalues of $G(x^{(k)})$ are positive, the direction $p^{(k)}$ will have the property that the corresponding steplength $\alpha^{(k)}$ must be positive. This can be shown by looking at the directional derivative of f in the direction of $p^{(k)}$, which is:

$$\frac{\partial}{\partial\alpha} f(x^{(k)} + \alpha\, p^{(k)}) = p^{(k)T}g(x^{(k)}) \qquad (2.4.2)$$

where $p^{(k)T}$ denotes the transpose of vector $p^{(k)}$. Because of (2.4.1), the right hand side of (2.4.2) can be written as:

$$-p^{(k)T} G(x^{(k)})p^{(k)} \qquad (2.4.3)$$

which is always negative if $G(x^{(k)})$ is positive definite and not all the elements of $p^{(k)}$ are zero. Therefore, a positive step in the direction $p^{(k)}$ must lead to a decrease in the objective.

When G is positive definite, a numerically stable method of solving (2.4.1) is first to factorize $G(x^{(k)})$ by the method of Cholesky into the form:

$$G(x^{(k)}) = L^{(k)} D^{(k)} L^{(k)T},$$

where $L^{(k)}$ is a lower-triangular matrix with unit diagonal elements and $D^{(k)}$ is a diagonal matrix. Then find vector y from:

$$L^{(k)}y = g(x^{(k)})$$

by forward substitution, vector z from:

$$D^{(k)}z = y$$

and finally $p^{(k)}$ from:

$$L^{(k)T} p^{(k)} = z$$

by backward substitution. The factorization of $G(x^{(k)})$ can be done using the method of Martin et al (1965). Their method has n major steps, where n is the number of variables, during each of which a column of $L^{(k)}$ and a diagonal element of $D^{(k)}$ are determined. Let g_{ij}, ℓ_{ij} and d_j denote the ij^{th} elements and jj^{th} element of $G(x^{(k)})$, $L^{(k)}$, and $D^{(k)}$ respectively. The j^{th} step of the decomposition is given by:

$$d_j = g_{jj} - \sum_{r=1}^{j-1} d_r \ell_{jr}^2, \qquad (2.4.4a)$$

and

$$\ell_{ij} = (g_{ij} - \sum_{r=1}^{j-1} d_r \ell_{ir} \ell_{jr})/d_j \qquad i=j+1,\ldots,n \qquad (2.4.4b)$$

It is advantageous to work with the auxiliary quantities c_{ij} defined by:

$$c_{ij} = \ell_{ij} d_j$$

and (2.4.4) becomes:

$$d_j = g_{jj} - \sum_{r=1}^{j-1} \ell_{jr} c_{jr}, \qquad (2.4.5a)$$

and

$$c_{ij} = g_{ij} - \sum_{r=1}^{j-1} \ell_{jr} c_{ir} \qquad i=j+1,\ldots,n \qquad (2.4.5b)$$

The numerical stability of this factorization method hangs on the positive definiteness of the matrix G: when this is the case, all the diagonals of $D^{(k)}$ are positive. When it is indefinite, or singular, i.e. one or more of its eigenvalues is less than or equal to zero, the factorization is no longer numerically stable even if the factors exist. In practice, a positive definite and yet very ill-conditioned matrix can become indefinite because

the factorization is carried out in finite arithmetic. In either situation, the direction $p^{(k)}$ obtained through (2.4.1) will not necessarily be a descent or 'downhill' direction because (2.4.3) is not always negative.

The modified Newton's method adopts a more sophisticated factorization technique in which, when the matrix $G(x^{(k)})$ is sufficiently positive definite (within the accuracy of the computer), the factors are identical to those by Cholesky's method, otherwise the factors are the Cholesky decomposition of a positive definite matrix:

$$\bar{G}^{(k)} = G(x^{(k)}) + E^{(k)}$$

where $E^{(k)}$ is a diagonal matrix with positive or zero elements. These elements are determined as the decomposition takes place such that the factors of $\bar{G}^{(k)}$ satisfy the following:

1) each diagonal element of $\bar{D}^{(k)}$ is always greater than a machine dependent small constant, which can be set to 2^{-t} when the computer has t-bit wordlength, and

2) the elements of $\bar{L}^{(k)}\bar{D}^{(k)\frac{1}{2}}$ are bounded by a constant β where:

$$\beta^2 = \max\left[\max_{1 \leq j \leq n} |g_{jj}| \, , \max_{\substack{1 \leq j \leq n \\ i > j}} |g_{ij}|/n \right]$$

The original paper by Gill and Murray (1972a) has the full details on the decomposition method and the rationale behind choosing the constants in (1) and (2).

When we use the modified Newton's method we are determining the search direction $p^{(k)}$ not by (2.4.1) but by:

$$\bar{G}^{(k)}\, p^{(k)} = -g(x^{(k)}) \qquad (2.4.6)$$

which is only equivalent to (2.4.1) if $E^{(k)}$ is a zero matrix, i.e. when $G(x^{(k)})$ is sufficiently positive definite. Unlike (2.4.3),

$$-p^{(k)T}\, \bar{G}^{(k)}\, p^{(k)} \qquad (2.4.7)$$

will always be negative irrespective of whether $G(x^{(k)})$ is positive definite. This means we will always have a descent direction which is determined in a numerically stable way.

The search direction will continuously be found by the use of (2.4.6) until

$$(g(x^{(k)})^T g(x^{(k)}))^{\frac{1}{2}} = \|g(x^{(k)})\|_2 < tol$$

and $E^{(k)}$ is non-zero, where tol is a small positive scalar. This means we are in the vicinity of a <u>saddle</u> point rather than a minimum. Since a saddle point has the property that $\|g(x^{(k)})\|_2 = 0$, (2.4.6) cannot be used to generate any useful direction and an alternative is needed. Gill and Murray (1972b) used the following strategy. They solved the vector y from:

$$\bar{L}^{(k)T} y = e_j$$

where e_j is the j^{th} column of an (nxn) identity matrix and j is an index

such that:

$$\bar{d}_j(k) - E_j(k) \leq \bar{d}_i(k) - E_i(k) \qquad i = 1,\ldots,n$$

with $E_j(k)$ denoting the j^{th} diagonal element of $E(k)$. Note that $\bar{d}_j(k) - E_j(k)$ is negative. The alternative search direction is defined as:

$$p(k) = \begin{cases} -\text{sign}(y^T g(x(k)))y & \text{if } \|g(x(k)\|_2 \neq 0 \\ y & \text{otherwise} \end{cases}$$

The reason for defining the alternative search direction as above is that this will be a descent direction even if $\|g(x(k)\|_2 = 0$, because under this situation the directional derivative of f in the direction $p(k)$ is equals to:

$$p(k)^T G(x(k))p(k) = p(k)^T(\bar{G}(k) - E(k))p(k)$$

$$= e_j^T \bar{D}(k) e_j - p(k)^T E(k) p(k)$$

$$= \bar{d}_j(k) - E_j(k) - \sum_{r=j+1}^{n} p_r^2(k) E_r(k) < 0$$

where $p_r(k)$ denotes the r^{th} element of $p(k)$.

The search terminates if $\|g(x(k))\|_2 < tol$ and $E(k)$ is a zero matrix, i.e. $G(x(k))$ is sufficiently positive definite.

The work involved in providing analytical second derivative in our multi-variate problems is quite substantial. Therefore the Hessian matrix $G(x(k))$ is approximated by finite differencing the derivatives as suggested

by the original authors. First we form the (nxn) unsymmetric matrix $Q^{(k)}$ whose j^{th} column $q_j^{(k)}$ is given by forward differencing g, i.e.:

$$q_j^{(k)} = (g(x^{(k)} + he_j) - g(x^{(k)}))/h$$

where h is the finite difference interval. The symmetric approximation is then given by:

$$\hat{G}(x^{(k)}) = (Q^{(k)} + Q^{(k)T})/2$$

In our program h is chosen to be 10^{-6}. The original authors suggested $2^{-t/2}$ but they also found the performance of the algorithm almost invariant amongst reasonable choices of h.

There remains to describe how the steplength $\alpha^{(k)}$ is determined in each iteration. There are broadly two descent strategies, optimal and non-optimal. In an optimal descent method the step is taken to the minimum in the descent direction. In a non-optimal descent method the step is taken whenever there is a sufficient decrease in the objective but not necessarily the minimum in that direction. The non-optimal descent strategy when applied with care is usually more efficient. The algorithm used here is proposed by Gill and Murray (1974).

The basic philosophy of the steplength algorithm being used is to proceed to compute the minimum of $f(x^{(k)} + \alpha p^{(k)})$ in α using a safeguarded polynomial approximation minimisation method and terminates the search when the function value at the new point is judged to be sufficiently lower than the current value.

Since we have both the function and gradient value available, cubic is used in the polynomial approximation. The basic strategies are identical to those in the quadratic case described in section 2.3. However, in the cubic case only 2 points x and w will be held at each iteration and x will coincide either with a or b. Therefore in order to use the same definition (2.3.3) for m in a comparison step, (2.3.3c) has to be modified to the following:

$$d_1 = w - x$$

$$d_2 = \begin{cases} b - x & \text{if } w < x \\ a - x & \text{if } w > x. \end{cases} \qquad (2.4.8)$$

The additional strategy of taking a comparison step whenever the predicted step lies outside [a,b], $|e| < tol(x)$ or $|p/q| > \frac{1}{2}|e|$ applies. Here the absolute error τ in the definition of tol(x) has to be adjusted by the division of $\|p^{(k)}\|_2$.

While the m defined by (2.3.3a), (2.3.3b) and (2.4.10) can be used in the comparison step, it is not optimal when x and w bracket the minimum, i.e. x and w coincide either with a and b or b and a. The optimal function comparison step in this case is bisection, i.e. m = (a+b)/2.

The minimisation will continue until we find an α such that:

$$|p^{(k)T} g(x^{(k)} + \alpha p^{(k)})| < -\eta p^{(k)T} g(x^{(k)})$$

and (2.4.9)

$$f(x^{(k)} + \alpha p^{(k)}) < f(x^{(k)})$$

where $\eta (0 < \eta < 1)$ is a prescribed constant. If at this point:

$$f(x^{(k)}) - f(x^{(k)} + \alpha p^{(k)}) > -\mu \alpha p^{(k)T} g(x^{(k)}) \qquad (2.4.10)$$

then the steplength $\alpha^{(k)} = \alpha$. Otherwise let s be the first member of the sequence $\{(\frac{1}{2})^j\}$ such that α_s satisfies:

$$f(x^{(k)}) - f(x^{(k)} + \alpha_s p^{(k)}) > -\mu \alpha_s p^{(k)T} g(x^{(k)}) \qquad (2.4.11)$$

and the steplength $\alpha^{(k)} = \alpha_s$. Condition (2.4.9) is to ensure that the objective is decreased sufficiently. Note that when $\eta = 0$ condition (2.4.9) is equivalent to requiring the minimum along $p^{(k)}$ to be found. Condition (2.4.10) is to prevent the situation as depicted in Figure 2.3, in which case the halving strategy will guarantee a more satisfactory value for $\alpha^{(k)}$.

In addition, an upper bound λ is imposed on α. If the best point obtained by the minimisation algorithm is λ and the directional derivative $p^{(k)T} g(x^{(k)} + \lambda p^{(k)})$ is negative, we will proceed to test condition (2.4.10) even if (2.4.9) is not satisfied.

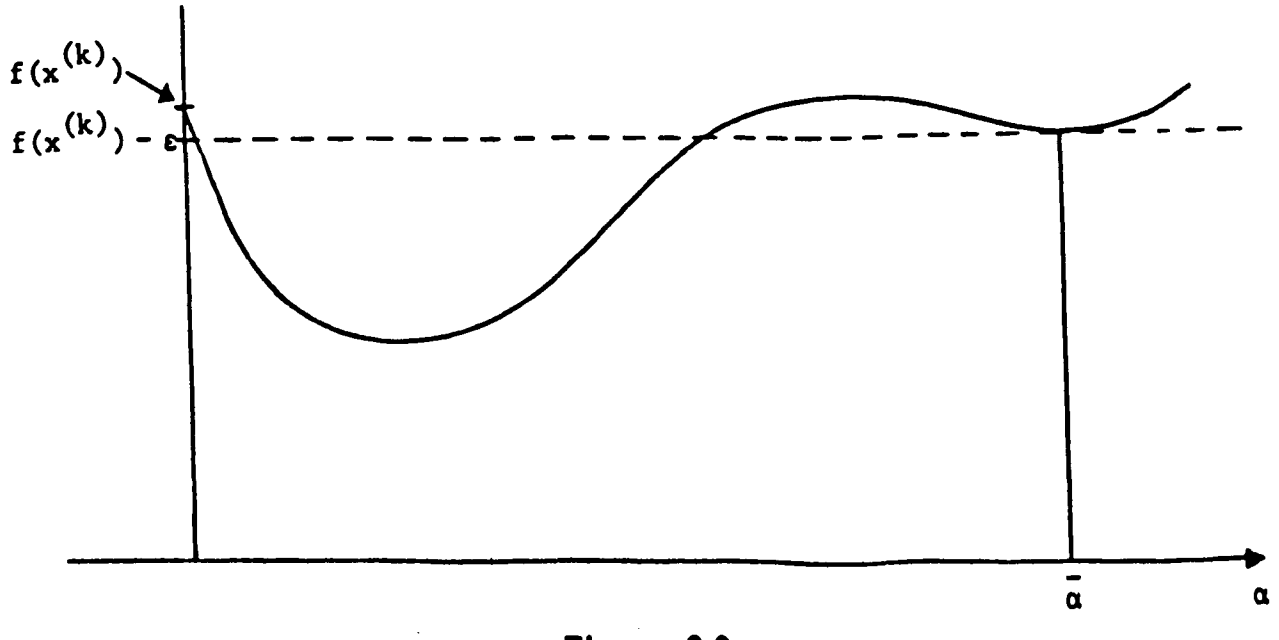


Figure 2.3.

In our programs we have chosen η to be 0.4 so that the minimum along $p^{(k)}$ is rarely found, and $\mu = 10^{-4}$ to avoid halving the steplength unnecessarily.

CHAPTER 3

IMPLEMENTATION OF THE MINIMISATION METHODS AND NUMERICAL RESULTS

3.1. INTRODUCTION

There are altogether 9 prediction systems included in this study. One of them has a Bayesian inference procedure and the remaining 8 use maximum likelihood. Of these 8 systems only one has a model with a likelihood function which can be maximised analytically, the rest can only be optimized using numerical techniques.

A section is devoted to each prediction system. When applicable, we first investigate the possibility of reducing the number of parameters in the model so that the search can be performed in a space of lower dimension. Since some or all of the parameters are constrained (for example, the parameter must be positive or bigger than a fixed number, etc.), the next step is to transform the constrained problem into an unconstrained one which is usually easier to solve. Although the Bayesian system does not rely on the methods described in Chapter 2 for its parameter estimation, a numerical algorithm is needed for the determination of its predicted median. Two methods for this purpose will be presented in the corresponding section.

The prediction systems were coded and tested on 7 sets of real data. 6 of these data sets come from Musa (1979) and the remaining one from

British Aerospace. The results provide us with considerable insight into the behaviour of the ML parameter estimate in each model.

Previously, the Jelinski and Moranda model (1972) was most studied and best understood. Littlewood and Verrall (1981) have shown that the MLE of the initial number of faults in the program according to the Jelinski and Moranda model can be infinite if a certain condition in the data is not met. A detailed proof can also be found in Joe and Reid (1985) and Moek (1983b). This "excursion to infinity" behaviour of the MLE was observed to be present in all the models. Obviously we cannot achieve the value of infinity on a computer but the parameters can assume value of magnitude which is so big that computation carried out in this range is beyond the accuracy of the machine. We will prove in the case of the Goel and Okumoto model (1979) that the likelihood function is unimodal and also obtain the condition under which the MLE of one of the parameters is at infinity. Details are given in Appendix 1.

Unfortunately similar proofs cannot be found for the remaining models, therefore we adopt the strategy of setting bounds on the parameters instead. This means the techniques described in Chapter 2 will have to be modified into methods of minimisation subject to bounds on variables. This problem has been investigated by Gill and Murray (1976).

We also found that the multi-variate problems can be very unevenly scaled, i.e. the contours of the function are packed much closer together in some directions, causing the steplength algorithm to fail. A strategy is incorporated into the multi-variate algorithm to overcome this difficulty.

The result of applying the final programs to all 7 sets of data are given in the last section.

3.2. THE JELINSKI AND MORANDA SYSTEM (JM)

The model used here is developed by Jelinski and Moranda (1972) and can justifiably claim to be the first software reliability model. It assumes that there are initially N bugs in the program each of which causes the program to fail according to a Poisson process with constant rate Φ, and the bug will be removed from the program once it causes a failure. Maximum likelihood is used to estimate the unknown parameters $N \geq i$ and $\Phi > 0$.

On observing i failures the likelihood function is:

$$f(t_1,\ldots,t_i/N,\Phi) = \prod_{j=1}^{i} (N-j+1)\Phi e^{-(N-j+1)\Phi t_j} \qquad (3.2.1)$$

and the natural log of this is:

$$\ell(t_1,\ldots,t_i/N,\Phi) = \sum_{j=1}^{i} \log(N-j+1) + i\log\Phi - \Phi \sum_{j=1}^{i} (N-j+1)t_j \qquad (3.2.2)$$

One can maximise (3.2.2) in two parameters N and Φ and eliminate Φ by expressing it as a function of N. This is done by differentiating (3.2.2) with respect to (w.r.t) Φ and equating to zero which yields:

$$\Phi = \frac{i}{\sum_{j=1}^{i} (N-j+1)t_j} \qquad (3.2.3)$$

Substituting (3.2.3) into (3.2.2) we get:

$$\hat{\ell}(t_1,\ldots,t_i/N) = \sum_{j=1}^{i} \log(N-j+1) + i\log i$$

$$-i\log\left[\sum_{j=1}^{i}(N-j+1)t_j\right] - i \qquad (3.2.4)$$

and $\hat{N}$ can be obtained by maximising (3.2.4) then $\hat{\Phi}$ from(3.2.3).

It is clear from (3.2.3) that if $\hat{N} \geq i$ then $\hat{\Phi} > 0$, therefore we only have to ensure that the constraint on $\hat{N}$ is satisfied. By letting $x^2 = N-i$ and expressing (3.2.4) in terms of x, the above constained problem is transformed into an unconstrained maximisation problem in x. The objective function is:

$$\hat{\ell}(t_1,\ldots,t_i/x) = \sum_{j=1}^{i} \log(x^2+i-j+i) + i\log i$$

$$-i\log\left[\sum_{j=1}^{i}(x^2+i-j+i)t_j\right] - i \qquad (3.2.5)$$

Littlewood and Verrall (1981) have shown that the MLE of N is infinity if

$$\frac{\sum_{j=1}^{i}(j-1)t_j}{\sum_{j=1}^{i}(j-1)} < \frac{\sum_{j=1}^{i} t_j}{i} \qquad (3.2.6)$$

in which case $\hat{\Phi} = 0$ and the MLE of $\lambda = N\Phi$ is:

$$\hat{\lambda} = \frac{i}{\sum_{j=1}^{i} t_j} \qquad (3.2.7)$$

Joe and Reid (1985) further proved that $\hat{N} = i$ if:

$$\frac{1}{\tau_i}\sum_{j=1}^{i}\tau_j < i/\left(\sum_{j=1}^{i} 1/j\right) \qquad (3.2.8)$$

and

$$\hat{\Phi} = \frac{i}{\sum_{j=1}^{i} (i-j+1)t_j} \qquad (3.2.9)$$

where τ_j is the total elapsed time before the j^{th} failure, i.e. $\tau_j = \sum_{\ell=1}^{j} t_\ell$.

Conditions (3.2.6) and (3.2.8) are tested before the numerical search. If a test is failed the MLE of the parameters will be set according to (3.2.7) or (3.2.9) and no further search is needed.

3.3. THE BAYESIAN JELINSKI AND MORANDA SYSTEM (BJM)

The model used here is essentially the same as JM in that the failure rate of the program depletes by an amount of Φ whenever a failure occurs. The only difference is that the initial failure λ is not necessarily an integer multiple of Φ. This modification was introduced to ease the inference, full details can be found in the paper by Littlewood and Sofer (1981).

In the calculation of the posterior and prediction distribution, the quantites $\{a_{j,i}\}$ are required. These are the x coefficients in the following product:

$$\prod_{j=1}^{i} (x-j) = \sum_{j=0}^{i} a_{j,i}\, x^{i-j} \qquad (3.3.1)$$

and are defined by the following recurrence relationship:

$$a_{j,i} = i a_{j-1,i-1} + a_{j,i-1} \qquad \text{for } j \geq 1 \qquad (3.3.2a)$$

with

$$a_{0,1} = 1, \quad a_{1,1} = 1 \text{ and } a_{0,i} = 1 \ \forall\, i \qquad (3.3.2b)$$

To delay overflow in computing the a's when i is big, we define:

$$a^*_{j,i} = \frac{(i-j)!}{i!}\, a_{j,i} \qquad (3.3.3)$$

and use a* in the calculation instead. A similar recurrence relationship for $a^*_{j,i}$ can be obtained through (3.3.2) where:

$$a^*_{j,i} = \frac{(i-j)!}{i!}\, i a_{j-1,i-1} + \frac{(i-j)!}{i!}\, a_{j,i-1}$$

$$= a^*_{j-1,i-1} + \frac{(i-j)}{i}\, a^*_{j,i-1} \qquad (3.3.4a)$$

with

$$a^*_{0,1},\ a^*_{1,1} = 1 \quad \text{and} \quad a^*_{0,i} = 1 \quad \forall\ i \qquad (3.3.4b)$$

Although the computation involved are complicated, they are all in finite closed form and will not take more time than the numerical search for MLE. However, if we want the predicted median, $\hat{m}_{i+1}$, we would have to solve:

$$\hat{F}_{i+1}(\hat{m}_{i+1}) = 0.5 \qquad (3.3.5)$$

where $\hat{F}_{i+1}$ is extremely complicated and does not have an analytical inverse function. Once again we have to rely on a numerical procedure.

A classical numerical problem is to find the zero of a function (Stoer and Bulirsch, 1980). To find the median, we define a univariate function:

$$h(t) = \hat{F}_{i+1}(t) - 0.5 \qquad (3.3.6)$$

then we find $\overset{*}{t}$ such that $h(\overset{*}{t}) = 0$ and our required median $\hat{m}_{i+1}$ is equal to $\overset{*}{t}$. We propose the following two methods for solving h(t) = 0 numerically.

(1) Newton's Method

This method is iterative and terminates when $\|h(t^{(k)})\|_2 < tol$, where tol is a pre-assigned small positive constant. If this condition is not satisfied, we calculate:

$$\tilde{t} = t^{(k)} - \frac{h(t^{(k)})}{h'(t^{(k)})} \qquad (3.3.7a)$$

and set:

$$t^{(k+1)} = \begin{cases} \tilde{t} & \text{if } \tilde{t} > 0 \\ t^{(k)}/2 & \end{cases} \qquad (3.3.7b)$$

The derivative of h, which is the predictive density $\hat{f}_{i+1}$, is also required in this method. The alternative in (3.3.7b) when $\tilde{t}$ becomes negative is necessary because the function $\hat{F}_{i+1}$, hence h, is only defined for positive values of t.

2. Secant Method

The ordinary Secant method involves fitting a straight line passing through the 2 points: $(t^{(k-1)},h(t^{(k-1)}))$ and $(t^{(k)},h(t^{(k)}))$. This line will cut the t-axis at point:

$$\tilde{t} = \frac{t^{(k-1)}h(t^{(k)}) - t^{(k)}h(t^{(k-1)})}{h(t^{(k)}) - h(t^{(k-1)})} \qquad (3.3.8)$$

Then $t^{(k+1)} = \tilde{t}$ and the search terminates if $\|h(t^{(k+1)})\|_2 < tol$, where tol is a pre-assigned small positive constant. Otherwise, the point $t^{(k-1)}$ is discarded and the process repeated with the two points $t^{(k+1)}$ and $t^{(k)}$. In our application it would be necessary to define $t^{(k+1)}$ as in (3.3.7b) with $\tilde{t}$ defined by (3.3.8) because $t^{(k+1)}$ cannot be negative. In this method, only one evaluation of h is required per iteration except for the first where two function values are required.

Newton's method has a higher rate of convergence which means it will need a smaller number of iterations to satisfy the termination condition. The Secant method is not very reliable if the starting value is far from the solution. But with a good starting value, it can be faster than Newton's method because in this particular application the computation involved for a function value or a gradient is roughly equal, therefore the effort per iteration, except the first, of the Secant method is roughly half that of the Newton's method. In our program we have used Newton's method for the calculation of the predicted lower quartile, median and upper quartile.

3.4. THE GOEL AND OKUMOTO SYSTEM (GO)

The model here is due to Goel and Okumoto (1979). It is a non-homogeneous Poisson process (NHPP) with a rate function defined as:

$$\lambda(\tau) = \mu\phi e^{-\phi\tau} \qquad (3.4.1)$$

where τ is the total elapsed time, i.e. the total execution time since the beginning of execution of the program. This model can also be obtained by allowing the parameter N in JM model to be distributed as a Poisson variate with mean μ.

On observing i failures the likelihood function is:

$$f(t_1,\ldots,t_i/\mu,\phi) = \left[\prod_{j=1}^{i} \mu\phi e^{-\phi\tau_j}\right] e^{-\mu[1-e^{-\phi\tau_i}]} \qquad (3.4.2)$$

where τ_j is the total elapsed time to the j^{th} failure, i.e. $\tau_j = \sum_{\ell=1}^{j} t_\ell$,

and the log likelihood is:

$$\ell(t_1,\ldots,t_i/\mu,\Phi) = i\log\mu + i\log\Phi - \Phi\sum_{j=1}^{i}\tau_j - \mu[1 - e^{-\Phi\tau_i}] \qquad (3.4.3)$$

At the maximum we can express μ in terms of Φ by differentiating (3.4.3) w.r.t. μ and equating to zero. This yields:

$$\mu = \frac{i}{1-e^{-\Phi\tau_i}} \qquad (3.4.4)$$

and when substituted into (3.4.3) gives:

$$\hat{\ell}(t_1,\ldots t_i/\Phi) = -i + i\log i - i\log[1-e^{-\Phi\tau_i}] + i\log\Phi - \Phi\sum_{j=1}^{i}\tau_j \qquad (3.4.5)$$

Therefore $\hat{\Phi}$ can be obtained by maximising (3.4.5) over $\Phi > 0$ and $\hat{\mu}$ from (3.4.4). To transform this problem into an unconstrained one, we define

$$x^2 = \Phi - \epsilon \qquad (3.4.6)$$

here ϵ is 2^{-t} when t-bit wordlength is used by the computer for the calculations. We can now express (3.4.5) in terms of x which is:

$$\hat{\ell}(t_1,\ldots,t_i/x) = -i + i\log i - i\log[1 - e^{-(x^2+\epsilon)\tau_i}] + i\log(x^2+\epsilon) - (x^2+\epsilon)\sum_{j=1}^{i}\tau_j \qquad (3.4.7)$$

and optimise this over x for $\hat{\Phi}$. The reason for the additional term ϵ in (3.4.6) is to avoid numerical difficulty in the log term in (3.4.7) when x^2 becomes too small. Note that $\hat{\mu} \to \infty$ when $\hat{\Phi} \to 0$ because of (3.4.4) which is very similar to the behaviour of $\hat{N}$ in JM model.

In fact it can be shown that the likelihood (3.4.5) is a <u>concave</u> function, and such a function has the property that there can only be one

maximum. Furthermore, this maximum occurs at $\hat{\Phi} = 0$, which implies $\hat{\mu} = \infty$, if:

$$\frac{\tau_i}{2} < \frac{1}{i} \sum_{j=1}^{i} \tau_j \qquad (3.4.8)$$

The proof of the above is given in Appendix 1. Condition (3.4.8) is the Laplace test for trend in the τ's (see Cox and Lewis, 1965).

Intuitively, the above condition cuts the total elapsed time interval into 2 halves. If the average of the elapsed time lies in the right-hand half, this means failures tend to occur late, which is in conflict with the growth situation where failures will tend to occur more frequently at the beginning of execution. When (3.4.8) is true the rate function $\lambda(\tau) \rightarrow \lambda$ with:

$$\hat{\lambda} = \frac{i}{\tau_i} \qquad (3.4.9)$$

i.e. when there is no evidence of growth in the data, the model behaves exactly as a Poisson process with a constant rate. Condition (3.4.8) is incorporated into our program, while the definition of x in (3.4.6) remains as an extra precaution.

3.5. THE MUSA AND OKUMOTO SYSTEM (MO)

The model in this prediction system was developed by Musa and Okumoto (1984). This is essentially an NHPP with a rate function:

$$\lambda(\tau) = \frac{\xi}{\beta + \tau} \qquad (3.5.1)$$

where τ is the total elapsed time.

After i failures have occurred, the likelihood function is:

$$f(t_1,\ldots,t_i/\xi,\beta) = \prod_{j=1}^{i} \frac{\xi(\beta+\tau_j)^{\xi}}{(\beta+\tau_{j-1})^{\xi+1}} \qquad (3.5.2)$$

where $\tau_j = \sum_{\ell=1}^{j} \tau_\ell$, $\tau_0 = 0$, and the log likelihood can be simplified into:

$$\ell(t_1,\ldots,t_i/\xi,\beta) = i\log\xi - \xi\log(1+\frac{\tau_i}{\beta}) - \sum_{j=1}^{i} \log(\beta+\tau_j) \qquad (3.5.3)$$

For the purpose of maximising (3.5.3), the parameter ξ can be eliminated from the log likelihood by differentiating (3.5.3) w.r.t. ξ and equating to zero. This gives:

$$\xi = \frac{i}{\log(1+\frac{\tau_i}{\beta})} \qquad (3.5.4)$$

and when substituted into (3.5.3) gives:

$$\hat{\ell}(t_1,\ldots,t_i/\beta) = i\log i - i - i\log[\log(1+\frac{\tau_i}{\beta})] - \sum_{j=1}^{i} \log(\beta+\tau_j) \qquad (3.5.5)$$

Since $\beta > 0$, we define:

$$x^2 = \beta - \epsilon \qquad (3.5.6)$$

where $\epsilon = 2^{-t}$ for a t-bit machine and $\hat{\beta}$ can be obtained by maximising:

$$\hat{\ell}(t_1,\ldots,t_i/x) = i\log i - i - i\log[\log(1+\frac{\tau_i}{(x^2+\epsilon)})] - \sum_{j=1}^{i} \log(x^2+\epsilon+\tau_j) \qquad (3.5.7)$$

over x. After which $\hat{\xi}$ can be obtained through (3.5.4).

3.6. THE DUANE SYSTEM (DU)

The model here has been studied by Duane (1964) and Crow (1977) and is again an NHPP which has a rate function defined as:

$$\lambda(\tau) = \lambda\delta\tau^{\delta-1} \qquad (3.6.1)$$

When $\delta < 1$, the rate function is decreasing in τ, it is constant when $\delta = 1$ and increasing when $\delta > 1$. This is the only model of those included in this study which has a likelihood function that can be maximised analytically.

After i failures, the likelihood function is:

$$f(t_1,..,t_i/\lambda,\delta) = \lambda^i\delta^i e^{-\lambda\tau_i^{\delta}} \prod_{j=1}^{i} \tau_j^{\delta-1} \qquad (3.6.2)$$

and the log likelihood:

$$\ell(t_1,...,t_i/\lambda,\delta) = i\log\lambda + i\log\delta - \lambda\tau_i^{\delta} + (\delta-1)\sum_{j=1}^{i} \log(\tau_j) \qquad (3.6.3)$$

The MLE of λ and δ is obtained by differentiating (3.6.3) w.r.t. λ and δ respectively, and equating to zero. This gives:

$$\hat{\lambda} = \frac{i}{\tau_i^{\hat{\delta}}} \qquad (3.6.4a)$$

and

$$\hat{\delta} = \frac{i}{i\log\tau_i - \sum_{j=1}^{i} \log(\tau_j)} \qquad (3.6.4b)$$

From (3.6.4a) it is clear that the constraint on λ being positive is automatically satisfied.

3.7. THE LITTLEWOOD SYSTEM (L)

The model here is due to Littlewood (1981). It assumes that the inter-failure times are independent exponentials with rate which initially is the sum of N independent and identically distributed (iid) Gamma variates with parameters α and β, and N is reduced by 1 every time a failure occurred.

On observing i failures the likelihood function is:

$$f(t_1,..,t_i/N,\alpha,\beta) = \prod_{j=1}^{i} \frac{(N-j+1)\alpha(\beta+\tau_{j-1})^{\alpha(N-i+1)}}{(\beta+\tau_j)^{\alpha(N-i+1)+1}} \qquad (3.7.1)$$

with τ_j defined as in MO, and the log likelihood can be simplified into:

$$\ell(t_1,..,t_i/N,\alpha,\beta) = \sum_{j=1}^{i} \log(N-j+1) - (1+\alpha)\sum_{j=1}^{i} \log(\beta+\tau_j)$$

$$+ i\log\alpha + N\alpha\log\beta - (N-i)\alpha\log(\beta+\tau_i) \qquad (3.7.2)$$

For the purpose of maximising the above, α can be expressed in terms of N and β by differentiating (3.7.2) w.r.t. α and equating to zero. This yields:

$$\alpha = \frac{i}{\sum_{j=1}^{i} \log(\beta+\tau_j) + (N-i)\log(\beta+\tau_i) - N\log\beta} \qquad (3.7.3)$$

which when substituted into (3.7.2) gives:

$$\hat{\ell}(t_1,...,t_i/N,\beta) = \sum_{j=1}^{i} \log(N-j-1) - \sum_{j=1}^{i} \log(\beta+\tau_j) - i + i\log i$$

$$-\log\left[\sum_{j=1}^{i} \log(\beta+\tau_j)+(N-i)\log(\beta+\tau_i)-N\log\beta\right] \qquad (3.7.4)$$

Furthermore, we define:

$$x_1^2 = N - i \tag{3.7.5a}$$

and

$$x_2^2 = \beta - \epsilon \tag{3.7.5b}$$

in order to transform the problem into unconstrained minimisation in the two dimensional space of x_1 and x_2.

3.8. THE LITTLEWOOD NHPP SYSTEM (LNHPP)

The model here can be obtained by letting N in the L model have a Poisson distribution with mean μ. Miller (1986) called it the Pareto NHPP and Moek (1983a) investigated the MLE of its parameters. The likelihood function after i failures is:

$$f(t_1,\ldots,t_i/\mu,\alpha,\beta) = \left[\prod_{j=1}^{i} \frac{\mu\alpha\beta^{\alpha}}{(\beta+\tau_j)^{\alpha+1}}\right] e^{-\mu\left[1-\left(\frac{\beta}{\beta+\tau_i}\right)^{\alpha}\right]} \tag{3.8.1}$$

the log of which is:

$$\ell(t_1,\ldots,t_i/\mu,\alpha,\beta) = i\log\mu + i\log\alpha + \alpha i\log\beta - (1+\alpha)\sum_{j=1}^{i} \log(\beta+\tau_j) - \mu\left[1 - \left(\frac{\beta}{\beta+\tau_j}\right)^{\alpha}\right] \tag{3.8.2}$$

Differentiating (3.8.2) w.r.t. μ and equating to zero gives:

$$\alpha = \frac{\log(1 - \frac{i}{\mu})}{\log(\frac{\beta}{\beta+\tau_j})} \qquad (3.8.3)$$

The reason for eliminating α instead of μ is that α is usually many orders of magnitude smaller than μ and β, therefore minimising in (α,β) space can be more difficult than in (μ,β) space because the former will be very poorly scaled. We further define:

$$x_1^2 = \mu - i - \epsilon \qquad (3.8.4a)$$

and

$$x_2^2 = \beta - \epsilon \qquad (3.8.4b)$$

and perform unconstrained minimisation in (x_1,x_2) space for $\hat{\mu}$ and $\hat{\beta}$. $\hat{\alpha}$ is then obtained from (3.8.3).

3.9. THE LITTLEWOOD AND VERRALL SYSTEM (LV)

The model in this system was formulated by Littlewood and Verrall (1973). They assume the inter-failure times to be independent exponentials and the j^{th} has a failure rate which is a Gamma variate with parameters $(\alpha,\psi(j))$. Growth or deterioration in reliability will depend on whether $\psi(j)$ is increasing or decreasing with j. Here we have used:

$$\psi(j) = \beta_1 + j\beta_2 \qquad (3.9.1a)$$

with

$$\beta_1 + \beta_2 > 0 \quad \text{and} \quad \beta_1 + i\beta_2 > 0 \tag{3.9.1b}$$

where i is the current number of total failures.

On observing i failures the likelihood function is:

$$f(t_1,\ldots,t_i/\alpha,\psi(j)) = \prod_{j=1}^{i} \frac{\alpha\psi(j)^{\alpha}}{(\psi(j)+t_j)^{\alpha+1}} \tag{3.9.2}$$

and the log likelihood is:

$$\ell(t_1,\ldots,t_i/\alpha,\psi(j)) = i\log\alpha + \alpha \sum_{j=1}^{i} \log\psi(j) - (1+\alpha) \sum_{j=1}^{i} \log(\psi(j)+t_j) \tag{3.9.3}$$

For the purpose of maximising (3.9.3) α can be eliminated from the above by differentiating (3.9.3) w.r.t. α and equating to zero. This gives:

$$\alpha = \frac{i}{\sum_{j=1}^{i} (\log[\psi(j) + t_j] - \log[\psi(j)])} \tag{3.9.4}$$

and the maximisation is now in the space of β_1 and β_2. To remove the constraints (3.9.1b) we consider:

$$\psi_1 = \beta_1 + \beta_2 \tag{3.9.5a}$$

and

$$\psi_2 = \beta_1 + i\beta_2 \tag{3.9.5b}$$

Clearly $\psi(j)$ can be expressed in terms of ψ_1 and ψ_2 as:

$$\psi(j) = \left(\frac{j-1}{i-1}\right) \psi_2 + \left(\frac{i-j}{i-1}\right) \psi_1 \tag{3.9.6}$$

By defining:

$$x_1^2 = \psi_1 - \epsilon \qquad (3.9.7a)$$

and

$$x_2^2 = \psi_2 - \epsilon \qquad (3.9.7b)$$

we can now maximise in the unconstrained (x_1,x_2) space by substituting (3.9.4), (3.9.6) into (3.9.3) with ψ_1 and ψ_2 defined by (3.9.7a) and (3.9.7b).

3.10 THE KEILER AND LITTLEWOOD SYSTEM (KL)

The model here is almost identical to LV model (Keiller et al, 1983). In this case the j^{th} inter-failure time is exponential with rate which is a Gamma variate with parameters $(\psi(j),\beta)$, i.e. the growth or deterioration in reliability is reflected through the shape parameter α rather than the scale parameter β.

The likelihood after i failures is:

$$f(t_1,\ldots t_i/\psi(i),\beta) = \prod_{j=1}^{i} \frac{\psi(j)\beta^{\psi(j)}}{(\beta+t_j)^{\psi(j)+1}} \qquad (3.10.1)$$

and the log of which is:

$$\ell(t_1,\ldots t_i/\psi(i),\beta) = \sum_{j=1}^{i} \log\psi(j) + \log\beta \sum_{j=1}^{i} \psi(j)$$

$$- \sum_{j=1}^{i} [\psi(j)+1]\log(\beta+t_j) \qquad (3.10.2)$$

Here $\psi(j)$ is defined as:

$$\psi(j) = \frac{1}{\alpha_1 + j\alpha_2} \tag{3.10.3a}$$

with

$$\frac{1}{\alpha_1+\alpha_2} > 0 \text{ and } \frac{1}{\alpha_1+ j\alpha_2} > 0 \tag{3.10.3b}$$

In order to reduce the dimension of the minimisation problem and to remove the constrains on the variables we define:

$$\gamma_1 = \frac{1}{\alpha_1+\alpha_2} \tag{3.10.4a}$$

and

$$\gamma_2 = \frac{\alpha_1+i\alpha_2}{\alpha_1+ \alpha_2} \tag{3.10.4b}$$

$\psi(j)$ can now be defined in terms of γ_1 and γ_2 as:

$$\psi(j) = \frac{\gamma_1}{(\frac{i-j}{i-1}) + \gamma_2(\frac{j-1}{i-1})} \tag{3.10.5}$$

Substituting (3.10.5) into (3.10.2) gives:

$$\ell(t_1,\ldots t_i/\gamma_1,\gamma_2,\beta) = i\log\gamma_1 + \gamma_1\log\beta \sum_{j=1}^{i} \frac{1}{(\frac{i-j}{i-1})+\gamma_2(\frac{j-1}{i-1})}$$

$$-\gamma_1 \sum_{j=1}^{i} \frac{\log(\beta+t_j)}{(\frac{i-j}{i-1})+\gamma_2(\frac{j-1}{i-1})} - \sum_{j=1}^{i} \log(\beta+t_j)$$

$$- \sum_{j=1}^{i} \log\left[\frac{(i-j)}{(i-1)} + \gamma_2 \frac{(j-1)}{(i-1)}\right] \tag{3.10.6}$$

By differentiating (3.10.6) w.r.t. γ_1 and equating to zero, we have:

$$\gamma_1 = \frac{i}{\sum_{j=1}^{i} \frac{\log(1 + \frac{t_j}{\beta})}{(\frac{i-j}{i-1}) + \gamma_2(\frac{j-1}{i-1})}} \qquad (3.10.7)$$

Finally, we define:

$$x_1^2 = \gamma_2 - \epsilon \qquad (3.10.8a)$$

and

$$x_2^2 = \beta - \epsilon \qquad (3.10.8b)$$

and $\hat{\alpha}_1$, $\hat{\alpha}_2$ and $\hat{\beta}$ can be obtained by first substituting (3.10.7) into (3.10.6), this function is then maximised over the unconstrained (x_1, x_2) space through definitions (3.10.8a) and (3.10.8b).

3.11. IMPLEMENTATION AND MODIFICATIONS

The safeguarded quadratic approximation minimisation algorithm (SQAMA) and the modified Newton's algorithm (MNA) described in Chapter 2 were coded as subroutines in Fortran 77 on an IBM PC-AT. The model programs used one or the other subroutine for the optimisation. 7 sets of data have been used to test the performance of these algorithms. The 7 sets of data are System 1, System 2, System 3, System 4, System 6 and System SS3 from Musa (1979) and BAe data from British Aerospace. These data are listed in Appendix 3. The results of these tests provided us

with considerable insight into the behaviour of the MLE of the parameters in each model. Furthermore, it is on the basis of the difficulties we have encountered during these tests that we incorporate certain changes to our original minimisation algorithms to make them more efficient for the more difficult problems. We begin by looking at the univariate problems first.

JM and GO model programs performed very well across all 7 data sets. However a potential difficulty exists in GO which is partly originated from the data itself.

There are often no dimensions given for software reliability data. They usually come as a sequence of numbers which might have already been scaled in some way which is convenient for recording and security. Therefore, the magnitudes of two sets of data can be very different, even if the programs are equally reliable, just because the data have been scaled differently. For example, it is fairly obvious that the magnitude of the failure times in System SS3 is bigger than that of System 1, but if one multiplies the inter-failure times in System SS3 by a factor of 10^{-3}, they would not look so dissimilar in magnitude anymore. Some but not all of the parameters in a software reliability model are <u>scale invariant</u>, i.e. the magnitude of the parameter does <u>not</u> change when a <u>positive</u> scale is applied to the data. Therefore the magnitude of those which are variant to scale will depend on the scale of the data.

Recall that in the case of GO model the minimisation variable is $x = \pm\sqrt{\phi+\epsilon}$ which is usually very small. Also recall that the search will terminate if $\max[b-x,x-a] < 2tol(x)$ with $tol(x) = \epsilon|x|+\tau$ as defined in section 2.3. This choice of tol(x) is satisfactory for $|x| > 1$ but becomes

unsatisfactory when $|x|$ is very small because τ will then act as a lower bound on tol(x). In the extreme situation where $\hat{x}$ is of the same magnitude as τ, the termination condition will cease to have any effect on the accuracy of $\hat{x}$, hence $\hat{\Phi}$. Merely setting smaller values for ϵ and τ will lead us beyond the accuracy of the computer.

One way to get round this problem is to optimise in μ rather than Φ. Alternatively, we can scale the data such that $|\hat{\Phi}| \sim 1$. We have adopted the second strategy because scaling the data proved to be useful also in other model programs for a different reason.

In the case of GO model, if a positive factor s is applied to the data, the magnitude of the new Φ parameter will become Φ/s. The method we have used to determine the scale factor s is dynamic. Let the factor used in the stage i minimisation be s_i and we obtained $\hat{\Phi}_i$, the factor for stage (i+1) is given by:

$$s_{i+1} = s_i \hat{\Phi}_i \qquad (3.11.1)$$

with the value of s for the first minimisation arbitrarily chosen to be the reciprocal of the total elapsed time up to when the analysis starts. The effect of using this scaling strategy is that not only is the accuracy of $\hat{\Phi}$ safeguarded, the efficiency of the minimisation does not suffer as a result. This is because the starting value, which is $\hat{\Phi}$ for the previous stage, is usually very close to the minimum, the value of which is under our control via scaling.

In the case of JM model, the parameter N is scale invariant and the

program is already efficient. Therefore, scaling the data serves no purpose and is not incorporated in the program.

Although the parameter β in MO model is variant to scale, if the data is multiplied by $s > 0$ the new β value will become $s\beta$, the size of which is usually bigger than 1 and therefore does not suffer the same accuracy problem as in GO model. But it has a problem of the opposite nature.

When we analysed the data sets using MO, we observed that the run of $\hat{\beta}$ from stage to stage within a set of data is usually smooth <u>if</u> the corresponding samples of inter-failure times showed evidence of reliability growth. Otherwise, the magnitude of $\hat{\beta}$ can fluctuate quite substantially and assumes a value which is so big that computation is no longer accurate.

Recall that the rate function of this NHPP is:

$$\lambda(\tau) = \frac{\xi}{\beta+\tau}$$

and the MLE of ξ is given by:

$$\hat{\xi} = \frac{i}{\log(1 + \frac{\tau_i}{\beta})}$$

where τ_i is the total elapsed time at the i^{th} failure. If $\hat{\beta}$ is very big, $\hat{\xi}$ will also be very big because of the above relationship. Looking at the rate function of this model, if we let $\xi \to \infty$ and $\beta \to \infty$ while $\xi/\beta \to \lambda$, this NHPP becomes a Poisson process with rate λ. This is analogous to the situation of $N \to \infty$, $\phi \to 0$ in JM model, or $\mu \to \infty$, $\phi \to 0$ in GO model when the data shows no sign of reliability growth. Unfortunately, we have not been able to prove results similar to those in the case of JM and GO so that we

can know when $\hat{\xi} \to \infty$ and $\hat{\beta} \to \infty$ and can avoid the numerical search. Here it is not only a matter of economising, but the value of $\hat{\beta}$ can be so big that the numerical overflowing and underflowing will either render the result of the search useless or cause the program execution to fail.

In the absence of such a test condition, the strategy we have adopted is to restrict the size of the parameter β so that we can stay clear from these numerical difficulties. This approach is further supported by the empirical observation that if the size of $\hat{\beta}$ is restrictd when it becomes too big, the detailed predictions that results from using the restricted value are not affected to any significant extent. Therefore, it seems that in these situations, the exact value of $\hat{\beta}$ or $\hat{\xi}$ is no longer significant, but:

$$\hat{\lambda}(\tau) = \frac{\hat{\xi}}{\hat{\beta}+\tau} \sim \hat{\lambda}$$

will be the single quantity of importance.

To choose a suitable upper bound on $\hat{\beta}$ for a given data set is not an obvious matter because the magnitude of $\hat{\beta}$ can be <u>big</u> for two reasons: no evidence of reliability growth in the data or the observed inter-failure times are large because of their scale. Therefore a suitable choice for one data set does not imply its suitability for use in another. We suggest that the upper bound should be determined on the basis of a trial run performed on a subset of the data. In the trial run we set a larger than expected value for the upper bound and then run the model program for a few stages. The result of this run is usually a sufficient aide for selecting a suitable value to restrict the size of $\hat{\beta}$. It will only fail if the data shows no growth at each successive stage being tested. In this

case, we can use the Laplace test to find a trial run sub-sample which shows the presence of reliability growth before we try again.

Having set an upper bound is only part of the preliminary work, the magnitude of $\hat{\beta}$ can still be unsatisfactorily large for the purpose of computation. In order to enhance the numerical accuracy, it is necessary to scale the data so that the scaled $\hat{\beta}$ remains within the desired range for computation. The result from the trial run is again useful for the purpose of choosing the scale factor s. As a general rule, we choose s so that the scaled upper bound of $\hat{\beta}$ is not greater than 10^7 when 64-bit double precision variables are being used in the computation. For example, if the trial run shows that $\hat{\beta} \sim 10^7$, we can set the upper bound at 10^{10} and s can be chosen as 10^5, which means the scaled $\hat{\beta} \sim 10^2$ and the scaled upper bound is 10^5.

The starting value of the search must now be within the set bounds. The usual choice is the minimum of the previous stage. But if this is equal to the upper bound value, it would not be a satisfactory choice. Therefore, we built in an option in the MO model program such that if $\hat{\beta}$ is greater than a pre-assigned value, this pre-assigned value will be used instead of $\hat{\beta}$ as the starting value for the minimisation in the next stage.

The setting of upper bound on the minimisation variable means that the univariate search method has to be modified. Gill and Murray (1974) in their original paper suggested a method which can be used for unimodal functions. While likelihood functions are usually well behaved, we are not entirely sure that the likelihood of MO model is unimodal, therefore we

adopt the following strategy instead. This method is applicable to problems with upper and lower bounds. The example we have used is for upper bound, the lower bound case can be dealt with similarly.

In the process of finding the initial interval of uncertainty [a,b], if the best point yet happens to be on the upper bound, we set b to be the upper bound and update a to the previous x value. Figure 3.1 shows how the points are configured.

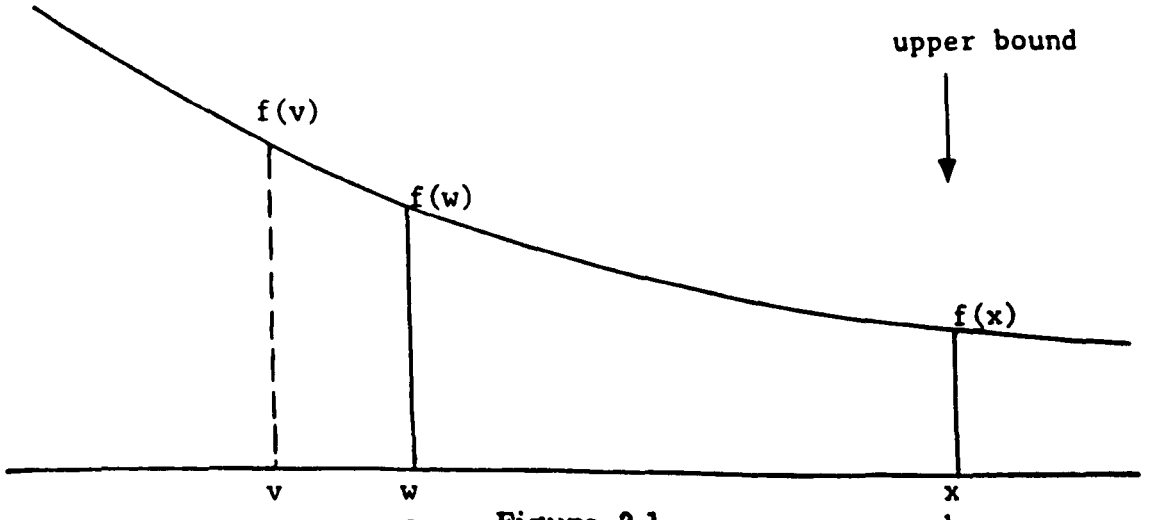


Figure 3.1

The SQAMA is then used to reduce this interval [a,b]. It is clear from Figure 3.1 that the use of this algorithm would only lead to a sequence of comparison steps. However, point x coincides with b in this situation (x will coincide with a if it would have been a lower bound) and definition (2.3.3c) cannot be used without modification to obtain the d_1 and d_2 required in defining a comparison step. We propose that the corresponding value of d, which is supposed to be set to (x-a) in (2.3.3c), be set to -tol(x) instead, if x=a. Similarly, the d which is supposed to be set to (b-x) in (2.3.3c) be set to tol(x) instead, if x=b. Note that x can only coincide with either a or b at any one time, therefore only one of the

d's will be 0 and hence be changed. By doing so, d_2 will always be set to the full interval [a,b] and the comparison step will be defined within it. The use of this strategy means that the objective will be evaluated at a few points, which are progressively closer to x, before we decide x is the bounded minimum. The number of such evaluations is usually small and in return we can be more confident that x is the minimum point within the initial interval [a,b].

This modified version of SQAMA is used by the final version of JM, GO and MO model programs. In all cases, the lower bound is set to be the minus of the upper bound which obviously can also be set to 0, but the larger range does not affect the efficiency of the programs. The upper bound in the case of JM and GO is usually set to an arbitrary big value because it is basically redundant. The scaling of data is done automatically in the case of GO model while no scaling is used in the JM model program. The alternative starting value option is not used in either, but the minimisation algorithm automatically checks and ensures that the starting value is away from the bounds. In the case of MO the modified minimisation algorithm, the scaling of the data and the alternative starting strategy are all contributive factors for the efficient determination of trustworthy MLE of the model parameters.

When we analyse the data with the remaining 4 models, we find that the LV and KL model programs performed very efficiently, but less so in the case of L and LNHPP. This is due to the MLE of the parameters in L and LNHPP being frequently very large. When this happens, the convergence criteria in the search algorithm are usually not satisfied,

resulting in a failure in the MNA. This behaviour was also observed to be present in the MLE of LV and KL model but only infrequently among the data sets we have analysed.

Miller (1986) has studied the behaviour of L and LNHPP (called the Pareto NHPP by Miller) model when their parameters take on various limiting values. In the case of L model, using an obvious notation:

$$\begin{aligned} &\underset{\substack{N\to\infty,\ \alpha\to 0\\ N\alpha\to\xi}}{\text{Lt}}\ L(N,\alpha,\beta) \to MO(\xi,\beta) \\ &\underset{\substack{\alpha\to\infty,\ \beta\to\infty\\ \alpha/\beta\to\phi}}{\text{Lt}}\ L(N,\alpha,\beta) \to JM(N,\phi) \\ &\text{and} \\ &\underset{\substack{N\to\infty,\ \beta\to\infty\\ N\alpha/\beta\to\lambda}}{\text{Lt}}\ L(N,\alpha,\beta) \to HPP(\lambda) \end{aligned} \qquad (3.11.2)$$

where HPP(λ) represents the homogeneous Poisson process with rate λ, the rest correspond to the various models and their respective parameters. Similarly, in the case of LNHPP,

$$\begin{aligned} &\underset{\substack{\mu\to\infty,\ \alpha\to 0\\ \mu\alpha\to\xi}}{\text{Lt}}\ LNHPP(\mu,\alpha,\beta) \to MO(\xi,\beta) \\ &\underset{\substack{\alpha\to\infty,\ \beta\to\infty\\ \alpha/\beta\to\phi}}{\text{Lt}}\ LNHPP(\mu,\alpha,\beta) \to GO(\mu,\phi) \\ &\text{and} \\ &\underset{\substack{\mu\to\infty,\ \beta\to\infty\\ \mu\alpha/\beta\to\lambda}}{\text{Lt}}\ LNHPP(\mu,\alpha,\beta) \to HPP(\lambda) \end{aligned} \qquad (3.11.3)$$

It is illuminating to compare the rate function of the models in the L family. The HPP is the only member with a constant rate, the others all have a rate function which is decreasing either with the number of failures occurred so far (JM model) or the elapsed time (MO model) or both (L model). Therefore, one can distinguish between these models by the different structure of reliability growth each model represents. From this viewpoint, the observed behaviour of the ML parameter estimate in this model is merely reflecting the structure of reliability growth which is present in the data. Therefore, if the likelihood of JM model is maximised when $N = \infty$ and $\hat{\beta}$ in MO is very big for a particular set of data, we can certainly expect $\hat{N}$ and $\hat{\beta}$ in L model to take very large values when we apply it to the same data.

The behaviour of the ML parameter estimate in LNHPP model is identical to L, but in the case of LV and KL, though we believe it is also related to certain structure in the failure data, they are not as well understood.

Whatever the underlying cause of this behaviour in the MLE may be, the prime concern here is to find an efficient way of obtaining the MLE of these models. Since each individual likelihood function is far too complicated for the purpose of obtaining conditions on data under which the various parameters in a model attain their possible limits, we opted for the same strategy as used in MO model, i.e. set upper bound on the model parameters.

Gill and Murray (1976) have outlined a method, which is based on the MNA, for solving minimisation problems subject to bounds on variables. Because there are only 2 variables in our problems here, we have used a

simpler approach which will not be applicable to problems with more than 2 variables. The following extra steps form a shell on the MNA and have to be performed every iteration.

1. For each of the two variables check whether it is a fixed or free variable. A fixed variable is one which hits the upper bound and has a negative gradient component or one which hits the lower bound and has a positive gradient component, otherwise it is a free variable.

2. If both variables are free, the search uses a usual MNA iteration.

3. If both variables are fixed or if one variable is fixed and the absolute value of the gradient component corresponding to the free variable is less than a pre-assigned small positive constant, tol, the search will terminate and the current point is the bounded minimum.

4. When one variable is fixed but the size of the gradient component of the free variable is not smaller than tol, an accurate line search will be performed in the direction of the free variable. If this line search fails to locate a lower point then the algorithm cannot find a bounded minimum which satisfies the termination condition and the search can either be terminated or restart at another point in the feasible region. Otherwise the search returns to step 1.

The above extra steps can be easily added to the MNA program. The only alteration required within the basic MNA is in the setting of the upper bound λ in the steplength algorithm. λ was chosen arbitrarily as 10^6 in

the unconstrained situation. Now it will be set to the largest possible step along the descent direction $p^{(k)}$ so that a bound is not crossed.

MODEL

	L	LNHPP	LV	KL
Model parameters	N,α,β	μ,α,β	$\alpha,\psi(j)=\beta_1+j\beta_2$	$\psi(j)=\frac{1}{\alpha_1+j\alpha_2},\beta$
Minimisation Variables	$x_1=\pm\sqrt{N-i}$	$x_1=\pm\sqrt{\mu-i-\epsilon}$	$x_1=\pm\sqrt{\beta_1+\beta_2-\epsilon}$	$x_1=\pm\sqrt{\frac{\alpha_1+\alpha_2}{\alpha_1+i\alpha_2}-\epsilon}$
	$x_2=\pm\sqrt{\beta-\epsilon}$	$x_2=\pm\sqrt{\beta-\epsilon}$	$x_2=\pm\sqrt{\beta_1+i\beta_2-\epsilon}$	$x_2=\pm\sqrt{\beta-\epsilon}$

Table 3.1. Relationship between the model parameters and the minimisation variables x_1 and x_2. i is the sample size and $\epsilon = 2^{-t}$ on a t-bit wordlength computer

Recall that the minimisation variables in each case are not necessarily the model parameters. Table 3.1 serves as a reminder of the relationship between the minimisation variables and the respective model parameters. In our programs, upper bounds are set on x_1^2 and x_2^2 instead of the model parameters.

To find suitable upper bounds for x_1^2 and x_2^2 is again not trivial. In the case of LV and KL, the program will usually analyse the data without much difficulty, we can set some big values for the bounds, and lower them to more suitable values only if the MLE at different stages make excursions to very large values.

For L and LNHPP model, the bound on $\hat{\beta}$ can be set to the bound used for $\hat{\beta}$ in MO model because of the limiting relationship between them. The other variable x_1^2 in these two models is invariant to scale. It has a physical interpretation as the number of bugs still remaining in the program, the upper bound of which can be set to a value judged to be too big for any practical program, e.g. 10^6.

In all cases, the scaling option is included so that we can control the size of the variables which are variant to scale, to be within the range suitable for computation. Also included is the option of starting the search at a pre-assigned point if the previous MLE is judged to be too close to the bound.

From our preliminary tests we also observed that very occasionally, when MLE were behaving normally, i.e. not exceptionally big, the steplength algorithm failed to locate a lower point, and because the current point did not satisfy the termination conditions, the MNA failed as a result. But if we perturb this point slightly, a converged solution can usually be found very near to the point at which difficulty first arose. When we investigate this in detail we find that the problems we are trying to solve can be extremely unevenly scaled. Geometrically, this means the contours of the two dimensional surface are highly elongated. When analytical Hessian $G(x^{(k)})$ is available, the scaling of the problem will not affect the performance of the modified Newton's algorithm. But in our case, since the Hessian is being approximated by forward differencing the gradient, the approximation error will have an effect on the performance of the algorithm. In one particular case we found the ratio of the gradient

components to be 10^4, with the smaller element being less than the set tolerance of 10^{-7}, at the point where the steplength algorithm failed. It is clear that differencing quantities of such order of magnitudes can introduce considerable error into the approximated $G(x^{(k)})$. The effect of the error in $\hat{G}(x^{(k)})$ can then cause the resulting $p^{(k)}$ not to point at the minimum. Recall that in the steplength algorithm, a new point must be at least a distance of tol away from any previously used points to avoid artificial modes. Therefore the steplength algorithm can still fail even though $p^{(k)}$ is a descent direction, if the situation is as depicted in Figure 3.2. In Figure 3.2, p represents the direction towards the minimum and $p^{(k)}$ is the MNA search direction. When the contours are highly elongated and the current point is situated as in Figure 3.2, the angle between the two search directions, θ, does not even have to be big to cause the steplength algorithm to fail.

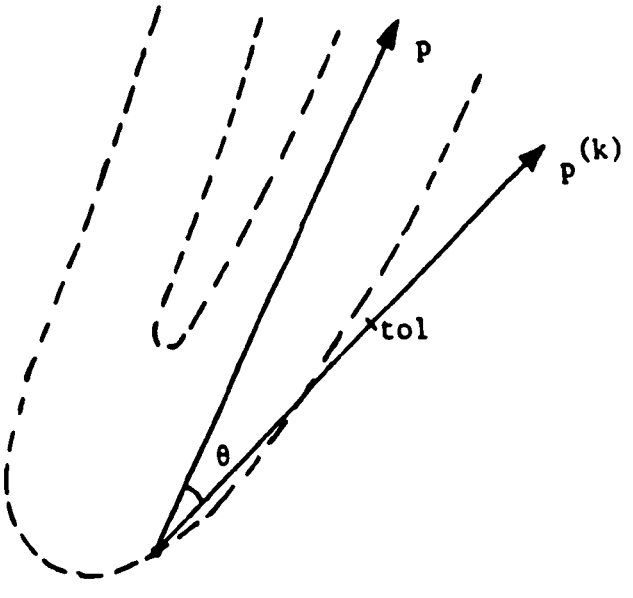


Figure 3.2.

One possible solution is to use a more accurate $\hat{G}(x^{(k)})$. We have tried using central difference instead of forward difference. This method requires twice as many gradient evaluations than the forward difference method but the approximation is usually more reliable. However, if this method were to be used throughout the search, the extra gradient evaluations are not justified because the difficulty with the steplength algorithm only arises infrequently. When it does occur, there is no guarantee that we can overcome it just by using a more accurate estimate of $\hat{G}(x^{(k)})$. In view of the above, we have adopted the following strategy instead.

Whenever the steplength algorithm fails to locate a lower point, we perform an accurate line search in the direction of the steepest descent. This direction is simply:

$$-g(x^{(k)})/\|g(x^{(k)})\|_2$$

and is a guaranteed descent direction which can be obtained easily. Another advantage of using this search direction is that when one gradient component is significantly bigger than the other, as in the case we have quoted above, the search will then be mainly in the variable with the bigger gradient. If this search produces a lower point then we continue with the steps in a usual iteration. Otherwise, we reject the critical point and start the MNA search at a perturbed point in the vicinity. The perturbation in the i^{th} variable, is arbitrarily chosen to be:

$$-10^{-3}\ \text{sign}(g_i(x^{(k)}))x_i^{(k)} \qquad \text{for } i = 1,2 \qquad (3.11.4)$$

where $g_i(x^{(k)})$ and $x_i^{(k)}$ is the i^{th} component of the gradient and the variable respectively. When the perturbed component lies outside its bounds, the sign of the perturbation defined by (3.11.4) is changed. The rejected point and its function value is stored so that if the search failed later for another reason at a different point, the point with the lowest function value will be returned. This strategy has proved to be effective in the cases we have tested. A converged solution was found either in the steepest descent search or roughly after 2 to 3 further iterations if a new point in the vicinity had to be chosen.

3.12. NUMERICAL RESULTS

The results are summarised in two main tables. Table 3.2 is for JM, GO and MO models which used the univariate search algorithm. Table 3.4 is for L, LNHPP, LV and KL models. All these programs were implemented on an IBM PC-AT and double precision variables with 64-bit wordlength were used in all the computations.

We have not timed the runs because it would be machine dependent. Instead we will report $\bar{n}$, which is the average number of function evaluations per stage in the case of Table 3.2 or the average effort per stage for Table 3.4, for each of the data sets. Effort is defined here as either a function or a gradient evaluation because they involve roughly the same amount of computation.

For the models in Table 3.2, the option of setting upper bound on the

		System 1 (101)	System 2 (41)	System 3 (28)	System 4 (40)	System 6 (53)	System SS3 (188)	BAe (127)
JM	x_B	10^8	10^8	10^8	10^8	10^8	10^8	10^8
	$\bar{n}$	11.4	11.8	13.6	12.4	13.2	12.8	12.8
($\epsilon=\tau=10^{-6}$)								
GO	x_B	10^8	10^8	10^8	10^8	10^8	10^8	10^8
	$\bar{n}$	10.7	11.0	14.0	11.4	14.2	11.8	12.7
($\epsilon=\tau=10^{-6}$)								
MO	x_B	10^7	10^7	10^7	10^7	10^7	10^{10}*	10^7
	x_s	10^6	10^6	10^6	10^6	10^6	10^9	10^6
	s	10^{-3}	10^{-3}	10^{-3}	10^{-4}	10^{-2}	10^{-6}	10^{-4}
	$\bar{n}$	10.5	11.8	12.8	11.3	12.7	12.6	11.2
($\epsilon=\tau=10^{-7}$)								

Table 3.2. Summary of the values used for the parameters in the minimisation algorithm
(x_B = upper bound of x^2 where x is the minimisation variable.
x_s = upper bound on the square of the starting value of x,
s = scale factor applied to data,
ϵ = relative error and
τ = absolute error used in defining
tol(x) = $\epsilon|x|+\tau$ for use in search termination)
and the associated average number of function evaluations per stage $\bar{n}$.
The number below the data name is the total number of stages.
If the upper bound was reached during any stage, it is followed by a *.

minimisation variable is only effective in MO model. Out of the 7 data sets, this option was required only for System SS3. Table 3.3 shows how $\hat{\beta}$ fluctuates in this case over a range of stages. The upper bound on $\hat{\beta}$ in this case was set to be 10^{10}.

Stage i	$\hat{\beta} \times 10^{-e}$
128	1.9401
129	6.8663
130	100.0000
131	100.0000
132	5.3996
133	100.0000
:	:
137	100.0000
138	3.7539
139	3.8847
140	8.7773
141	100.0000
:	:
:	:

Table 3.3. MLE of $\hat{\beta}$ in MO model for System SS3 data. Upper bound on $\hat{\beta}$ is 10^{10}.

On the basis of the results in Table 3.2, it is clear that the method we propose for obtaining the MLE of the parameters in these 3 models is very efficient indeed. The highest average number of function evaluations per stage is 14.

In the case of those models in Table 3.4, it is also clear that the modified Newton's method is efficient in solving the parameter estimation

		System 1 (101)	System 2 (41)	System 3 (28)	System 4 (40)	System 6 (53)	System SS3 (188)	BAe (127)
L	x_{B1}, x_{B2}	10^7	10^{7*}	10^{7*}	10^{7*}	10^{7*}	10^{7*} 10^{10*}	10^{7*}
	x_{s1}, x_{s2}	10^6	10^6	10^6	10^6	10^6	10^6 10^9	10^6
	s	10^{-3}	10^{-3}	10^{-2}	10^{-2}	10^{-1}	10^{-5}	10^{-3}
	$\bar{n}$	59.9	47.4	84.5	149.8	236.7	63.9	150.9
LNHPP	x_{B1}, x_{B2}	10^{7*}	10^{7*}	10^{7*}	10^{7*}	10^{7*}	10^{7*} 10^{10*}	10^{7*}
	x_{s1}, x_{s2}	10^6	10^6	10^6	10^6	10^6	10^6 10^9	10^6
	s	10^{-3}	10^{-3}	10^{-2}	10^{-2}	10^{-1}	10^{-6}	10^{-3}
	$\bar{n}$	55.4	59.8	53.0	90.6	61.7	53.2	109.3
LV	x_{B1}, x_{B2}	10^8	10^{6*} 10^{8*}	10^8	10^8	10^8	10^9	10^8
	x_{s1}, x_{s2}	10^7	10^5 10^7	10^7	10^7	10^7	10^8	10^7
	s	10^{-3}	10^{-4}	10^{-3}	10^{-2}	10^{-2}	10^{-6}	10^{-4}
	$\bar{n}$	26.5	74.3	21.8	21.1	18.4	16.2	18.0
KL	x_{B1}, x_{B2}	10^{7*}	10^{7*}	10^8	10^{7*}	10^8	10^9	10^8
	x_{s1}, x_{s2}	10^6	10^6	10^7	10^8	10^7	10^8	10^7
	s	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-2}	10^{-6}	10^{-4}
	$\bar{n}$	59.0	26.8	22.0	41.2	19.4	22.8	19.4

Table 3.4 .Summary of the value used for the parameters in the minimisation algorithm
(x_B = upper bound on x_i^2 where x_i is the i^{th} component of the minimisation variable,
x_s = upper bound on the square of the starting value of x_i,
s = scale factor applied to data,
$\epsilon = \tau = 2.5 \times 10^{-10}$ for calculating tol(α) in the steplength algorithm and the minimisation is said to have converged at $\hat{x}$ if $\|g(x)\|_2 < 10^{-7}$ and the associated average effort per stage $\bar{n}$.
If at least one of the bounds was reached at any stage, it is followed by a *.

problem in LV and KL model, but less so in the case of L and LNHPP. However, the algorithm is not the cause of this worse performance.

The main reason for the large amount of effort involved in L and LNHPP is because the program was constructed to continue searching until it finds a point which converges (satisfies the termination condition) or terminates if the maximum effort allowed, which is 300, is exhausted. Should the algorithm fail before the maximum effort allowed is exceeded, the search will restart in another point in the vicinity. Therefore the final point is either a converged solution or the best point yet within the total effort spent. Thus, when there is a large proportion of non-converging stages in a data set, the average number of effort will tend to be high. Had a lower figure been used as the maximum allowed, the average figures in Table 3.4 will come down in the case of L and LNHPP. However, the risk of using a low value for the maximum allowed is that the accuracy of the solution to more difficult problems could be affected.

Another reason is the large fluctuation which is present in the MLE of the parameters in these models, noticeably in L. The following examples are chosen to illustrate the extent of fluctuation exhibited by the MLE in each model. The first is L on System 1 data over stages 81 to 103, given in Table 3.5.

The column of n is the effort required for that stage. When the algorithm did not converge, it is followed by a *. The upper bound on (N-i) and β were set to be 10^7. It is spectacular how the parameter estimates fluctuate from stage to stage. By this behaviour, L model was

System 1(L)

Stage i	$\hat{N}$	$\hat{\alpha}$	$\hat{\beta} \times 10^{-4}$	n
81	1×10^{7}	0.4529×10^{-5}	0.4218	44
82	1×10^{7}	0.4639×10^{-5}	0.4386	24
83	0.1058×10^{7}	0.3996×10^{-4}	0.3769	16
84	0.4504×10^{7}	0.9141×10^{-5}	0.3605	26
85	95	0.6569×10^{3}	0.7665×10^{3}	302*
86	95	0.3883×10^{3}	0.4514×10^{3}	302*
87	96	0.6054×10^{3}	0.7216×10^{3}	304*
88	97	0.2683×10^{3}	0.3262×10^{3}	304*
89	285547	0.1394×10^{-3}	0.3409	212
90	679705	0.5842×10^{-4}	0.3396	22
91	698491	0.5711×10^{-4}	0.3423	18
92	108	0.3435×10^{1}	0.4425×10^{1}	136
93	98	0.8095×10^{3}	0.9993×10^{3}	302*
94	99	0.1631×10^{3}	0.2061×10^{3}	302*
95	105	0.4552×10^{1}	0.5734×10^{1}	62
96	125	0.1236×10^{1}	0.1691×10^{1}	38
97	166	0.5170	0.8586	36
98	145	0.7345	1.1064	26
99	168	0.5050	0.8495	28
100	167	0.5124	0.8586	52
101	322	0.1655	0.4845	40
102	0.5572×10^{7}	0.6949×10^{-5}	0.3261	128
103	1×10^{7}	0.3967×10^{-5}	0.3408	40

Table 3.5. MLE of the unknown parameters in L for System 1 data over a selected range.

switched between MO (when $\hat{N} \sim 10^7$), JM (when $\hat{\beta} \sim 10^7$) and itself. When this happens a larger effort is required because the starting value is not a good guess of the minimum anymore.

Over the same range, the MLE of LNHPP does not behave as violently as in the case of L, it only does the following switches as shown in Table 3.6.

System 1 (LNHPP)

Stage i	$\hat{\mu}$	$\hat{\alpha}$	$\hat{\beta}\times10^{-4}$	n
90	1×10^{7}	0.3970×10^{-5}	0.3395	52
91	1×10^{7}	0.3989×10^{-5}	0.3422	36
92	1×10^{7}	0.3783×10^{-5}	0.3122	40
93	0.7633×10^{5}	0.4682×10^{-3}	0.2823	78
94	0.2063×10^{3}	0.3017	0.5659	130
95	$0.1615sx10^{4}$	0.2330×10^{-1}	0.3122	84
96	1×10^{7}	0.3661×10^{-5}	0.2823	106
97	1×10^{7}	0.3741×10^{-5}	0.5659	44

Table 3.6. MLE of the unknown parameters in LNHPP for System 1 data over a selected range.

However, it does not mean that it is incapable of behaving like L. Table 3.7 is a range of stages when LNHPP was applied to BAe data.

The upper bound option was only used for LV in analysing System 2 data. Table 3.8 shows the sudden change in the magnitude of the MLE which took place between stages 42 and 43.

BAe (LNHPP)

Stage i	$\hat{\mu}$	$\hat{\alpha}$	$\hat{\beta}\times10^{-4}$	n
100	1×10^{7}	0.3381×10^{-4}	0.1342×10^{1}	130
101	1×10^{7}	0.1566×10^{-4}	0.5522	44
102	0.9492×10^{7}	0.1467×10^{-4}	0.4782	70
103	1×10^{7}	0.1403×10^{-4}	0.4824	46
104	0.1777×10^{3}	0.1056×10^{-4}	0.6521×10^{3}	304*
105	0.1877×10^{3}	0.1506×10^{4}	1×10^{7}	124
:	:	:	:	:
:	:	:	:	:
120	0.1894×10^{3}	0.1034×10^{4}	0.9681×10^{3}	302*
121	0.2078×10^{3}	0.9840×10^{1}	0.7330×10^{1}	110
122	1×10^{7}	0.1318×10^{-4}	0.4477	290

Table 3.7. MLE of the unknown parameters in LNHPP for BAe data over a selected range.

System 2 (LV)

Stage i	$\hat{\alpha}$	$\hat{\beta}_1$	$\hat{\beta}_2$	n
40	0.6080×10^{4}	0.7241×10^{6}	0.2759×10^{6}	96
41	0.5933×10^{4}	0.7406×10^{6}	0.2594×10^{6}	64
42	0.6138×10^{4}	0.7192×10^{6}	0.2808×10^{6}	54
43	0.2137×10^{2}	0.1941×10^{4}	0.1130×10^{4}	62
44	0.1417×10^{2}	0.1441×10^{4}	0.6975×10^{3}	54
45	0.1032×10^{2}	0.1182×10^{4}	0.4693×10^{4}	48

Table 3.8. MLE of the unknown parameters in LV for System 2 data over a selected range.

Prior to stage 43, the $\Psi(i)$ values were all on the upper bound. The change in the parameter estimate for KL also occurred between stages 42 and 43. This is shown in Table 3.9.

System 2 (KL)

Stage i	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\beta}$	n
40	0.1207×10^{-4}	0.4421×10^{-5}	1×10^{7}	26
41	0.2833×10^{-3}	0.9331×10^{-4}	0.4538×10^{6}	36
42	0.1189×10^{-4}	0.4462×10^{-5}	1×10^{7}	80
43	0.1398×10^{-1}	0.5485×10^{-2}	0.7967×10^{4}	40
44	0.1841×10^{-1}	0.6125×10^{-2}	0.6629×10^{4}	20
45	0.2434×10^{-1}	0.6797×10^{-1}	0.5512×10^{4}	20

Table 3.9. MLE of the unknown parameters in KL for System 2 data over a selected range.

The switching behaviour of the ML parameter estimate in all these models is because at each stage, the inference procedure selects a structure of reliability growth, permitted by the model being used, to best fit the data. On observing the next inter-failure time, the parameters have to be re-estimated on the basis of all the data now available. Certain asepcts of the data might have changed as a result of including this extra data point. For example, the enlarged data set might now fail the test for finite $\hat{N}$ in JM, in which case $\hat{N}$ will jump from some finite value to ∞ from one stage to the next. Thus, this fluctuating behaviour in the ML parameter estimates is in response to the change in certain characteristic of the data from one stage to another.

We have compared the estimates obtained from our programs with those obtained using the programs coded by Abdel-Ghaly for his thesis (1986). His programs used the Nelder-Mead Simplex search (Nelder and Mead,1965) for the ML parameter estimation. This non-gradient minimisation is known to be robust and is easy to implement but is extremely inefficient. We found, however, that our programs succeeded in locating estimates with lower objective function value which were missed by the other programs. In fact in the estimates obtained from the Nelder-Mead programs, the switching behaviour was not detected and the algorithms simply terminated incorrectly.

As for efficiency, it is difficult to compare between the bi-variate minimisation models because we have used a minimisation method which requires gradient, but those in Table 3.2 are undoubtedly better. However, Abdel-Ghaly's programs were not constructed with the intention that they should be efficient, so it would not be a fair basis for comparison. Nonetheless, we can safely conclude that our methods are efficient when the corresponding problem is relatively easy to solve and reliable when it is difficult.

CHAPTER 4

MEASUREMENT AND ANALYSIS OF THE PREDICTIVE QUALITY OF ORDINARY AND ADAPTIVE PREDICTION SYSTEMS

4.1. INTRODUCTION

In this Chapter we shall first look at ways of measuring the success of a prediction system in performing its task of prediction. The idea behind most of these predictive quality measurement procedures is to compare the actual predictions made in the past with the actual outcomes when they become available later. The current techniques are by no means complete, but jointly they can provide us with a lot of useful information on the past performance of a prediction system when used on a particular data set.

With this information at hand, it may help us to decide whether or not to trust the future predictions on the same program from the prediction system in question. When more than one system is being used simultaneously on a single data set, it would be even more important to be able to identify the better ones. After all, how can we justify using one system instead of another if we cannot show that there are advantages in doing so?

The methods that we shall present here are only a subset of those reported in Abdel-Ghaly et al (1986), but they are sufficiently informative for our current purposes.

The idea of adapting future predictions on the basis of past prediction error in software reliability context is due to Keiller and Littlewood (1984). They have reported encouraging results in their paper on the basis of two predictive quality measurement procedures only. When we analyse the results using a new measurement tool, the prequential likelihood, we obtained conflicting conclusions. After further investigations, we have identified the cause and the details are given in the following.

4.2. MEASURING PREDICTIVE QUALITY

So far we have been dealing with the estimation of the unknown parameters. The next step is to incorporate the result from the inference into the prediction phase.

For a Bayesian system, the predictive distribution for T_{i+1} will be its posterior distribution conditional on the data $t_1,\ldots,t_i$. For a maximum likelihood system, the ML estimate of the unknown parameters based on data $t_1,\ldots t_i$ will be substituted into the distribution of T_{i+1} as if they were the true parameters. The cumulative distribution function (cdf) of T_{i+1} is:

$$F_{i+1}(t) = Pr[T_{i+1} < t] \qquad (4.2.1)$$

The estimate of which, based on $t_1,t_2,\ldots,t_i$, is our predictive distribution $\hat{F}_{i+1}(t)$.

Once we have obtained $\hat{F}_{i+1}(t)$, we can calculate quantities which are of interest to reliability measurement. The common statistics for current reliability are the mean or median time to the next failure, the rate of occurrence of failures (ROCOF), etc. Naturally, it is important to know whether these predictions are in good accord with reality. Therefore, our next task is to examine the closeness of the predictions to reality. As these statistics are derived from $\hat{F}_{i+1}(t)$, a good estimate of the latter which is close to the true $F_{i+1}(t)$ will enable us to have good related estimates of any kind. Hence we shall focus our attention on examining the closeness of $\hat{F}_{i+1}(t)$ and $F_{i+1}(t)$.

The obvious difficulty in analysing the closeness of $\hat{F}_{i+1}(t)$ and $F_{i+1}(t)$, which is also shared by other statistics we have mentioned, arises from our never knowing what the truth is, even at a later stage. In fact, all we will ever observe is the single realisation of T_{i+1} when the program next fails, and we must base all our analysis of the predictive capability of a prediction system on these pairs $\{\hat{F}_{i+1}(t), t_{i+1}\}$.

Consider the transformations:

$$u_{i+1-i_o} = \hat{F}_{i+1}(t_{i+1}) \qquad \text{for } i \geq i_o \qquad (4.2.2)$$

which is the estimated cdf at the actual observed failure time, and i_o is the initial stage at which we begin reliability prediction. If each $\hat{F}_{i+1}(t)$ is identical to the corresponding $F_{i+1}(t)$ generating the observation t_{i+1}, then according to formal theory, the u's would be realisations of independent and identically distributed (iid) Uniform (0,1) (U(0,1)) random variables

(Rosenblatt, 1952, and Dawid 1984a). Consequently, our task of examining the closeness of $\hat{F}_{i+1}(t)$ and $F_{i+1}(t)$ can be reduced to one of examining whether the sequence of u's is behaving like a random sample from U(0,1). If it does not, it must cast doubts on the suitability of the prediction system in question for the given data.

Amongst many possible ways of examining the uniformity of a sequence of (0,1) random variables, here we will use the following two probability plotting procedures.

4.2.1. The u-plot Procedure

Assume that we have n u values generated at n successive stages. Each of these u's is obtained via a transformation as defined by (4.2.2). If they are really observations from U(0,1) random variables, their sample cdf should be close to the 45° line. The sample cdf here is the step function which is defined on (0,1) and increases by $^1/_n$ at each of the n order statistics of the u's.

The two-sided Kolmogorov-Smirnov (KS) distance is used to signify departure of the sample cdf from the line of unit slope. This distance is simply the greatest vertical distance between the sample cdf and the 45° line. Significance levels for the statistic can be found in Kendall and Stuart (1977) or Miller (1956).

Intuitively, this procedure aims at examining whether there is any biasedness in the u's. If the KS distance is significant, it would suggest

that the u's are biased and would not have come from the uniform distribution. This would in turn imply the unsuitability of the underlying prediction system to be used for the particular data set. It is important to bear in mind that we are not merely trying to establish the success or failure of the model concerned, but rather of the prediction system as a whole.

However, the order in which the u's are realised will be lost in the process of constructing the sample cdf. Thus we can, for example, have a situation where the first half of the unordered u's are biased towards low values and the latter half is reversely biased, and when they are combined together they look perfectly uniform. In this situation, the u-plot will not be able to detect the presence of such trend in the u's. The y-plot procedure, however, can prevent this kind of behaviour in the u's going undetected.

4.2.2. The y-plot Procedure

The foundation of this procedure is that if the u's are indeed the realisations of iid U(0,1) random variables, the transformations:

$$x_i = -\log(1 - u_i) \qquad i=1,\ldots,n \qquad (4.2.2.1)$$

will then be realisations of a sequence of n iid unit exponentials. When these x's are normalised by defining:

$$y_i = \frac{\sum_{r=1}^{i} x_r}{\sum_{r=1}^{n} x_r} \qquad i=1,\ldots,n \qquad (4.2.2.2)$$

the sequence of y's will be the order statistics of the realisations of n iid U(0,1) random variables.

If trend is present in the y's, the x's will no longer be realisations of n iid unit exponentials, thus resulting in the y's being non-uniform (Cox and Lewis, 1966).

The y-plot procedure compares the sample cdf of the y's with the line of unit slope. The KS distance is again used to signify departure from the 45° line.

4.2.3. The Prequential Likelihood

Miller (1983) in private communication has pointed out that:

> "...a good u-plot reflects unbiasedness or being well-calibrated and a good y-plot reflects a good fit of trend. But there is a third aspect to quality of predictions: how noisy is the predictor? Two different predictors could both have very good u-plots and y-plots but differ significantly in quality because of noise".

To illustrate this point further, he constructed the following example:

He assumed that the unconditional distribution of T_{i+1} is exponential with rate λ_{i+1}. Furthermore, this rate is estimated as:

$$\hat{\lambda}_{i+1} = \frac{2}{t_{i-1} + t_i} \qquad (4.2.3.1)$$

Therefore, the predictive cdf is:

$$\hat{F}_{i+1}(t) = 1 - e^{-\hat{\lambda}_{i+1}t}$$
$$= 1 - e^{-\frac{2t}{t_{i-1}+t_i}} \qquad (4.2.3.2)$$

It can be shown that:

$$\Pr[\hat{F}_{i+1}(T_{i+1}) \leq u] = \frac{\log(1-u)[\log(1-u)-4]}{[2 - \log(1-u)]^2} \qquad (4.2.3.3)$$

for $u\epsilon(0,1)$, which is the expected u-plot of this prediction system if the underlying assumptions of the predictor are true for the data. As it is not uniform, we can adapt using (4.2.3.3) and the adaptor is:

$$\hat{G}_{i+1}(u) = \frac{\log(1-u)[\log(1-u)-4]}{[2 - \log(1-u)]^2} \qquad (4.2.3.4)$$

This means the adapted predictive cdf $\hat{F}^*_{i+1}(t)$ will have $\hat{G}_{i+1}$ and $\hat{F}_{i+1}$ as defined in (4.2.3.4) and (4.2.3.2) respectively, and can be simplified into:

$$\hat{F}^*_{i+1}(t) = \frac{t[t+2(t_i+t_{i-1})]}{[t_{i-1}+t_i+t]^2} \qquad \text{for } t > 0 \qquad (4.2.3.5)$$

It can be shown that:

$$\Pr[\hat{F}^*_{i+1}(T_{i+1}) \leq u] = u$$

which means the u-plot should be very good. Note that in the above the original predictor is being adapted. Details on adaptive modelling are given in the next Section.

In practical situations, even though the underlying assumptions of this predictor are unrealistic, we can still expect the u-plot to be good because

of the way it is constructed; the trend is estimated using very local information, therefore the y-plot should also be good. But it is fairly obvious that there is considerable noise in this prediction system - successive predictors can fluctuate a great deal.

Table 4.1. shows the u- and y- plot distances and their corresponding significance level for the various prediction systems, and the predictor by Miller, when applied to System 1 data. The significance level of the distances is denoted by one of the letters from A to E. The range of values they represent are as follows:

A - above 20%
B - between 20% - 10%
C - between 10% - 5%
D - between 5% - 1%
E - below 1%

(4.2.3.6)

Therefore, in the case of JM model, the u-plot is significant at 1%, GO is significant at 5% but not at 1% and MO is not significant even at 20%.

As we can see from Table 4.1, MO, LNHPP and the Miller prediction system have the best u-plot and y-plot distances. The noise which is present in the last prediction system has gone undetected. Therefore, we must look for some means of measuring the variability or noise of a prediction system.

	u-plot distance	y-plot distance	-log PL
JM	.1874E	.1202C	770.253
BJM	.1702E	.1161B	770.694
GO	.1525D	.1245C	768.568
MO	.0805A	.0642A	761.393
DU	.1590D	.0931A	765.299
L	.1089B	.0732A	762.975
LNHPP	.0805A	.0643A	761.439
LV	.1437D	.1099B	764.868
KL	.1378D	.1156B	765.066
Miller	.0757A	.0686A	790.802

Table 4.1. u-plot and y-plot KS distances and log prequential likelihood of the respective prediction systems for Musa's System 1 data. Total number of predictions is 101 in all cases

In a series of important papers Dawid (1982, 1984a, 1984b) dealt with various theoretical issues concerning the validity of forecasting systems. In particular, he introduced the idea of prequential likelihood (PL) which can be used to investigate the relative plausibility of the predictions emanating from two or more different systems.

The definition of PL is as follows. The predictive probability density function (pdf) of the random variable T_{i+1} is:

$$\hat{f}_{i+1}(t) = \frac{d}{dt} \hat{F}_{i+1}(t) \qquad (4.2.3.7)$$

After a sequence of n predictions beginning at stage i_o, the prequential likelihood is:

$$PL_n = \prod_{i=i_o+1}^{i_o+n} \hat{f}_i(t_i) \qquad (4.2.3.8)$$

When there are two such systems A and B, a comparison between them can be made via their prequential likelihood ratio:

$$PLR_n = \prod_{i=i_o+1}^{i_o+n} \frac{\hat{f}_i^A(t_i)}{\hat{f}_i^B(t_i)} \qquad (4.2.3.9)$$

Dawid shows that if $PLR_n \to \infty$ as $n \to \infty$, prediction system B is discredited in favour of A, and when $PLR_n \to 0$ as $n \to \infty$, prediction system A is discredited in favour of B.

To get an intuitive feel for the behaviour of the prequential likelihood, we consider the following example. Let us assume, for the sake of simplicity, that we are trying to predict a sequence of iid random variables, i.e. $F_{i+1}(t) = F(t)$ and $f_{i+1}(t) = f(t)$ $\forall$ i. The extension to our non-stationary case is trivial.

Figure 4.1 depicts a sequence of predictive densities and the true density. The predictive densities are all biased towards the left relative to the true density. Observations which will tend to fall within the body of the true distribution, will tend to lie in the right hand tail of the predictive densities. Thus the prequential likelihood will tend to be small.

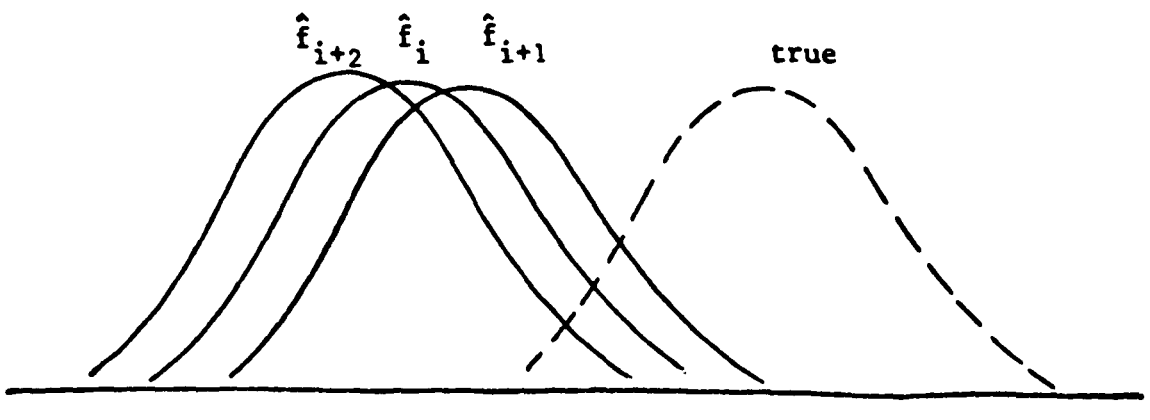


Figure 4.1

Another situation, depicted in Figure 4.2, is when the predictive densities exhibit a lot of variation, i.e. they are very noisy, but on average they are roughly unbiased. Here again the observations will tend to fall in the body of the true distribution which corresponds to the tails of the predictive densities and the prequential likelihood will again tend to be small. Thus the prequential likelihood can in principle detect predictors which are either too noisy or biased or both.

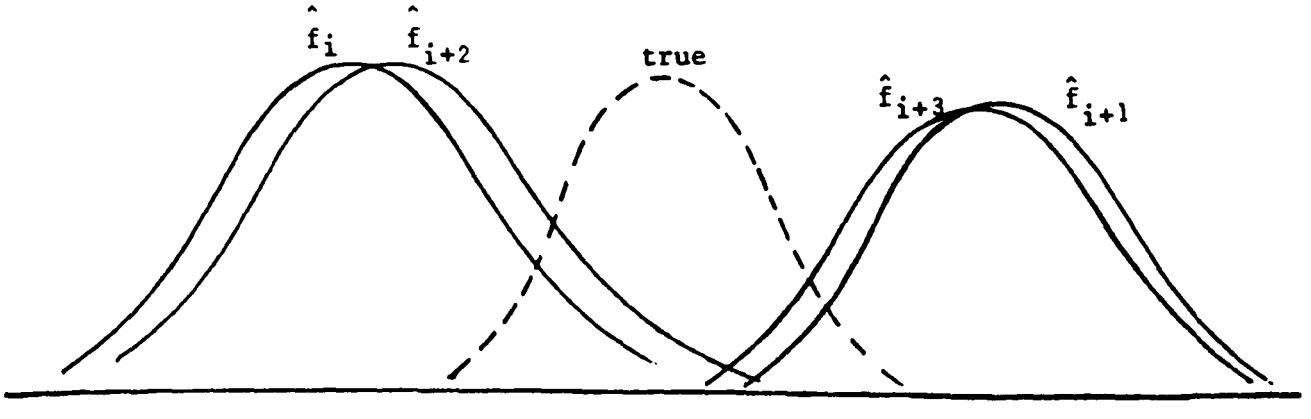


Figure 4.2

Returning to the example of Miller's prediction system, the -log of the prequential likelihood of the respective prediction systems is given in the last column of Table 4.1. While the PL confirms the superiority of MO and LNHPP for this data, it also points out the shortcoming in the Miller predictor. The PLR of MO against Miller is e^{29}. Even BJM, which has the lowest PL, against Miller is e^{20}. Thus in all cases, it is highly probable that the Miller system is discredited.

Another situation which can occur in the context of software reliability is that, while the predictors are smooth, the true distributions fluctuate for different values of i due, for example, to bad fixes. In this case the observations will tend to fall in the body of the noisy yet true distributions which correspond to the tails of the predictors, and again we can expect the prequential likelihood to detect it.

Although prequential likelihood is a global measure of predictive quality, it does not mean we can rely on the PL alone because given the PL of a set of predictions is worse than another, we cannot separate out which of bias, noise or wrong trend is responsible for this worse PL. With the u-plot and y-plot procedures, we can gain insight into what is objectively wrong with these predictions.

If two Bayesian systems are being compared, the PLR can be the posterior odds ratio of one system against the other (Abdel-Ghaly et al, 1986). Although we are not always dealing with Bayesian systems, odds ratio seems to be a useful interpretation of the PLR, and we can always bear this informal interpretation in mind.

In practical situations we would not know the location of the true distributions. Therefore, when we have more than one sequence of predictors, we can take the sequence with the highest prequential likelihood as being the closest to the truth. Other predictive distributions which are significantly different can then be judged as being too noisy or too smooth or biased.

4.2.4. Median Plots

For the purpose of comparing the location of different predictive distributions and their respective variability over different stages, it is informative to plot the predicted medians against stage i for each prediction system. The predicted median $\hat{m}_{i+1}$ is the t value which corresponds to the 50% point in the predictive cdf, i.e.

$$\hat{F}_{i+1}(\hat{m}_{i+1}) = 0.5 \qquad (4.2.4.1)$$

4.2.5. Summary

To summarise, we shall use the u-plot procedure to check the unbiasedness of the u's and the y-plot procedure to check if they are trend free. The prequential likelihood and prequential likelihood ratio shall be used to compare globally the relative plausibility of different prediction systems. The predicted medians shall be plotted to provide information on the variability or noise of the predictive distributions. This will be the basis we adopt for the assessment of predictive quality of different prediction systems.

4.3. THE u-PLOT AND ADAPTIVE MODELLING

Apart from detecting the existence of bias in the u's, the u-plot has one further useful feature. Suppose we have a prediction system which is consistently optimistic relative to reality. The actual observations will

tend to be smaller than expected according to this prediction system. Consequently, the sequence of u's which results from a number of such predictions will have a high proportion of small u values. The sample cdf of such a set of u's will lie predominantly above the 45° line.

In the opposite situation when the system is pessimistic, the set of u's will have a high proportion of big u values. The u-plot of these u's will lie predominantly below the 45° line.

Applying this observation in reverse, we can deduce whether a prediction system is pessimistic or optimistic for a particular data set by inspecting whether the u-plot is predominantly below or above the line of unit slope. This kind of bias, i.e. simple optimism or pessimism, is only used as an example and the u-plot could pick up other consistent departures from reality.

Keiller and Littlewood (1984) utilised this idea further and constructed a general adaptive procedure which allows current predictions to be improved in the light of past predictive behaviour of the system. Their method aims at improving the future predictions of a prediction system which has a good y-plot but poor u-plot. In other words, the method should work best when we are reasonably confident that the trend (reliability growth) in the data is being captured by the system, but there is still considerable biasedness.

The rationale of this adaptive approach is as follows. In theory, if

one knows the true cdf $F_{i+1}(t)$, one can always express this as a function of the estimated predictive cdf, i.e.:

$$F_{i+1}(t) = G_{i+1}(\hat{F}_{i+1}(t)) \qquad (4.3.1)$$

The role of G_{i+1} can be viewed as correcting the estimated cdf $\hat{F}_{i+1}(t)$. If $\hat{F}_{i+1}(t)$ is indeed the true cdf, then G_{i+1} would simply be the line of unit slope.

It is conceivable that the estimated cdf is wrong only in being biased, in this case G_{i+1} will recalibrate the predictive cdf. As we have just established that the u-plot contains information on the nature of biasedness which is present in a prediction system, this information could then be used to recalibrate future prediction by the same system on the assumption that this bias is expected to persist. To insist on having a good y-plot aims to ensure that any non-uniformity in the u-plot is only due to biasedness and not wrong trend.

The adapted predictive cdf is defined as:

$$\hat{F}^*_{i+1}(t) = \hat{G}_{i+1}(\hat{F}_{i+1}(t)) \qquad (4.3.2)$$

where $\hat{G}_{i+1}$ is the estimated calibration curve for stage i based on the u's obtained from past predictions prior to the present stage. This is repeated with a new $\hat{G}$ being constructed at each stage. Note that this adaptive prediction method forms a genuine prediction system because it only uses past data to predict.

Keiller and Littlewood (1984) suggested the following two ways of constructing the $\hat{G}$ functions.

Method 1. Here $\hat{G}$ is essentially the u-plot of the u's obtained prior to the current stage. In order that this function be continuous and the adpated predictions are unique, the vertical increment is $1/(n+1)$ instead of $1/n$ at each of the n u values, and $\hat{G}$ is the function which interpolates this modified u-plot. In other words, $\hat{G}$ is the joined up sample cdf of the u's with step height $1/(n+1)$.

Method 2. The $\hat{G}$ function here is constructed in the same way as in Method 1, but the u's are calculated using the most recently estimated model parameters. Therefore, at stage i of a ML system, the first step is to estimate the model parameters using data $t_1,\ldots,t_i$. These parameters are then substituted into the cdf's of the past stages to recalculate the u's. In other words, the u's are being retrodicted. $\hat{G}$ will then be constructed as in Method 1 with these new u's.

The authors reported encouraging results on the basis of the KS distances of the u-plot and y-plot when these two methods were applied to a selection of prediction systems and real data sets. Another feature in their results is that the second method of adapting is not performing as well as the first method.

This is hardly surprising because the $\hat{G}$ function obtained by the second method does not contain the actual predictive error which the system has made in the past. In fact, none of the u's used for the

estimation of $\hat{G}$ is genuinely predictive because all these u's were constructed using data which at least partly would not be available if they were in a predictive position. For example, the parameters in u_j $(j > 0)$ were estimated using data $t_1,\ldots,t_{i_o+j}$, which if it were genuinely predictive it could only have used data $t_1,\ldots t_{i_o+j-1}$.

One might be inclined to think that the $\hat{G}$ obtained via Method 2 should be superior because the parameters are estimated on the basis of more data. However, this is not necessarily the case, at least for some of the models, because the behaviour of their ML parameter estimates are not known. With those models which specify a finite number of failures, the usual asymptotic properties of MLE will not apply because of the existence of a finite ceiling on the total population. Finite sample properties of these model parameter estimates are invariably difficult to obtain even for the simplest model. Hence we shall concentrate on the application of Method 1 only.

4.4. THE ANALYSIS OF PREDICTIVE QUALITY OF ADAPTIVE PREDICTION SYSTEMS

In the previous section we have discussed how to construct an adaptive predictor. For reasons given there, we will only consider adapting with $\hat{G}$ functions constructed using Method 1. Since $\hat{G}_{i+1}$ is constructed on the basis of past data $t_1,\ldots,t_i$ only, it is therefore a genuine prediction system. This means we can assess the performance of a sequence of adapted predictions using the techniques we have discussed in section 4.2.

Keiller and Littlewood (1984) have reported results on the basis of the u-plot and y-plot distances only. We shall present the results of our wider choice of prediction systems and an extra data set. Table 4.2 summarises the total number of data points, the stage at which analysis of predictive quality starts (i_o), the number of u's in the analysis of predictive performance (n) and the number of u's in the first adaptor $\hat{G}_{i_o+1},(n_o)$. Therefore, for each of the data sets, the parameter estimation begins at stage (i_o-n_o) and actual predictions by the adaptive systems only start at stage i_o.

For the purpose of analysing the predictive quality of an adaptive system, we use the u- and y-plot procedures on the set of u^*'s defined by:

$$u^*_j = \hat{F}^*_{i_o+j}(t_{i_o+j}) \qquad j=1,\ldots,n \qquad (4.4.1)$$

The prequential likelihood contribution from stage $i \geq i_o$, which is the adapted predictive pdf evaluated at the observed failure time t_{i+1}, is:

$$\hat{f}^*_{i+1}(t_{i+1}) = \frac{d}{dt}\hat{G}_{i+1}(\hat{F}_{i+1}(t))\Big|_{t=t_{i+1}}$$
$$= \hat{g}_{i+1}(u_{i-i_o+1})\hat{f}_{i+1}(t_{i+1}) \qquad \text{for } i=i_o,\ldots,i_o+n-1 \qquad (4.4.2)$$

where $\hat{g}$ is the derivative of $\hat{G}$. Note the PL for stage i is the product of the raw PL and the gradient of $\hat{G}$ at the value of u defined by the raw prediction system for stage i. The PL after n predictions will simply be:

$$\prod_{i=i_o+1}^{i_o+n} \hat{f}^*_i(t_i) \qquad (4.4.3)$$

with each of the $\hat{f}^*$'s as defined by (4.4.2).

DATA	Total no. of failures	Stage at which analysis starts (i_o)	Total no. of predictions (n)	Number of u's in the first adaptor $\hat{G}_{i_o+1}(n_o)$
System 1	136	50	86	15
System 2	54	23	31	10
System 3	38	20	18	10
System 4	53	23	30	10
System 6	73	35	38	15
System SS3	278	105	173	15
BAe	207	95	112	15

Table 4.2. Summary of the total number of failures, i_o, n and number of u's in the first adaptor.

If we want to examine the improvement over the raw predictor brought about by this adaptive procedure, we can calculate the PLR of the adapted against raw predictions which is:

$$\prod_{i=i_o+1}^{i_o+n} \frac{\hat{f}^*_i(t_i)}{\hat{f}_i(t_i)} = \prod_{i=i_o+1}^{i_o+n} \frac{\hat{g}_i(u_{i-i_o})\hat{f}_i(t_i)}{\hat{f}_i(t_i)} = \prod_{i=i_o+1}^{i_o+n} \hat{g}_i(u_{i-i_o}) \qquad (4.4.4)$$

which is simply the product of the gradient of the successive $\hat{G}$ functions at the corresponding u value given rise by the raw system.

The predicted median of the adapted distribution is obtained by solving:

$$\hat{F}^*_i(\hat{m}^*_i) = 0.5 \qquad i = i_o+1,\ldots i_o+n \qquad (4.4.5)$$

which presents no difficulties.

Table 4.3 contains the u- and y-plot KS distances for the 9 prediction systems before and after * adapting using Method 1. The corresponding level of significance is given by (4.2.3.6). These results are not identical to those of Keiller and Littlewood because the ranges over which predictions were made in each data set were different. There is clear evidence that the adapting procedure does improve the results from the raw predictors, in some cases quite considerably, for example, System 6 and SS3 data. Only on one occasion does the u^*-plot have a marginally worse KS distance, but this is still not significant at 10%. Note also that the y-distance in this particular case is very poor indeed, which suggests that the trend in the data is not being captured in the first place. Thus the u-plot, hence $\hat{G}$, will contain error information which is not only due to biasedness alone.

DATA (n)		JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
	u	.2049E	.1871E	.1773E	.0982A	.1567D	.1123A	.0982A	.1504D	.1457D
Sys.1	u*	.1188B	.1226B	.1341C	.0499A	.0752A	.0499A	.0499A	.0894A	.0901A
(86)	y	.1156B	.1148B	.1190B	.0795A	.1029A	.0904A	.0793A	.1148B	.1173B
	y*	.1018A	.1016A	.1076A	.0775A	.0808A	.0893A	.0768A	.0901A	.0916A
	u	.2604D	.2325C	.2181C	.1518A	.2317C	.1554A	.1518A	.2388D	.2219C
Sys.2	u*	.1637A	.1666A	.1736A	.1423A	.1337A	.1401A	.1617A	.1332A	.1115A
(31)	y	.1858A	.1853A	.1989B	.1898B	.1620A	.1772A	.1743A	.1325A	.1451A
	y*	.1500A	.1511A	.1581A	.1909B	.1812A	.2045B	.1476A	.1747A	.1880B
	u	.7038E	.3354D	.2705B	.1877A	.3556D	.2556B	.1531A	.4260E	.3908E
Sys.3	u*	.3294D	.2696B	.2730B	.1121A	.1333A	.1590A	.1059A	.2070A	.2009A
(18)	y	.6808E	.3900E	.4445E	.2234A	.2012A	.3075C	.2430A	.1112A	.1135A
	y*	.3436D	.3193D	.2841C	.1874A	.2113A	.1839A	.2006A	.1576A	.1303A
	u	.1711A	.2185C	.1328A	.1143A	.1415A	.1712A	.1211A	.1955B	.2014B
Sys.4	u*	.1259A	.1854A	.1627A	.1805A	.1975B	.1253A	.1805A	.2156B	.1994B
(30)	y	.4647E	.1399A	.1989B	.3418E	.4887E	.2709D	.2695D	.2420D	.2010B
	y*	.2110B	.1535A	.1440A	.2360C	.4567E	.1838A	.2002B	.1754A	.1495A
	u	.2924E	.3010E	.2812E	.2845E	.2856E	.2853E	.2845E	.1658A	.1731B
Sys.6	u*	.0821A	.0803A	.0786A	.0639A	.0846A	.0923A	.1030A	.1248A	.0925A
(38)	y	.3969E	.3486E	.3870E	.4017E	.4010E	.3978E	.4026E	.2020C	.2069C
	y*	.2373D	.2405D	.2421D	.2573D	.2366D	.2404D	.2501D	.2066C	.2160D
	u	.2717E	.2713E	.2705E	.2645E	.2596E	.2717E	.2704E	.2382E	.2372E
SS3	u*	.0982C	.1042D	.0978C	.1057D	.1122D	.0987C	.0997C	.0864B	.1043D
(173)	y	.1273E	.1379E	.1263E	.1435E	.1835E	.1291E	.1300E	.0346A	.0500A
	y*	.0577A	.0664A	.0579A	.0631A	.0968C	.0561A	.0558A	.0415A	.0596A
	u	.0775A	.0726A	.0697A	.0713A	.1270D	.0763A	.0655A	.1039B	.1151C
BAe	u*	.0623A	.0626A	.0613A	.0876A	.1126B	.0617A	.0636A	.1016B	.0931A
(112)	y	.0890A	.0787A	.0906A	.0793A	.0744A	.0873A	.0790A	.0673A	.0687A
	y*	.0753A	.0689A	.0763A	.0711A	.0682A	.0743A	.0728A	.0765A	.0818A

Table 4.3. Kolmogorov-Smirnov distance of the u-plot and y-plot of the respective prediction systems before and after * adapting (joined-up function adaptor).
See (4.2.3.6) for the corresponding significance level.

A surprising feature is the improvement in the y-plot KS distances. In most cases there is marginal improvement in the y-plot after adapting. Only two adapted predictors have marginally poorer y^*-plot on System 2 data. The most significant improvement is on System SS3 data.

Table 4.4 gives the -log prequential likelihood of the raw systems for each set of data. In the cases of JM and L model on System 3 and 4 data, the - ∞ log prequential likelihood is due to the occurrence of the event that at some stage i, $\hat{N}$ equals the number of failures seen, i. According to these two models, it means that the program should be free of any faults. But it promptly failed after executing for a period of t_{i+1}. Therefore, according to these prediction systems an impossible event which has zero measure has occurred, thus the prequential likelihood immediately goes to zero.

The prequential likelihood is unforgiving in reacting to this kind of behaviour in a prediction system, because once the PL is zero it will remain at zero no matter how good or bad the other predictions are. Such a system is simply totally rejected. After all, it is not desirable to use a prediction system which assigns zero probability to an event which promptly occurs at a later stage. By having a zero PL, not only are we reminded of the occurrence of this event in the past, we are also constantly warned that this might happen again in the future.

In fact, whenever $\hat{N} = i$, where i is the number of failures occurred, the predictive cdf in these predictive systems will be:

$$\hat{F}_{i+1}(t) = \begin{cases} 0 & 0 < t < \infty \\ 1 & t = \infty \end{cases} \qquad (4.4.6)$$

DATA (n)	JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
Sys.1 (86)	668.944	669.147	667.267	660.061	663.715	661.664	660.107	663.348	663.212
Sys.2 (31)	286.183	285.546	284.313	279.918	283.425	282.010	280.532	281.882	282.244
Sys.3 (18)	∞	173.779	172.848	164.140	169.089	∞	165.365	170.955	169.367
Sys.4 (30)	∞	233.691	239.356	242.512	253.617	∞	241.838	233.390	232.233
Sys.6 (38)	210.007	204.807	208.211	207.407	203.618	209.659	207.587	191.395	191.554
SS3 (173)	2300.37	2298.09	2300.12	2301.12	2303.07	2300.47	2300.49	2263.79	2267.31
BAe (112)	637.352	636.835	637.419	637.265	641.053	637.566	637.572	637.969	638.694

Table 4.4. -log prequential likelihood of the respective prediction systems.

DATA (n)	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
Sys.1 (86)	680.743	693.229	693.978	690.742	689.099	682.513	690.044	685.545	688.851
Sys.2 (31)	292.269	291.094	288.235	287.037	289.994	288.623	287.127	285.069	287.203
Sys.3 (18)	∞	179.596	177.455	172.803	168.808	∞	173.382	168.985	175.417
Sys.4 (30)	∞	243.954	254.317	252.966	258.466	∞	254.092	244.884	244.789
Sys.6 (38)	208.735	205.540	207.166	203.425	211.886	207.996	207.406	200.758	202.255
SS3 (173)	2274.06	2266.05	2271.95	2268.83	2281.41	2267.57	2270.89	2274.62	2288.64
BAe (112)	674.711	681.492	676.400	676.652	682.754	677.950	667.049	671.112	672.455

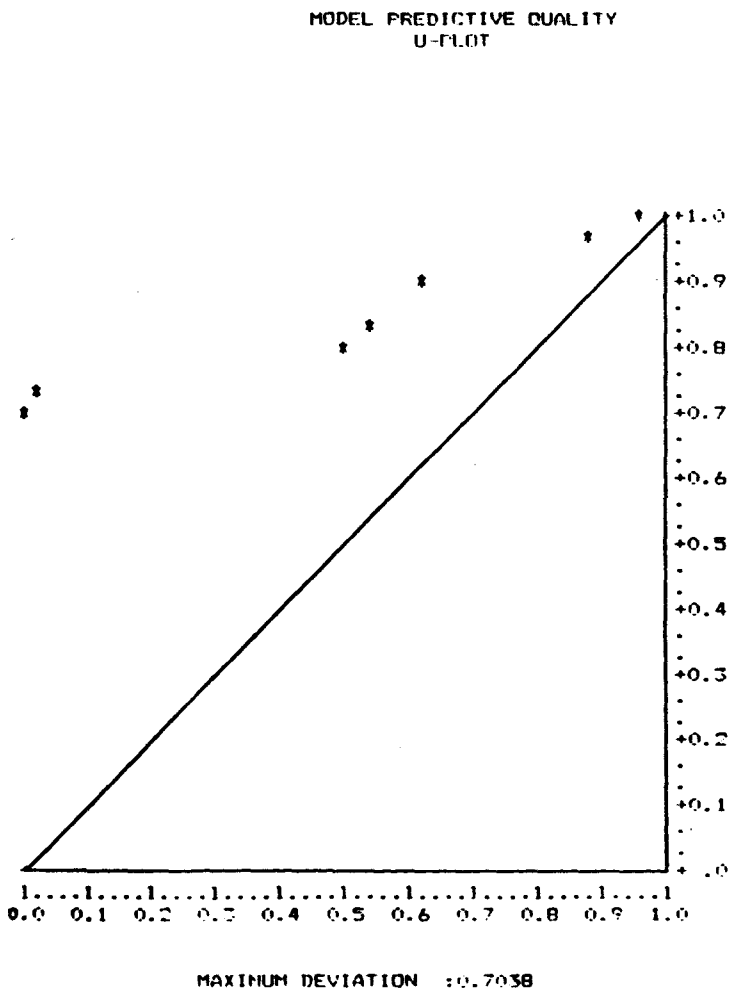
Table 4.5. -log prequential likelihood of the respective adpated (joined-up function adaptor) prediction systems.

which means the adapted probability for any finite failure time is equal to the value of $\hat{G}_{i+1}$ at the origin. When $\hat{N} = i$ has happened for previous value(s) of i, it would mean that there is a discrete component to $\hat{G}_{i+1}$ at the origin. Thus the adapting procedure might seem to be able to correct the raw prediction, but in reality the adapted predictor is just as practically useless as before: it has probability $\hat{G}_{i+1}(0)$ for any finite failure time and probability $(1-\hat{G}_{i+1}(0))$ at infinity. Merely looking at the u-plot and u^*-plot could be very misleading. As an example, Figure 4.3 shows the u-plot for JM on System 3 data before and after adapting. (These are line printer plots and only provide an approximate picture to the true plots).

When there is a discrete component to $\hat{G}$ at point 0, it means the derivative at 0 will be infinite. Since the PL of the adapted prediction is now the product (4.4.2) with a term being zero, it will also be 0.

However, there are zero failure times in some of Musa's data, for example, System 1 data. In this case, it means that there will be a discrete component to $\hat{G}$, due to these zero failure times which will give rise to repeated zero u values. Because the raw PL is non-zero in this case, the PL of the adapted predictor will also be infinite. This implies that all other prediction systems, except those also with infinite PL, are inferior.

This behaviour in the PL is due to our mixing discrete and continuous probabilities in the adapted predictive cdf. The PL as defined is for continuous random variables and utilises the pdf which is the derivative



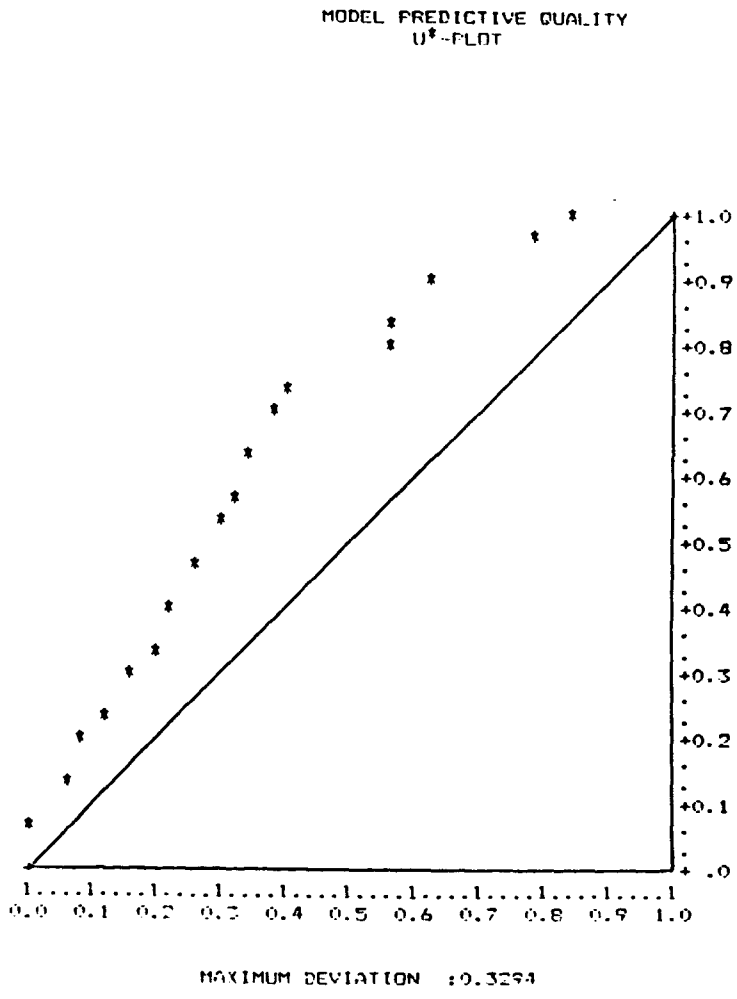


Fig. 4.3 The u-plot before and after * adapting JM predictions on System 3 data using a joined-up adaptor.

of the cdf and is a continuous function. For discrete random variables, the PL is defined in terms of the probability mass function which is a discrete function obtained by differencing, instead of differentiating, the cdf at consecutive points in the domain. This is essentially a conceptual confusion which can be avoided if $\hat{G}$ is constrained to be a function without a discrete component.

Table 4.5 is the -log prequential likelihood of the adpated predictions on the various data sets. The gradient of $\hat{G}$ at 0 is taken to be the gradient at 0+ in order to avoid the situation mentioned above. Adapted predictions for System SS3 have in most cases better prequential likelihood value than raw predictions. The improvement in System 6 data is only negligible in those cases which are better. In all other cases the adapted predictions have worse prequential likelihood than the raw predictions.

This is counter-intuitive. Take System 1 data for example. The KS distance of u^*-plots have all been improved by the adapting process which means the new predictions are not as biased as before. The KS distances of the y^*-plots are all very good, which means the trend in the data is being captured. The adapted predicted medians are in closer agreement than before.

Table 4.6 gives the predicted medians by the various raw prediction systems at selected stages. We can clearly see that the first three systems are always predicting high values, the last two systems are always

predicting low values and the rest are in the middle. The disagreement is profound between the highest and the lowest predictions, especially at the later stages.

Table 4.7 gives the predicted medians by the adaptive systems. As we can see there are hardly any differences among all of them up to stage 80. Subsequent disagreement is much less severe than before adapting, and there is close agreement amongst the last six adaptive systems. Yet even in the presence of all this evidence, the prequential likelihood insists on the superiority of the raw predictions.

Since both bias and trend are good in the adpated predictions, can it be the case that a lot of noise was somehow introduced into the adaptive systems? If this is true, as the raw predictive cdf and the adapted predictive cdf are related only via the $\hat{G}$ function, it must be the source of all this extra noise in the new system.

Successive $\hat{G}$ functions are different only because of an extra u being included in the basis for constructing the latter. Therefore, it should be fairly slow changing over different stages, particularly when there are many u's in the basis already, and should not fluctuate in any substantial way.

A possibility is when the number of u's used in the basis for constructing $\hat{G}$ is small, the effect of an extra u would be larger and thus causing fluctuations in the early stages. If this is the sole reason, all we

Stage i	JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
60	344	331	316	302	230	302	302	242	247
70	377	372	357	336	255	336	336	274	281
80	460	449	433	385	288	385	385	318	334
90	900	873	841	577	401	577	577	418	428
100	1729	1676	1615	854	563	1032	854	534	538
110	1502	1452	1408	906	595	906	906	570	575
120	1320	1250	1217	931	613	931	931	613	621
130	2314	2197	2137	1242	793	1242	1242	662	668

Table 4.6. Predicted median for System 1 data at selected stages by respective prediction systems.

Stage i	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
60	265	268	273	250	249	250	250	255	252
70	298	301	308	279	276	279	279	296	299
80	363	363	374	330	318	326	326	340	342
90	763	763	809	627	568	515	627	558	553
100	1467	1461	1529	974	868	921	974	740	715
110	1155	1171	1202	965	886	778	955	790	767
120	994	982	993	952	899	791	896	848	817
130	1732	1684	1722	1195	1124	1051	1190	894	890

Table 4.7. Predicted median for System 1 data at selected stages by respective adapted (joined-up function adaptor) prediction systems.

have to do is to start with a reasonable number of u's for constructing $\hat{G}$ and we should see better PL from the adapted systems. It is easy to check from Tables 4.8 and 4.9 that it is again not possible to obtain better PL just by starting at a later stage. Hence, we must conclude that this cannot be the sole reason for the extra noise in the new systems.

The other kind of noise which could possibly be present is internal to the $\hat{G}$ function. If we examine closer how a $\hat{G}$ function is constructed, it will become apparent where all the unwarranted noise is coming from.

Since $\hat{G}_{i+1}$ is a joined-up function, its derivative $\hat{g}_{i+1}$ at the joint of two lines with different slopes is discontinuous. This means $\hat{f}^*_{i+1}$ is also discontinuous over its domain because:

$$\hat{f}^*_{i+1}(t) = \frac{d}{dt}\hat{F}^*_{i+1}(t)$$

$$= \hat{g}_{i+1}(\hat{F}_{i+1}(t))\hat{f}_{i+1}(t) \qquad (4.4.7)$$

Figure 4.4 is the plot of $\hat{G}_{278}$ for adapting LV on Musa's System SS3 data which has 188 u's values. This might look rather smooth but if we look at Figure 4.5 which is the derivative of $\hat{G}_{278}$, we see how 'spiky' it is. In fact the larger the number of u's, the more spiky $\hat{g}$ becomes, in which case, the adapted predictive density for the failure time is also becomming more discontinuous. As there is no apparent reason why the pdf of T_{i+1} should be discontinuous in such a fashion, the PL points out this flaw in the adaptive systems.

In the case of System SS3 data, the gain from correcting the bias in

Stage i	JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
60	73.580	73.409	73.022	72.542	71.294	72.542	72.542	71.639	71.924
70	143.593	143.494	142.882	142.120	140.635	142.120	142.120	141.101	141.574
80	215.410	215.669	214.769	213.895	212.639	213.895	213.895	213.255	213.766
90	294.487	295.140	294.006	292.918	293.344	293.562	292.918	293.318	293.582
100	377.702	378.464	377.060	374.652	376.638	376.173	374.698	376.443	376.759
110	458.548	459.042	457.444	452.562	453.751	454.146	452.608	453.482	453.851
120	537.944	538.225	536.437	529.442	528.924	531.025	529.488	528.654	529.025
130	624.160	624.203	622.444	615.300	617.282	616.883	615.347	616.350	616.536
135	668.944	669.147	667.267	660.061	663.715	661.644	660.107	663.348	663.212

Table 4.8. -log prequential likelihood of the respective prediction system at selected stages of System 1 data.

Stage i	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
60	77.294	77.356	76.910	73.951	74.835	73.951	73.921	75.453	75.556
70	149.269	149.529	150.038	146.572	147.803	146.573	146.550	147.957	148.457
80	222.968	220.555	222.620	222.541	223.647	222.541	222.538	219.722	219.785
90	305.369	303.411	303.540	302.641	306.581	304.624	302.643	303.878	304.616
100	389.930	392.141	392.086	389.896	388.443	390.513	389.557	389.894	387.019
110	471.203	474.660	474.846	474.166	471.135	471.378	472.768	470.041	468.941
120	550.554	558.058	558.855	555.166	549.723	550.056	554.483	549.851	548.909
130	639.199	649.416	663.298	644.162	641.162	637.016	643.466	638.591	639.184
135	680.743	693.229	693.978	690.742	689.099	682.513	690.044	685.545	688.851

Table 4.9. -log prequential likelihood of the adpated (joined-up function adaptor) prediction systems at selected stages of System 1 data.

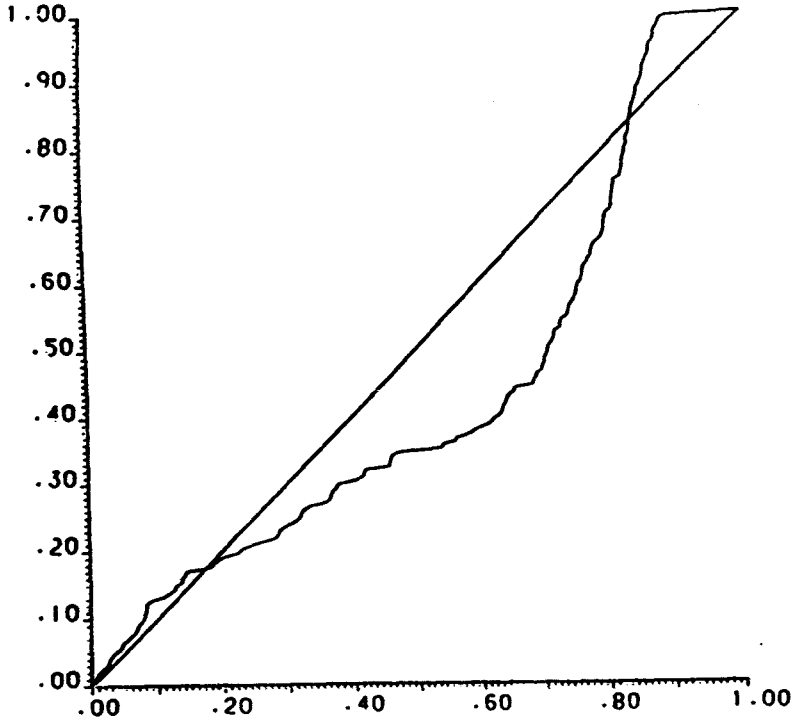


Fig. 4.4 The joined-up $\hat{G}_{278}$ based on 188u values for adapting LV on Musa's System SS3 data.

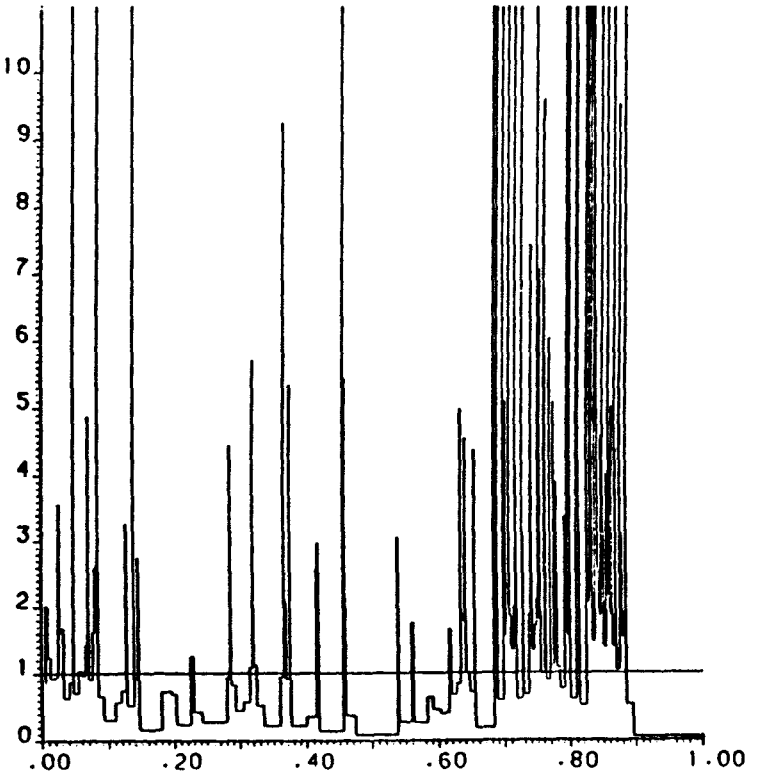


Fig. 4.5 The derivative of the joined-up $\hat{G}_{278}$ based on 188u values for adapting LV on Musa's System SS3 data.

the predictions is more than sufficient to compensate for the introduction of this internal noise, hence better PL is observed in most of the adaptive systems. Apparently this is not the case in most other situations.

The obvious solution is to use a smooth adapting function $\hat{G}$, which is the subject of the next Chapter.

CHAPTER 5

THE DETERMINATION OF THE PARAMETRIC SPLINE ADAPTOR AND THE ANALYSIS OF THE RESULTS OF THEIR APPLICATION

5.1. INTRODUCTION

In this Chapter, we shall present a non-parametric method of constructing a smooth adapting function $\hat{G}$. The function we have used is a parametric spline consisting of two suitably constrained least-squares cubic splines.

Two representations of a cubic spline will be given. The first is the redundant representation which has a clear physical interpretation. The second is the B-spline representation which is not as obvious as the first but has many advantages in practical applications. With the use of the B-spline representation, it is possible to fit an over-constrained least-square cubic spline efficiently and in a numerically stable way.

The parametric spline adaptor is then used to adapt those cases reported in Chapter 4. Detailed results are given in the last section.

5.2. THE SMOOTH ADAPTIVE CURVE

We have established in the previous Chapter that it is desirable to have a smooth adapting function $\hat{G}$. We shall look at the properties which such a smooth function must possess:

1) $\hat{G}$ is a function defined on (0,1) and the range of which is also (0,1), i.e.:

$$\hat{G} : (0,1) \rightarrow (0,1)$$

2) The derivative of $\hat{G}$ must be positive for all values within its domain, i.e.:

$$\hat{G}'(u) > 0 \quad \forall u \in (0,1)$$

3) $$\hat{G}(0) = 0 \text{ and } \hat{G}(1) = 1$$

The above conditions are automatically satisfied by the cdf of a random variable defined on the interval (0,1). This is hardly surprising since the u-plot is just the sample cdf of the u's. Therefore one could view the problem as one of obtaining a smooth estimate of the cdf of a random variable defined on (0,1), based on a finite random sample.

A possibility is to choose a parametric family of distributions and use the u's as data to estimate the unknown parameters. However, the family of distributions must be for a random variable which is defined on a finite

interval, otherwise a transformation on the variable would be required. Once the domain of the distribution is a finite interval, it would be an easy matter to make this interval into (0,1).

An example of such a family is the Beta distribution with two parameters. It has a pdf:

$$f(u) = \frac{\gamma(\alpha + \beta)}{\gamma(\alpha)\ \gamma(\beta)}\ u^{\alpha-1}(1 - u)^{\beta-1}$$

for $\alpha,\beta > 0$ and $u\epsilon(0,1)$. $\gamma(x)$ is the Gamma function which is defined as:

$$\gamma(x) = \int_0^\infty u^{x-1}e^{-u}\,du \qquad \text{for } x > 0$$

Although the shape of this distribution is relatively flexible among parametric distributions, it might still be unable to give a close fit to the shape of the joined-up adapting curve. See Figure 4.4 for one such adaptor.

Another more serious difficulty in using a Beta distribution, which is also true for many parametric distributions, is in the evaluation of the cdf at a given point. This can usually be done only by numerical approximations and can be very difficult with certain values of α and β. This means we will have difficulties calculating the u's or the predicted medians.

In view of the above, we can add the following two extra requirements on the smooth $\hat{G}$ function.

4) The function must be flexible enough to fit very different shapes.

5) It must be easy to compute the derivative and the value of $\hat{G}$ at any point in the interval (0,1).

Because most parametric distributions will have difficulty in satisfying conditions (4) and (5), we have not pursued the use of parametric distributions any further. Instead, we have chosen to use a parametric spline which is composed of two suitably constrained least-squares cubic splines.

A parametric spline is based on the following parametric representation. Take a general dependent variable y, say, with x being the independent variable. Let the function f define their relationship, i.e.:

$$y = f(x)$$

If we introduce a parameter p such that:

$$x = x(p)$$

we can also express y in terms of p:

$$y = y(p)$$

Then we will have a parametric representation of x and y in terms of p (x(p),y(p)). When x(p) and y(p) are splines, the resulting function is a parametric spline.

The parametric spline is widely used in computer graphics and computer aided design. The main reason for its use in the context of adaptive modelling is because of its flexibility. The shape in Figure 4.4 can easily be reproduced by a parametric spline.

Epstein (1976) has supported the use of the cumulative chord joining up the discrete data points as the parameter. If (x_i,y_i) for i = 1,..., r, denote the r pairs of data to which we want to fit the parametric spline, the cumulative chord is:

$$p_i' = p_{i-1}' + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \qquad i=1,\ldots,r \tag{5.2.1}$$

with $p_o' = 0$, $x_o = 0$ and $y_o = 0$. In our application, x_i is the i^{th} order statistic of the u's and y_i is the height of $\hat{G}$ at x_i. However, we have, for convenience, used the normalised cumulative chord:

$$p_i = \frac{p_i'}{p_r'} \qquad i=1,\ldots,r \tag{5.2.2}$$

so that both parametric functions will have domain [0,1].

Having introduced the parameter p, we now have two groups of data: (p_i,x_i) and (p_i,y_i) for i=1,...,r. To each group of data we shall fit a least-squares cubic spline. This spline function is constrained such that conditions (1) to (3) in section 5.2 are satisfied. It is clear that if the px spline and the py spline both satisfy conditions (1) to (3) the resulting xy function will also satisfy these conditions.

It might seem to be a waste of effort to use a parametric spline

because we could have simply used one instead of two such constrained least-squares spline for the xy function. Practical results, however, have shown that the shape of the $\hat{G}$ can be very fast changing. If we were to simply fit a cubic spline, we would have to use many knots, hence increasing the dimension of the problem (to be explained later), in order that the cubic spline could reproduce such characteristics in the adapting function. But the parametric spline can do so with fewer number of knots being used. Furthermore, the introduction of the parameter p helps to smooth out local oscillations in the data and as a result the parametric spline does not oscillate as much as the joined-up $\hat{G}$ function.

5.3. THE REDUNDANT REPRESENTATION AND THE LEAST-SQUARES SOLUTION OF THE CONSTRAINED CUBIC SPLINE

Since the parametric spline is made up of two cubic splines defined on [0,1], we shall derive in this section the redundant representation of a cubic spline on the interval [0,1]. Other variants of this representation could be found in Ahlberg et al (1967).

In order to define a cubic spline over the interval, it is necessary to choose a knot sequence:

$$0 = \lambda_0 < \lambda_1 \ldots\ldots < \lambda_m < \lambda_{m+1} = 1 \qquad (5.3.1)$$

where m is the number of interior knots in the sequence. This knot

sequence also defines (m+1) sub-intervals in [0,1]. The choice of knots will affect the fitted spline function. We shall discuss knot placement in more detail as we proceed.

Let z_j and w_j denote the function value and the second derivative of the spline at the j^{th} knot, for j=0,1...m+1. We can define the cubic spline in terms of these 2(m+2) quantities in the following way.

The property of a cubic spline is that it is continuous up to its second derivative over the entire [0,1] interval, and between two successive knots it is a cubic. Let $S_k(p)$ be the cubic spline between the knots λ_{k-1} and λ_k for k=1,...,m+1. Because $S_k(p)$ is a cubic for $p\epsilon[\lambda_{k-1},\lambda_k]$, this implies that the second derivative must be a straight line, i.e.:

$$S_k''(p) = \frac{(p-\lambda_{k-1})}{h_k} w_k + \frac{(\lambda_k-p)}{h_k} w_{k-1} \qquad (5.3.2)$$

where h_k is the k^{th} interknot spacing:

$$h_k = \lambda_k-\lambda_{k-1} \qquad k=1,\ldots,m+1 \qquad (5.3.3)$$

Integrating (5.3.2) gives:

$$S_k'(p) = \frac{(p-\lambda_{k-1})^2}{2h_k} w_k - \frac{(\lambda_k-p)^2}{2h_k} w_{k-1} + a_k \qquad (5.3.4)$$

and integrating again gives:

$$S_k(p) = \frac{(p-\lambda_{k-1})^3}{6h_k} w_k + \frac{(\lambda_k-p)^3}{6h_k} w_{k-1} + a_k(p-\lambda_{k-1}) + b_k \qquad (5.3.5)$$

where a_k and b_k are constants of integration.

Now $S_k(\lambda_{k-1}) = z_{k-1}$, so from (5.3.5) we have:

$$b_k = z_{k-1} - \frac{h_k^2}{6} w_{k-1} \qquad (5.3.6)$$

Similarly, $S_k(\lambda_k) = z_k$ and from (5.3.5) we have:

$$a_k = \frac{z_k}{h_k} - \frac{h_k}{6} w_k - \frac{b_k}{h_k} \qquad (5.3.7)$$

Combining (5.3.5) with (5.3.6) and (5.3.7) gives:

$$\begin{aligned} S_k(p) = {} & \frac{(p-\lambda_{k-1})}{6}\left[\frac{(p-\lambda_{k-1})^2}{h_k} - h_k\right] w_k \\ & + \frac{(\lambda_k-p)}{6}\left[\frac{(\lambda_k-p)^2}{h_k} - h_k\right] w_{k-1} \\ & + \frac{(p-\lambda_{k-1})}{h_k} z_k + \frac{(\lambda_k-p)}{h_k} z_{k-1} \quad k=1,\ldots,m+1 \end{aligned} \qquad (5.3.8)$$

If we utilise the continuity of derivative condition, i.e. $S_k'(\lambda_k) = S_{k+1}'(\lambda_k)$, we have:

$$\begin{aligned} & \frac{h_k}{6} w_{k-1} + \frac{(h_k + h_{k+1})}{3} w_k + \frac{h_k}{6} w_{k+1} \\ & = \frac{z_{k+1} - z_k}{h_{k+1}} - \frac{z_k - z_{k-1}}{h_k} \quad \text{for } k=1,\ldots,m \end{aligned} \qquad (5.3.9)$$

which forms m constraints on the z's and the w's. This is the reason why this representation is called <u>redundant</u> because instead of requiring 2(m+2) variables, we only need (m+4) variables to uniquely define the cubic spline for a given knot sequence.

If we have data (p_i,y_i) for $i=1,\ldots,r$, to which we want to fit a least-squares cubic spline, the problem is to find the unknowns (w_j,z_j) for $j=0,\ldots, m+1$, which minimise the residual sum of squares and also satisfy the m equations defined by (5.3.9). Note that if there are m interior knots in the knot sequence, there will be $(m+4)$ independent variables, therefore the maximum number of interior knots must not be greater than $(r-4)$. Thus, the larger the number of knots being used, the higher is the dimension of the minimisation problem. In fact, there are restrictions on the position of the knots such that the least-square solution is unique (Cox, 1975, Cox and Hayes, 1973). For our purpose, we will ensure that there is at least one data point between any two knots.

Corresponding to each p_i, let $\hat{y}_i$ denote the fitted value of y_i given by (5.3.8). Therefore the r vector of fitted value $\hat{y}$ can be written as:

$$\hat{y} = Aw + Bz \qquad (5.3.10)$$

where w and z are the $(m+2)$ vector of w_k and z_k respectively, A and B are the $(rx(m+2))$ matrix of coefficients of w and z respectively. The constraint equation (5.3.9) can also be written as:

$$Cw = Dz \qquad (5.3.11)$$

where C and D are the $(mx(m+2))$ matrix of the coefficients of w and z given by (5.3.9).

If there are no further constraints on the spline function, the least-squares problem has linear constraints and is:

$$\min_{w,z} \tfrac{1}{2}\|\hat{y} - y\|_2^2 \quad \text{subject to } Cw = Dz \tag{5.3.12}$$

We can write down the Lagrangian:

$$= \tfrac{1}{2}(Aw + Bz - y)^T (Aw + Bz - y) - \lambda^T(Cw - Dz) \tag{5.3.13}$$

where λ is the m vector of Lagrange multipliers. By differentiating w.r.t. w, z and λ and equating to zero, we get the following system of equations:

$$\begin{pmatrix} A^TA & A^TB & -C^T \\ B^TA & B^TB & D^T \\ -C & D & \underset{\sim}{0} \end{pmatrix} \begin{pmatrix} w \\ y \\ \lambda \end{pmatrix} = \begin{pmatrix} A^Ty \\ B^Ty \\ \underset{\sim}{0} \end{pmatrix} \tag{5.3.14}$$

which we can solve for the unknowns $\hat{w}$, $\hat{z}$ and $\hat{\lambda}$. However, in our application, it is necessary that the least-squares spline should satisfy conditions (1) to (3) of section 5.2.

To impose condition (3) presents no problem because these are linear equality constraints and we simply have to include:

$$z_o = 0 \quad \text{and} \quad z_{m+1} = 1 \tag{5.3.15}$$

in the equality constraints of (5.3.9).

Condition (1) is automatically observed if (2) and (3) are jointly satisfied because condition (2) will ensure the function to be monotonically increasing, thus the function is a one-to-one map with range [0,1].

To impose condition (2) causes considerable difficulty because this condition cannot be expressed in terms of a finite number of linear equations (Cox and Jones, 1985). To see this, we look at interval $[\lambda_{k-1},\lambda_k]$ in more detail.

Now $S_k'(p)$ is a quadratic over this interval $[\lambda_{k-1},\lambda_k]$ and condition (2) can be separated into two parts:

a) $S_k'(\lambda_{k-1})$ and $S_k'(\lambda_k)$ must both be positive. These two requirements can be expressed as two linear inequality constraints and can be handled without great difficulty.

b) The minimum of $S_k'(p)$, if it is within $[\lambda_{k-1},\lambda_k]$, must also be positive. This requirement, however, cannot be expressed as a linear inequality of the variables and cannot be imposed easily.

We have previously used the Nelder-Mead simplex search method (Nelder and Mead, 1962) to solve this non-linear inequality constrained least-squares problem, whenever the least-squares solution violates conditions (a) or (b) within any sub-intervals. This method is very heavy in computational terms whenever the search is invoked and is not a practical way of determining such a constrained cubic spline function. However, if we use the B-spline representation instead of the redundant representation of the cubic spline, we can find a practical solution to the spline fitting problem very efficiently.

5.4. THE B-SPLINE REPRESENTATION OF THE CUBIC SPLINE

Just as polynomials can be expressed as a linear combination of certain basis polynomials, such as Chebyshev, it is possible to express a spline in terms of a basis - the B-spline basis. We will show by using the B-spline representation, we can derive a numerically stable and efficient method for obtaining a practical solution to the constrained cubic spline fitting problem. Our discussion here is specifically related to our problem only, more detailed and general discussions on various aspects of B-splines and their applications could be found in Cox (1975).

In order to define the B-spline basis for a cubic spline, it is necessary to extend the knot sequence to include 3 extra exterior knots at either end of the interval [0,1]. Therefore, the new knot sequence is:

$$0 = \lambda_{-3} \ldots = \lambda_0 < \lambda_1 \ldots < \lambda_{m+1} = \lambda_{m+2} \ldots = \lambda_{m+4} = 1 \qquad (5.4.1)$$

The reason for the above choice of exterior knots will become apparent later.

Given a knot sequence the normalised B-splines satisfy the following recurrence relation:

$$N_{n,k}(p) = \left[\frac{p - \lambda_{k-n}}{\lambda_{k-1} - \lambda_{k-n}}\right] N_{n-1,k-1}(p) + \left[\frac{\lambda_k - p}{\lambda_k - \lambda_{k-n+1}}\right] N_{n-1,k}(p) \qquad \text{for } n > 1 \qquad (5.4.2a)$$

with

$$N_{1,k}(p) = \begin{cases} 1 & \text{if } p \in [\lambda_{k-1}, \lambda_k) \\ 0 & \text{otherwise} \end{cases} \qquad (5.4.2b)$$

where n is the order of the B-spline, which is 4 in the case of cubic B-splines. In order to define the B-splines on the entire interval [0,1], we have to extend the last sub-interval to be closed on the right hand side as well, which means:

$$N_{i,m+1}(p) = \begin{cases} 1 & \text{if } p\epsilon[\lambda_m,\lambda_{m+1}] \\ 0 & \text{otherwise} \end{cases} \qquad (5.4.2c)$$

Curry and Schoenberg (1966) has shown that the normalised B-splines defined by (5.4.2) for $k=1,\ldots,m+n$, form a basis for splines of order n over the interval with which the knot sequence $(\lambda_0,\ldots,\lambda_{m+1})$ is prescribed. Thus if S(p) is such a spline, it has the B-spline representation:

$$S(p) = \sum_{j=1}^{m+n} c_j N_{n,j}(p) \qquad \text{for } p\epsilon[\lambda_0,\lambda_{m+1}]$$

where the c_j's are the B-spline coefficients.

In the case of a cubic spline over the interval [0,1], this means the spline function S(p) can be defined as:

$$S(p) = \sum_{j=1}^{m+4} c_j N_{4,j}(p) \qquad p\epsilon[0,1] \qquad (5.4.3)$$

From (5.4.2), it is clear that $N_{4,j}(p)$ is non-zero if the value of p falls within the interval $(\lambda_{j-4},\lambda_j)$ for $j=1,\ldots,m+1$. Therefore, if $p\epsilon[\lambda_{k-1},\lambda_k)$ for $k=1,\ldots,m$ or $p\epsilon[\lambda_{k-1},\lambda_k]$ for $k=m+1$, $N_{4,k}(p),\ldots,N_{4,k+3}(p)$ will be the only 4 non-zero normalised B-splines. Thus:

$$S(p) = \sum_{j=k}^{k+3} c_j N_{4,j}(p) \qquad (5.4.4)$$

when p lie within the k^{th} sub-interval $[\lambda_{k-1},\lambda_k)$ with the last $[\lambda_m,\lambda_{m+1}]$ also closed on the right hand side.

The r vector of fitted values $\hat{y}$ can now be written as:

$$\hat{y} = Nc \tag{5.4.5}$$

where N is the (rx(m+4)) matrix of non-zero normalised B-splines corresponding to the data vector p and c is the (m+4) vector of B-spline coefficients. The least squares fitting problem is to:

$$\min_c \| \hat{y} - y \|_2^2$$

which is simply:

$$\min_c \|Nc - y\|_2^2 \tag{5.4.6}$$

Note that (5.4.6) is much simpler compared to the least-square problem (5.3.12) when the redundant representation is being used. Furthermore, because of (5.4.4), the non-zero elements of the matrix N have a band structure with bandwidth 4. This structure can be taken advantage of when solving the least squares problem (5.4.6). Details of how this is done are given in the next section.

Recall that our cubic spline has to be constrained to pass through the points (0,0) and (1,1). If we use the recurrence relation of the normalised B-splines (5.4.2) and (5.4.4) to evaluate S(0) and S(1) we will find that:

$$\left.\begin{aligned} S(0) &= c_1 \\ \text{and} \quad S(1) &= c_{m+4} \end{aligned}\right\} \tag{5.4.7}$$

because the exterior knots at both ends of the interval [0,1] are chosen to

be coincidental. The advantage of this choice of exterior knots is now apparent: condition (3) of section 5.2, i.e. S(0)=0 and S(1)=1 will then be requiring:

$$c_1 = 0 \quad \text{and} \quad c_{m+4} = 1 \tag{5.4.8}$$

which is only a pair of simple equality constraints.

As for the positivity requirement on the derivative of the spline within the entire interval, it is still not possible to express it in a finite number of linear equations. Hanson (1979) has suggested solving the least-squares problem by imposing the positive derivative constraint at a finite number of points within the interval. This set of points is built-up iteratively until the spline has positive derivative over the entire interval. Apart from a brief description similar to the one we have just given, Hanson has not disclosed further algorithmic details on how this could be done. On the surface of his suggestion, we do not envisage much practical value in his approach because there is too much vagueness in how the set of points could be updated in each iteration, and it is doubtful if the constrained solution obtained in this way is optimal.

One of the properties of the B-spline representation is that the derivative of the cubic spline can be expressed in a way similar to that of the spline in terms of the c_j's. By differentiating (5.4.3) it can readily be shown that:

$$S'(p) = \sum_{j=1}^{m+3} c_j^{(1)} N_{3,j}(p) \qquad p \in [0,1] \tag{5.4.9a}$$

with

$$c_j^{(1)} = 3\,\frac{c_{j+1} - c_j}{\lambda_j - \lambda_{j-3}} \qquad j=1,\ldots,m+3 \qquad (5.4.9b)$$

in the cubic case. Because there can only be 3 non-zero normalised B-splines - $N_{3,k}(p)$, $N_{3,k+1}(p)$ and $N_{3,k+2}(p)$ - when $p\in[\lambda_{k-1},\lambda_k)$, the derivative $S'(p)$ will only be a product sum of these three terms. Note also that the derivative at 0 is simply $c_1^{(1)}$ and at 1 is $c_{m+3}^{(1)}$, again as a result of our using coincidental exterior knots.

Furthermore, the derivative satisfies <u>locally</u> the following:

$$\min c_j^{(1)} \leqslant S'(p) \leqslant \max c_j^{(1)}$$

when $p\in[\lambda_{k-1},\lambda_k)$, the min and max are taken over j=k, k+1, k+2 (Cox, 1975). Therefore, if we ensure that all the $c_j^{(1)}$'s are positive, it would guarantee the positivity of the derivative over the entire interval. As the $c_j^{(1)}$'s are linear in the B-spline coefficients, this requirement can be translated into imposing (m+3) linear inequality constraints on the least-squares problem (5.4.6).

Now the <u>optimally constrained</u> problem will have two linear inequality constraints at the end points of the interval, i.e. $c_1^{(1)} > 0$ and $c_{m+3}^{(1)} > 0$. But within the interval the inequality constraints are not linear in the c_j's. Furthermore, all the $c_j^{(1)}$'s, except the first and the last, do not necessarily have to be positive in order that the derivative is positive over the interior of the interval. To insist on all the $c_j^{(1)}$'s being positive might <u>over-constrain</u> the fitted spline. The derivative of the two constrained splines might look like those in Figure 5.1.

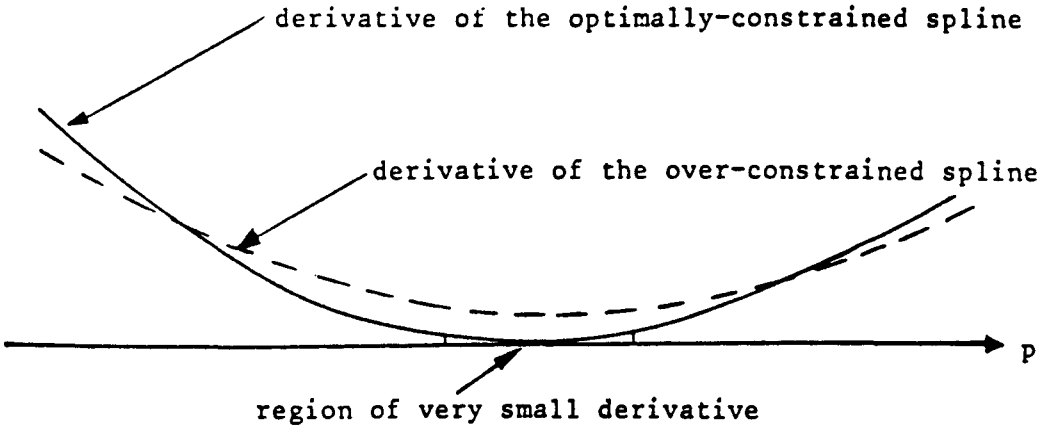


Figure 5.1.

However, in our application this is not a disadvantage because the optimally constrained spline might entail a region with very small derivative like that of Figure 5.1. This corresponds to a shoulder in the cubic spline. With the over-constrained spline, the presence of such a region is unlikely, thus the spline will tend to be smoother. A by-product of using the over-constrained spline is that we can avoid having very small prequential likelihood if the spline in Figure 5.1 is the py function, or very big prequential likelihood if it is the px function.

In private communication, Cox (1986) has also recommended the over-constrained spline as the most practical solution to fitting monotonic splines with order greater than 3.

5.5. METHOD FOR THE DETERMINATION OF THE OVER-CONSTRAINED CUBIC SPLINE

The least-squares problem which we have to solve is:

$$\min_{c} \|Nc - y\|_2^2 \qquad (5.5.1a)$$

subject to the equality constraints:

$$\begin{aligned} c_1 &= 0 \\ c_{m+4} &= 1 \end{aligned} \qquad (5.5.1b)$$

and the inequality constraints:

$$A^T c > \underset{\sim}{0} \qquad (5.5.1c)$$

where A is the ((m+4)x(m+3)) matrix with elements:

$$a_{ij} = \begin{cases} -\dfrac{3}{\lambda_j - \lambda_{j-3}} & i=j \\ \dfrac{3}{\lambda_j - \lambda_{j-3}} & i=j+1 \\ 0 & \text{otherwise} \end{cases} \qquad (5.5.1d)$$

for $j=1,..,m+3$ and $\underset{\sim}{0}$ is a $(m+3)$ vector of zeros. Note that $A^T c = c^{(1)}$ which is the $(m+3)$ vector of $c_j^{(1)}$'s, and the j^{th} column of A is made up of the non-zero coefficients defining $c_j^{(1)}$ as given in (5.4.9b).

The approach we shall adopt to solve (5.5.1) consists of two stages:

(1) The sub-problem (5.5.1a) subject to (5.5.1b) is solved and c^* is used to denote the solution.

(2) If all the inequality constraints in (5.5.1c) are satisfied, i.e.:

$$A^T c^* > \underset{\sim}{0}$$

then the solution to the constrained least-squares problem (5.5.1), $\hat{c}$, is equal to c^*. Should any of (5.5.1c) not be satisfied, we will find an adjustment vector ε^* such that the constrained solution:

$$\hat{c} = c^* + \varepsilon^*$$

The sub-problem corresponding to the first stage can be solved by the orthogonal triangularization of N. Here we shall only give an outline of the principles involved. Further details could be found in Cox (1975, 1980) and an Algol implementation could be found in Cox and Hayes (1973).

Suppose Q is an orthogonal (rxr) matrix such that:

$$Q^T Q = I$$

and

$$Q^T N = \begin{bmatrix} R \\ \underset{\sim}{0} \end{bmatrix} \qquad (5.5.2)$$

where R is an upper-triangular matrix whose order is identical to the rank of N. (This is one of the criteria governing the choice of the m interior knots: the rank of N must be full, i.e. (m+4), in order that the B-spline coefficients can be uniquely determined. The condition on the position of a knot sequence which will guarantee N to have full rank and methods to deal with rank deficiency could be found in Cox (1975, 1982). Here we shall insist upon having at least one data point within any two non-coincidental knots, which is more than sufficient to guarantee N to have full rank).

We can write:

$$e = Nc - y \qquad (5.5.3a)$$

and let

$$e = Q^T y \qquad (5.5.3b)$$

Furthermore, we shall separate the vector e into e_1 and e_2 such that e_1 has $(m+4)$ and e_2 has $(r-m-4)$ elements. Then:

$$e = Q^T \begin{bmatrix} Rc - e_1 \\ - e_2 \end{bmatrix} \qquad (5.5.4)$$

which means:

$$e^T e = \|Rc - e_1\|_2^2 + \|e_2\|_2^2 \qquad (5.5.5)$$

because of the orthogonality of Q. It is clear from (5.5.5) that $e^T e$ is minimised if:

$$\|Rc - e_1\|_2^2 = 0$$

or

$$Rc = e_1 \qquad (5.5.6)$$

and the residual sum of squares is simply $\|e_2\|_2^2$. Since N has full rank, R is $((m+4)x(m+4))$ and upper-triangular which means c can be solved easily by back-substitutions.

To impose the constraints (5.5.1b), we note that:

$$e = Nc - y = N \begin{bmatrix} 0 \\ \tilde{c} \\ 1 \end{bmatrix} - y$$

$$= \tilde{N}\tilde{c} + N_{m+4} - y \qquad (5.5.7)$$

where $\tilde{c}^T = (c_2, c_3, \ldots, c_{m+3})$, $\tilde{N}$ is the $(rx(m+2))$ matrix composed of the second to the $(m+3)^{th}$ columns of N and N_{m+4} is the $(m+4)^{th}$ column of N. Therefore we have:

$$e = \tilde{N}\tilde{c} - \tilde{y} \tag{5.5.8}$$

where $\tilde{y} = y - N_{m+4}$ and $\tilde{c}$ can be determined by the orthogonal triangularization of $\tilde{N}$ just as before. Thus the solution of the first stage sub-problem is:

$$c^* = \begin{bmatrix} 0 \\ \tilde{c} \\ 1 \end{bmatrix} \tag{5.5.9a}$$

with

$$\tilde{R}\tilde{c} = \tilde{e}_1 \tag{5.5.9b}$$

where $\tilde{R}$ comes from the factorization:

$$\tilde{Q}^T\tilde{N} = \begin{bmatrix} \tilde{R} \\ \underset{\sim}{0} \end{bmatrix} \tag{5.5.9c}$$

and $\tilde{e}_1$ from:

$$\tilde{Q}^T\tilde{y} = \begin{bmatrix} \tilde{e}_1 \\ \\ \tilde{e}_2 \end{bmatrix} \tag{5.5.9d}$$

The dimension of $\tilde{R}$ is ((m+2)x(m+2)) and $\tilde{e}_1$ is a (m+2) vector. From now on we shall drop the ~ and all matrices and vectors will correspond to the constrained situation of (5.5.7) to (5.5.9).

The factorization of N is performed via repeated use of Givens rotation matrix which is:

$$Q_{ij} = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & i & & j & & \end{array} \\ \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \ddots & & & & & \\ & & & & C & & S & & \\ & & & & & 1 & & & \\ & & & & & & \ddots & & \\ & & & & -S & & C & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{bmatrix} \end{array}$$

where C and S denote Cose and Sine, respectively, and θ is chosen such that the ji^{th} element of the matrix:

$$N' = Q_{ij}N$$

is zero. It is easy to verify that the appropriate values of C and S are given by:

$$\left.\begin{aligned} C &= n_{ii}/h \\ S &= n_{ji}/h \end{aligned}\right\} \qquad (5.5.10)$$

where

$$h = (n_{ji}^2 + n_{ji}^2)^{1/2}$$

and n_{ij} denotes the ij^{th} element of N. Here we assumed $n_{ji} \neq 0$, which ensures that h is non-zero. If n_{ji} is already zero then no rotation is needed.

The effect of multiplying the Givens matrix onto N is that the:

i^{th} row of N' = C x(i^{th} row of N) + S x (j^{th} row of N)

j^{th} row of N' = C x(j^{th} row of N) - S x (i^{th} row of N)

with the ji^{th} element being reduced to zero, i.e.:

$$n'_{ii} = h; \quad n'_{ji} = 0 \qquad (5.5.10a)$$

$$\left.\begin{aligned} n'_{ik} &= C\, n_{ik} + S\, n_{jk} \\ n'_{jk} &= -S\, n_{ik} + C\, n_{jk} \end{aligned}\right\} \quad k=i+1,\ldots,n \qquad (5.5.10b)$$

Since there are 4 multiplications in (5.5.10b) it is termed the 4 multiplication rule.

By using a series of Q_{ij} matrices, we can triangularize N into the required form:

$$\begin{bmatrix} R \\ \underset{\sim}{0} \end{bmatrix}$$

The advantage of this method is that the conditioning of the matrix N is not worsened in the factorization process because of the use of orthogonal matrices. This method is unconditional stable (Wilkinson, 1963) and, most importantly, we can take advantage of the band structure in N and economize considerably on the amount of work required for the triangularization.

The matrix N corresponding to our constrained sub-problem has the following structure:

$$N = \begin{bmatrix} xxx & & & & & & & & & \\ & xxx & & & & & & & & \\ & xxxx & & & & & & & & \\ & & xxxx & & & & & & & \\ & & xxxx & & & & & & & \\ & & & xxxx & & & & & & \\ & & & xxxx & & & & & & \\ & & & & \ddots & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & \ddots & & \\ & & & & & & & & xxxx & \\ & & & & & & & & xxxx & \\ & & & & & & & & xxxx & \\ & & & & & & & & xxxx & \\ & & & & & & & & xxx & \\ & & & & & & & & & xxx \end{bmatrix} \qquad (5.5.11)$$

with x denoting a non-zero element. As a result, R would have the following structure:

$$R = \begin{bmatrix} xxxx & & & & & & & & & \\ & xxxx & & & & & & & & \\ & & xxxx & & & & & & & \\ & & & \ddots & & & & & & \\ & & & & \ddots & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & \ddots & & \\ & & & & & & & & xxxx & \\ & & & & & & & & xxx & \\ & & & & & & & & xx & \\ & & & & & & & & & x \end{bmatrix} \qquad (5.5.12)$$

The rows of N are rotated into R, successively. To show how this is done, we use the following pictorial illustration which Cox (1980) has used. In each of these diagrams, (5.5.13) and (5.5.14), the first block represents the situation immediately before the reduction process. Each subsequent block illustrates R and the row of N immediately after its leading element has been reduced to zero using a Givens rotation matrix. The

corresponding elements of e_1 and y are to the right in each block. In each case a previously established zero is represented by a period, a zero element that has just been created is denoted by 0, an element which is affected by the rotation is defined by m and a non-zero element which is unaffected during the last rotation is denoted by x. (5.5.13) shows how a 4 elemented row of N is reduced to zeros sequentially. A 3-elemented row is processed in the same way.

```
xxxx x      mmmm m      xxxx x      xxxx x      xxxx x
 xxx x       xxx x       mmm m       xxx x       xxx x
  xx x        xx x        xx x        mm m        xx x      (5.5.13)
   x x         x x         x x         x x         m m

xxxx x      0mmm m      .0mm m      ..0m m      ...0 m
```

The corresponding element of y is rotated into e_1 by the same sequence of Givens rotations. During the reduction process, if the element of R corresponding to the non-zero leading element of the row of N is zero, the two rows are <u>swapped</u>, including the corresponding e_1 element and the y element, and we continue to process the next row of N. This situation is depicted in (5.5.14) when the third row of N is being processed. We have assumed that this is also the first 4-elemented row in N, i.e. $p_3 \epsilon [\lambda_1, \lambda_2)$.

```
xxx. x    mmmm m    xxxx x    xxxx x
 xx. x     xx. x     mmm m     xxx x
  .. .      .. .      .. .      mm m      (5.5.14)
   . .       . .       . .       . .
xxxx x    0mmm m    .0mm m    ..00 0
```

Because of the band structure of N, it means that there will be no more than 4 rotations required for the reduction of each row of N. Furthermore, the constraints $c_1 = 0$ and $c_{m+4} = 1$ can be incorporated

without extra effort being required to obtain the constrained solution to this sub-problem. The residual sum of squares is simply the cumulated value of the square of the rotated y elements (now being elements of e_2) corresponding to those situations of (5.5.13), but excluding those of (5.5.14) where a row swapped is involved.

Once N is triangularized, c^* can be determined easily by back-substitutions. The next step is to check whether:

$$A^T c^* > \underset{\sim}{0} \tag{5.5.15}$$

If condition (5.5.15) is satisfied then the solution to the constrained least-squares problem (5.5.1) is $\hat{c} = c^*$. Otherwise, it would be necessary to invoke the second stage process which involves finding a vector δ^* such that $\hat{c} = c^* + \delta^*$. Note that since the first and the last B-spline coefficients are already fixed, the corresponding elements of δ^* must be zero. Therefore, returning to the ~ notation used earlier, we have:

$$\delta = \begin{bmatrix} 0 \\ \tilde{\delta} \\ 0 \end{bmatrix} \tag{5.5.16a}$$

and

$$\hat{c} = \begin{bmatrix} 0 \\ \tilde{c}^* + \tilde{\delta}^* \\ 1 \end{bmatrix} \tag{5.5.16b}$$

where $\tilde{\delta}^*$ has (m+2) elements.

Before we begin discussing how $\tilde{s}^*$ can be found, there are two preliminary adjustments which we shall make. Firstly, we shall rewrite the inequality constrains (5.5.1c) into:

$$A^T \begin{bmatrix} 0 \\ \tilde{c} \\ 1 \end{bmatrix} \geq d \qquad (5.5.17)$$

where d is a (m+3) vector with elements equal to some small constant ϵ. We have chosen ϵ to be 10^{-6}. The effect of replacing (5.5.1c) by (5.5.17) is that we shall restrict the $c_j^{(1)}$'s, hence the derivative of the cubic spline, to be bigger than ϵ rather than zero. Since the derivative, if it exists, of a function with the parametric representation of (x(p),y(p)) is:

$$\frac{dy}{dx} = \frac{dy}{dp} \Big/ \frac{dx}{dp} \qquad (5.5.18)$$

Thus by setting a lower bound on the derivative of the respective splines, we have avoided possible division by too small a value or having too small a derivative when combining two such functions to form the parametric spline.

Secondly, we have to find the $\tilde{A}$ matrix and $\tilde{d}$ vector corresponding to the variable vector $\tilde{c}$. It is easy to see that:

$$A^T \begin{bmatrix} 0 \\ \tilde{c} \\ 1 \end{bmatrix} = \tilde{A}^T\tilde{c} + \begin{bmatrix} \underset{\sim}{0} \\ \dfrac{3}{\lambda_{m+3}-\lambda_m} \end{bmatrix} \geq d \qquad (5.5.19)$$

which can be expressed as:

$$\tilde{A}^T\tilde{c} \geq \tilde{d} \qquad (5.5.20)$$

where $\tilde{A}$ is composed of the 2nd to the $(m+2)^{th}$ row of A and the $(m+3)^{th}$ element of $\tilde{d}$ is:

$$(\epsilon - \frac{3}{\lambda_{m+3} - \lambda_m})$$

We can now formally specify the second stage sub-problem as:

$$\min_{\tilde{c}} \tfrac{1}{2}\|\tilde{R}\tilde{c} - \tilde{e}_1\|_2^2 \quad \text{subject to} \quad \tilde{A}^T\tilde{c} \geq \tilde{d} \tag{5.5.21}$$

The $\|e_2\|_2^2$ term is left out because it is just a constant. Again we shall drop the ~ notation and all the matrices and vectors mentioned below will correspond to the situation of (5.5.21).

The method we have used to solve the quadratic programming problem is iterative. It requires solving a series of sub-problems of the form:

$$\min_{c} \tfrac{1}{2}\|Rc - e_1\|_2^2 \quad \text{subject to} \quad \hat{A}^Tc = \hat{d} \tag{5.5.22}$$

where $\hat{A}$ is $((m+2)xt)$ and $\hat{d}$ is a t vector for $t < (m+3)$. $\hat{A}$ is composed of the columns of A corresponding to the t active constraints in (5.5.22): the j^{th} constraint is said to be active if $c_j^{(1)} = \epsilon$. This sub-problem is solved in the following way.

First we define the Lagrangian:

$$= \tfrac{1}{2}(Rc - e_1)^T(Rc - e_1) - \lambda^T(\hat{A}^Tc - \hat{d}) \tag{5.5.23}$$

where λ is the t vector of Lagrange multipliers. Differentiating (5.5.23) w.r.t. the c_j's and equating to zero yields the following set of equations:

$$R^TRc - R^Te_1 = \hat{A}\lambda \tag{5.5.24}$$

Differentiating (5.5.23) w.r.t. the λ_j's and equating to zero yields another set of equations:

$$\hat{A}^T c = \hat{d} \tag{5.5.25}$$

One way of obtaining the solution of (5.5.24) and (5.5.25), c' and λ' is to solve:

$$\begin{pmatrix} R^TR & -\hat{A} \\ \hat{A}^T & 0 \end{pmatrix} \begin{pmatrix} c \\ \lambda \end{pmatrix} = \begin{pmatrix} R^Te_1 \\ \hat{d} \end{pmatrix} \tag{5.5.26}$$

This approach, however, does not take advantage of the triangular structure of R and also requires the inversion of a ((m+2+t)x(m+2+t)) matrix. However, the method we have used, as suggested by Cox (1975), takes full advantage of the structure of R and the unknowns are found in an efficient and numerically stable way.

Clearly, we can express the vector of solution c' as:

$$c' = c^* + \delta' \tag{5.5.27}$$

where δ' can be viewed as an adjustment vector to the solution of the first stage sub-problem. If we substitute $c = c^* + \delta$ into (5.5.24) and (5.5.25) we shall have:

$$R^TRc^* - R^Te_1 + R^TR\delta = \hat{A}\lambda \tag{5.5.28a}$$

and

$$\hat{A}^T\delta = \hat{d} - \hat{A}^Tc^* \tag{5.5.28b}$$

Since

$$Rc^* = e_1$$

therefore (5.5.28) becomes:

$$R^T R\varepsilon = \hat{A}\lambda \qquad (5.5.29a)$$

and

$$A^T\varepsilon = d' \qquad (5.5.29b)$$

where $d' = \hat{d} - \hat{A}^T c^*$.

From (5.5.29a),

$$\varepsilon = (R^T R)^{-1}\,\hat{A}\lambda \qquad (5.5.30)$$

which when substituted into (5.5.29b) gives:

$$\hat{A}^T(R^T R)^{-1}\,\hat{A}\lambda = d' \qquad (5.5.31)$$

To solve (5.5.31) for λ', we first find the ((m+2)xt) matrix V from:

$$R^T V = \hat{A} \qquad (5.5.32)$$

because:

$$V^TV = \hat{A}^T(R^TR)^{-1}\hat{A}$$

Therefore, in terms of V, (5.5.31) is:

$$V^TV\lambda = d' \qquad (5.5.33)$$

The matrix V is then triangularised:

$$Q_V{}^TV = \begin{bmatrix} R_V \\ \underset{\sim}{0} \end{bmatrix}$$

by the multiplication of the orthogonal matrix $Q_V{}^T$ and (5.5.33) becomes:

$$R_V{}^TR_V\lambda = d' \qquad (5.5.34)$$

The vector λ' can now be found by forward-substitutions:

$$R_V{}^Tu = d' \qquad (5.5.35a)$$

for vector u and then by back-substitutions:

$$R_V\lambda' = u \qquad (5.5.35b)$$

Once λ' is found, $\mathcal{S}'$ can be solved from (5.5.29a) again by forward- and back-substitutions.

The amount of work involved in finding $\mathcal{S}'$ for each sub-problem (or each iteration) might seem to be very heavy by this approach. However, we have adopted an updating technique by Gill et al (1972) which only changes one column of $\hat{A}$ in every iteration, thus V does not have to be recalculated from the beginning in each iteration and the triangularization of which only requires little extra effort. The remaining work all involves trivial forward- and back-substitutions.

This updating technique utilises the property that if:

$$\hat{c} = \begin{bmatrix} 0 \\ c' \\ 1 \end{bmatrix}$$

i.e. the solution to the quadratic programming problem defined by (5.5.21) has been found, then λ' corresponding to the t' active constraints are all positive and there is no violation of the inactive constraints. Should the j^{th} constraint be violated, i.e. $c_j^{(1)} < \epsilon$ we shall set that constraint to be active by including the j^{th} column of A into $\hat{A}$ and the corresponding element of d into $\hat{d}$ and solve for the new $\mathcal{S}'$ and λ'.

Let A_j denote the j^{th} column of A. The next step in adding a constraint is to find the corresponding enlarged V matrix. If A_j is always the last column of the new $\hat{A}$, then the new V matrix will also have an extra new (t+1) column with the rest of the columns unchanged. This new column of V is given by:

$$R^T v_{t+1} = A_j \qquad (5.5.36)$$

by forward-substitutions.

Once the new V matrix is determined, we shall have to triangularize it. Let $Q_V^{(k-1)T}$ be the orthogonal matrix for the triangularization of the old V matrix, $V^{(k-1)}$, where the superscript in brackets is used to denote the iteration number and the present iteration is k. If we multiply $Q_V^{(k-1)T}$ onto $V^{(k)}$ we shall have:

$$Q_V^{(k-1)T} V^{(k)} = \begin{bmatrix} R_V^{(k-1)} & u \\ \underset{\sim}{0} & z \end{bmatrix} \qquad (5.5.37a)$$

where:

$$\begin{matrix} u \\ z \end{matrix} \Big) \begin{matrix} (t+1) \\ (m+1-t) \end{matrix} = Q_V^{(k-1)T} v_{t+1} \qquad (5.5.37b)$$

All we have to do now is to rotate the vector z into a vector of zeros and:

$$R_V^{(k)} = \left[R_V^{(k-1)} \quad Gu \right] \qquad (5.5.38)$$

where G is the product of the sequence of Givens rotation matrices required for the reduction. (5.5.39) is a pictorial illustration when t=3 and m=5. The characters used here bear the same meaning as those in (5.5.13) and (5.5.14).

$$
\begin{array}{l}
\left.\begin{array}{l} x\,x\,x\,x \\ x\,x\,x \\ x\,x \\ x \end{array}\right\} u \\
\left.\begin{array}{l} x \\ x \\ x \end{array}\right\} z
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ x\,x \\ m \\ 0 \\ x \\ x
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ x\,x \\ m \\ . \\ 0 \\ x
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ x\,x \\ m \\ . \\ . \\ 0
\end{array}
\qquad (5.5.39)
$$

The $Q_V^{(k-1)}$ matrix is also updated in the process and stored for use in the next iteration. Once $R_V^{(k)}$ is found, we can proceed to determine λ' and ε'.

At the end of an iteration if none of the inactive constraints are violated, but a Lagrange multiplier is negative, we shall release this constraint in the next iteration. To release a previously active constraints, we have to delete a column in $\hat{A}$, hence a column in V and the corresponding element in $\hat{d}$. The effect on R_V when a column of V is being deleted is that R_V is no longer upper triangular. Again we shall use a series of Givens rotations to triangularize R_V. (5.5.40) illustrates how the triangularization is performed when t=s and the constraint corresponding to the second column of $\hat{A}$ is being deleted.

$$
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ x\,x\,x \\ x\,x \\ x
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ m\,m\,m \\ 0\,m\,m \\ x\,x \\ x
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ .\,m\,m \\ 0\,m \\ x
\end{array}
\qquad
\begin{array}{c}
x\,x\,x\,x \\ x\,x\,x \\ .\,x\,x \\ .\,m \\ 0
\end{array}
\qquad (5.5.40)
$$

Again Q_V is also updated and stored for use in the next iteration.

Should there be more than one violation of the constraints, the one with the biggest violation will be set active. Similarly, if there are more than one Lagrange multiplier being negative, the constraint with the most

negative multiplier will be released. This will be repeated until we find the ε', such that c' does not violate the constraints and the Lagrange multipliers λ' are all positive. Then $\varepsilon^* = \varepsilon'$ and:

$$\hat{c} = \begin{bmatrix} & 0 & \\ c^* & + & \varepsilon^* \\ & 1 & \end{bmatrix} \qquad (5.5.41)$$

which is the solution to the constrained least-squares problem (5.5.1).

The whole fitting procedures were coded into 12 Fortran 77 subroutines on an IBM PC-AT. In the next section, we shall report the results of using parametric splines to adapt our predictions systems.

5.6. QUALITY OF THE PREDICTIONS BY PARAMETRIC SPLINE ADAPTIVE PREDICTION SYSTEMS

Here in this section, we shall present the results of adapting our 9 prediction systems on 7 data sets using parametric spline adaptors. The number of interior knots (m) we have used is 3 in all the following examples, i.e. 4 sub-intervals within [0,1]. This number was first chosen arbitrarily in our experimental fit, using the Nelder-Mead search method, of an optimally constrained least-squares spline. Results of this have been published in Chan (1986). Therefore, when we adopt the over-constrained version, we have chosen to use the same number of knots so that we can compare the effect of over-constraining the least-squares splines.

These 3 interior knots are placed within (0,1) such that a roughly equal number of data points are distributed in each of the 4 sub-intervals. The reason for doing this is that if the data points are very unevenly distributed amongst the sub-intervals, the fitted spline might fail to capture the shape characteristic in a sub-interval with many data, and oscillates excessively in a sub-interval with few data points.

Considerable effort has been devoted to devising knot placement strategies in fitting spline functions (see Cox, 1982, for some suggestions). Here we shall contend that the results we have achieved could possibly be improved upon, if another strategy for knot placement is used.

On the basis of our results, we found that it has negligible effect on the predictions when the splines of the parametric spline adaptor are over-constrained. This is partly due to the fact that introducing the parameter p has a smoothing effect on the raw data, which makes it less likely for the constituent functions to oscillate within [0,1] and have negative gradient.

Table 5.1 summarizes the u-plot and y-plot KS distances before and after adapting the same examples used in Chapter 4 with a parametric spline adaptor. Comparing the distances here with those by using a joined-up adaptor given in Table 4.3, we can see that they are very similar in terms of their significance levels. In the case of JM, the spline adaptor has worse u^*- and y^*-plot distances for System 3 data. We have already commented on the fact that on this data set, JM has $\hat{N}$ equal to the number of failures seen at various stages. The joined-up adaptor

DATA (n)		JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
Sys.1 (86)	u	.2049E	.1871E	.1773E	.0982A	.1567D	.1123A	.0982A	.1504D	.1457D
	u*	.1168B	.1197B	.1277B	.0511A	.0794A	.0507A	.0526A	.1027A	.1053A
	y	.1156B	.1148B	.1190B	.0795A	.1029A	.0904A	.0793A	.1148B	.1173B
	y*	.1109A	.1126A	.1102A	.0852A	.0762A	.0715A	.0853A	.0878A	.0916A
Sys.2 (31)	u	.2604D	.2325C	.2181C	.1518A	.2317C	.1554A	.1518A	.2388D	.2219C
	u*	.1595A	.1665A	.1747A	.1488A	.1421A	.1428A	.1561A	.1305A	.1203A
	y	.1858A	.1853A	.1989B	.1898B	.1620A	.1772A	.1743A	.1325A	.1451A
	y*	.1635A	.1514A	.1629A	.1865A	.1877B	.2087B	.1551A	.1867A	.1947B
Sys.3 (18)	u	.7038E	.3354D	.2705B	.1877A	.3556D	.2556B	.1531A	.4260E	.3908E
	u*	.6939E	.2522B	.2701B	.1067A	.1347A	.1949A	.1067A	.2058A	.1847A
	y	.6808E	.3900E	.4445E	.2234A	.2012A	.3075C	.2430A	.1112A	.1135A
	y*	.6473E	.3108D	.3101D	.1882A	.1987A	.2015A	.1963A	.1628A	.1497A
Sys.4 (30)	u	.1711A	.2185C	.1328A	.1143A	.1415A	.1712A	.1211A	.1955B	.2014B
	u*	.1333A	.1576A	.1551A	.1466A	.1905B	.1333A	.1362A	.2199C	.2162B
	y	.4647E	.1399A	.1989B	.3418E	.4887E	.2709D	.2695D	.2420D	.2010B
	y*	.4487E	.1249A	.1788A	.4766E	.4691E	.4480E	.4580E	.2093B	.1747A
Sys.6 (38)	u	.2924E	.3010E	.2812E	.2845E	.2856E	.2853E	.2845E	.1658A	.1731B
	u*	.0787A	.0806A	.0850A	.0819A	.1039A	.0812A	.0840A	.1531A	.1210A
	y	.3969E	.3486E	.3870E	.4017E	.4010E	.3978E	.4026E	.2020C	.2069C
	y*	.2708E	.2549D	.4241E	.2818E	.3010E	.2715E	.2764E	.2063C	.2120C
SS3 (173)	u	.2717E	.2713E	.2705E	.2645E	.2596E	.2717E	.2704E	.2382E	.2372E
	u*	.0820B	.0822B	.0782A	.0901B	.0916B	.0859B	.0846B	.0834B	.1006C
	y	.1273E	.1379E	.1263E	.1435E	.1835E	.1291E	.1300E	.0346A	.0500A
	y*	.0573A	.0693A	.0560A	.0632A	.1016C	.0571A	.0557A	.0352A	.0452A
BAe. (112)	u	.0775A	.0726A	.0697A	.0713A	.1270D	.0763A	.0655A	.1039B	.1151C
	u*	.0731A	.0809A	.0728A	.0826A	.0974A	.0730A	.0707A	.0853A	.0812A
	y	.0890A	.0787A	.0906A	.0793A	.0744A	.0873A	.0790A	.0673A	.0687A
	y*	.0725A	.0680A	.0733A	.0690A	.0656A	.0717A	.0704A	.0671A	.0721A

Table 5.1. Kolmogorov-Smirnov distance of the u-plot and y-plot of the respective prediction systems before and after * adapting (parametric spline adaptor). See (4.2.3.6) for the corresponding significance level.

accumulates a discrete component at the origin and adapts a zero u to that value. Thus u^* will have a non-zero value which will help the u^*-plot to become more uniform but the corresponding predictor is practically useless. The parametric spline $\hat{G}$ function, however, is constrained to be zero at the origin, therefore u^* will be zero whenever u is zero. Hence, the u^*- and y^*-plot KS distances corresponding to the spline adaptor are not better than those of the joined-up adaptor because of the presence of this group of zeros. This is also the case for System 4 data.

In the case of L, since $\hat{N}=i$ occurred 3 times between stages 25 and 27 amongst the 28 predictions on System 3 data, the difference in the KS distances is not as apparent as in the case of JM^*, which has 12 occurrences of $\hat{N}=i$ amongst the 28 predictions on the same data. But with System 4 data, the y^*-plot KS distance of the spline adapted predictions is much worse than the corresponding distance of the prediction from a joined-up adaptor.

In the majority of cases, the two adapting methods have produced very similar u^*- and y^*-plot KS distances, with the exception of System 6 and SS3 data where the spline adaptor has produced even better results. Thus on the basis of these distances, the spline adaptor seems to be at least as capable of adapting predictions as the joined-up adaptor, and is more reliable when a situation like JM on System 3 data occurs. Furthermore, by using the spline adaptor we no longer have to concern ourselves with having infinite prequential likelihood when there are zero failure times in the data, which will be the case if a joined-up adaptor is used.

The use of the parametric spline has also removed the internal noise which is present in a joined-up adaptor. Corresponding to the example given in Chapter 4, i.e. the joined-up $\hat{G}$ function for LV^* on System SS3 data and its derivative in Figure 4.2 and Figure 4.3, the parametric spline adaptor for the same situation is given in Figure 5.2 and its derivative in Figure 5.3. We can clearly see the shape of $\hat{G}$ in Figure 4.2 being reproduced by the spline without unwarranted oscillation in the middle section. The shape of the derivative in Figure 5.3 will be difficult for most approximating polynomials or functions to reproduce. Since the parametric spline is made up of two cubic splines, it encompasses a much wider class of functions and has a unique combination of flexibility and smoothness.

In Chapter 4 we have given, in Tables 4.6 and 4.8, the predicted median and the prequential likelihood of the raw predictors at selected stages of System 1 data. We shall now make use of this information and perform a detailed analysis on the effect of our spline adaptive procedure on this data set.

From Table 5.2 we can clearly see that the magnitude of predicted medians from the spline adaptor are very close to those from the joined-up adaptor (Table 4.7). When compared with the raw medians in Table 4.6, we can see that the adapted medians are in closer agreement than before. It is more informative for the purpose of comparison to plot the raw and adapted medians against the stage number i. These plots are grouped under Appendix 4 for ease of comparison here and with the results in the next Chapter.

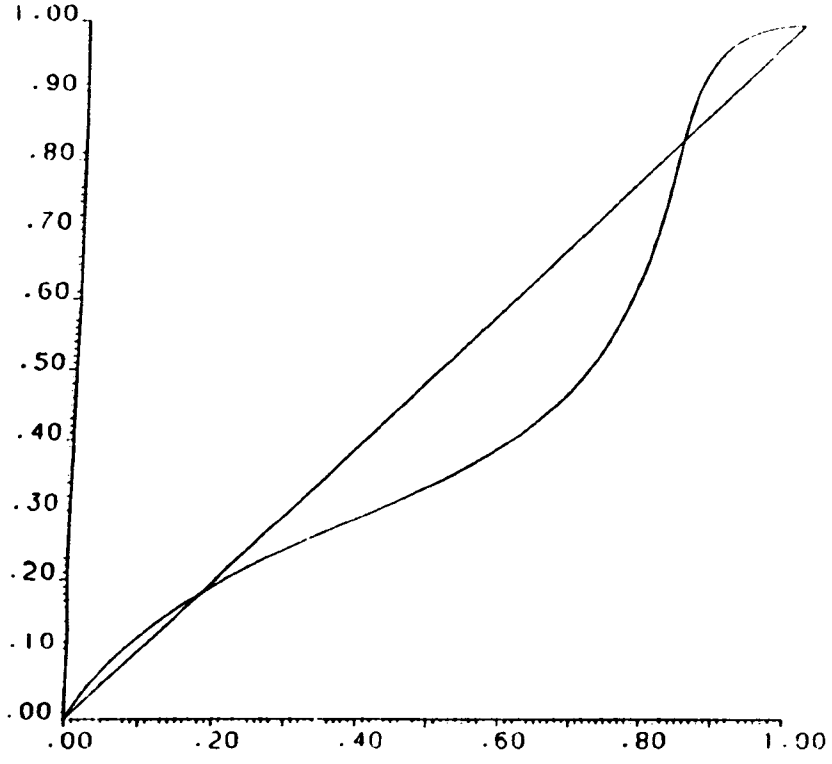


Fig. 5.2 The parametric spline $\hat{G}_{278}$ based on 188u values for adapting LV on Musa's System SS3 data.

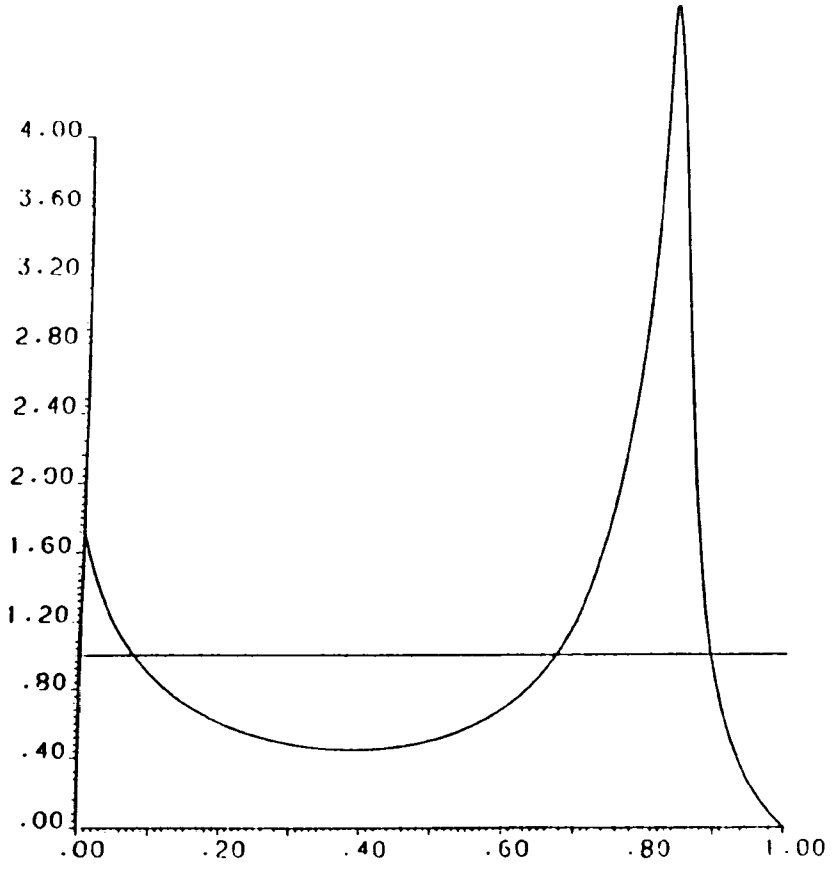


Fig. 5.3 The derivative of the parametric spline $\hat{G}_{278}$ based on 188u values for adapting LV on Musa's System SS3 data.

Stage i	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
60	259	259	264	250	250	250	250	255	252
70	321	327	334	309	305	309	309	296	299
80	383	385	394	349	342	349	349	340	342
90	799	797	812	593	537	557	593	558	553
100	1482	1484	1509	934	824	1003	927	740	715
110	1168	1175	1196	937	840	853	930	790	767
120	968	956	972	919	838	852	913	848	817
130	1641	1611	1638	1188	1058	1110	1181	894	890

Table 5.2. Predicted median for System 1 data at selected stages by respective adapted (parametric spline adaptor) prediction systems.

Stage i	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
60	74.155	73.874	74.083	72.820	72.386	72.820	72.820	72.490	72.318
70	142.571	142.383	142.364	141.364	140.875	141.365	141.364	140.812	140.687
80	216.185	215.896	216.015	214.564	213.785	214.564	214.564	213.847	213.626
90	293.965	293.910	293.915	293.585	293.218	293.084	293.585	293.380	293.282
100	375.727	375.652	375.702	374.648	374.690	375.327	374.726	375.653	376.114
110	454.340	454.367	454.563	451.552	451.466	451.552	451.628	452.615	453.167
120	531.869	532.076	532.311	528.334	527.460	528.150	528.430	528.163	528.625
130	616.966	617.547	617.604	613.415	612.998	612.771	613.475	614.067	614.789
135	662.644	663.210	663.298	658.489	658.035	657.867	658.535	659.846	660.642

Table 5.3. -log prequential likelihood of the adapted (parametric spline adaptor) prediction systems at selected stages.

In Plot A4.1a, we have the raw predicted medians for System 1 data from the 9 different raw prediction systems. We can clearly see that after stage 80 there is wide disagreement amongst several groups of raw predictions. Roughly we have JM, BJM, GO consistently predicting very large values, DU, LV and KL consistently predicting very low values and the remaining systems predicting values in-between. Incidentally, the u-plot of the first 3 systems lies entirely above the 45° line and the u-plot of the last 3 systems lies predominantly below the 45° line. Thus there is evidence that the first 3 systems are optimistic and the last 3 are pessimistic.

It is also clear from the median plot that after stage 80 the medians from JM, BJM, GO and L (not after stage 100 in the last case) become more noisy than those from LNHPP and MO, and even more so than those from DU, LV and KL.

Inspecting the PL of the raw predictions at selected stages given in Table 4.8 in the previous Chapter, it seems that LNHPP and MO are most likely to be the closest to the truth (these predictions are almost identical because LNHPP behaved like MO nearly throughout the entire sequence of predictions). The closeness here is in the sense of the whole distribution rather than just the median. The median is only a point in the distribution and two different distributions could have identical medians, but differ in other distributional aspects, for example in the spread of the distribution.

In Plot A4.1b we can see the predicted medians of LV^*, KL^* and DU^* have all been adjusted upwards and are now in much closer agreement with $LNHPP^*$ and MO^* than before. The predicted medians of JM^*, BJM^* and GO^* after stage 80 are still bigger than the others, but they have all been adjusted downwards by the adapting process. Evidence exists in the u^*-plot to support the observation that JM^*, BJM^* and GO^* are still optimistic, although less seriously than their raw counterparts. Figure 5.4 is the u-plots of JM predictions on System 1 data before and after adapting. We can see that the raw u-plot lies entirely above the line of unit slope. The u^*-plot is still exhibiting this behaviour but is not as severe as before.

The switching behaviour of L between JM, MO and itself between stages 80 and 100 is still very visible in the predicted medians of L^*. It is also visible that the predicted medians of MO^* and $LNHPP^*$ are more noisy than before. The introduction of some noise into the adaptive system is the cost we have to pay for having to estimate the $\hat{G}$.

The prequential likelihood of the adapted prediction at selected stages are given in Table 5.3. If we compare these with the raw PL in Table 4.8, we can see that in PL terms, all the adapted predictions are better than before. It is also fair to say that on the basis of PL, there is little to choose between the last 6 adaptive systems for this data set. The first 3 adaptive systems have nearly identical and poorer PL. Thus confirming our observation earlier from the median plot and u^*-plot that these predictions are still optimistic for this data.

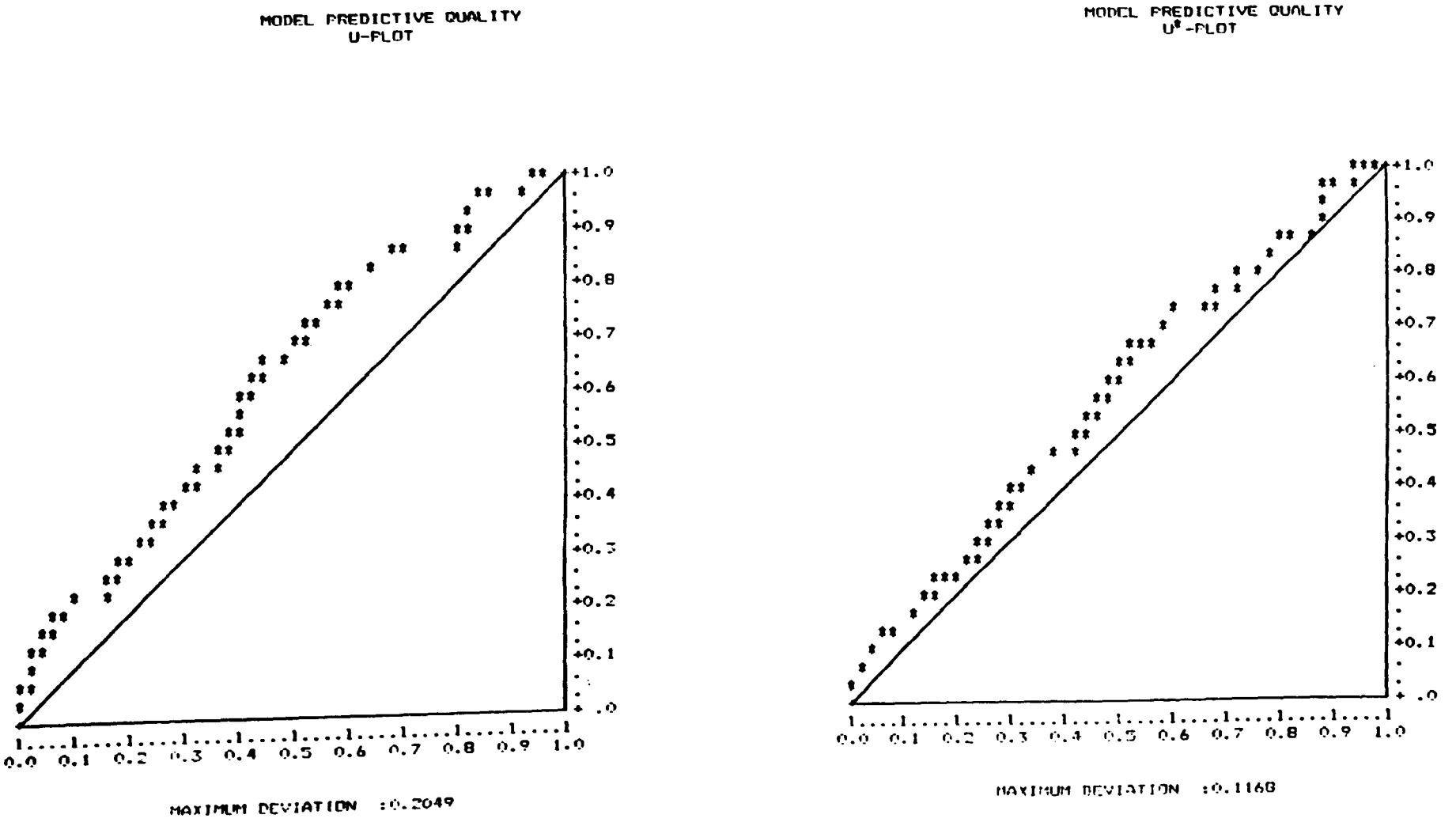


Fig. 5.4 The u-plot before and after * adapting JM predictions on System I data.

Among the 6 better systems L^* has the best PL. But from the median plot, L^* has very nosiy predicted medians especially between stages 80 to 100. If we check through the PL of L^* given in Table 5.3, we can see that prior to stage 100, the PL of L^* is indeed worse than those of DU^*, MO^* and $LNHPP^*$ because of the extra noise. This means L^* has gained more than the lost ground in the last 35 predictions. This is confirmed in the median plot, where the L^* predictions after stage 100 are indeed much less noisy than before, because the raw L predictions have switched to those of MO.

Thus we conclude that this method of adapting has improved the quality and accuracy of some of the raw predictions on System 1 data. In the case of good raw predictions like those from MO and LNHPP, this adapting process has not worsened the quality of these predictions to any appreciable extent. The price we have to pay for adapting these predictions, i.e. the introduction of noise into the predictors, is either insignificant or being out-weighed by the gain from correcting the bias.

The next set of results which we shall analyse is on System 2 data. From Table 4.4, the raw predictions from MO have the best PL. This is closely followed by LNHPP because their predictions are again identical except for those between stages 27 to 35. This is clearly visible from Plot A4.2a too.

The behaviour of the predicted medians here is similar to those in System 1 data. We can see from the median plot that DU, LV and KL predictions are small compared to the others. They all have a u-plot

which lies mostly below the 45° line, thus they are pessimistic for this data. JM, BJM and GO predictions are usually relatively big and all of them have a u-plot which lies entirely above the 45° line, hence they are optimistic. The predicted medians from MO and LNHPP lie in between these two groups, while the medians of L switched between its limits. On the basis of the PL, it seems most likely that the MO predictions are closest to the truth.

The PL of the adapted predictions in Table 5.4 confirmed this observation because DU* has the best PL amongst all the adapted predictions and it is clear from Plot A4.2b that its predicted medians are indeed very close to those of MO's. The predicted medians of MO* have become noisier but their locations have not been changed by any substantial amount.

After stage 35, the adapted medians are in closer agreement than before. We can see from Plot A4.2a and Plot A4.2b, that the optimistic predictions have been effectively adapted downwards and the pessimistic ones being adapted upwards. Before stage 35, the adapted medians of optimistic systems like JM, BJM and GO have all been adapted upwards. This behaviour is very difficult to avoid because we have to estimate the adapting function on the basis of the past u values. Sampling fluctuation in the u's could lead to a wrong $\hat{G}$ being estimated, especially when there are relatively few u's available. Therefore it is safer to start adapting when more u's become available. Unfortunately there is limitation to this strategy in practice because the length of the data might not permit us to do so.

DATA (n)	JM*	BJM*	GO*	MO*	DU*	L*	LNHPP*	LV*	KL*
Sys.1 (86)	662.264	663.210	663.298	658.489	658.035	657.867	658.535	659.846	660.642
Sys.2 (31)	282.137	283.786	284.635	279.330	278.943	280.247	281.210	279.959	281.042
Sys.3 (18)	∞	175.099	174.463	166.624	165.604	∞	167.892	164.890	166.727
Sys.4 (30)	∞	238.166	241.809	242.592	251.467	∞	244.770	238.672	235.982
Sys.6 (38)	200.288	196.536	200.223	198.885	197.797	199.380	199.241	194.352	193.943
SS3 (173)	2210.49	2211.47	2210.43	2210.69	2213.09	2211.21	2211.03	2214.06	2216.80
BAe (112)	644.116	643.596	644.290	643.457	643.189	644.208	644.042	643.459	643.761

Table 5.4. -log prequential likelihood of the respective adapted (parametric spline adaptor) prediction systems.

A common feature in those cases where adapting has improved the predictions is that they all have good y-plot and poor u-plot. For MO and LNHPP, adapting has not brought about any improvement because their u-plots are already very good in the first place. In the latter case, the PL has even gone slightly worse. Nonetheless, the adapting procedure does improve some of the predictions on this data set, especially at the later stages and has brought them into closer agreement.

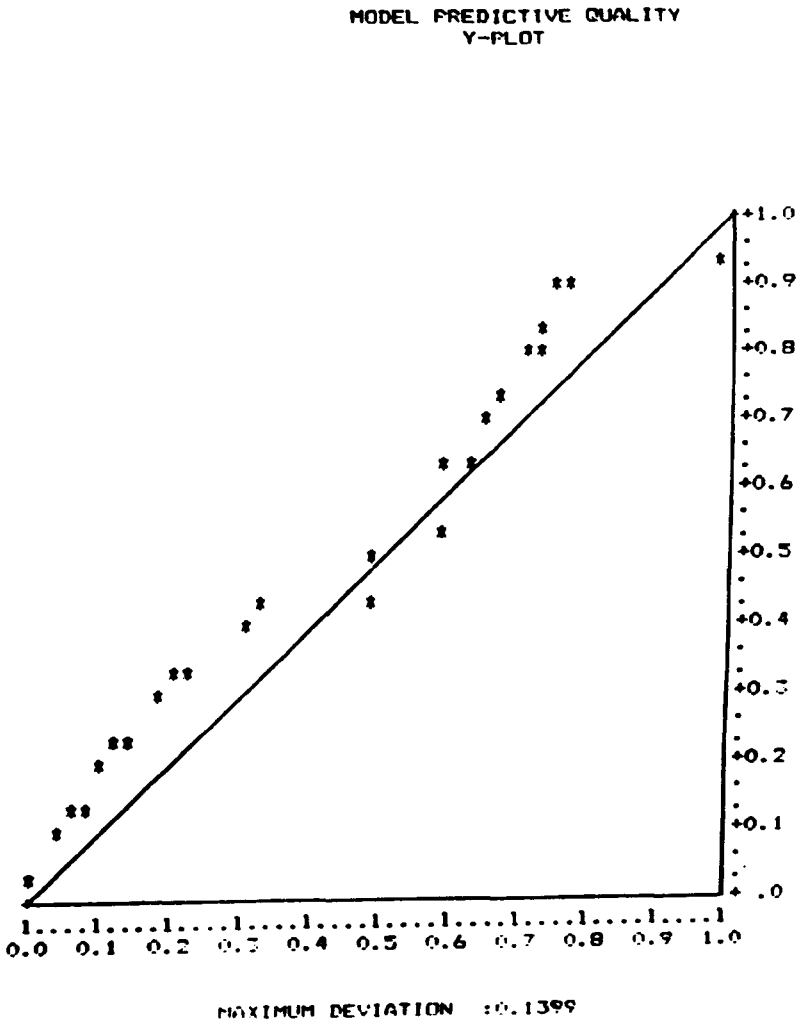
Looking at the u-plot and y-plot KS distances of System 3 data in Table 5.1 one would immediately expect DU, LV and KL to be the most suitable candidates for adapting. Indeed, the PL of these adapted predictions are all better than before. The raw MO and LNHPP predictions are very good in the first place and adapting them has led to worse PL. In the cases of JM and L we have $\hat{N}=i$ occurring to both of them and adapting cannot change the degenerate nature of these raw predictions. In the remaining cases of BJM and GO, their y-plots are very poor which means the trend in the data has not been captured adequately by these raw predictions. In fact the y-plots are slightly reverse s-shaped: the early section of the plot lies above and the later section lies below the 45° line. This means the early u's are consistently smaller than the later ones. Therefore, the u-plots are non-uniform not only because of being biased, but also because of an ill-captured trend, and adapting cannot be expected to improve these predictions.

From the median plots - Plot A4.3a and Plot A4.3b - we can see that the raw predicted medians corresponding to JM, BJM, GO and L are extremely noisy. Adapting has little effect on the predicted medians from

these systems. It is clear that the predicted medians from MO* and LNHPP* are more noisy than before and their PL's have become worse. The predicted medians from DU*, LV*, and KL*, however, are nearer to those from MO, and all of these 4 sets of predictions have very close PL values. Therefore, we conclude that these 4 sets of predictions are most likely to be closest to the truth for this data set. In the other cases, the raw predictions are either degenerate or have trend in the y-plots, which cannot be improved upon by adapting.

For System 4 data, we can see from Table 4.1 that the y-plot distance of most of the raw predictions are poor. Even in the case of BJM and GO where the distance is insignificant, the y-plots have clear visible trend. Figure 5.5 contains the y-plots of these two sets of predictions. We can clearly see that the y-plot in either case is broken at two points: one near the middle and one near the end. This is also very clear from the predicted medians in Plot A4.4a. Furthermore, the y-plots are slightly reverse s-shaped.

Given these observations, we cannot expect the adapting procedure to be able to improve upon these predictions. Indeed, the u*-plot distance is even worse than before for DU* and LV*. In the case of GO*, although the u*-plot is better, the PL has deteriorated, which must be due to the systematic bias in the raw predictions. The only case where there is a gain in PL is DU*, but this PL value is still very much lower than those of LV and KL. Incidentally, LV* and KL* have noiser predicted medians which are of similar magnitude to the raw medians. This accounts for the worse PL after LV and KL are adapted. Thus we conclude that our



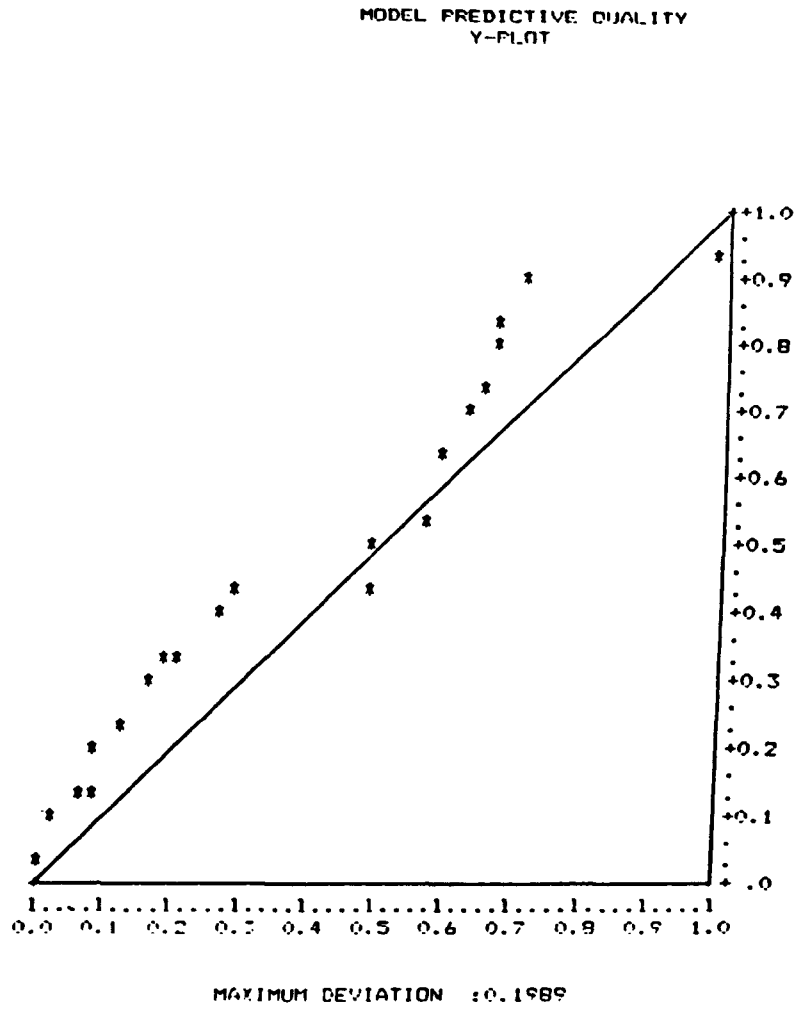


Fig. 5.5 The y-plot of BJM (left) and GO (right) predictions on System 4 data.

adapting approach cannot improve the worse raw predictions for this data because they themselves are not only being biased. It seems that the exponential failure time is unsuitable for this data set. The Pareto distribution of LV and KL, which has more likelihood for extreme values, seems to be much better.

On checking the y-plot distances of the raw predictions on System 6 data, one might think immediately that this would be like the previous example because all the y-plot distances are very poor. But if we examine the y-plots in more detail, we will find that the poor y-plot is due to a very large observation t_{69} (refer to Appendix 3 for a listing of the data).

This cause of a poor y-plot is quite different from the systematic bias we have seen in the previous data set. Let us consider the example of adapting the predictions with a reverse s-shaped y-plot. First of all, we shall assume that the predictions are optimistic at the early stages and pessimistic later. Therefore, during the early stages the $\hat{G}$ function will lie mostly above the 45° line because of the optmistic (small) u's seen so far. When the change in systematic bias occurs, the $\hat{G}$ will be of the wrong shape until there are sufficient pessimistic (big) u's to influence $\hat{G}$ to be below the 45° line. It is conceivable that $\hat{G}$ might still be wrong at the end of the prediction sequence because the u's simply averaged out and together they look perfectly uniform. As a result, the adapted predictions are clearly wrong. This situation is observed in System 4 data where the u-plots are all very good but the y-plots have visible trend.

Therefore our approach of estimating G in the light of past u's would work best when the G functions from stage to stage are stationary, i.e. the shape of G should not vary from one stage to another. Should the G's be non-stationary, the fact that we know its shape is changing does not enable us to estimate it, because of the lack of information concerning its new shape. Thus in practice, our requirement of a good y-plot is not strictly to have a non-significant KS distance, but an absence of trend in the y-plot itself. If this is accompanied by a steady shaped and non-uniform u-plot, we can expect the adaptive procedure to improve the raw predictions.

In the case of System 6, where the poor y-plot is caused by an exceptionally large inter-failure time, the effect on the shape of the u-plot might not be significant. In this particular case, the effect of this data on the shape of the adaptor is further diminished because this large value is very near the end of the data stream, which means there are already quite a number of u's in the basis for the estimation of G.

If we refer to the u^*-plot KS distances in Table 5.1, we will find all the distances are now dramaticaly improved to be non-significant even at 20%, the y^*-plot distances are still poor in general, may be except LV and KL and their adapted predictions, because of the large inter-failure time, t_{89}.

The PL has agreed with the KS distances that the adapted predictions are better except for LV and KL, which have the best PL amongst all raw and adapted predictions. Although the PL of LV^* and KL^* are worse than

their raw counterparts, they are still the best amongst all adapted predictions. The u*-plot for LV* and KL* are mainly above the 45° line which indicates that these adapted predictions are optimistic for this data. This plus the extra noise, which is evident in the median plot (Plot A4.5b), in the adapted predictions is responsible for the deterioration in the PL after adapting. As for the other adapted predictions, we can see from Plot A4.5a and Plot A4.5b that they are now in remarkable agreement with the raw predicted medians of LV and KL. Once again, the adapting procedure has improved the biased predictions for this data.

The next data set we shall analyse is System SS3 which is also the biggest data set in our study. The u-plot and y-plot distances are very poor for the first seven sets of raw predictions. The y-plot for the last two sequences of predictions are very good, but their u-plots are poor.

The predicted medians in Plot A4.6a revealed extreme disagreement between these two groups of predictors. The first group has extremely large median values and those from the latter group are much smaller. Examining the shape of the u-plots shows that the first group of predictions are very optmistic and the latter pessimistic although to a lesser extent.

Plot A4.6b shows the adapted medians. It is very clear that the pessimistic predictions are adapted upwards, at the cost of much more noise in the new medians, and the optimistic ones are adapted downwards considerably, also at a cost of having more noise in the new medians as evident in the median plot. However, there is now much closer agreement among these adapted predictions than before.

The u^*-plot and y^*-plot distances have improved all round. The gain in PL is most significant in the first group of predictors all of which has a prequential likelihood ratio of around e^{90} against their raw counterparts over these 173 predictions. The gain by LV^* and KL^* is appreciable but less substantial than the first group of predictors. In fact, the PL of LV^* and KL^* turned out to be worse than the others, most likely caused by the extra noise being introduced as evident in the adapted median plot. But still it is incredible that their PL should be so close after 173 predictions.

To demonstrate the extent of bias in these predictions being removed, Figure 5.6 and Figure 5.7 are the u-plots for JM and LV before and after adapting. We can see in both cases that the shape of the u-plot has been dramatically changed and both u^*-plots are much more uniform than before. Thus we conclude that the adapting procedure has been very successful in improving the preditions on this data set.

Finally, we shall look at the effect of adapting the BAe data. The u-plot and y-plot distances for this data are all good, may be with the exception of the u-plot of DU. Therefore, we cannot expect to gain much by adapting, at least in the cases of JM, BJM, GO, MO, L and LNHPP. Another reason for not expecting to gain anything by adapting is that the PL for all these raw predictions are incredibly close after 112 predictions, and we have yet to succeed in improving upon a set of predictions with good u-plot and y-plot and relatively good PL by adapting.

Although the u^*-plot distances showed improvement in the cases of DU, LV and KL, the PL registered no gain for any of the adapted

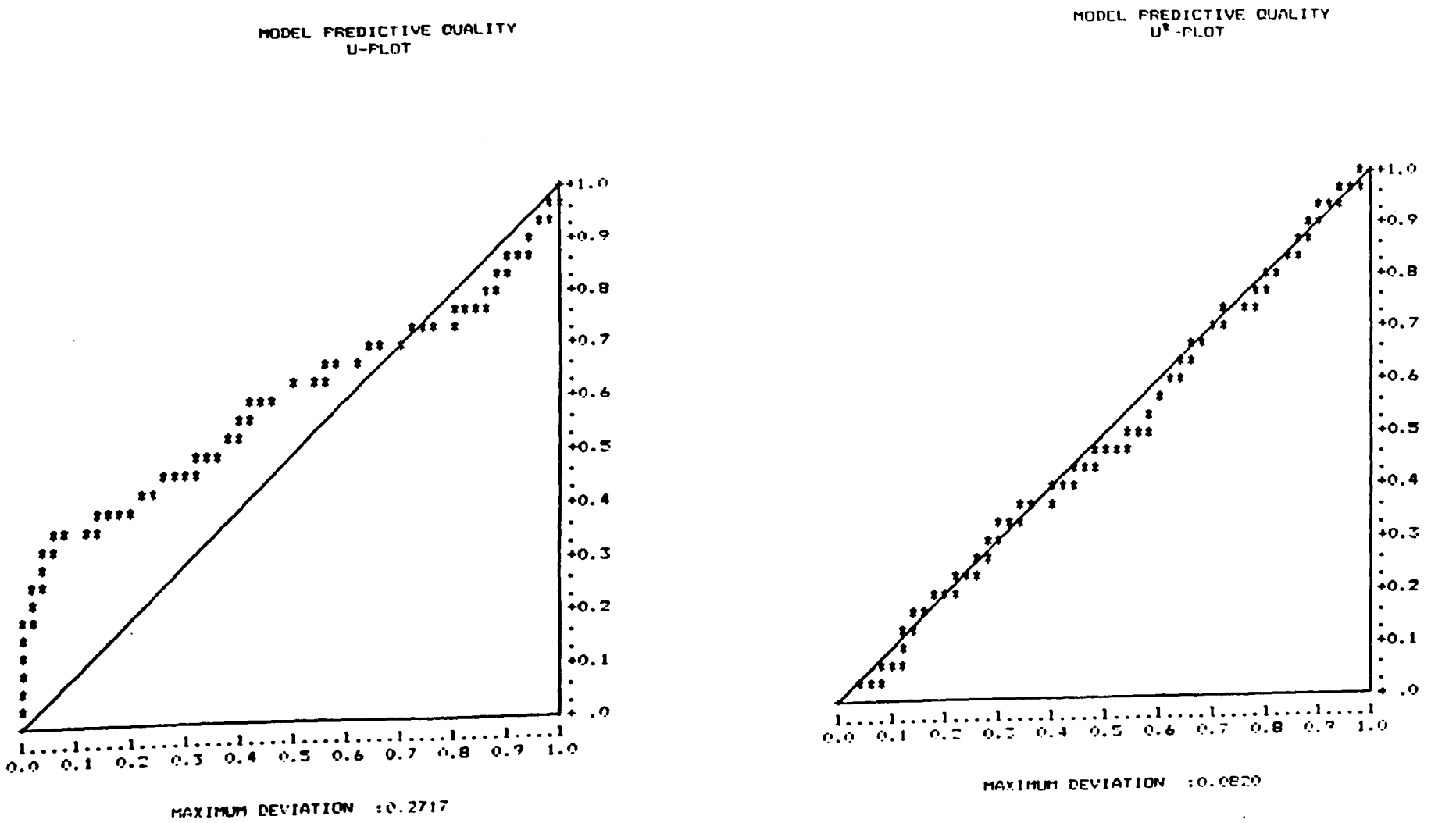


Fig. 5.6 The u-plot before and after * adapting JM predictions on System SS3 data using a parametric spline.

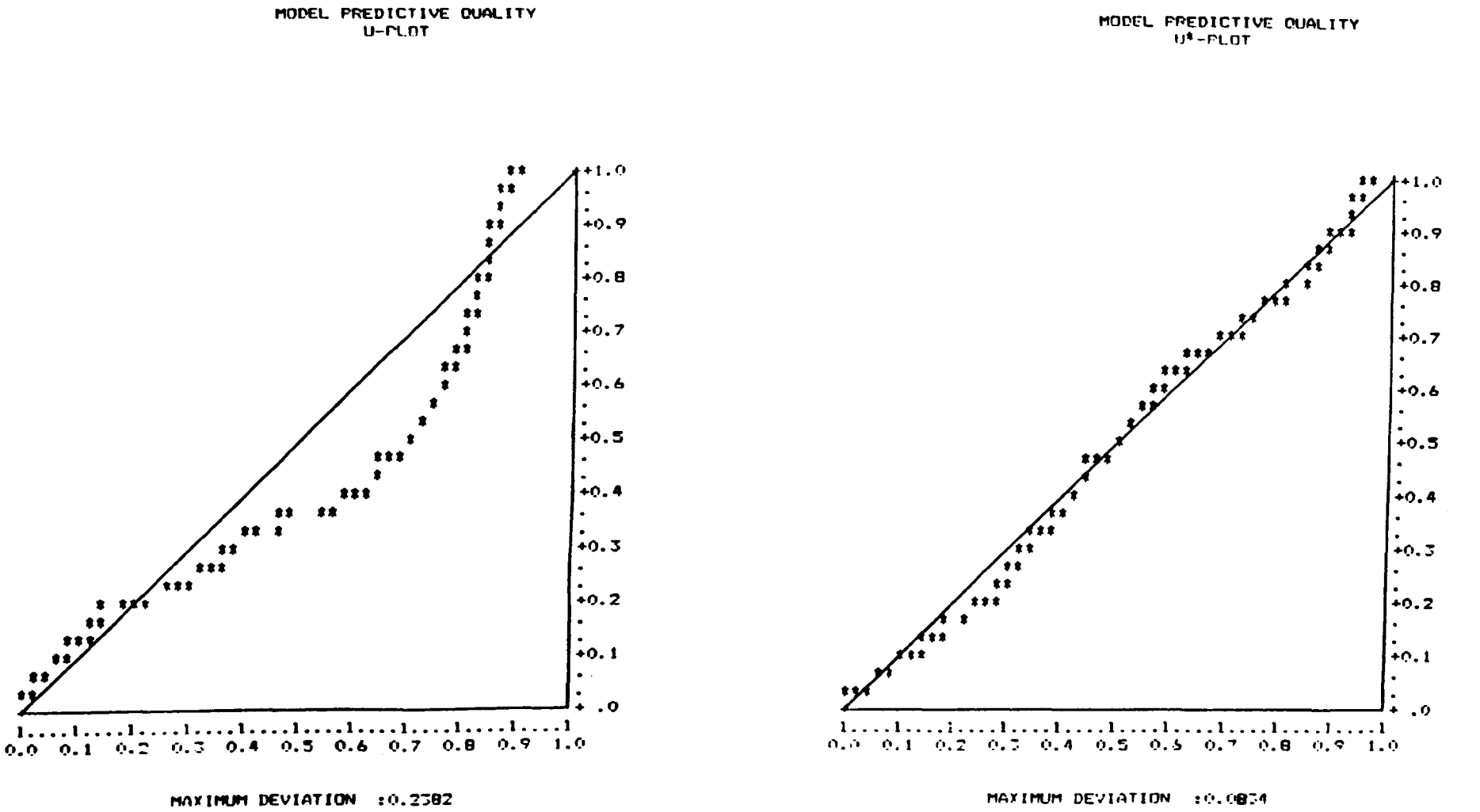


Fig. 5.7 The u-plot before and after * adapting LV predictions on System SS3 data using a parametric spline adaptor.

predictions. If we compare the predicted medians in Plot A4.7a and Plot A4.7b, we can see that the adapted medians, although in closer agreement now, are also noiser than before. The magnitudes of the medians have not been changed in any significant way, but those of DU^*, LV^* and KL^* have fluctuated above their raw predictions. Thus we conclude that since the bias in the raw predictions here is not serious, adapting has increased the noise and produced worse (more noisy) predictions. But in return we now have remarkable agreement amongst all the adapted predictions, as evident in Plot A4.7b.

CHAPTER 6

SOFTWARE RELIABILITY PREDICTION SYSTEMS WITH NON-PARAMETRIC RATE ESTIMATES AND THE ANALYSIS OF THEIR PERFORMANCE

6.1. INTRODUCTION

A major criticism of conventional software reliability models is that they are highly parameterized. The evolution of the failure rate of a program is highly structured through the modelling assumptions underlying the model being used.

Miller (1986) found that the rate function of most existing software reliability models has the complete montonicity property. Based on this observation, Miller and Sofer (1986a) formulated a non-parametric approach to estimate the failure rate of a program. Their basic assumption is that the failure rate should be completely monotone up to a specified order d. They found that the resulting problem of least-squares regression under order restrictions warrants a careful method of solution because the constraint matrix is very ill-conditioned (Miller and Sofer, 1986b).

They have investigated the performance of their method and a number of exponential models based on data simulated from the latter (Miller and Sofer, 1986a). Here we shall use the non-parametric rates estimated by their method to make predictions on the 7 real data sets. These predictions are then adapted with our parametric spline adaptor. The quality of these predictions are analysed and compared with those obtained in the previous Chapter. Another non-parametric approach to estimate

the failure rate is formulated in Appendix 2. Constraints based on empirical observation of the evolution of the failure rate of a program undergoing debugging can also be imposed.

6.2. ESTIMATION OF THE COMPLETELY MONOTONE RATES

Let $N(\tau)$ denote the number of failures observed in $[0,\tau]$ and $M(\tau) = E[N(\tau)]$ be the expected number. The rate function of the failure process is defined as:

$$r(\tau) = \frac{d}{d\tau} M(\tau) \qquad \text{for} \quad \tau \geq 0 \qquad (6.2.1)$$

Note that τ is the elapsed time.

The rate function is said to be completely monotone if and only if it possesses derivates of all order ($j \geq 0$) and

$$(-1)^j \frac{d^j}{d\tau^j} r(\tau) \geq 0 \qquad \text{for } \tau > 0 \qquad (6.2.2)$$

Miller (1986) has found that the rate functions of wide classes of exponential models (including all the exponential models we have used) possess this complete monotone property. A detailed exposition can be found in his original paper. On the basis of this observation, the complete monotone requirement is being imposed on the non-parametric estimate of the failure rate of a program.

Instead of using a completely monotone function, they have formulated the problem in terms of a completely monotone sequence. This means that the resulting rate function is piecewise constant. We shall briefly go through how this is done.

Consider a set of software failure data which consists of n failures in [0,T]. The elapsed time at the i^{th} failure is denoted by τ_i with $\tau_o = 0$. [0,T] is then separated into k equal intervals: let $\Delta s = T/k$ and $s_i = i\Delta s$ for $i = 0,T,\ldots,k$.

Now the function corresponding to a completely monotone sequence of rates $\{r_i\}$ defined over the partition of $[0, \infty]$ in steps of Δs will be of the form:

$$r(\tau) = r_i \qquad \text{if } s_{i-1} \leqslant \tau < s_i \qquad \text{for } i = 1,2, \ldots\ldots \tag{6.2.3}$$

with the set of r_i's satisfying:

$$(-1)^j \Delta^j r_i \geqslant 0 \qquad \text{for } i \geqslant j+1 \text{ and } j \geqslant 0 \tag{6.2.4}$$

where Δ^j is the j^{th} backward difference operator.

$$\left.\begin{aligned} &\Delta^0 r_i = r_i; \\ &\Delta^1 r_i = r_i - r_{i-1}; \\ &\text{and} \\ &\Delta^j r_i = \Delta^{j-1} r_i - \Delta^{j-1} r_{i-1} \qquad \text{for } j > 1 \end{aligned}\right\} \tag{6.2.5}$$

The estimation problem is to find a sequence of rate estimates $\{r_i\}$ which in some sense best fits the available data and also satisfies the constraints (6.2.4).

In practice, the number of future intervals is restricted to ℓ (rather than ∞) and the order of the difference constraints j is restricted to d (rather than ∞). Thus the constraints (6.2.4) become:

$$(-1)^j \Delta^j r_i \geq 0 \qquad \begin{array}{l} \text{for } k + \ell \geq i \geq j + 1 \\ \text{and} \quad d \geq j \geq 0 \end{array} \qquad (6.2.6)$$

The rates are estimated as follows:

1. The expected number of failures is constructed as a continuous function on the basis of the data:

$$\hat{M}(\tau) = \begin{cases} i + (\tau - \tau_i)/(\tau_{i+1} - \tau_i) & \text{if } \tau_i \leq \tau \leq \tau_{i+1} \\ & \text{for } i = 0,\ldots n-1 \\ n + 0.5(\tau - \tau_n)/(T - \tau_n) & \text{if } \tau_n \leq \tau \leq T. \end{cases}$$

Note that there is half a failure being accounted for should the period [0,T] not end with a failure.

2. For each of the k intervals, the raw data is defined as:

$$\hat{r}_i = (\hat{M}(s_i) - \hat{M}(s_{i-1}))/\Delta s \qquad i = 1,2,\ldots,k \qquad (6.2.7)$$

3. The fitted rates $\{r_i\}$ minimise the weighted sum of squared deviations which is:

$$D(r,\hat{r}) = \sum_{i=1}^{k} w_i(r_i - \hat{r}_i)^2$$

subject to the constraints in (6.2.6) with specified values of ℓ and d.

6.3. PREDICTION SYSTEMS WITH MONOTONE RATE ESTIMATES

Miller and Sofer (1986a) compare the estimated with the actual rates because the data are being simulated and the truth is known. In our case, such a direct comparison is not possible because the true state of nature underlying any of the real data sets is not known to us. Our approach is to define a prediction system which utilises the non-parametric rate estimates to predict the current reliability of the program, just as the prediction systems in the last Chapter. The quality of these predictions are then analysed using the techniques of Chapter 4. This provides us with some indication concerning the practical value of such non-parametric estimates.

The 3 components of such a prediction system are:

1. At stage i, after we have seen i failures, we assume the time to the next failure of the program to be exponentially distributed.

2. The data available up to and including the i^{th} failure will be used to estimate a completely monotone rate sequence with difference constraintsof order d (=1,2,3 and 4).

3. The k^{th} member of the sequence estimated in (2) is taken as the rate of the failure time distribution in (1) to make predictions.

Although the success of such a prediction system will be a joint effort of all components rather than just (2) alone, it is doubtful that the predictions emanating from such a system could be of good quality when the estimated rates are wrong. Thus the analysis of the quality of these predictions should be a good guide to the quality of the rate estimates for a given data set. The analysis could also be viewed as simply forming a basis of comparison between these new predictions and those in the previous Chapter.

We are grateful to Miller and Sofer for providing us with the software for estimating the rates in our examples. For each of the 7 data sets, a sequence of predictions is generated for each of 1 to 4 orders of difference constraints. We shall use dDIF to denote the prediction system with exponential failure times and completely monotone rate estimates subject to d difference constraints. At each stage the number of intervals k is 30 which means the dimension of the least-squares problem is always 30. The number of intervals into the future ℓ is always fixed at 5 and unit weights are used.

6.4. ANALYSIS OF THE QUALITY OF THE PREDICTIONS USING MONOTONE RATE ESTIMATES

The first set of results we shall look at is on System 1 data. From Table 6.1 the KS distances of the u-plot for 2DIF, 3DIF and 4DIF are all

Data (n)		1DIF	2DIF	3DIF	4DIF
Sys.1 (86)	u	.1689D	.0870A	.0784A	.0821A
	u*	.0715A	.0601A	.0764A	.0780A
	y	.0939A	.1144B	.1243B	.1251B
	y*	.0815A	.1165B	.1346C	.1360C
Sys.2 (31)	u	.1392A	.1220A	.1184A	.1215A
	u*	.1222A	.1446A	.1394A	.1392A
	y	.1417A	.1783A	.1956B	.1960B
	y*	.1652A	.2192C	.2382D	.2372C
Sys.3 (18)	u	.1542A	.1978A	.1904A	.1747A
	u*	.0901A	.0891A	.0919A	.0919A
	y	.1968A	.1848A	.1753A	.1818A
	y*	.1744A	.1650A	.1531A	.1552A
Sys.4 (30)	u	.1446A	.1245A	.1527A	.1521A
	u*	.1606A	.1694A	.1180A	.1068A
	y	.5028E	.4352E	.4291E	.4307E
	y*	.4674E	.4493E	.4402E	.4421E
Sys.6 (38)	u	.3308E	.2774E	.2769E	.2769E
	u*	.0993A	.0922A	.1061A	.1012A
	y	.3164E	.4007E	.4009E	.4028E
	y*	.2259D	.4311E	.4313E	.4311E
SS3 (173)	u	.2771E	.2612E	.2632E	.2632E
	u*	.0698A	.0899B	.0908B	.0882B
	y	.1223D	.1777E	.1796E	.1780E
	y*	.0657A	.0977C	.0988C	.0978C
BAe. (112)	u	.1614E	.0891A	.0670A	.0651A
	u*	.1060B	.0732A	.0685A	.0689A
	y	.0625A	.0831A	.0955A	.0951A
	y*	.0461A	.0772A	.0818A	.0818A

Table 6.1. Kolmogorov-Smirnov distance of the u-plot and y-plot of the exponential monotone rate prediction systems before and after * adapting (parametric spline adaptor). See (4.2.3.6) for the corresponding significance level.

very good and their y-plot distances are also good. But the u-plot distance for 1DIF is rather poor and the plot itself lies entirely above the 45° line, which indicates that these predictions are optimistic for this data.

Incidentally, this coincides with the result of the simulation study by Miller and Sofer (1986a) that the current failure rate estimate with 1 difference constraints has large negative bias.

The PL in Table 6.2 also suggests that the predictions from 1DIF are inferior to those from the other 3 prediction systems. Since the y-plot in the case of 1DIF is very good, we can use our parametric spline adapting procedure to improve these predictions. The y-plot distances show that 2DIF captures the trend better than the higher difference predictions, which explains its slightly better PL.

Indeed, the KS distances of the u^*-plot have improved in all 4 cases. The most significant improvement being 1DIF: from D to A. The PL in Table 6.3. agrees that the predictions of 1DIF* are better than those from 1DIF, but they are still not better than the predictions from 2DIF, 3DIF or 4DIF.

The reason for this will be apparent if we refer to the raw medians plotted in Plot A4.1c. From the median plot, we can see that the 1DIF predicted medians are extremely noisy. Each of the peaks corresponds to a small estimated rate in the predictive distribution. We have already established that the predictions are optimistic. But this optimism is also because of the current failure rate estimated with 1 difference constraints is highly affected by the last data point.

Data (n)	1DIF	2DIF	3DIF	4DIF
Sys.1 (86)	668.907	661.425	662.233	662.601
Sys.2 (31)	279.498	282.102	281.834	282.136
Sys.3 (18)	164.739	165.832	166.929	167.001
Sys.4 (30)	245.590	249.413	308.927	281.357
Sys.6 (38)	207.685	206.312	206.617	206.864
SS3 (173)	2298.23	2306.31	2306.97	2306.95
BAe (112)	642.590	638.912	639.115	639.018

Table 6.2. -log prequential likelihood of the prediction systems with monotone failure rate and exponential failure time distribution.

Data (n)	1DIF*	2DIF*	3DIF*	4DIF*
Sys.1 (86)	664.599	658.714	661.509	661.812
Sys.2 (31)	280.346	282.716	283.403	283.439
Sys.3 (18)	167.792	167.542	167.470	168.313
Sys.4 (30)	250.392	250.937	309.769	282.012
Sys.6 (38)	195.304	199.207	198.508	198.399
SS3 (173)	2211.27	2213.75	2214.19	2214.00
BAe (112)	645.417	643.987	645.009	644.969

Table 6.3. -log prequential likelihood of the adapted (parametric spline adaptor) exponential monotone rate prediction systems.

The least-squares monotone rate estimation subject to 1 difference constraints is identical to isotonic regression (Barlow et al, 1972), which has been applied to estimating system reliability by Campbell and Ott (1979). There is a simple algorithm for fitting these rates: they are the slopes of the smallest concave envelope (least concave majorant, Barlow et al, 1972) of the graph of the cumulative sum of $\hat{r}_i \Delta s$ against s_i, where $\hat{r}_i$, Δs and s_i are as defined in section 6.2. Because this cumulative sum is a step function, the envelope will be made up of a series of straight lines with positive but successively decreasing slopes. If the last data $\hat{r}_k$ happens to be relatively small, then the corresponding piece of straight line in the envlope is likely to be flat, which means the current failure rate will also be small. When $\hat{r}_k$ is not relatively small, the corresponding piece of straight line in the envelope might be joining up more than 1 interval, thus the current failure rate might be smoothed and no longer be too small. As a result, we see the peaks in the median plot corresponding to the first situation which we have described above.

This kind of bias is not consistent in the sense we have described in the last Chapter because between two peaks the estimated rates might not be as optimistic. Thus, even though the u-plot of 1DIF lies entirely above the 45° line and the medians of $1DIF^*$ are being adapted downwards, as evident in Plot A4.1d, there is still considerable noise in them. Hence the PL is worse than the other less noisy predictions.

Note that by imposing 1 more difference constraint the raw medians are much smoother than those of 1DIF. This is clearly visible in Plot A4.1c. Another feature which is also apparent in Plot A4.1c is that the predicted medians of 1DIF, 3DIF and 4DIF are very close, which implies that the estimated failure rate in each of these cases is also close.

This again coincides with the result of the simulation study of Miller and Sofer (1986a) that for data simulated from moderate growth models, the estimates are not improved as the order of the difference restrictions increases beyond 2.

The PL of 2DIF* has shown improvement over the raw 2DIF predictions. Indeed, the u*-plot distance is even smaller than before even though the u-plot is already very good. We can see from Plot A4.1d that the medians of 2DIF* are adapted slightly upwards.

For the remaining two sets of adapted predictions the improvement in PL is small, probably due to the trend in these adapted predictions being marginally worse than before as evident in their y*-plot distances.

If we compare these non-parametric predictions with those in the previous Chapters (Tables 4.3, 5.1 and 5.3) we will find that L* has the smallest u*- and y*-plot distances and the highest PL. MO* has the 2nd best distances and PL, which is followed very closely by LNHPP* in 3rd place with 2DIF* in 4th. The differences between their PL's are small. All of them have very good u-plot distances, but 2DIF* has a slightly worse y-plot distance with a significance level of between 10 and 20%. If we examine Plots A4.1b and A4.1d, the worse y*-plot in the case of 2DIF* is probably due to the predictions between stages 100 to around 127: the predicted medians from 2DIF* are noticeably smaller than those for the 3 better systems.

On comparing the median plots of the raw prediction systems, Plots A4.1a and A.4.1c, we observed the following features:

a) The predicted medians from DIF's with 2 or more difference constraints which are bigger than the medians of MO, behaved very similarly to the medians from L.

b) The predicted medians from DIF's with 2 or more difference constraints are nearly always bounded above by those from JM and below by those from DU.

From Plot A4.1a we can clearly see a space between the predicted medians of MO and DU. We postulate that if we would have used a prediction system with a Generalised Power Law NHPP (Miller, 1986) which has a rate function:

$$\lambda(\tau) = \lambda\delta\ (\beta + \tau)^{\delta-1} \qquad (6.4.1)$$

the predicted medians for this data from this prediction system (GP) will lie between those from MO and DU because:

$$\lim_{\beta\to 0} G(\lambda,\delta,\beta) \to DU(\lambda,\delta)$$

$$\lim_{\substack{\lambda\to\infty,\ \delta\to 0 \\ \lambda\delta\to\xi}} G(\lambda,\delta,\beta) \to MO(\xi,\beta) \qquad (6.4.2)$$

and

$$\lim_{\delta\to 1} G(\lambda,\delta,\beta) \to HPP(\lambda)$$

in an obvious notation.

We conjecture that the predicted medians from GP which are between those from MO and DU will be very close to those from 2DIF, 3DIF and 4DIF which lie below the medians from MO; specifically after stage 100 of this data set.

If this is indeed the case, it would provide more support for this non-parametric approach from the point of view of model flexibility because by using 1 such prediction system (with d $\geq$ 2) we are in effect using a meta-system consisting of 5 systems, namely JM(GO), L(LNHPP), MO, GP and DU.

However, from a prediction point of view, a prediction system with a more flexible model does not guarantee to produce better predictions. For this particular data set MO and LNHPP are better than 2DIF on the basis of PL; although 2DIF has a smaller u-plot distance, the y-plot distance is smaller in the case of MO and LNHPP.

Nonetheless, if we can achieve high flexibility with relatively relaxed assumptions, it is certainly an advantage. This type of non-parametric approach deserves further investigation.

The next set of results we shall look at is for System 2 data. On the basis of the u-plot distances in Table 6.1, the predictions from the non-parametric rate prediction systems are all very good. From Plot A4.2c we can see that the predicted medians from 2DIF, 3DIF and 4DIF are very close indeed. Those medians from 1DIF are slightly bigger than the others up to stage 28 and become much bigger and noisier after stage 41.

However, the PL in Table 6.2 while confirming that 2DIF, 3DIF and 4DIF are indeed similar, suggests that 1DIF is better than the others. If we refer to Appendix 3 for the listing of the data, we will find that the inter-failure times are occasionally very large, for example, t_{43} and those near the end of the data. This could be the main reason why even though the predictions from 1DIF are noisier than the others, its PL is still better because it captured the trend in the data better than the others. This is evident from the y-plot distances where 1DIF's is the smallest amongst all 4. However, it seems that 1DIF is performing better here by coincidence: its known optimistic bias is associated with an increase in the t's.

Because the u-plots are already good, it seems unlikely that these predictions can be improved to any appreciable extent, if at all, by adapting. In fact by adapting these predictions, the resulting u^*-plot distances have gone worse in all but one case. In the case of $1DIF^*$, the u^*-plot distance is smaller than that of 1DIF but the y^*-plot distance is worse, and as a result the PL of $1DIF^*$ is slightly worse than the PL of 1DIF. In all the remaining cases both the y^*-plot distance and the PL have worsened. A possible reason for no improvement here is that the raw predictions are noisy in the first place which cannot be improved simply by adapting.

If we compare these predictions with those in the previous Chapter we will find, on the basis of PL, that DU^* is the best, followed by MO^* with 1DIF in 3rd place. The differences between these PL's are very small indeed. However, on the basis of the u- and y- plot KS distances, 1DIF

is the best. The worst PL in the case of lDIF is probably caused by the noise in its predictions which we can clearly see by comparing the medians in Plots A4.2b and A.4.2c.

Finally, if we compare Plot A4.2a with Plot A4.2c, we will find that the medians from dDIF with d $\geq$ 2 are again bounded by those from JM and DU. This time the DIF medians are close to those from LNHPP which are greater than the MO medians. Their PL are also very close.

Plot A4.3c shows that the predicted medians for System 3 data are in close agreement amongst the 4 non-parametric rate prediction systems. The u- and y-plot distances are very good in all cases. According to the PL, lDIF has produced the best predictions for this data set.

Since the u-plots are already very good, we cannot expect to improve these predictions further by adapting. Although the u^*- and y^*-plot distances are even better after we have adapted these predictions, the PL disagrees that they are better than before. If we check the median plots we can see quite clearly that the adapted medians are more noisy than before: the peaks are more pronounced in all cases. Once again the non-parametric rate prediction systems are performing well on this data set and adapting does not improve the raw predictions.

We have seen in the previous Chapter that some prediction systems performed rather poorly on this data set (JM, L). We can see from Plot A4.3a that the predicted medians from a number of prediction systems are extremely noisy. Surprisingly, however, the predicted medians from the

non-parametric rate prediction systems are not as noisy, they are in fact behaving similar to those from LNHPP. Overall, the predictions with the best PL for this data set are from MO, closely followed by 1DIF and LV*. The noise in the 1DIF predictions is probably the reason why its PL is not better than MO's even though it has better u- and y-plot KS distances.

In the previous Chapter, when we analyse the predictions on System 4 data, we have found trend to be present in the y-plots of the parametric prediction systems with exponential failure time distribution. This situation is similar with the predictions from the non-parametric rate prediction systems here. We can see from Table 6.1 that the u-plot distances are all very good, but the y-plot distances are all extremely poor.

The PL for 3DIF is particularly poor because its estimated current failure rate at stage 51 is extremely small after seeing the exceptionally large inter-failure time t_{51}. 4DIF behaved similarly at stage 51 but not as extreme as 3DIF, hence the PL is better than 3DIF yet noticeably worse than 1DIF and 2DIF.

Under these circumstances, we cannot expect to be able to improve these predictions by adapting. Indeed, the PL in Table 6.3 confirms our belief: the PL of 1DIF* and 2DIF* are worse than before. Overall, the best predictions for this data set come from LV and KL with the Pareto failure time distribution.

From Table 6.1 we can see that the u- and y-plot distances for the predictions of System 6 data are all very poor. We have already commented on the cause of these poor y-plots in the previous Chapter: this is due to an exceptionally large inter-failure time t_{69}. The u-plot in all cases lies entirely above the 45° line. The level of significance in all the u-plot distances indicates that the optimistic bias is very severe.

After these predictions have been adapted, we find the u^*-plots to have improved considerably in all cases. The y^*-plots are still poor because of the large inter-failure time t_{69}.

The PL's in Table 6.2 and 6.3 suggest that all the adapted predictions have improved over their raw counterparts, of which lDIF* has improved most significantly and has the highest PL amongst all 4 sets of adapted predictions.

Comparing Plot A4.5c with Plot A4.5d, we can see that all the adapted medians are being adapted downwards. The medians from lDIF* are still the noisiest amongst the adapted medians, although the range of values is smaller than those from lDIF.

If we compare the raw medians here with those in the previous Chapter, we will find that the predictions from dDIF with d $\geq$ 2 are again behaving very similarly to those from LNHPP. After they have been adapted, we can see from Plots A4.5b and A4.5d that the medians from dDIF* with d $\geq$ 2 are incredibly close to the medians from those adapted parametric prediction systems with exponential failure time distribution. This agreement is also observed in their PL's.

Of all the predictions being made on this data set, those from LV and KL are still the best in PL terms. 1DIF* has the 3rd highest PL, the noise in its predictions has reflected the noise in this data. Again the Pareto distribution in LV and KL seems to be most suitable for data with exceptionally large or small values.

For System SS3 data, the u-plot distances in Table 6.1 are poor for all the predictions. The PL's in Table 6.2 prefer the 1DIF predictions for this data set, probably because of its better trend capturing as evident in the y-plot distances. The good performance of 1DIF here seems to be again coincidental and similar to the situation of System 2 data: its optimistic bias is matched by an increase in the t's. We can see from Plots A4.6a and A4.6c that after stage 220 there is a rapid increase in all the predicted medians.

The u-plot for 1DIF lies entirely above the line of unit slope and is extremely non-uniform. The u-plots in the other cases also lie predominantly above the 45° line and are very non-uniform. Thus it seems adapting can improve these predictions.

Indeed we see quite dramatic improvement in all the adapted predictions. The u^*- and y^*-plot distances are improved in all cases, the best being 1DIF* which are now insignificant even at 20%.

Plot A4.6c reveals that there is remarkably little difference amongst the raw medians before stage 220. Those from 1DIF are much bigger and noisier after this stage. After they have been adapted, we can see from

Plot A4.6d that all the medians are now smaller than before. Those from lDIF* are still noisier and bigger than the other adapted medians after stage 220, but seems to be less seriously so than before.

According to the PL in Table 6.3, lDIF* is the best with the other adapted predictions trailing closely behind. The improvement in PL through adapting is very substantial in all cases. The PLR of lDIF* against lDIF in nearly e^{87}, which clearly rejects lDIF for this data set.

In the previous Chapter, we have achieved very similar improvement by adapting the parametric predictions on this data set. Pooling all the results together, we find that the PL of lDIF* is very close to those better predictions of JM*, GO*, MO*, L*, and LNHPP*, while the PL of dDIF* with $d \geq 2$ is close to those of DU* and LV*.

If we compare the raw medians in Plot A4.6a and Plot A4.6c, they are not as closely behaving as in the previously analysed data sets. We can see from the raw medians that the trend is decreasing before stage 220. In the case of lDIF, we can see its predicted medians are quite different from the other raw medians depicted in Plot A4.6a. But after they have been adapted, the lDIF* medians are incredibly close to the medians from those adapted parametric systems with very similar PL.

Finally, we shall investigate the performance of the non-parametric rate prediction systems on BAe data. The u- and y-plot KS distances in Table 6.1 are very good for dDIF with $d \geq 2$. The u-plot for lDIF is very poor and lies entirely above the 45° line, which seems to be a good candidate for adapting.

Indeed the KS distance of the u^*-plot for lDIF* has improved. However, the PL's in Table 6.2 and 6.3 dismiss any improvement through adapting in all 4 cases. It is not surprising that the PL should register no gain if we adapt predictions which are already good. But in the case of lDIF, we would expect the adapted predictions to be better.

If we refer to Plot A4.7c for the lDIF medians, we will find that they are extremely noisy and 'spiky' like those in System 1 data. Although the location of the lDIF* medians is in close agreement with that of the better predictions, they are even more noisy than the raw medians, as we can see from Plot A4.7d. Thus the PL becomes smaller because the predictions from lDIF* are very noisy.

In the previous Chapter, we also find that the raw parametric predictions are generally good and cannot be improved further by adapting. When we compare the raw medians in Plot A4.7a and Plot A4.7c, we find that the medians from dDIF with $d \geq 2$ are not behaving closely to those from L or LNHPP, even though they seem to be once again bounded by the JM and DU medians. The DIF medians are visibly more noisy and this extra noise is likely to be the reason why the PL for dDIF with $d \geq 2$ are worse than the PL of the raw parametric prediction systems with the only exception of DU.

As for the adapted predictions, we can see from Plots A4.7b and A.47d that the medians from dDIF* ($d \geq 2$) are now very close to those from the adapted parametric prediction systems. Furthermore, their PL's also show remarkable agreement.

CHAPTER 7

SUMMARY AND CONCLUSIONS AND FUTURE RESEARCH POSSIBILITIES

7.1. SUMMARY AND CONCLUSIONS

We have presented two numerical methods for the minimisation of univariate and bivariate functions with bound restrictions on the variables. According to the numerical results we have obtained, these methods have proved to be both efficient and reliable for determining the maximum likelihood estimate of the unknown parameters in the 7 software reliability models in this study.

The success of our methods is also due to the two preliminary steps which we have taken:

1. Reduce the number of variables in the minimisation problem and optimise over a lower dimensional space.

2. Transform the variables in the minimisation problem so as to remove their constraints. We believe that the square transformation we have used in our examples might also have contributed to the efficiency of our methods because the shape of the objective function after such a transformation will be more curved and hence more suited to the minimisation methods.

These parameters are then used in the respective predictive distributions to predict the current reliability of a program. We argued that since our aim is to predict, the only sensible way of assessing our success is by a direct analysis of the quality of these predictions.

This is done by using:

1. The prequential likelihood as a global measure of the goodness of the predictions.

2. The u-plot procedure to check the biasedness in the predictions.

3. The y-plot procedure to check whether the predictions have captured the trend in the data adequately.

4. The predicted median plot to indicate the level of noise in the predictions.

Our ability to measure the quality of the predictions also enables us to make use of the information concerning past prediction error to correct the future predictions on the same data. A naive approach based on a joined-up adaptor can produce good probability predictions but at the same time introduces a lot of internal noise into the adapted predictors. As a result, we are unable to use PL to analyse the success of the adaptive procedure.

By using a smooth adaptor, based on a parametric spline, we have avoided the introduction of internal noise into the adapted predictors and are able to assess the effect of adapting. This is done by analysing the quality of the predictions before and after adapting, and comparing the results of the analyses.

On the basis of our 7 data sets and 9 different predictions systems we conclude that:

1. Adapting is most effective if the bias is consistent.

2. In order to have improved adapted predictions, it is not necessary to have a good y-plot to ensure that trend in the data is well captured. We have encountered situations where the y-plot is poor, maybe due to one or two exceptionally large observations, but their adapted predictions are improved. However, we must make sure that the bias is fairly stationary and there is no systematic trend in the y-plot which could be due to a reversal in the nature of the bias.

3. The y-plot might also be improved as a result of the improvement in the adapted predictions.

4. The adapted predictions are invariably more noisy than the raw predictions - the price we have to pay for having to estimate the adaptor from the data.

5. Predictions which are already very good would usually not gain by adapting mainly because of the induced noise. But there are a few cases in which we have observed a slight improvement in the prequential likelihood even when the raw predictions are already very good.

We then use the completely monotone rate estimates with 1 to 4 difference constraints and assume the failure times to be exponentially distributed to predict. We applied these 4 prediction systems to our 7 data sets and adapted all these predictions. We observed that:

1. The predictions from 1DIF are usually noisier than the others and they are usually optimistic.

2. The predictions from 3DIF and 4DIF are usually very similar.

3. On data sets with occasional exceptionally large data points, 1DIF or its adapted version, 1DIF*, is usually best in PL terms.

4. When the raw predictions are very noisy as well as being biased, adapting might not be able to improve them.

When we compare these predictions with those in which a parametric model is used, we find that:

1. The predictions from DIF with 2 or more difference constraints can be very similar to those based on parametric models with exponential failure times.

2. In none of the cases we have looked at are the predictions from DIF (raw or adapted) best, although the best set of DIF predictions (raw or adapted) is usually close to the overall best (raw or adapted).

In practice, a user can choose the prediction with the best past predictive analysis results for his/her use.

7.2. FUTURE RESEARCH POSSIBILITIES

This study is based on 7 sets of real data. The problem with using real data is that the underlying true state of nature is not known, thus it is very difficult to fully understand and interpret the results. Our predictive quality measurement tools can help us up to a point but are far from perfect. Therefore, we should organise a large scale simulation study in order to fully appreciate the performance of the different parametric and non-parametric prediction systems; the capability of our adaptive procedure in correcting biased predictions; the situations and conditions in which adapting will succeed or fail; the effect of the introduction of noise in the use of adapted predictions.

So far we have been analysing continuous data. The other type of data which is usually easier to collect consists of non-overlapping intervals of execution times and count of failures observed within each interval: the discrete data.

Abdel-Ghaly (1986) has implemented a number of prediction systems for this type of data. The unknown parameters in the respective models are estimated using maximum likelihood. The optimisation involved is done by

the Nelder-Mead simplex search which is very inefficient. We can most certainly improve the efficiency by using the same methods we have used for the continuous cases. It may also be possible to extend the adaptive idea to this type of data.

As for the predictive quality measurement tools, the theoretical justification for the u- and y-plot procedures to be used for discrete predictions is lacking. Their use in the discrete case is empirically motivated: they do provide the correct information. Therefore, more effort is needed to find suitable and effective measurement tools for discrete data. The search for predictive quality measurements is by no means restricted only to discrete data, more measurement procedures are also needed for continuous data. After all, it is very important to be able to measure what we have or have not achieved in our predictions, otherwise, we cannot justify any alternative or modification method like adapting.

The approach using non-parametric rate estimates seems to be generating predictions which are noisier than the better parametric predictions, but usually manage well in capturing trend. Based on the results of our study, it does seem that this approach can generate plausible predictions. It has the further advantage that the assumptions are more relaxed. It might be possible to reduce the noise in these predictions, perhaps at the cost of more bias or worse trend capturing. Maybe the formulation in Appendix 2 can generate less noisy rates because all the inter-failure times are used. Or maybe too much smoothing would be involved. The performance of such a non-parametric formulation should repay investigation.

Amongst all the data we have analysed there are siutations where the PL for two or more sets of predictions are very similar and yet the predictions are still dissimilar. It might be possible to combine these predictions to form a meta-prediction in some optimal way. It is conceivable that in doing so we can improve further the quality of the predictions. We have previously attempted to combine predictions using the past PL, but have not obtained consistent results (Abdel-Ghaly et al, 1985). This is also a topic which deserves more investigation.

APPENDIX 1

UNIQUENESS AND CONDITION FOR FINITE MAXIMUM LIKELIHOOD PARAMETER ESTIMATE IN THE GOEL AND OKUMOTO MODEL

APPENDIX 1

UNIQUENESS AND CONDITION FOR FINITE MAXIMUM LIKELIHOOD PARAMETER ESTIMATE IN THE GOEL AND OKUMOTO MODEL

We shall prove that the likelihood function of the Goel and Okumoto model has one unique maximum and this maximum is at finite μ and α provided that:

$$\frac{\tau_i}{2} > \frac{1}{i} \sum_{j=1}^{i} \tau_j \qquad \text{(A1.1)}$$

where τ_j is the total elapsed time as at the j^{th} failure and i is the total number of failures observed. If (A1.1) is not satisfied, then the maximum of the likelihood is at finite $\lambda = \mu\Phi$ and infinite μ.

Proof

In section 3.4. we have established that the MLE of Φ can be obtained by maximising:

$$\hat{\ell}(t_1,\ldots t_i/\mu,\Phi) = i\log i - i - i\log\left[1-e^{-\Phi\tau_i}\right]+i\log\Phi - \Phi \sum_{j=1}^{i} \tau_j \qquad \text{(A1.2)}$$

over $\Phi > 0$ and $\hat{\mu}$ is given by:

$$\hat{\mu} = \frac{i}{1-e^{-\Phi\tau_i}} \qquad \text{(A1.3)}$$

We will show that $\hat{\ell}$ is strictly concave, which is a necessary and sufficient condition for its maximum to be unique. Note that the reverse is not true - a function with a unique maximum is not necessarily strictly concave.

A function is strictly concave if its second derivative is always negative. The second derivative of (A1.2) is:

$$\frac{\partial^2 \hat{\varrho}}{\partial \phi^2} = -\frac{i}{\phi^2} + \frac{i\tau_i^2 e^{-\phi\tau_i}}{\left[1-e^{-\phi\tau_i}\right]^2} \qquad \text{(A1.4)}$$

which can be factorized into:

$$\frac{i}{\phi^2}\left\{\frac{\phi\tau_i e^{-\phi\frac{\tau_i}{2}}}{1-e^{-\phi\tau_i}} + 1\right\}\left\{\frac{\phi\tau_i e^{-\phi\frac{\tau_i}{2}}}{1-e^{-\phi\tau_i}} - 1\right\} \qquad \text{(A1.5)}$$

The sign of $\partial^2\hat{\varrho}/\partial\phi^2$ is the same as the sign of the last term in (A1.5) because the remaining terms are both positive for $\phi > 0$.

Now

$$\text{sign}\left\{\frac{\phi\tau_i e^{-\frac{\phi\tau_i}{2}}}{1-e^{-\phi\tau_i}} - 1\right\} = \text{sign}\left\{\frac{\phi\tau_i}{2}\left[\frac{2}{e^{\frac{\phi\tau_i}{2}} - e^{-\frac{\phi\tau_i}{2}}}\right] - 1\right\}$$

$$= \text{sign}\left\{\frac{\phi\tau_i}{2} - \sinh\left(\frac{\phi\tau_i}{2}\right)\right\} \qquad \text{(A1.6)}$$

By the MacLaurin's series:

$$\sinh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots\ldots$$

$$\therefore \text{sign}\left\{\frac{\partial^2\hat{\varrho}}{\partial\phi^2}\right\} = -\text{sign}\left\{\frac{(\phi\tau_i)^3}{3!} + \frac{(\phi\tau_i)^5}{5!} + \ldots\ldots\right\} \qquad \text{(A1.7)}$$

which is clearly -ve $\forall\ \phi > 0$. This means (A1.2) is strictly concave for $\phi > 0$, hence the likelihood function can have only one maximum.

Since the MLE of μ is given by:

$$\hat{\mu} = \frac{i}{1-e^{-\phi\tau_i}} \qquad \text{(A1.8)}$$

it will be infinite if $\hat{\phi} = 0$. To investigate the behaviour of the likelihood function at $\phi = 0$, we use the reparameterization of $\lambda = \mu\phi$. The log likelihood in terms of (λ,ϕ) is:

$$\hat{\ell}(t_1,\ldots,t_i/\lambda,\phi) = n\log\lambda - \phi\sum_{j=1}^{i}\tau_j - \frac{\lambda}{\phi}\left[1 - e^{-\phi\tau_i}\right] \qquad \text{(A1.9)}$$

Expanding $e^{-\phi\tau_i}$ by the use of MacLaurin's series, (A1.9) becomes:

$$\hat{\ell}(t_1,\ldots,t_i/\lambda,\phi) = n\log\lambda - \phi\sum_{j=1}^{i}\tau_j - \frac{\lambda}{\phi}\left[\phi\tau_i - \frac{(\phi\tau_i)^2}{2!} + \frac{(\phi\tau_i)^2}{3!}\right] + O(\phi^3)$$

$$= n\log\lambda - \lambda\tau_i + \phi\left[\frac{\lambda\tau_i^2}{2} - \sum_{j=1}^{i}\tau_j\right] - \frac{\lambda\phi^2}{3!}\tau_i^3 + O(\phi^3) \qquad \text{(A1.10)}$$

where $O(\phi^3)$ denotes the terms of order ϕ^3. Obviously:

$$\lim_{\phi\to 0}\ \ell(t_1,\ldots,t_i/\lambda,\phi) \to n\log\lambda - \lambda\tau_i \qquad \text{(A1.11)}$$

which is the log likelihood of the homogeneous Poisson process.

It is clear that if a small <u>feasible</u> step is taken from $(\lambda,0)$, for $\forall\lambda > 0$, in the direction of ϕ, the effect on ℓ will be determined by the first order term in (A1.10):

$$\frac{\lambda\tau_i^2}{2} - \sum_{j=1}^{i}\tau_j \qquad \text{(A.1.12)}$$

If (A1.12) is positive at the current value of λ, then a small feasible (positive) step in Φ will increase the value of ℓ, otherwise, such a step will lead to a decrease. When (A1.12) is equal to zero, the second order term is:

$$-\frac{\lambda\Phi^2}{3!}\tau_i^3 \qquad \text{(A1.13)}$$

which is always negative. If the latter situation is true, the maximum of ℓ will be equal to:

$$\max_{\lambda} [n\log\lambda - \lambda\tau_i]$$

which yields:

$$\hat{\lambda} = \frac{i}{\tau_i}$$

Because of the uniqueness of the maximum, it suffices to check the sign of (A1.12) only at $(\frac{i}{\tau_i},0)$ in order to decide whether $\hat{\mu}$ is finite or not.

Hence if,

$$\frac{\tau_i}{2} < \frac{1}{i}\sum_{j=1}^{i}\tau_j \qquad \text{(A1.14)}$$

then $\hat{\mu} = \infty$, $\hat{\Phi} = 0$, and $\hat{\lambda} = \frac{i}{\tau_i}$, otherwise $\hat{\mu}$ and $\hat{\Phi}$ will both be finite.

APPENDIX 2

NON-PARAMETRIC APPROACH TO ESTIMATE FAILURE RATES OF A PROGRAM UNDERGOING DEBUGGING

APPENDIX 2

A NON-PARAMETRIC APPROACH TO ESTIMATE THE FAILURE RATES OF A PROGRAM UNDERGOING DEBUGGING

We shall outline a non-parametric method for determining the failure rates of a program on the basis of past inter-failure time data $t_1,\ldots,t_n$.

Our aim is to estimate these rates such that the trend in the data is adequately captured. We shall do this by minimising the y-plot Kolmogorov-Smirnov Distance of $\{r_j t_j\}$ subject to some suitable constraints being imposed on the r_j's.

Since $\{r_j t_j\}$ is a sequence, the corresponding y-plot will be a step function which means there are two distances to consider:

$$D^+(r) = \max_{1 \leq i \leq n} \left| \frac{\sum_{j=1}^{i} r_j t_j}{\sum_{j=1}^{n} r_j t_j} - \frac{i}{n} \right| \qquad \text{(A2.1)}$$

and

$$D^-(r) = \max_{1 \leq i \leq n} \left| \frac{\sum_{j=1}^{i} r_j t_j}{\sum_{j=1}^{n} r_j t_j} - \frac{(i-1)}{n} \right| \qquad \text{(A2.2)}$$

Here we shall only deal with $D^+(r)$, the case of $D^-(r)$ can be dealt with in a similar way.

In order to transform (A2.1) and (A2.2) to be linear in the variables, r_j, we impose the following equality constraint:

$$\sum_{j=1}^{n} r_j t_j = n \qquad \text{(A2.3)}$$

Thus our estimation problem can be written as:

$$\left.\begin{aligned} &\min_{r} \left[\max_{1 \leqslant i \leqslant n} \left| \sum_{j=1}^{i} r_j t_j \right| \right] \\ &\text{subject to:} \\ &\sum_{j=1}^{n} r_j t_j = n \\ &\text{and} \\ &r_j \geqslant 0 \quad 1 \leqslant j \leqslant n \end{aligned}\right\} \qquad \text{(A2.4)}$$

which is a classical L_∞ fit problem in the n variables (Barrodale and Young, 1966) and can be transformed into a linear programming problem in the following way.

Let $\{r_j^*\}$ denote the solution to the above problem, and d^* denote the minimum positive distance, i.e.

$$d^* = \max_{1 \leqslant i \leqslant n} \left| \sum_{j=1}^{i} r_j^* t^j - i \right| \leqslant \max_{1 \leqslant i \leqslant n} \left| \sum_{j=1}^{i} r_j t_j - i \right| \qquad \text{(A2.5a)}$$

with

$$\sum_{j=1}^{n} r_j^* t_j = n \qquad \text{(A2.5b)}$$

Note that (A2.5a) is true for all non-negative $\{r_j\}$ which satisfies $\sum_{j=1}^{n} r_j t_j = n$.

If we let d be the positive distance corresponding to a set of r_j's then clearly:

$$\left.\begin{array}{l} \sum_{j=1}^{i} r_j t_j - i + d \geqslant 0 \\ \text{and} \\ \sum_{j=1}^{i} r_j t_j - i - d \leqslant 0 \end{array}\right\} \quad 1 \leqslant i \leqslant n \qquad \text{(A2.6)}$$

because d must be greater than or equal to all the positive deviations and -d must be less than or equal to all the negative deviations. Therefore, problem (A2.4) can now be solved by minimising d subject to the constraints in (A2.4) and (A2.6) with $d \geqslant 0$, which is a linear programming problem in (n+1) positive variables, d, $r_1,\ldots,r_n$.

In the context of reliability growth, the failure rate of a program should be decreasing with the number of bugs found. Thus we might want to impose the following constraints on the r_j's:

$$\Delta r_j = r_j - r_{j-1} \leqslant 0 \qquad 2 \leqslant j \leqslant n \qquad \text{(A2.7)}$$

Furthermore, we can reasonably expect early fixes would contribute more towards improving the reliability of the program, thus we can impose the following constraints to reflect this:

$$\left.\begin{array}{ll} \Delta r_j = r_j - r_{j-1} \leqslant 0 & 2 \leqslant j \leqslant n \\ \Delta^2 r_j = \Delta r_j - \Delta r_{j-1} \leqslant 0 & 3 \leqslant j \leqslant n \end{array}\right\} \qquad \text{(A2.8)}$$

These constraints, in the same spirit as those of Miller and Sofer (1986a,b), specify that the failure rates are decreasing (or non-increasing) with j and the amount of decrease is progressively smaller.

Therefore, we can impose further linear constraints onto the basic problem and solve it using a linear programming package.

APPENDIX 3

LISTING OF THE SOFTWARE RELIABILITY DATA

Musa's System 1 Data
(read left to right)

3.	30.	113.	81.	115.
9.	2.	91.	112.	15.
138.	50.	77.	24.	108.
88.	670.	120.	26.	114.
325.	55.	242.	68.	422.
180.	10.	1146.	600.	15.
36.	4.	0.	8.	227.
65.	176.	58.	457.	300.
97.	263.	452.	255.	197.
193.	6.	79.	816.	1351.
148.	21.	233.	134.	357.
193.	236.	31.	369.	748.
0.	232.	330.	365.	1222.
543.	10.	16.	529.	379.
44.	129.	810.	290.	300.
529.	281.	160.	828.	1011.
445.	296.	1755.	1064.	1783.
860.	983.	707.	33.	868.
724.	2323.	2930.	1461.	843.
12.	261.	1800.	865.	1435.
30.	143.	108.	0.	3110.
1247.	943.	700.	875.	245.
729.	1897.	447.	386.	446.
122.	990.	948.	1082.	22.
75.	482.	5509.	100.	10.
1071.	371.	790.	6150.	3321.
1045.	648.	5485.	1160.	1864.
4116.				

Musa's System 2 Data
(read left to right)

191.	222.	280.	290.	290.
385.	570.	610.	365.	390.
275.	360.	800.	1210.	407.
50.	660.	1507.	625.	912.
638.	293.	1212.	612.	675.
1215.	2715.	3551.	800.	3910.
6900.	3300.	1510.	195.	1956.
135.	661.	50.	729.	900.
180.	4225.	15600.	0.	0.
300.	9021.	2519.	6890.	3348.
2750.	6675.	6945.	7899.	

Musa's System 3 Data
(read left to right)

115.	0.	83.	178.	194.
136.	1077.	15.	15.	92.
50.	71.	606.	1189.	40.
788.	222.	72.	615.	589.
15.	390.	1863.	1337.	4508.
834.	3400.	6.	4561.	3186.
10571.	563.	2770.	652.	5593.
11696.	6724.	2546.		

Musa's System 4 Data
(read left to right)

5.	73.	141.	491.	5.
5.	28.	138.	478.	325.
147.	198.	22.	56.	424.
92.	520.	1424.	0.	92.
183.	10.	115.	17.	284.
296.	215.	116.	283.	50.
308.	279.	140.	678.	183.
2462.	104.	2178.	285.	171.
0.	643.	887.	149.	469.
716.	604.	0.	774.	256.
14637.	18740.	1526.		

Musa's System 6 Data
(read left to right)

3.	14.	59.	32.	8.
52.	2.	25.	2.	3.
4.	1.	30.	21.	196.
265.	6.	3.	8.	1.
12.	36.	38.	1.	74.
43.	236.	121.	18.	9.
23.	1.	672.	189.	83.
52.	8.	1.	41.	7.
43.	1.	4.	5.	1.
16.	70.	60.	2.	2.
3.	169.	29.	88.	55.
27.	24.	27.	140.	33.
5.	36.	74.	40.	2.
86.	221.	6.	891.	23.
4.	437.	66.		

Musa's System SS3 Data
(read left to right)

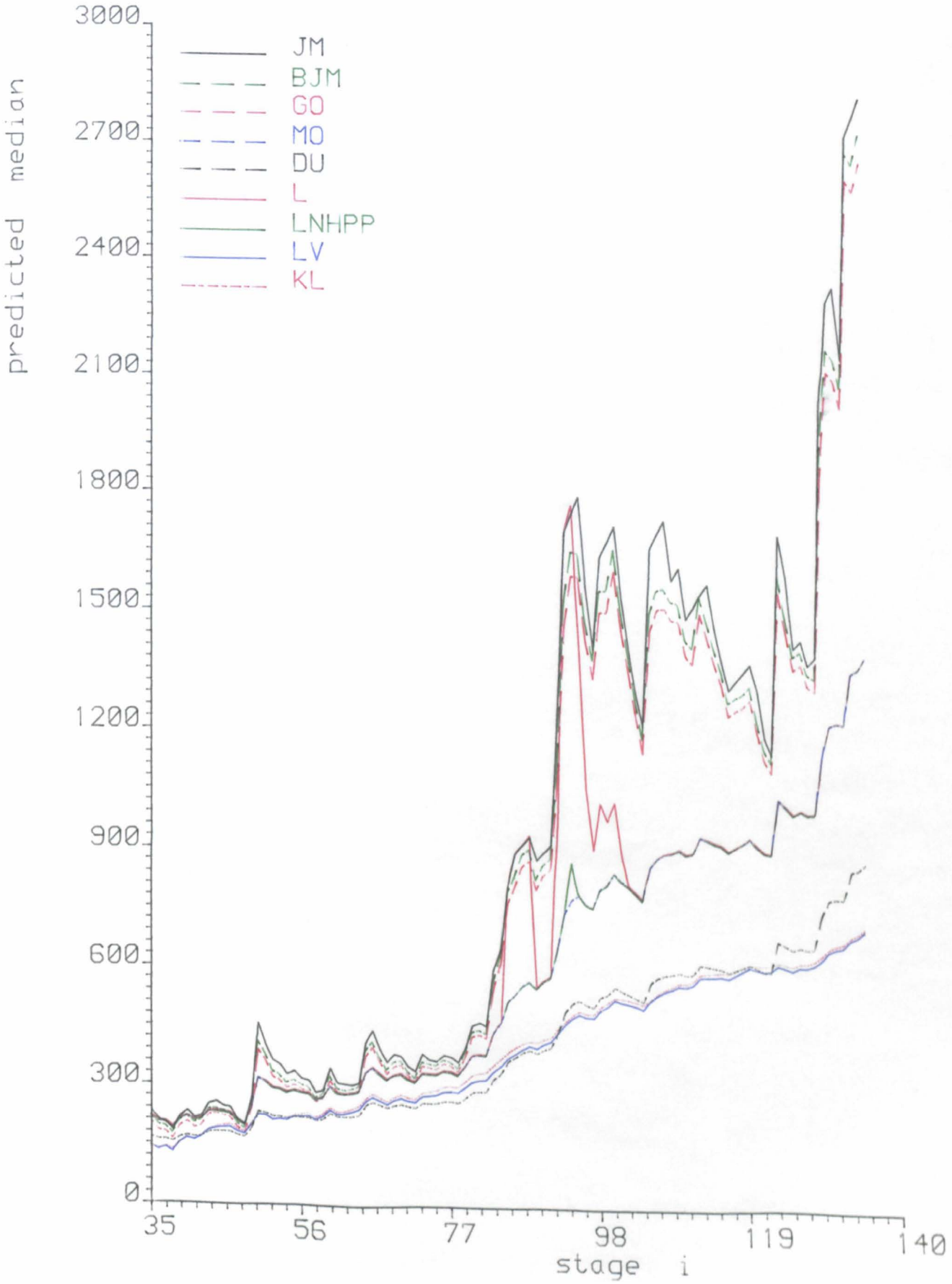
107400.	17220.	180.	32880.	960.
26100.	44160.	333720.	17820.	40860.
18780.	960.	960.	79860.	240.
120.	1800.	480.	780.	37260.
2100.	72060.	258704.	480.	21900.
478620.	80760.	1200.	80700.	688860.
2220.	758880.	166620.	8280.	951354.
1320.	14700.	3420.	2520.	162480.
520320.	96720.	418200.	434760.	543780.
8820.	488280.	480.	540.	2220.
1080.	137340.	91860.	22800.	22920.
473340.	354901.	369480.	380220.	848640.
120.	3416.	74160.	262500.	879300.
360.	8160.	180.	237920.	120.
70800.	12960.	300.	120.	558540.
188040.	56280.	420.	414464.	240780.
206640.	4740.	10140.	300.	4140.
472080.	300.	87600.	48240.	41940.
576612.	71820.	83100.	900.	240300.
73740.	169800.	1.	302280.	3360.
2340.	82260.	559920.	780.	10740.
180.	430860.	166740.	600.	376140.
5100.	549540.	540.	900.	521252.
420.	518640.	1020.	4140.	480.
180.	600.	53760.	82440.	180.
273000.	59880.	840.	7140.	76320.
148680.	237840.	4560.	1920.	16860.
77040.	74760.	738180.	147000.	76680.
70800.	66180.	27540.	55020.	120.
296796.	90180.	724560.	167100.	106200.
480.	117360.	6480.	60.	97860.
398580.	391380.	180.	180.	240.
540.	336900.	264480.	847080.	26460.
349320.	4080.	64680.	840.	540.
589980.	332280.	94140.	240060.	2700.
900.	1080.	11580.	2160.	192720.
87840.	84360.	378120.	58500.	83880.
158640.	660.	3180.	1560.	3180.
5700.	226560.	9840.	69060.	68880.
65460.	402900.	75480.	380220.	704968.
505680.	54420.	319020.	95220.	5100.
6240.	49440.	420.	667320.	120.
7200.	68940.	26820.	448620.	339420.
480.	1042680.	779580.	8040.	1158240.
907140.	58500.	383940.	2039460.	522240.
66000.	43500.	2040.	600.	226320.
327600.	201300.	226980.	553440.	1020.
960.	512760.	819240.	801660.	160380.
71640.	363990.	9090.	227970.	17190.
597900.	689400.	11520.	23850.	75870.
123030.	26010.	75240.	68130.	811050.
498360.	623280.	3330.	7290.	47160.
1328400.	109800.	343890.	1615860.	14940.
680760.	26220.	376110.	181890.	64320.
468180.	1568580.	333720.	180.	810.
322110.	21960.	363600.		

British Aerospace Data
(read left to right)

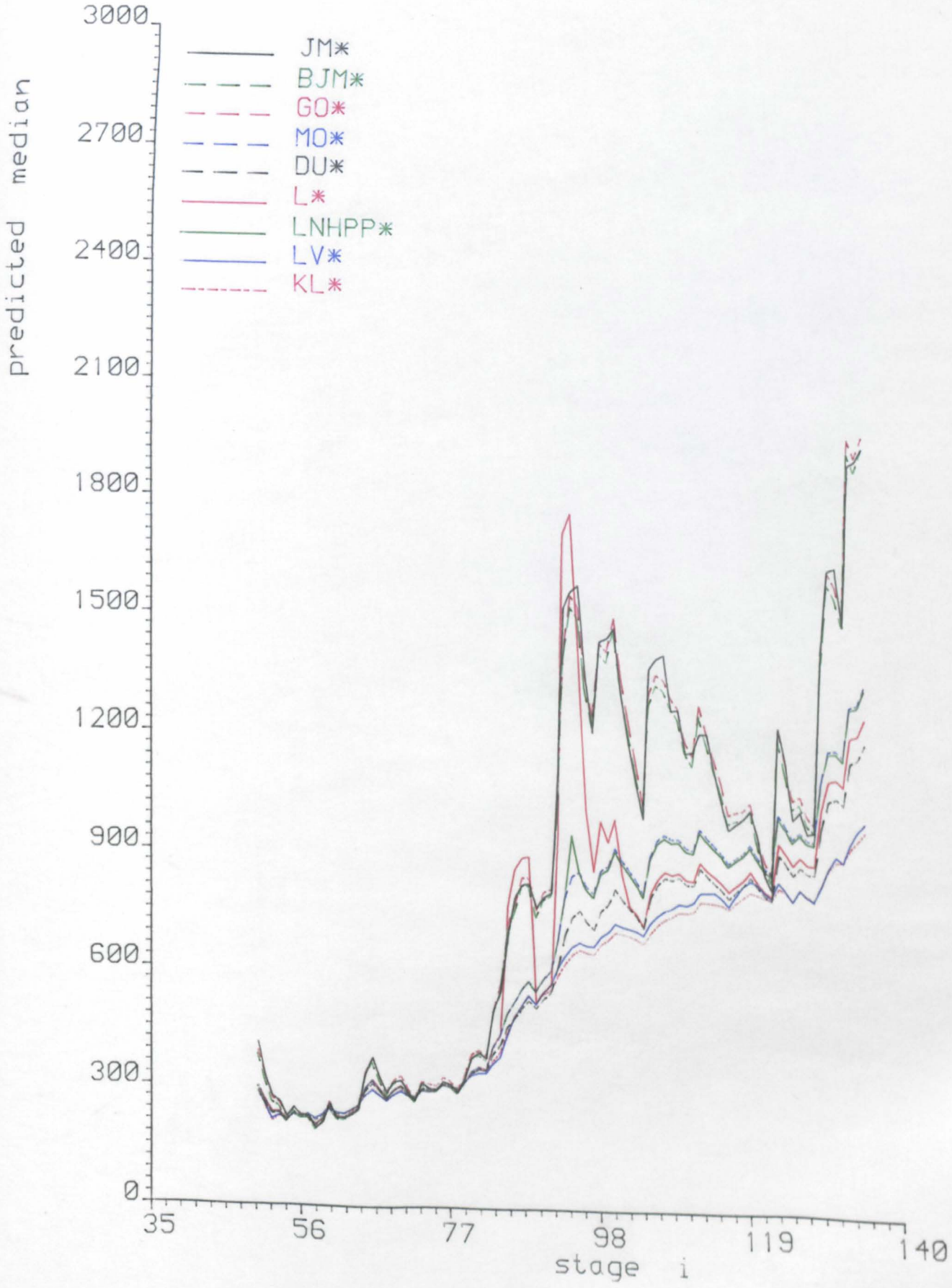
39.	10.	4.	36.	4.
5.	4.	91.	49.	1.
25.	1.	4.	30.	42.
9.	49.	44.	32.	3.
78.	1.	30.	205.	5.
129.	103.	224.	186.	53.
14.	9.	2.	10.	1.
34.	170.	129.	4.	4.
35.	5.	5.	22.	36.
35.	121.	23.	33.	48.
32.	21.	4.	23.	9.
13.	165.	14.	22.	41.
12.	138.	95.	49.	62.
2.	35.	89.	90.	69.
22.	15.	19.	42.	14.
11.	41.	210.	16.	30.
37.	66.	9.	16.	14.
24.	12.	159.	89.	118.
29.	21.	18.	2.	114.
37.	46.	17.	1.	150.
382.	160.	66.	206.	9.
26.	62.	239.	13.	4.
85.	85.	240.	178.	34.
102.	9.	146.	59.	48.
25.	25.	111.	5.	31.
51.	6.	193.	27.	25.
96.	26.	30.	30.	17.
320.	78.	39.	13.	13.
19.	128.	34.	84.	40.
177.	349.	274.	82.	58.
31.	114.	39.	88.	84.
232.	108.	38.	86.	7.
22.	80.	239.	3.	39.
63.	152.	63.	80.	245.
196.	46.	152.	102.	9.
228.	220.	208.	78.	3.
83.	6.	212.	91.	3.
10.	172.	21.	173.	371.
40.	48.	126.	90.	149.
30.	317.	500.	673.	432.
66.	168.	66.	66.	120.
49.	332.			

APPENDIX 4

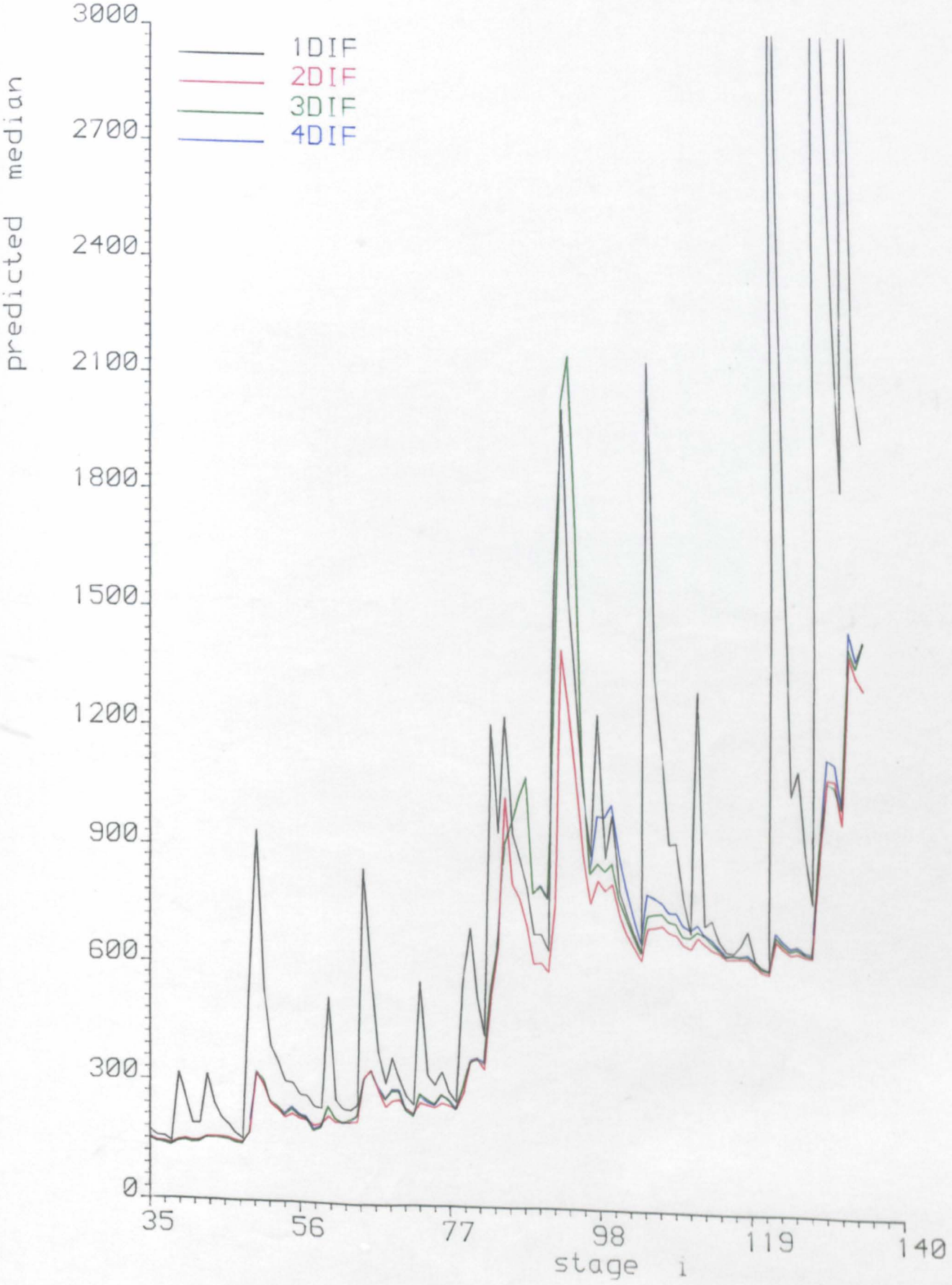
MEDIAN PLOTS



lot A4. 1a Predicted median plot for Musa's System 1 data

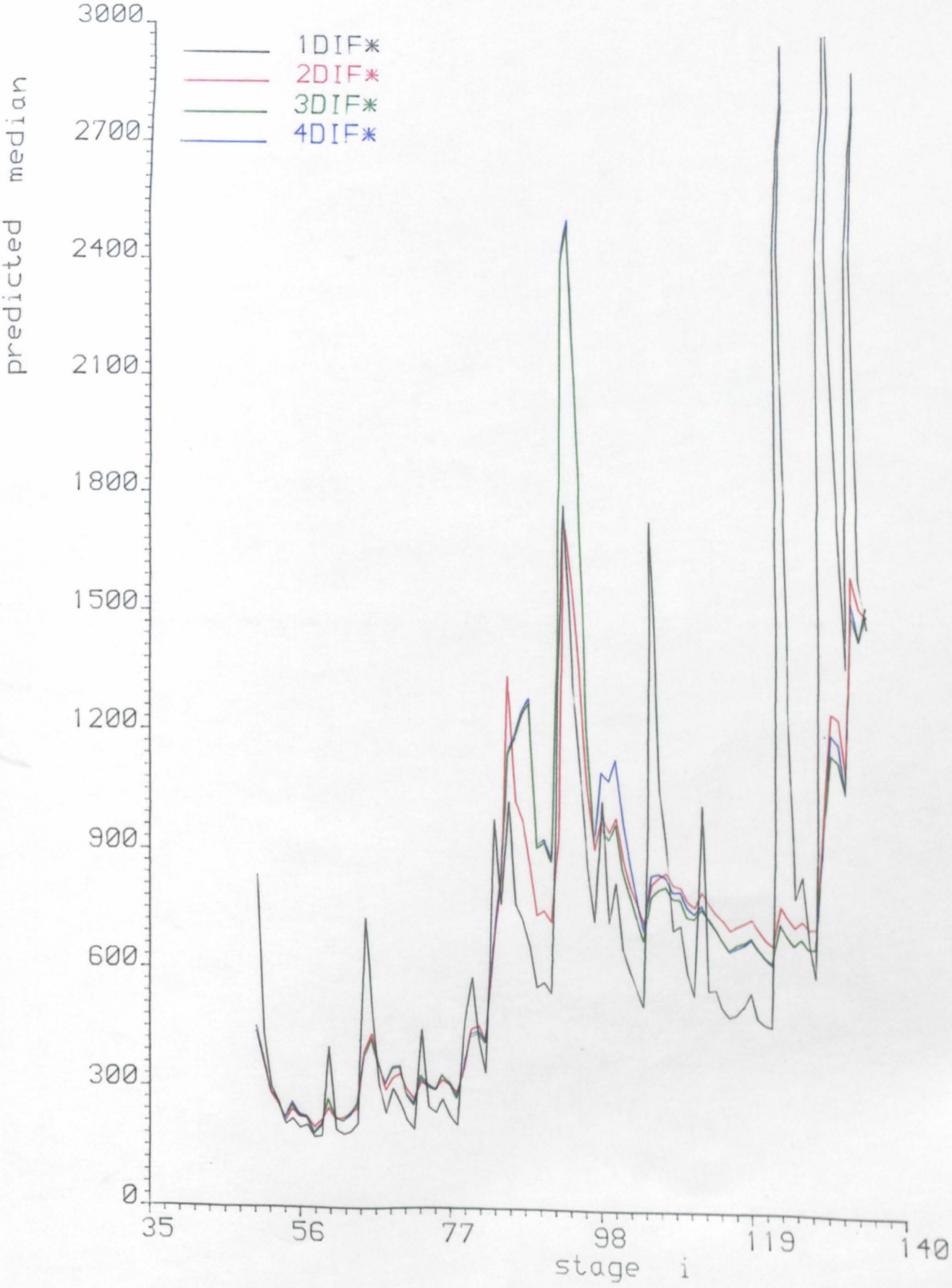


Plot A4.1b Predicted median plot for Musa's System 1 data
Models adapted using parametric spline



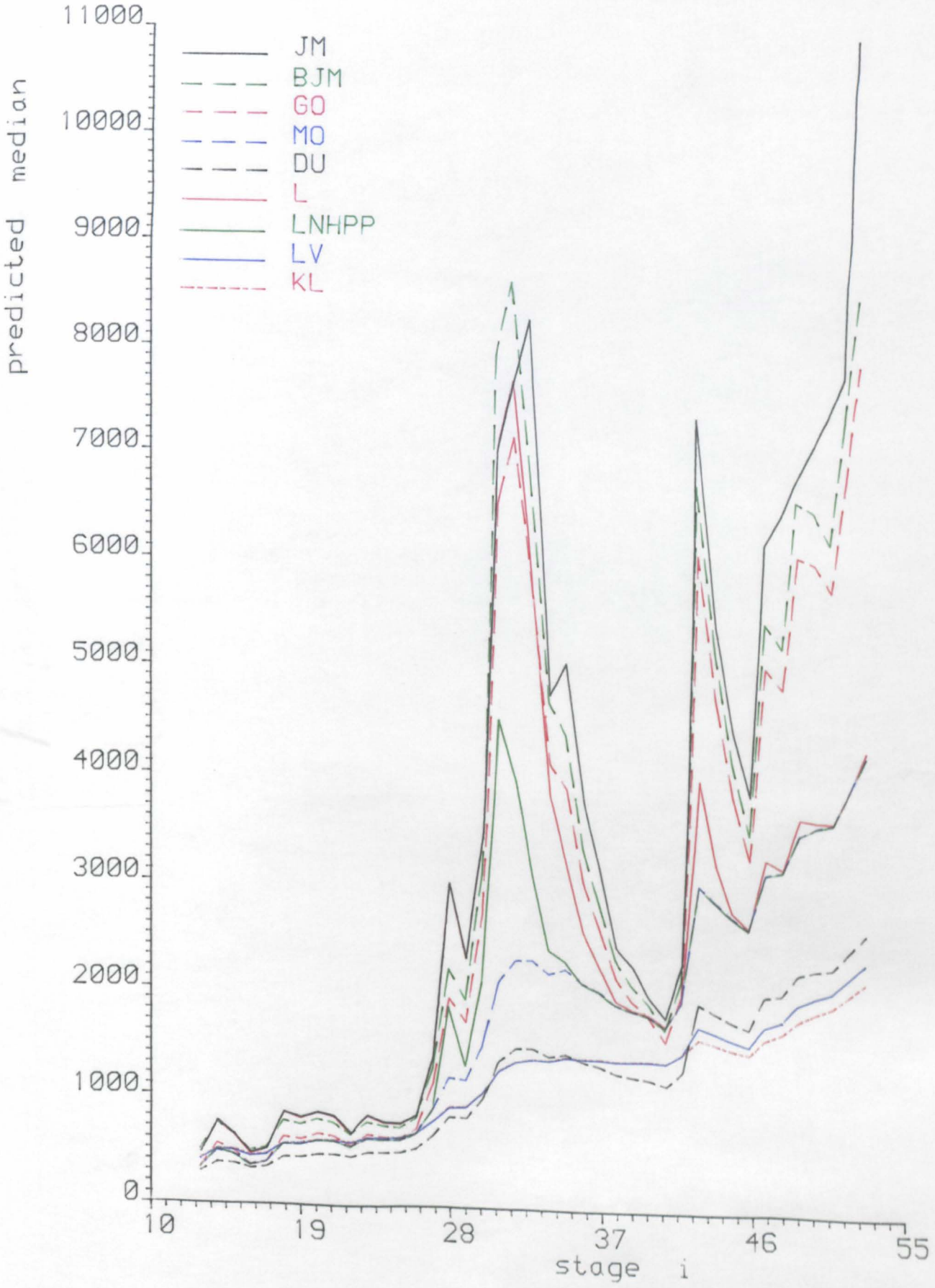
Plot A4.1c

Predicted median plot for Musa's System 1 data
Monotone rate estimates with exponential
failure time distribution

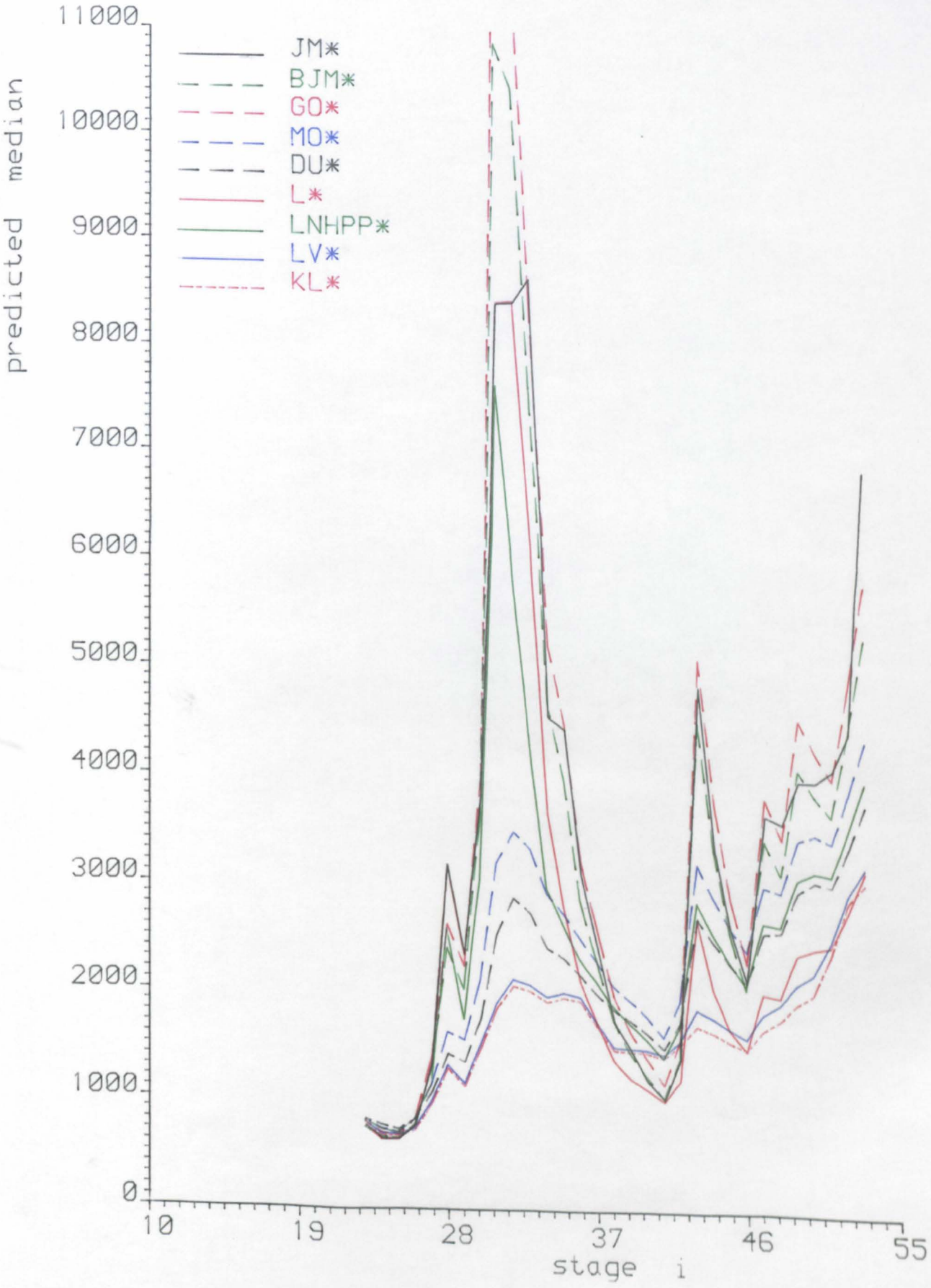


Plot A4.1d

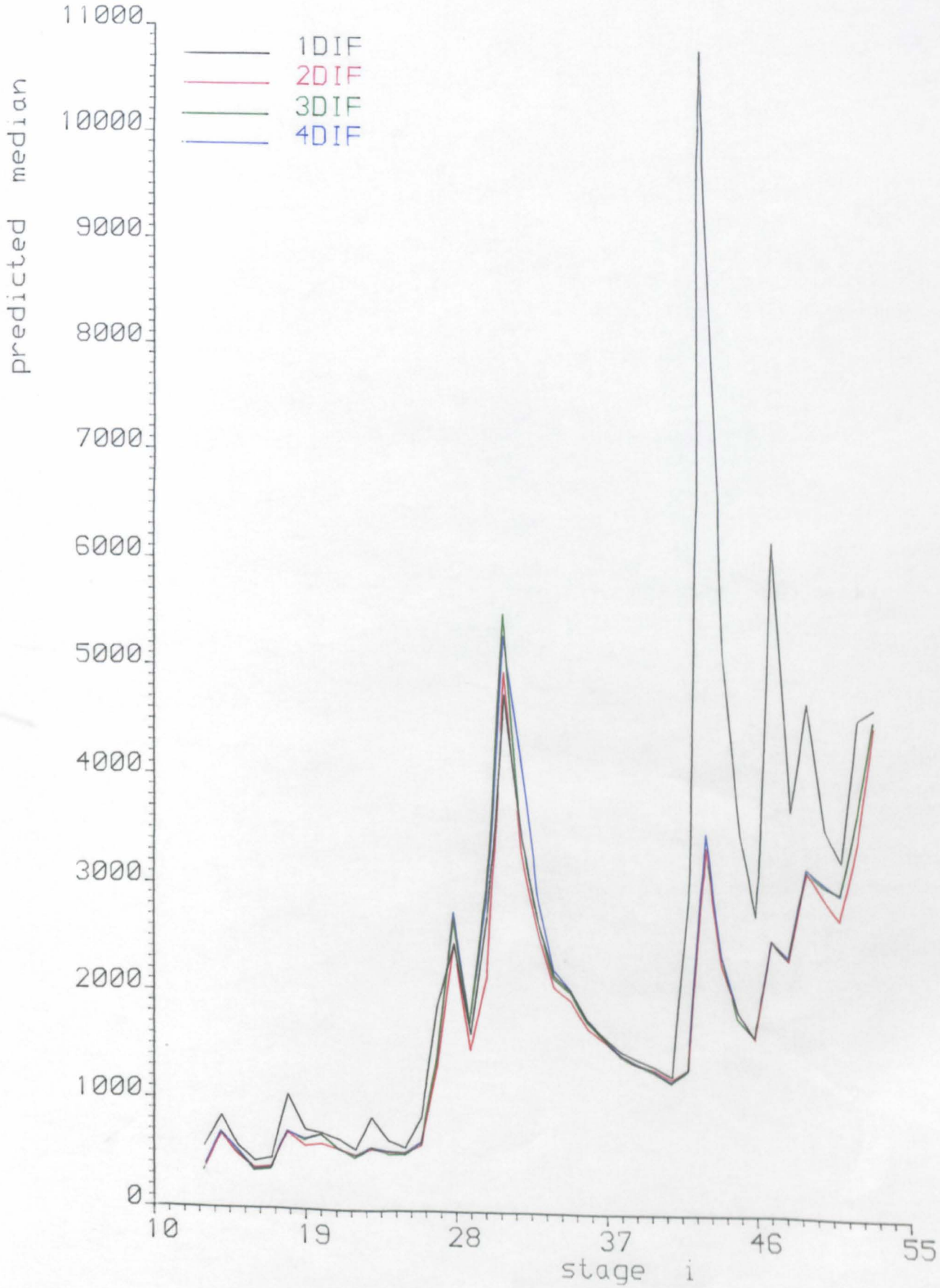
Predicted median plot for Musa's System 1 data
Monotone rate estimates with exponential and parametric spline adapted failure time distribution



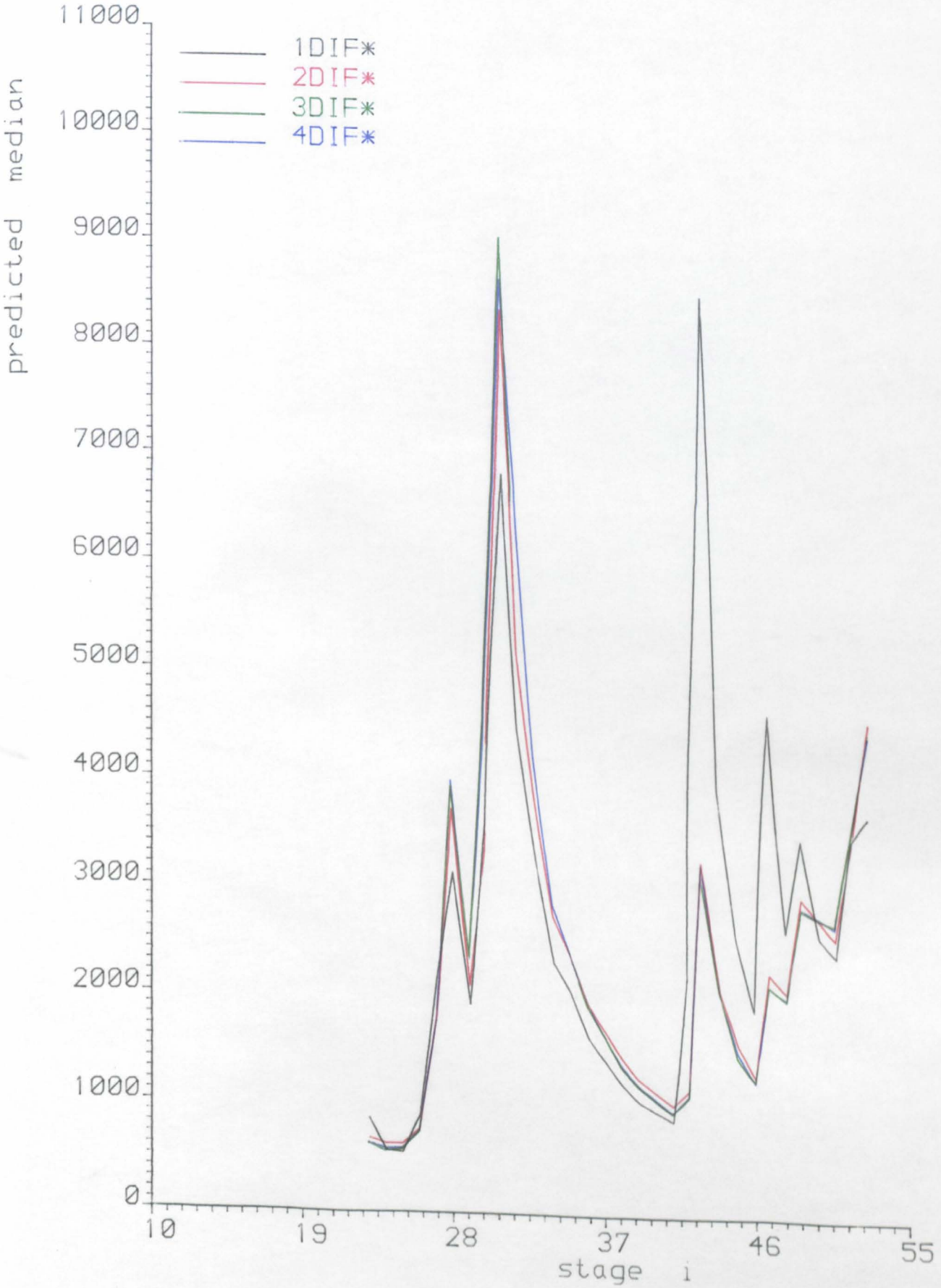
Plot A4.2a Predicted median plot for Musa's System 2 data



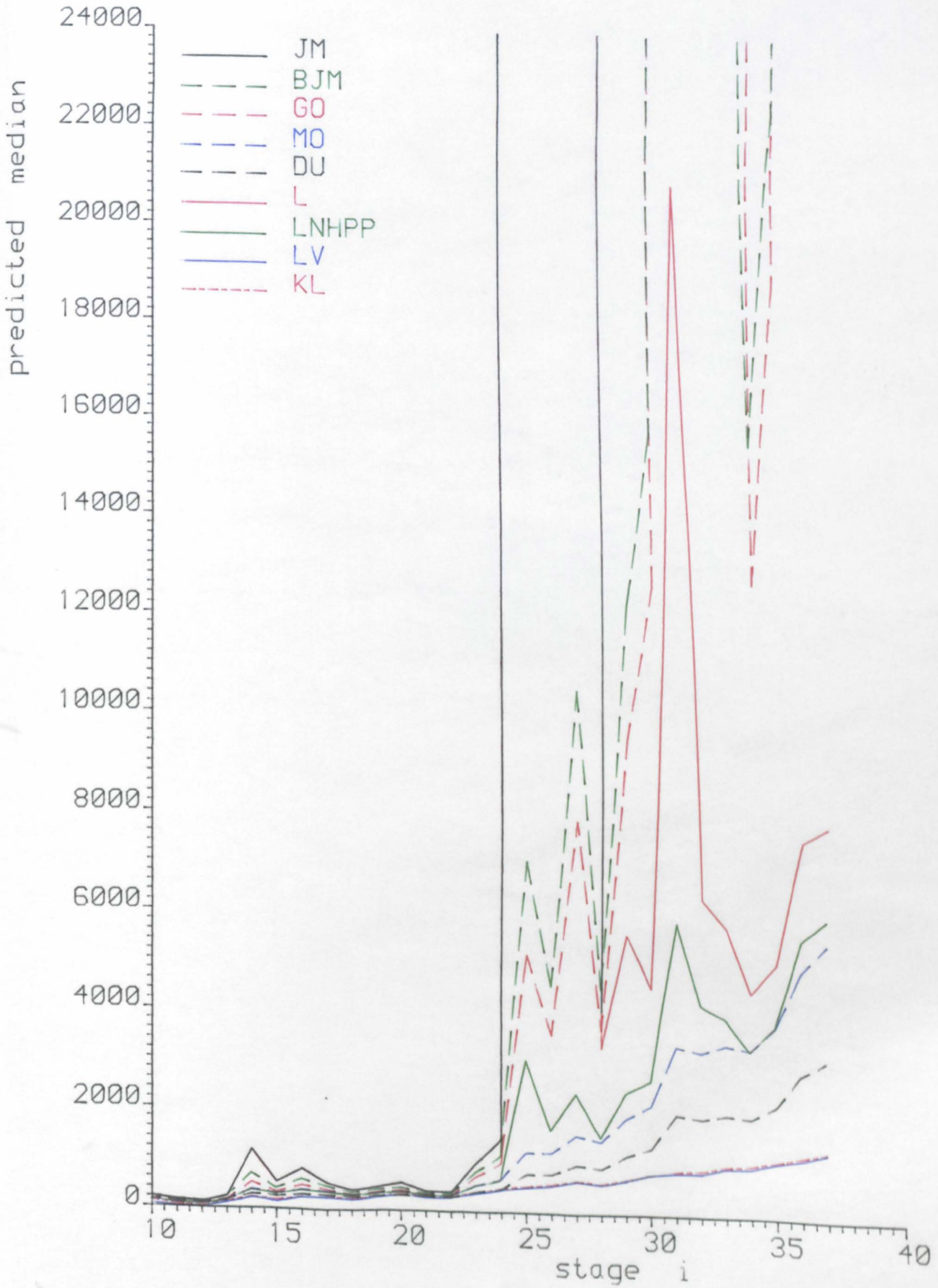
Plot A4.2b Predicted median plot for Musa's System 2 data
Models adapted using parametric spline



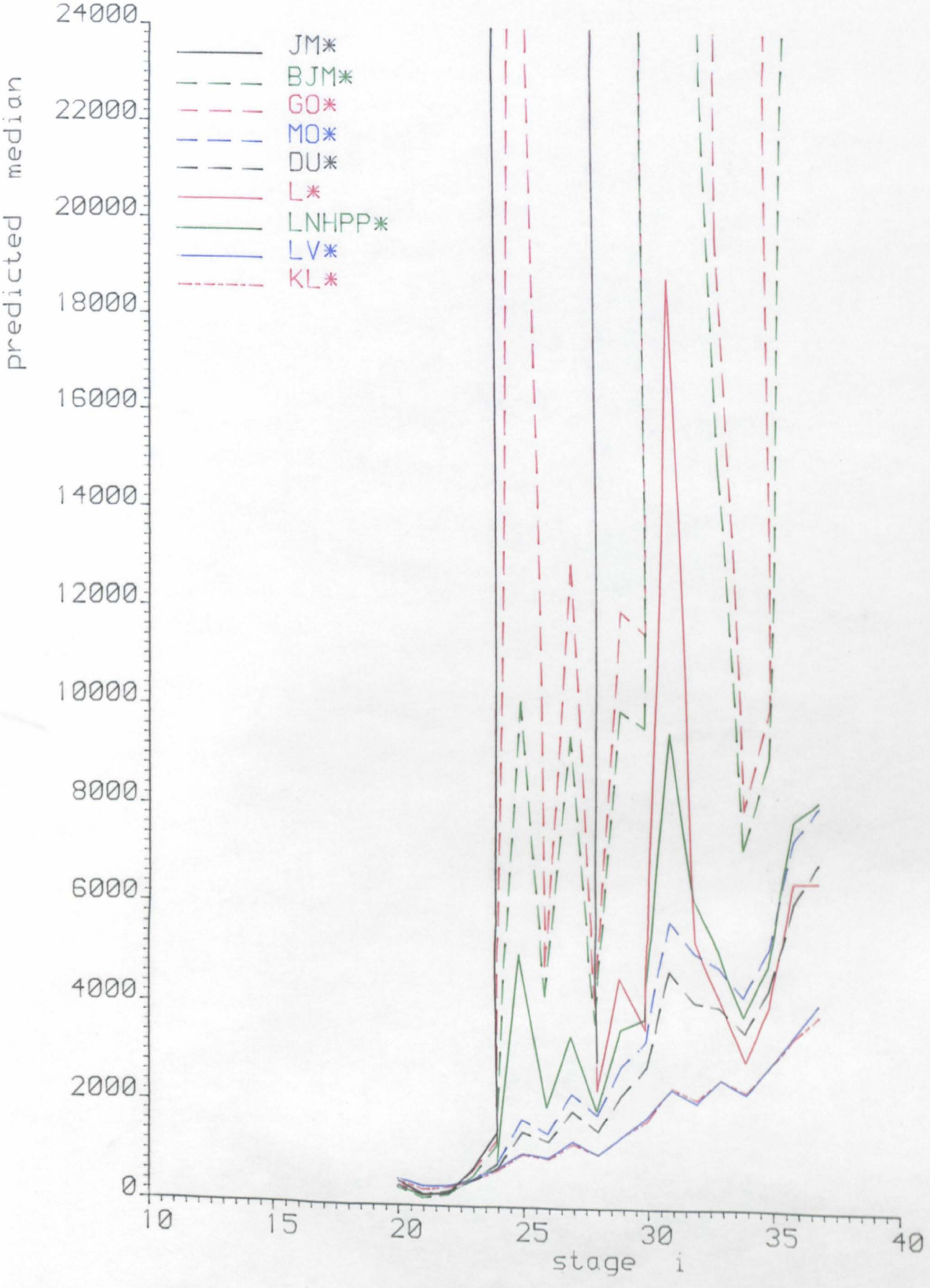
Plot A4.2c Predicted median plot for Musa's System 2 data
Monotone rate estimates with exponential
Failure time distribution



Plot A4.2d Predicted median plot for Musa's System 2 data Monotone rate estimates with exponential and parametric spline adapted failure time distribution

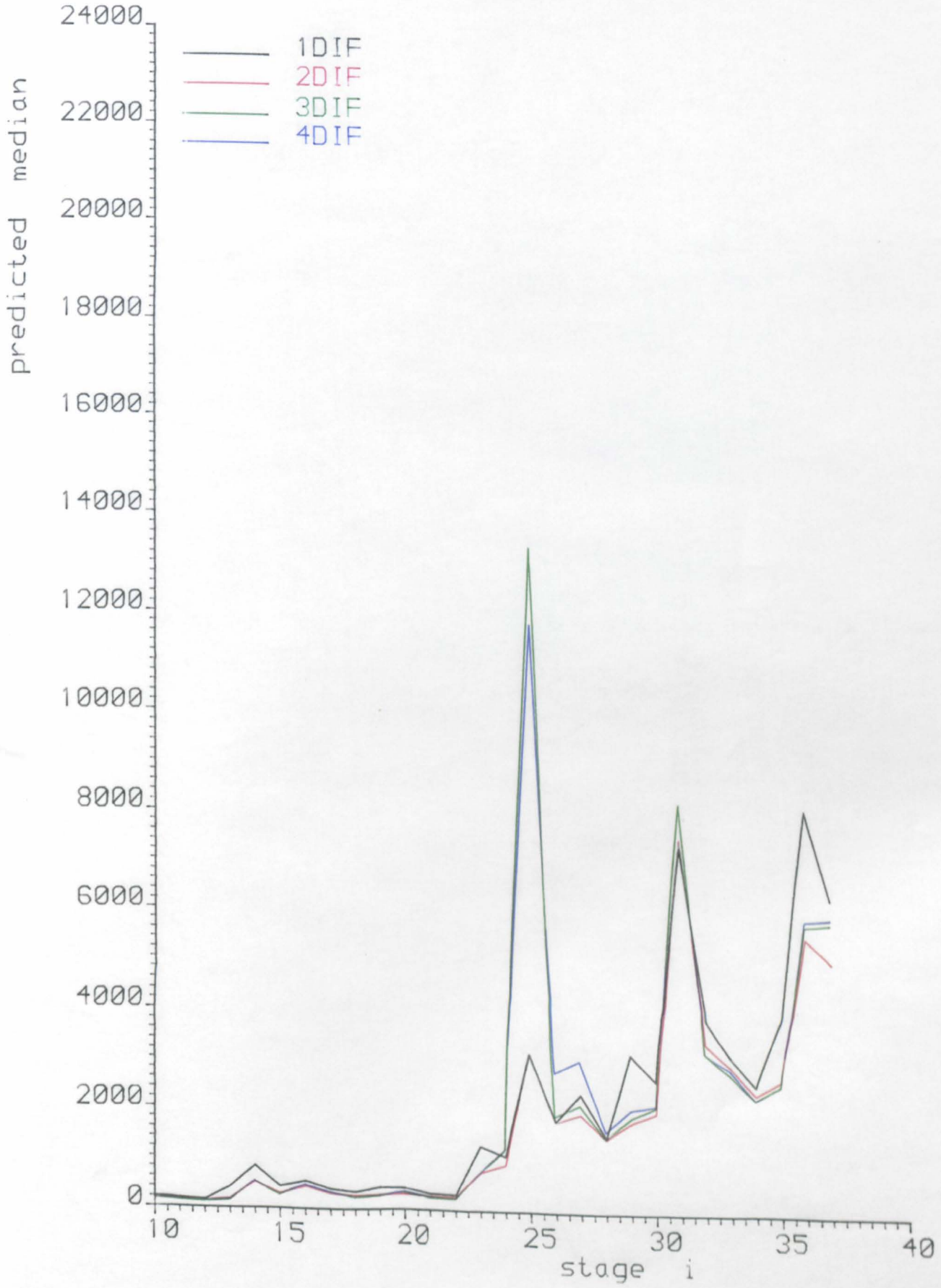


Plot A4.3a Predicted median plot for Musa's System 3 data

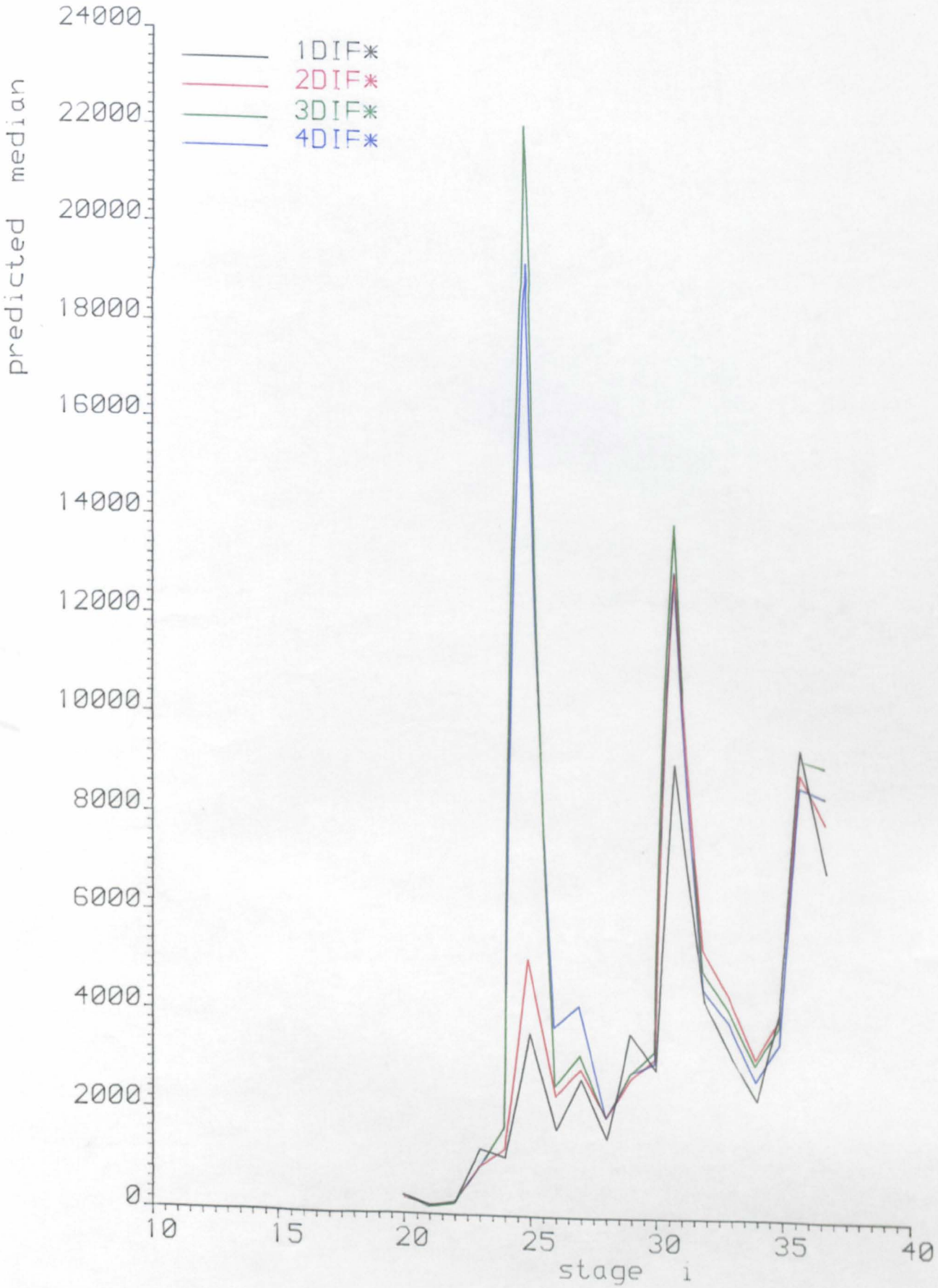


Plot A4.3b

Predicted median plot for Musa's System 3 data
Models adapted using parametric spline

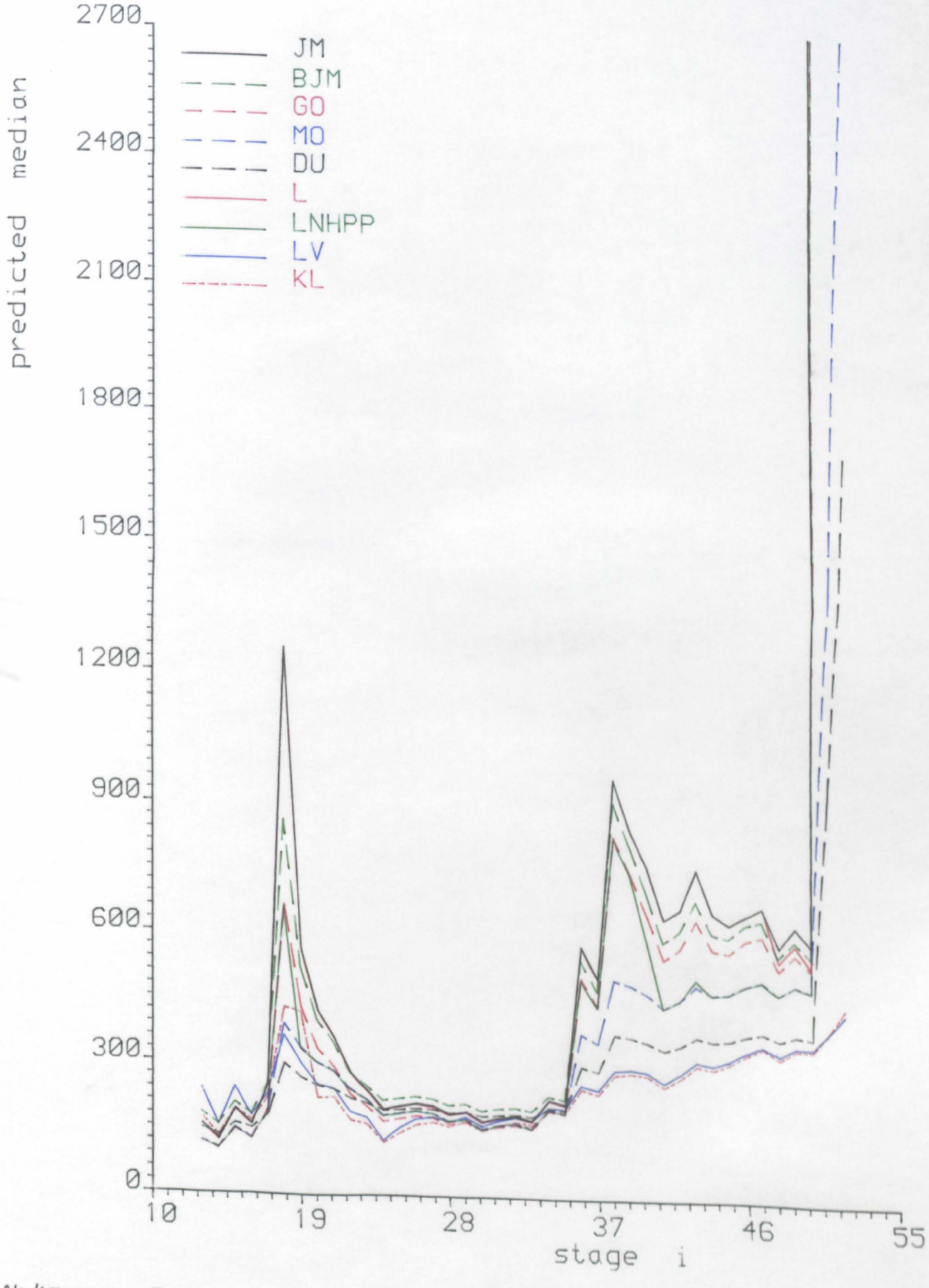


Plot A4.3c Predicted median plot for Musa's System 3 data
Monotone rate estimates with exponential
Failure time distribution

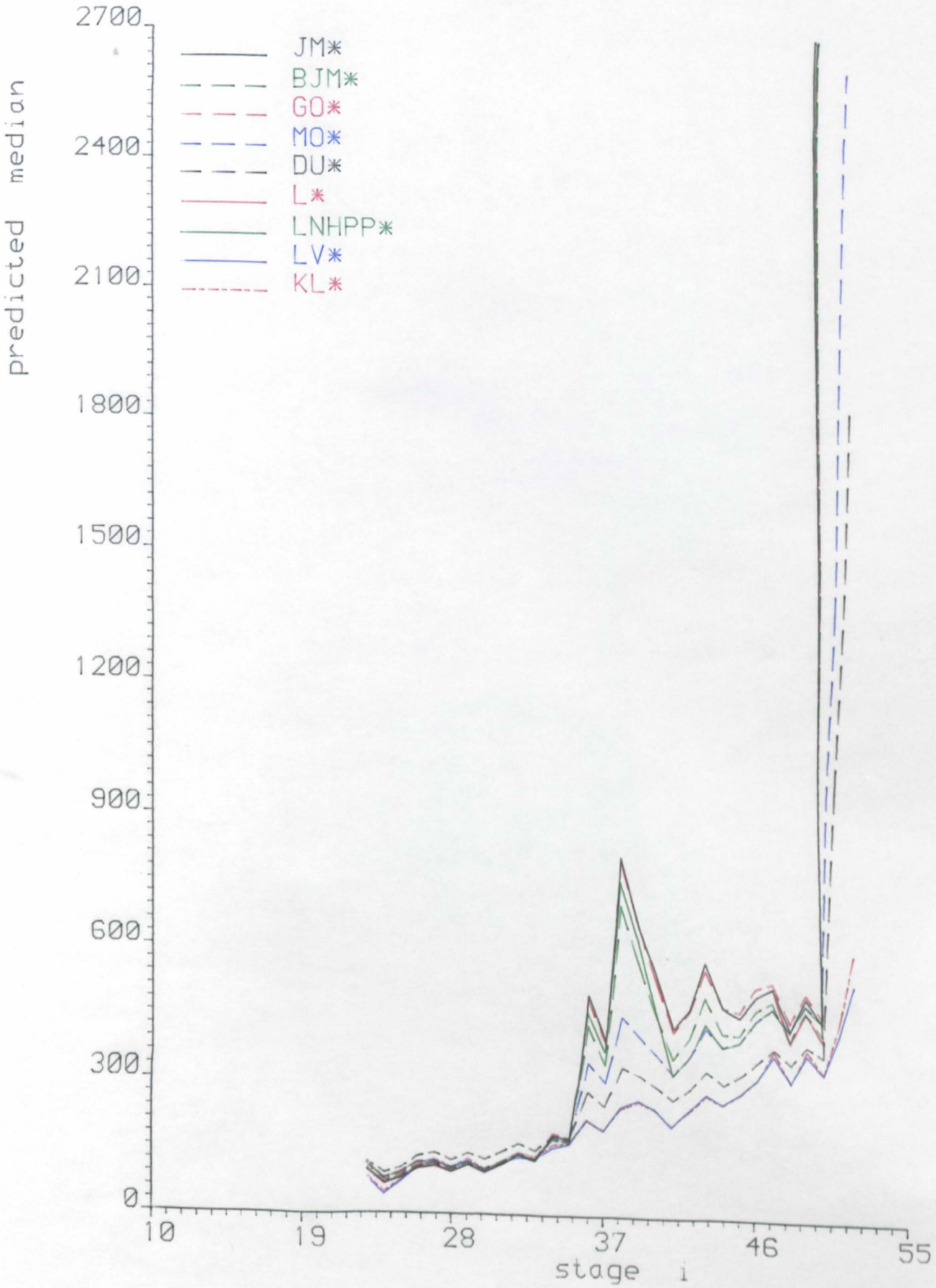


Plot A4. 3d

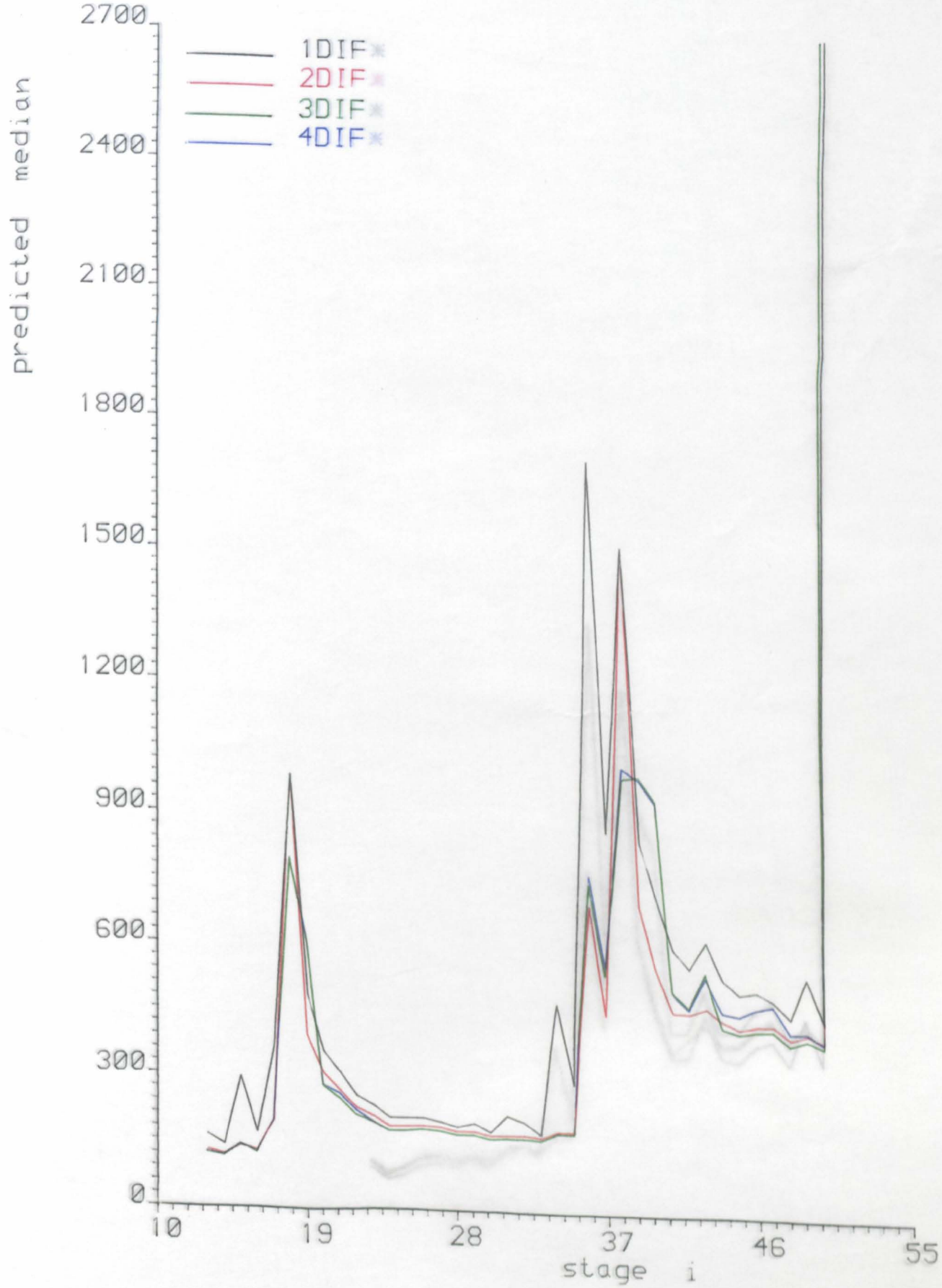
Predicted median plot for Musa's System 3 data Monotone rate estimates with exponential and parametric spline adapted failure time distribution



Plot A4.4a Predicted median plot for Musa's System 4 data

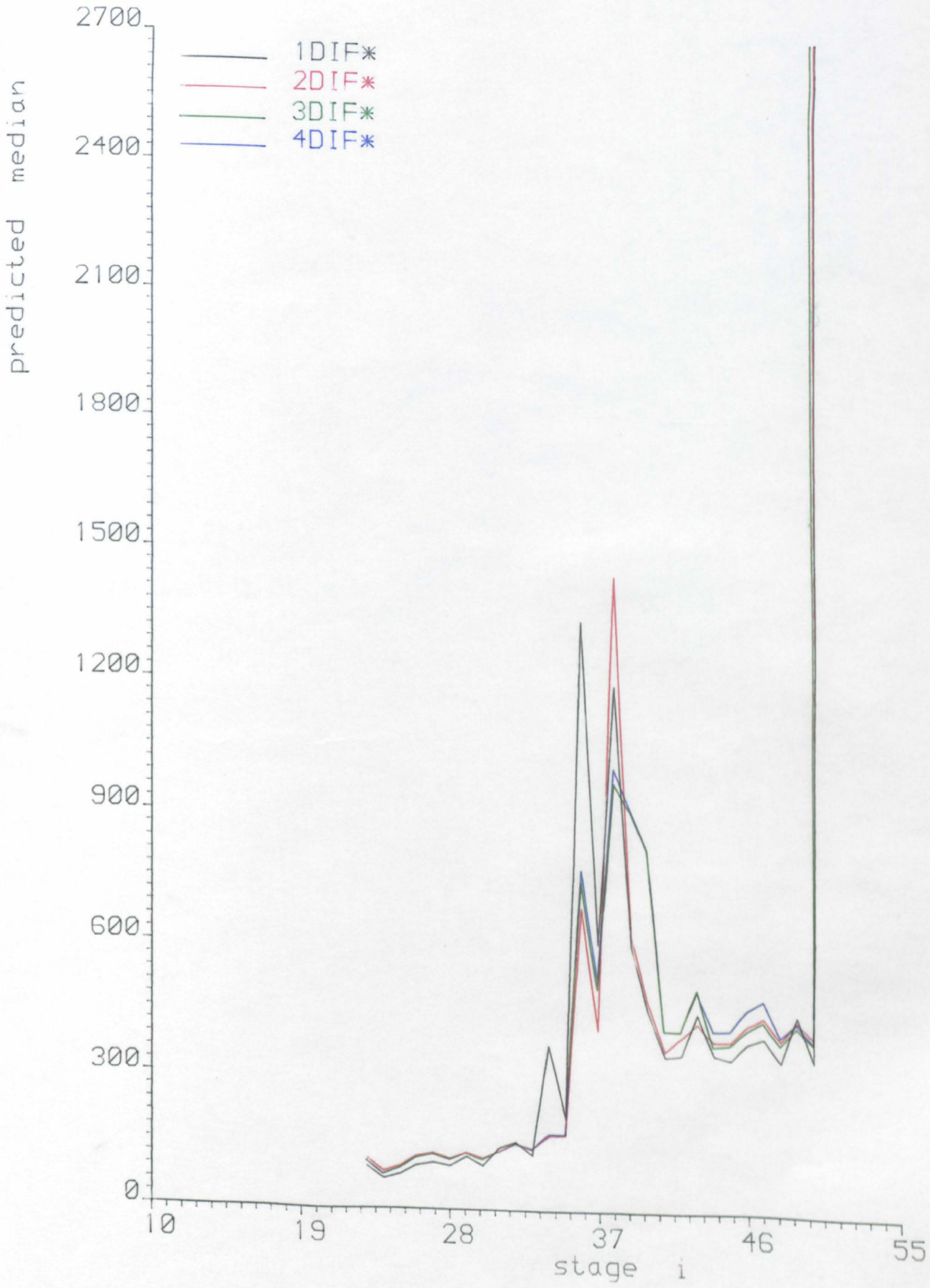


Plot A4.4b Predicted median plot for Musa's System 4 data
Models adapted using parametric spline



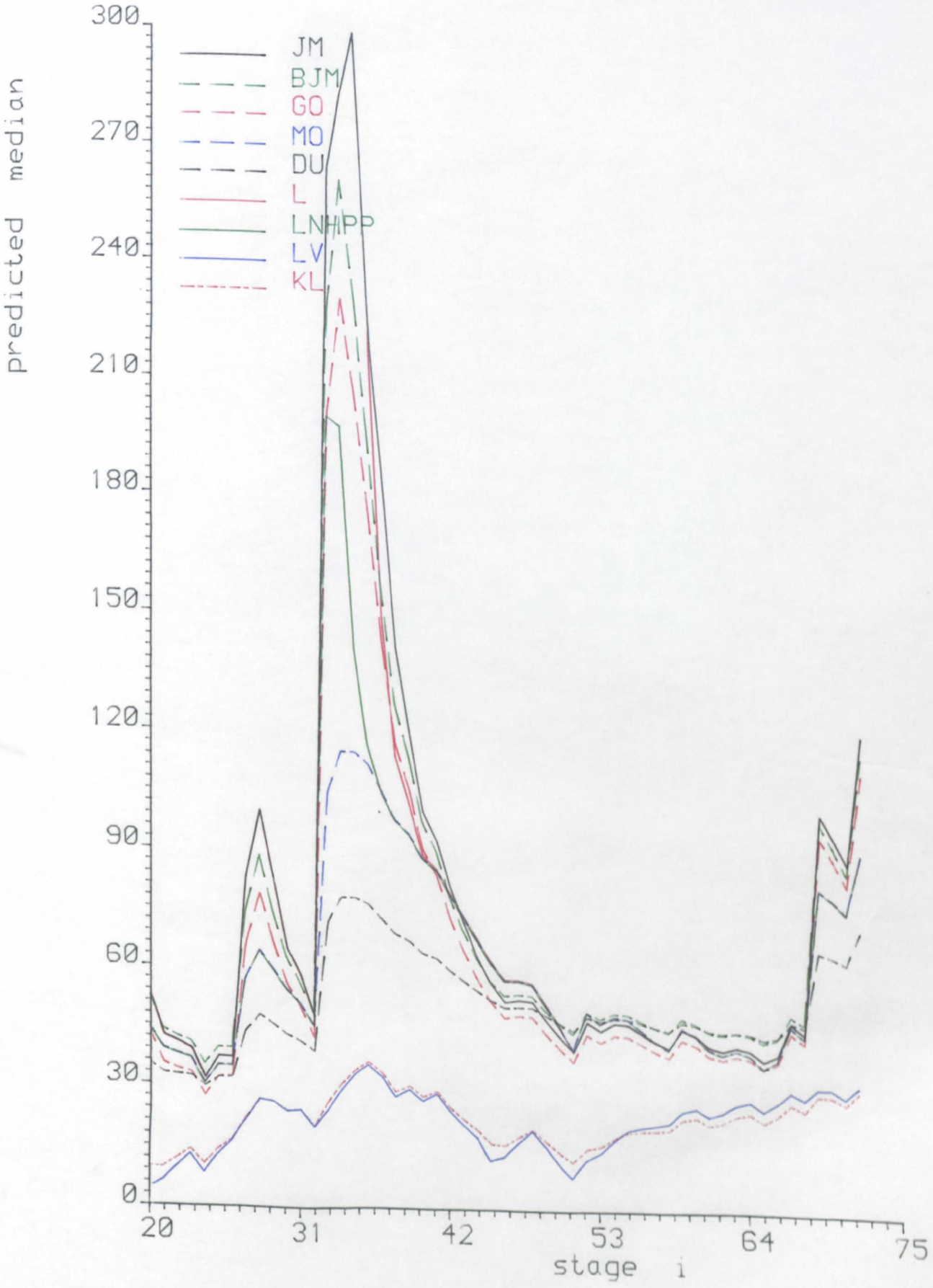
Plot A4.4c

Predicted median plot for Musa's System 4 data
Monotone rate estimates with exponential and
Failure time distribution

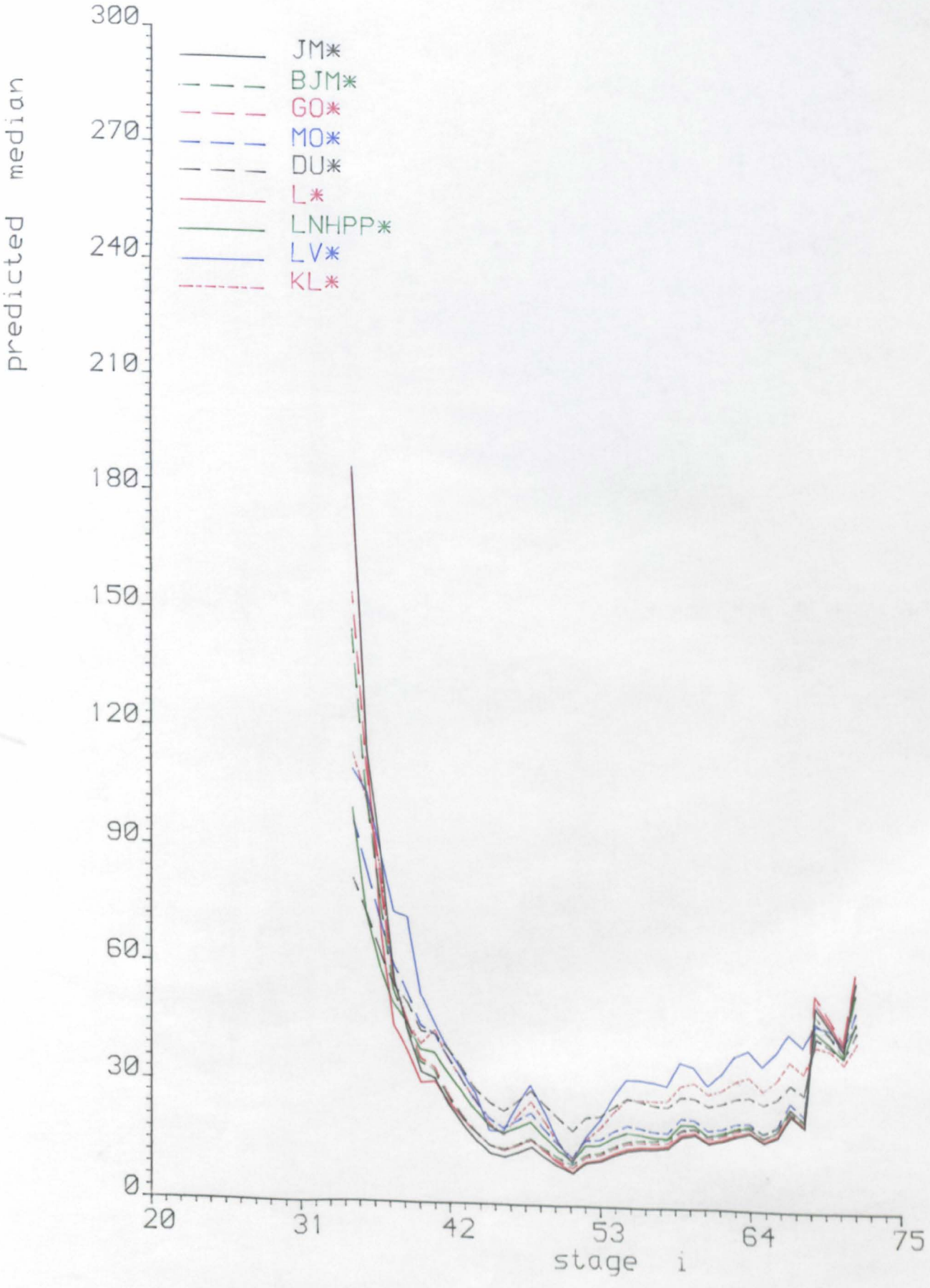


Plot A4.4d

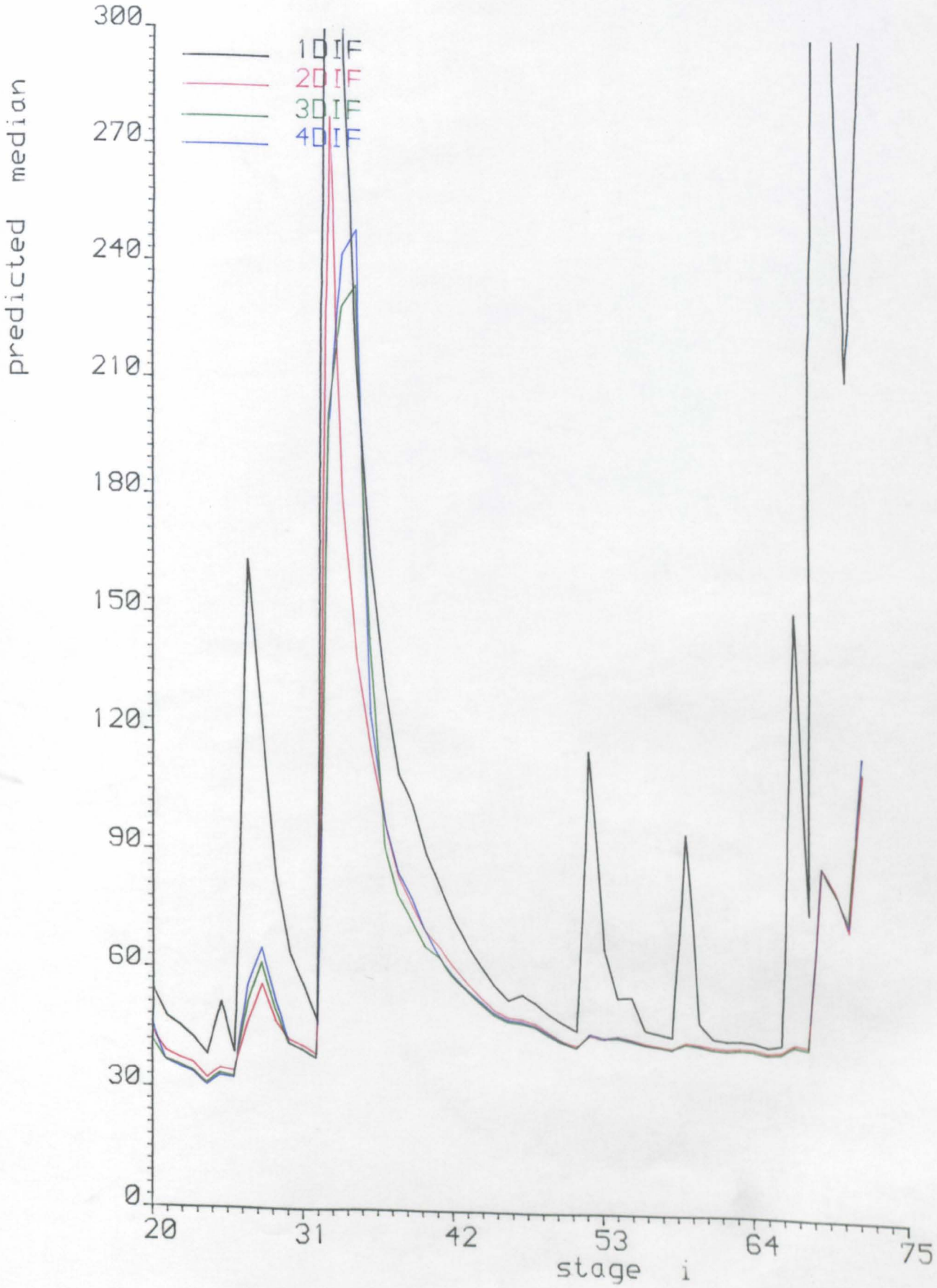
Predicted median plot for Musa's System 4 data Monotone rate estimates with exponential and parametric spline adapted failure time distribution



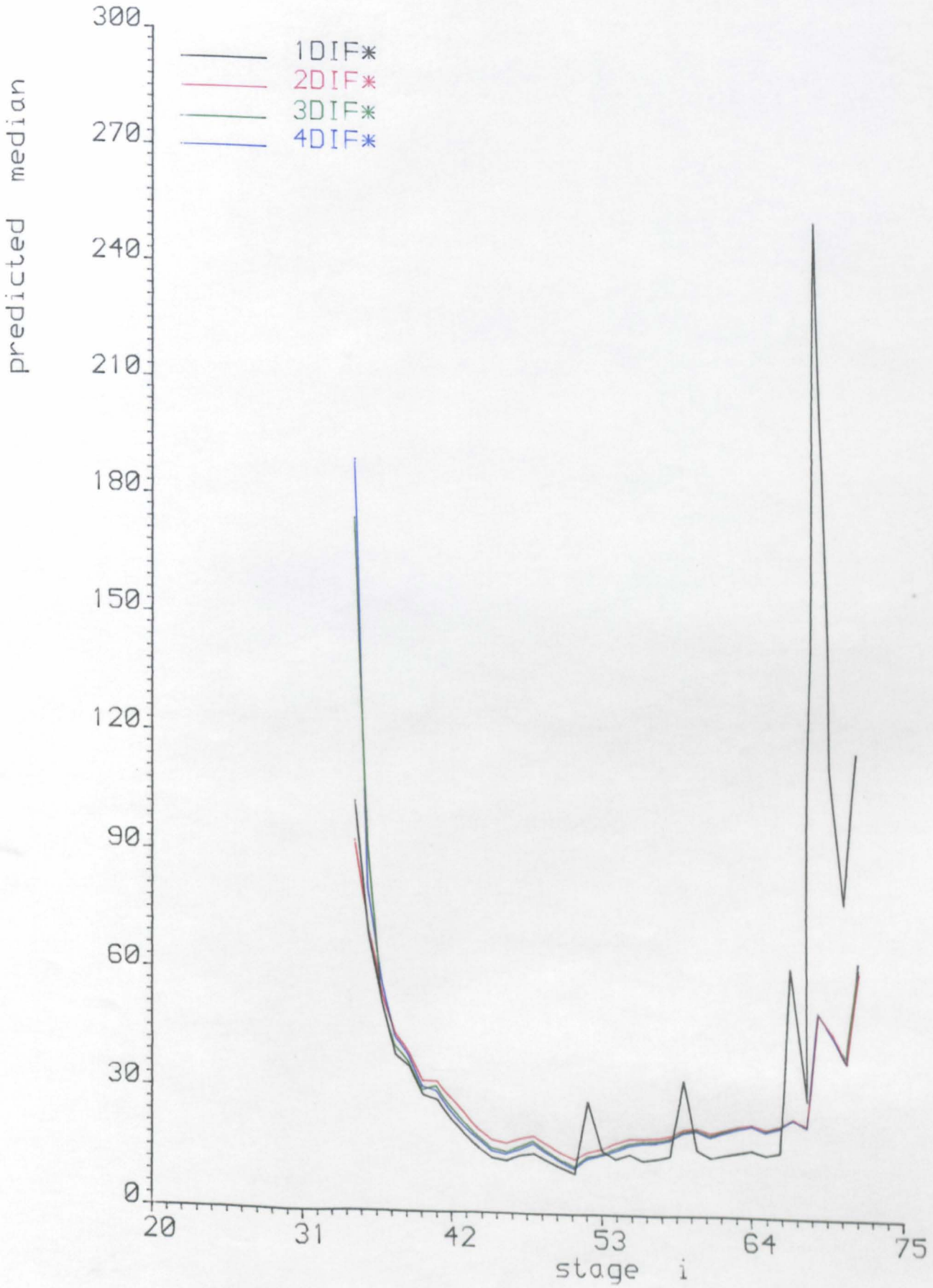
Plot A4.5a Predicted median plot for Musa's System 6 data



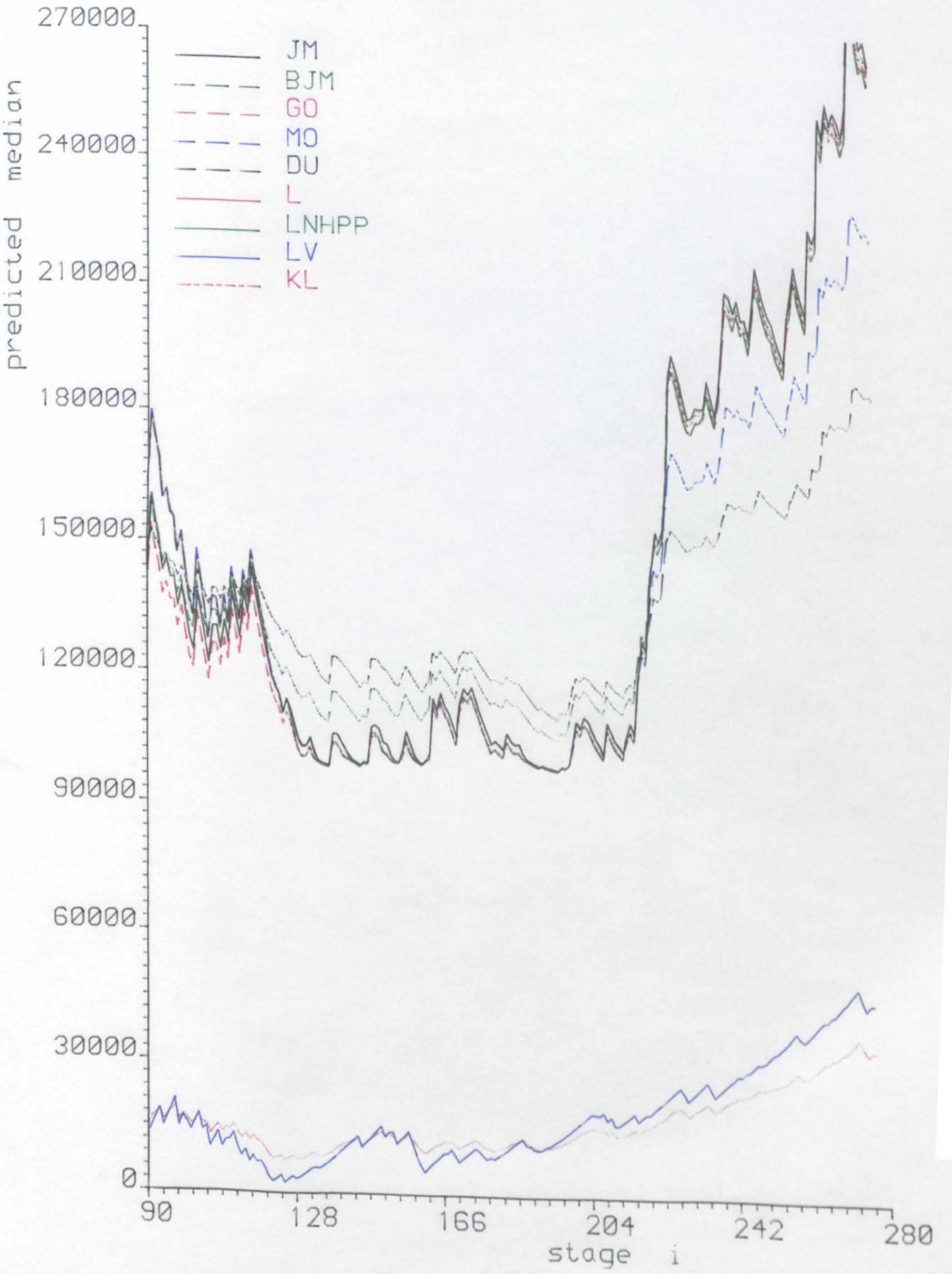
Plot A4.5b Predicted median plot for Musa's System 6 data
Models adapted using parametric spline



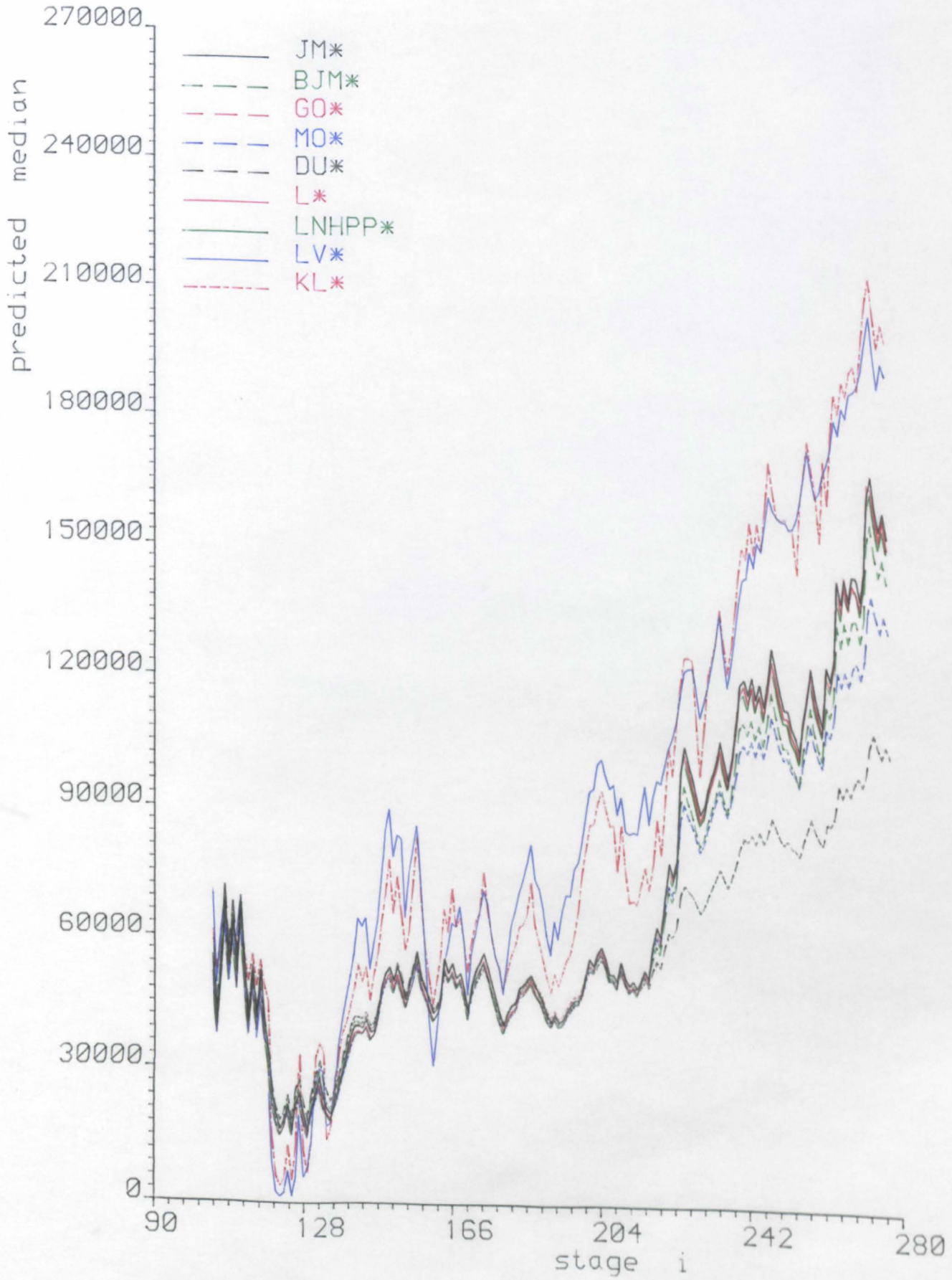
Plot A4.5c Predicted median plot for Musa's System 6 data
Monotone rate estimates with exponential
Failure time distribution



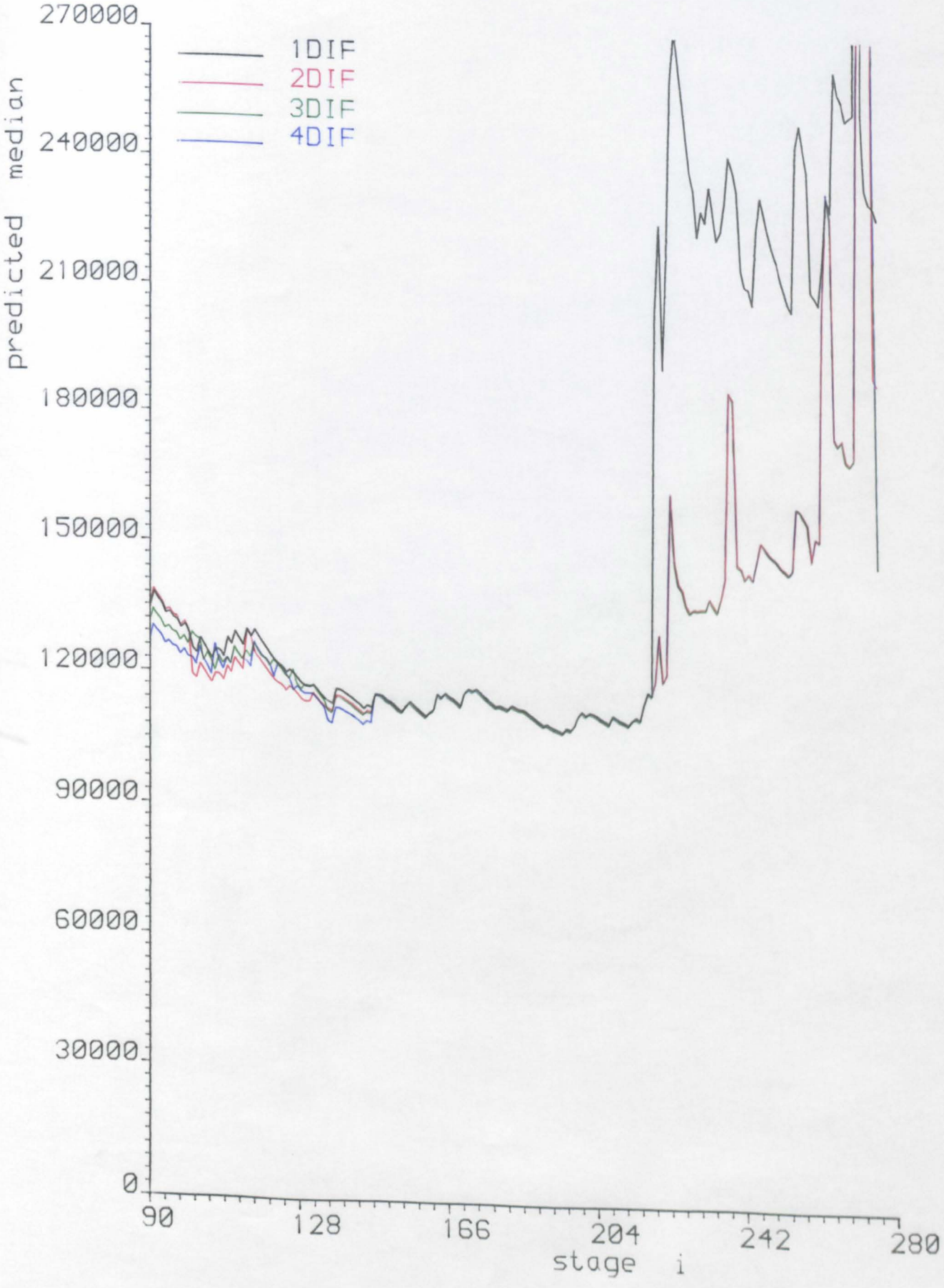
Plot A4.5d Predicted median plot for Musa's System 6 data Monotone rate estimates with exponential and parametric spline adapted failure time distribution



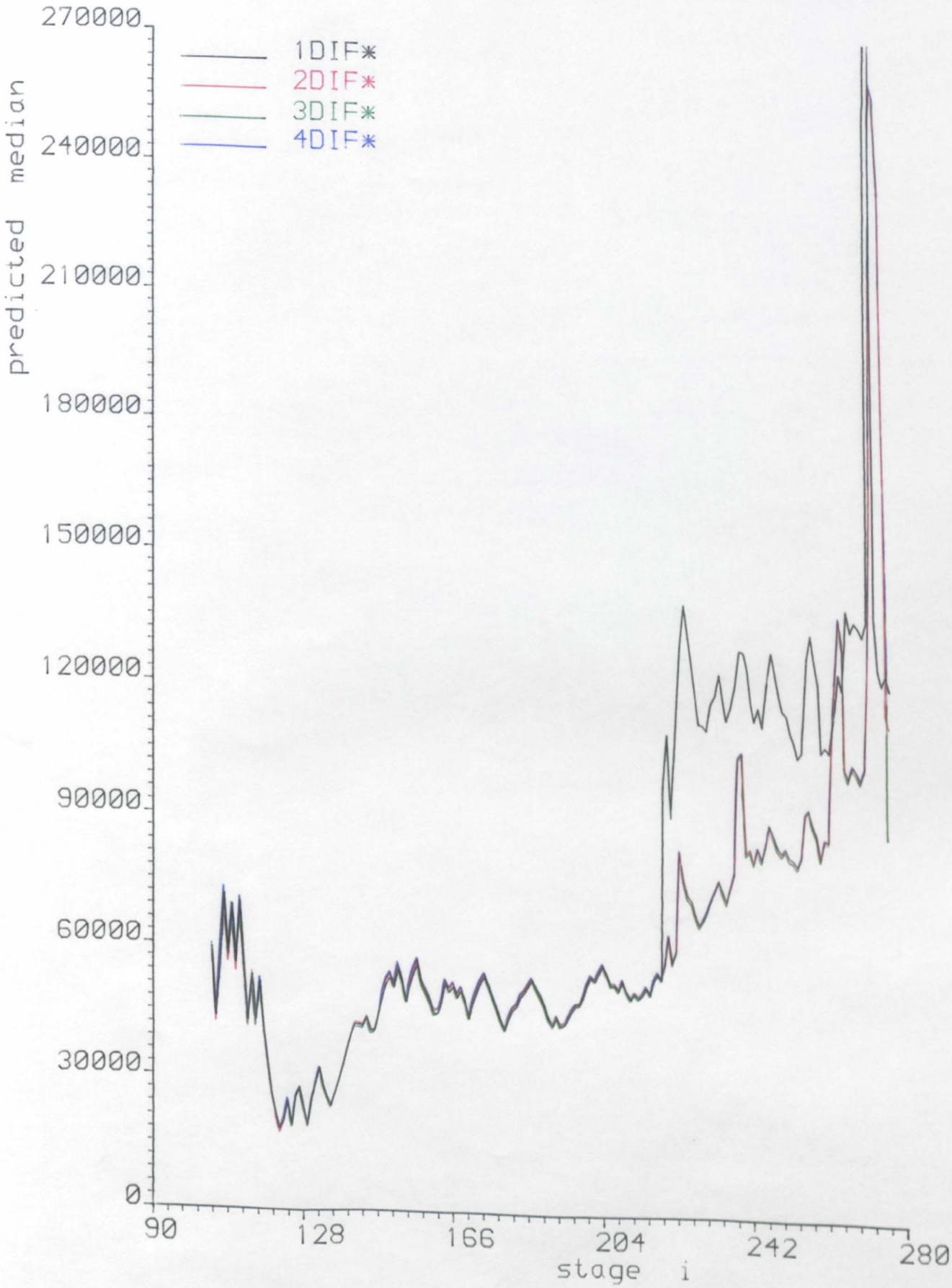
Plot A4. 6a Predicted median plot for Musa's System SS3 data



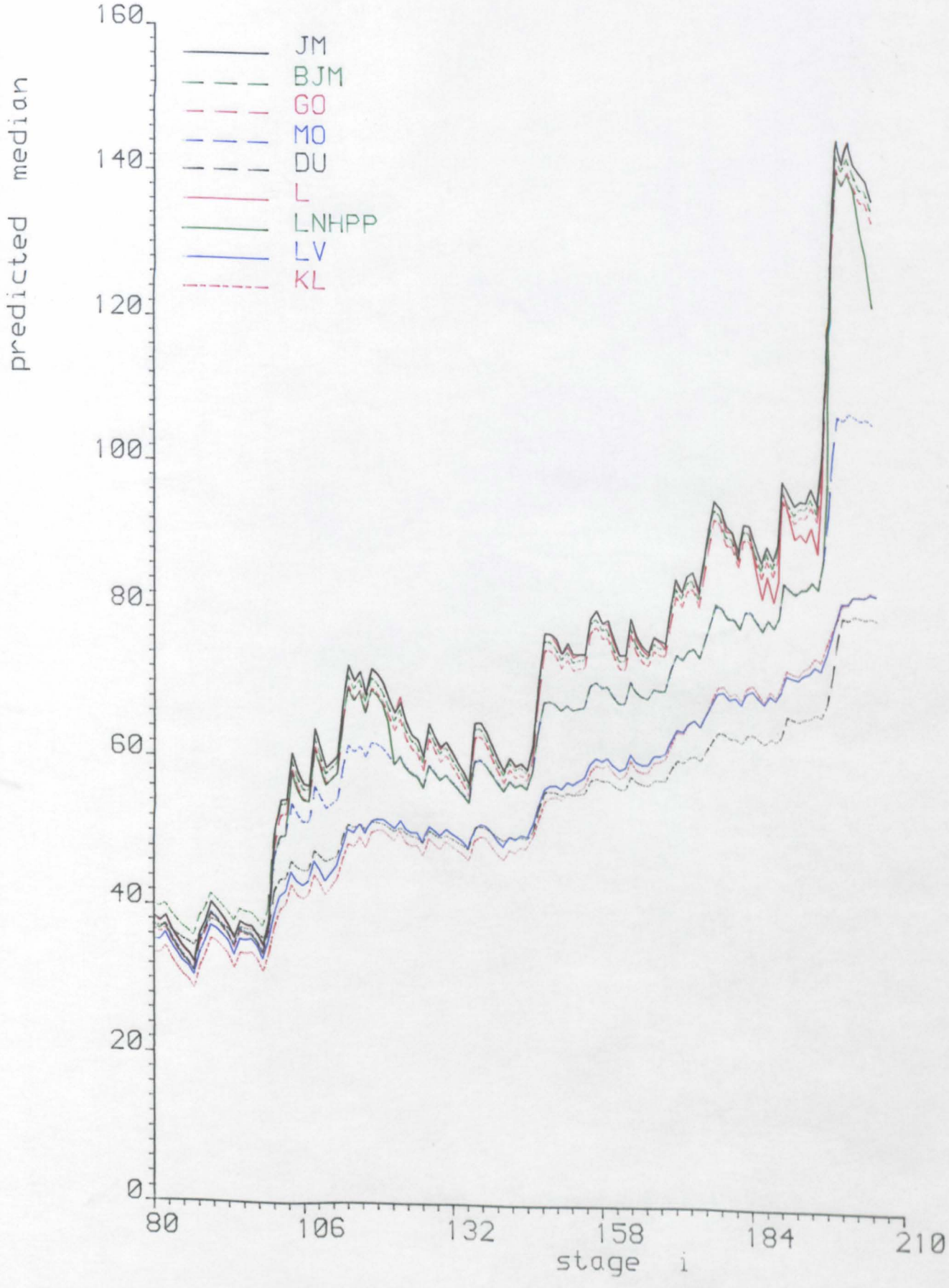
lot A4.6b Predicted median plot for Musa's System SS3 data
Models adapted using parametric spline



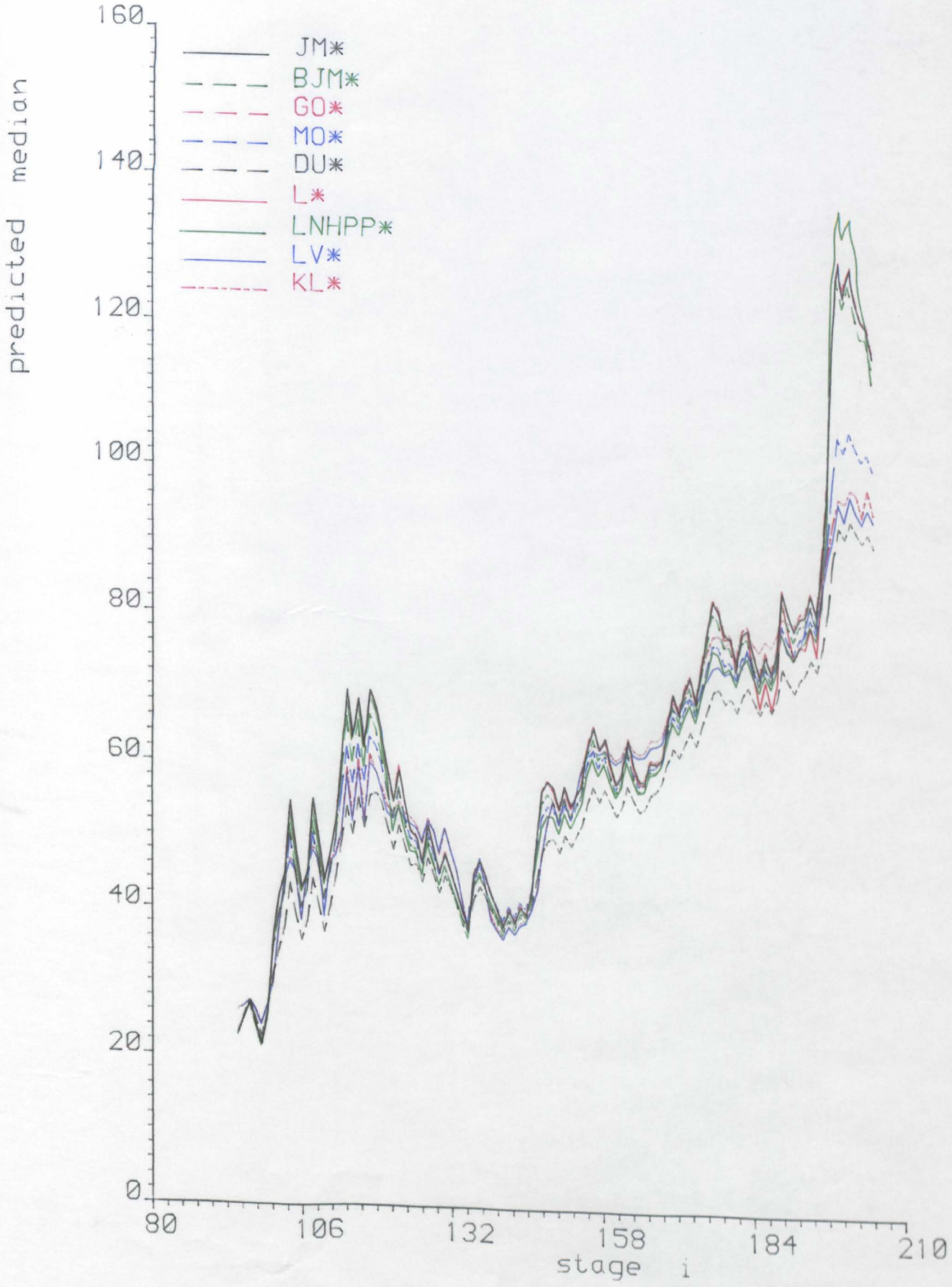
Plot A4.6c Predicted median plot for Musa's System SS3 data
Monotone rate estimates with exponential
Failure time distribution



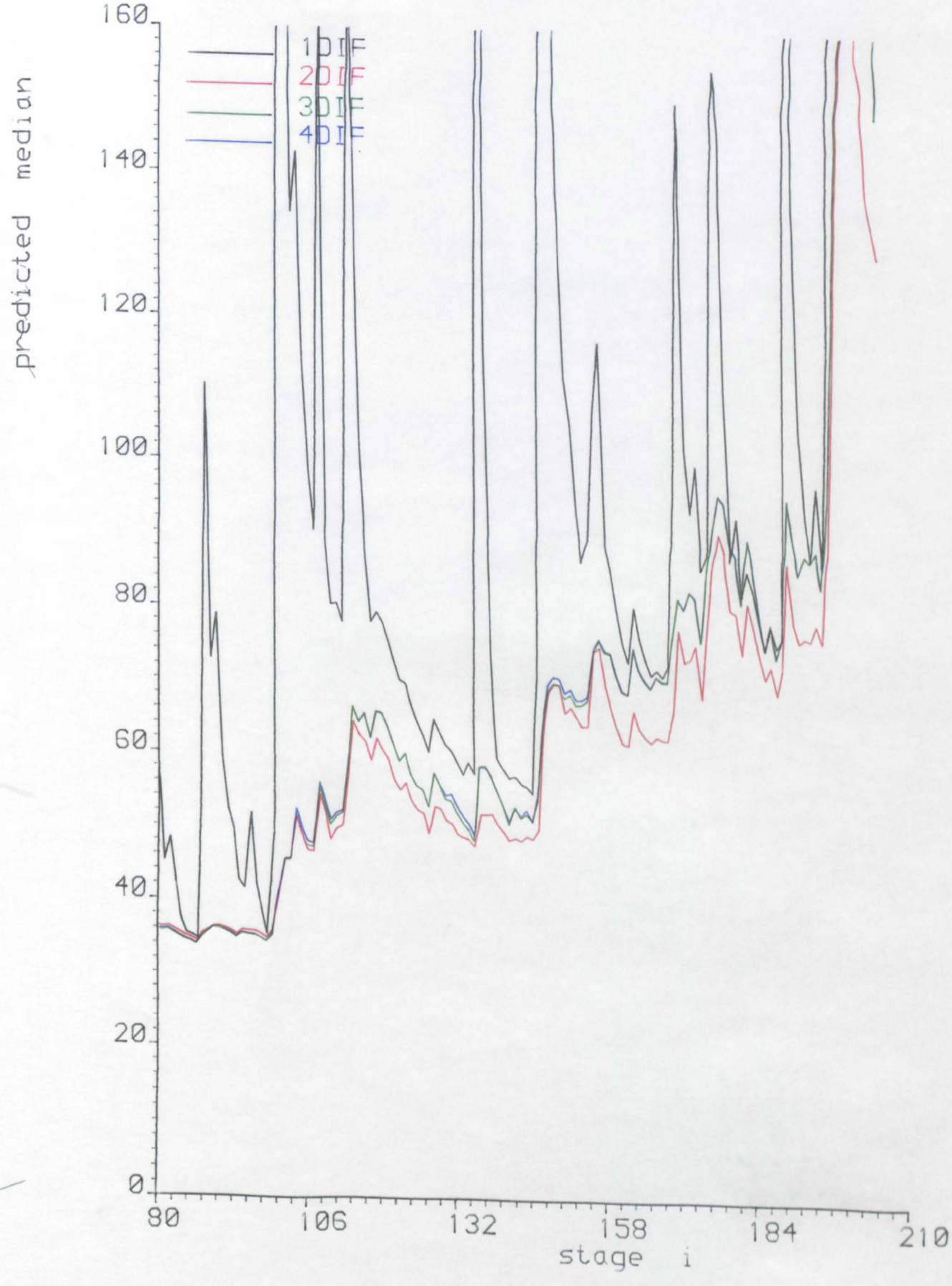
Plot A4.6d Predicted median plot for Musa's System SS3 data Monotone rate estimates with exponential and parametric spline adapted failure time distribution



Plot A4. 7a Predicted median plot for British Aerospace data

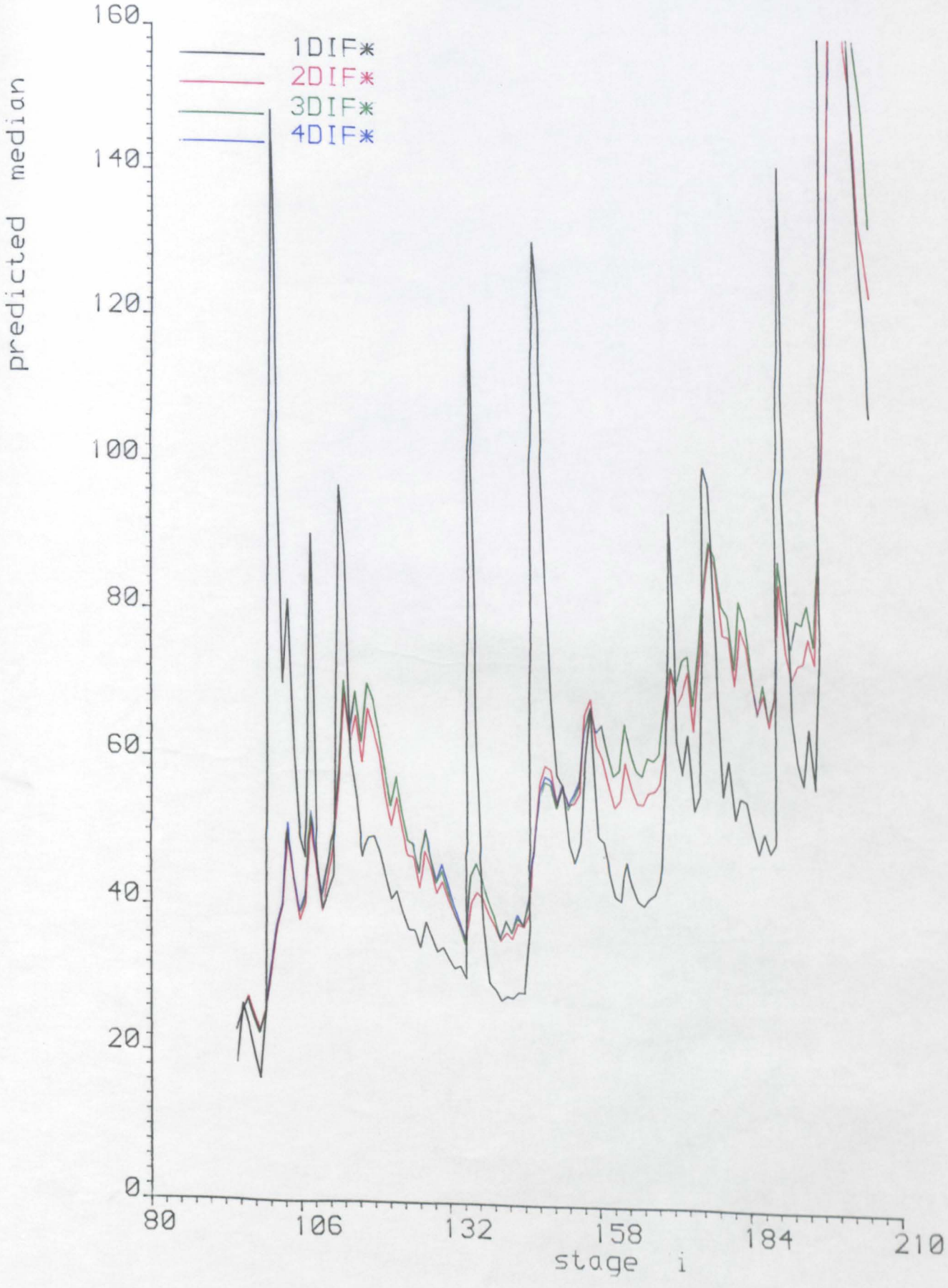


Plot A4.7b Predicted median plot for British Aerospace data
Models adapted using parametric spline



Plot A4.7c

Predicted median plot for British Aerospace data
Monotone rate estimates with exponential
Failure time distribution



Plot A4.7d Predicted median plot for British Aerospace data Monotone rate estimates with exponential and parametric spline adapted failure time distribution

REFERENCES AND BIBLIOGRAPHY

ABDEL-GHALY, A.A. (1986)
Analysis of predictive quality of software reliability models. Ph.D. Thesis, The City University, Department of Mathematics.

ABDEL-GHALY, A.A., CHAN, P.Y. and LITTLEWOOD, B. (L985)
Tools for the analysis of the accuracy of software reliability predictions, Software System Design Methods. NATO ASI Series F: Computer and Systems Sciences, Vol.22, (J.K.Skwirzynski, Ed.), Springer Verlag, Heidelberg.

ABDEL-GHALY, A.A., CHAN, P.Y. and LITTLEWOOD, B. (1986)
Evaluation of computing software reliability. IEEE Trans. on Software Eng., Vol.SE-12, No.9, Sept., pp.950-967.

AHLBERG, J., NILSON, E., and WALSH, J. (1967)
The Theory of Splines and their Applications. Academic Press, New York, U.S.A.

AITCHISON, J. and DUNSMORE, I.R. (1975)
Statistical Prediction Analysis. Cambridge University Press, Cambridge, U.K.

ASCHER, H. and FEINGOLD, H. (1984)
Repairable systems reliability. Lecture notes in Statistics, No.7, Marcel Dekker, New York, U.S.A.

BARLOW, R., BARTHOLOMEW, D.J., BREMMER, J.M. and BRUNK, H.D. (1972)
Statistical Inference under Order Restrictions. John Wiley & Sons, Inc., New York, U.S.A.

BARRODALE, I. and YOUNG, A. (1966)
Algorithms for best L_1 and L_∞ approximation on a discrete set. Numerische Math., 8, pp.295-306.

BRENT, R.P. (1973)
Algorithms for minimization without derivatives. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, U.S.A.

CAMPBELL, G., and OTT, K.O. (1979)
Statistical evaluation of major human errors during the development of new technological systems. Nuclear Sci. and Eng., 71, pp.267-279.

CHAN, P.Y. (1986)
Adaptive models. Pergamon Infotech State of the Art Report on Software Reliability. Pergomon Infotech, Ltd., pp.5-18.

COX, A.P. and LEWIS, P.A.W. (1966)
Statistical Analysis of Series of Events. Methuen, London, U.K.

COX, M.G. and HAYES, J.G. (1973)
Curve fitting: A Guide and suite of algorithms for the non-specialist user. NPL Report NAC 26, December.

COX, M.G. (1975)
Numerical methods for the interpolation and approximation of data by spline functions. Ph.D. Thesis, The City University, Department of Mathematics.

COX, M.G. (1980)
The least squares solution of overdetermined linear equations having band or augmented band structure. NPL Report DNACS 26/80, March. Also in J. Inst. Math. Appl., 21, pp.135-143.

COX, M.G. (1982)
Practical spline aproximation. NPL Report DITC 1/82, February.

COX, M.G. and JONES, H.M. (1985)
Shape-preserving spline appoximation in the ℓ_1-norm. NPL Report DITC 68/85, December.

COX, M.G. (1986)
Private communication.

CROW, L.H. (1977)
Confidence interval procedures for reliability growth analysis. U.S. Army Material Syst. Anal. Activity, Aberdeen, MD, Tech. report 197.

CURRY, H.B. and SCHOENBERG, I.J. (1966)
On Pólya frequency functions IV: the fundamental spline functions and their limits. J. Analyse Math., 17, pp.71-107.

DALE, C.J. and HARRIS, L.N. (1981)
Reliability aspects of microprocessor systems. British Aerospace Dynamics Group, ST-25358, July. Available from Dept. of Industry, Report No.T816164.

DAWID, A.P. (1982)
The well calibrated Bayesian, with discussion. J. Amer. Stat. Assoc., 77, pp.605-613.

DAWID, A.P. (1984a)
Statistical theory, the prequential approach. J. Royal Stat. Soc. A., 147, pp.278-292.

DAWID, A.P. (1984b)
Calibration based empirical probability. Research Report 36, Dept. of Stat. Sci., University College, London, U.K.

DAWID, A.P. (1985)
Probability forecasting. Encylopedia of Statistical Sciences, Vol.6, (S.Kotz, N.L.Johnson, and C.B.Read, Eds.) Wiley-Interscience.

DE BOOR, C. (1978)
A Practical Guide to Splines. Springer-Verlag, New York, U.S.A.

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

EPSTEIN, M.P. (1976)
On the influence of parameterization in parametric interpolation. SIAM J. Numer. Anal. Vol.13, No.2, April, pp.261-268.

GILL, P.E. and MURRAY, W. (1972a)
Two methods for the solution of linearly constrained and unconstrained optimization problems. NPL Report NAC 25, November.

GILL, P.E. and MURRAY, W. (1972b)
The implementation of two modified Newton algorithms for unconstrained optimizatin. NPL Report NAC 24, August.

GILL, P.E., MURRAY, W., and PITFIELD, R.A. (1972)
The implementation of two revised quasi-Newton algorithms for unconstrained optimization. NPL Report NAC 11, April.

GILL, P.E. and MURRAY, W. (1974)
Safeguarded steplength algorithms for optimization using descent methods. NPL Report NAC 37.

GILL, P.E., GOLUB, G.H., MURRAY, W. and SAUNDERS, M.A. (1974)
Methods for modifying matrix factorizations. Math. Comput., 28, pp.505-535.

GILL, P.E. and MURRAY, W. (1976)
Minimization subject to bounds on the variables. NPL Report NAC 72, December.

GILL, P.E. and MURRAY, W. (1977)
The computation of Lagrange-multiplier estimates for constrained minimization. NPL Report NAC 77, March.

GILL, P.E., MURRAY, W. and WRIGHT, M.H. (1978)
Introduction to the fundamental theory of optimization. NPL Report DNACS 4/78, October.

GOEL, A.L. and OKUMOTO, K. (1979)
Time-dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. Reliability, Vol.R28, pp.206-211.

HANSON, R.J. (1979)
Constrained least-squares curve fitting to discrete data using B-splines- A users' guide. Sandia Laboratories Report No. SAND78-1291.

JACOBY, S.L.S., KOWALIK, J.S. and PIZZO, J.T. (1972)
Iterative methods for non-linear optimization problems. Prentice-Hall Inc., Englewood Cliffs, New Jersey, U.S.A.

JELINKSI, Z. and MORANDA, P. (1972)
Software reliability research. Statistical Computer Performance Evaluation, Academic Press Inc., London, U.K. pp.465-484.

JOE, H. and REID, N. (1985)
Estimating the number of faults in a system. J. Amer. Stat. Assoc., Vol.80, pp.222-226.

KEILLER, P.A., LITTLEWOOD, B., MILLER, D.R. and SOFER, A. (1983)
On the quality of software reliability prediction. Electronic system Effectiveness and Life Cycle Costing, NATO ASI Seris, Vol.F3, (J.K.Skwirzynski, Ed.) Springer-Verlag, Heidelberg, Germany.

KEILLER, P.A. and LITTLEWOOD, B. (1984)
Adaptive software reliability modelling. Digest FTCS-14, (14th International Conference on Fault-Tolerant Computing), IEEE Computer Soc., Silver Spring, pp.108-113.

KENDALL, M. and STUART, A. (1977)
The Advanced Theory of Statistics. Vol.1. "Distribution Theory", 4th Edition, Charles Griffin & Co. Ltd., London, U.K.

LAPRIE, J.C. (1984)
Dependability evaluation of software systems. IEEE Trans. Software Eng., Vol.SE-10, No.6, November, pp.701-714.

LITTLEWOOD, B. and VERRALL, J.L. (1973)
A Bayesian reliability growth model for computer software. J. Royal Stat. Soc. C., Vol.22, No.3, pp.332-346.

LITTLEWOOD, B. (1979)
How to measure software reliability and how not to. IEEE Trans. Reliability, Vol.R-28, No.2, pp.103-110.

LITTLEWOOD, B. (1980)
Theories of software reliability: How good are they and how can they be improved? IEEE Trans. Software Eng., Vol.SE-6, No.5, pp.489-500.

LITTLEWOOD, B. (1981)
Stochastic reliability growth: A model for fault removal in computer programs and hardware design. IEEE Trans. Reliability, Vol.R-30, No.4, pp.313-320.

LITTLEWOOD, B. and VERRALL, J.L. (1981)
Likelihood function of a debugging model for computer software reliability. IEEE Trans. Reliability, Vol.R-30, No.2, pp.145-148.

LITTLEWOOD, B. and SOFER, A. (1981)
A Bayesian modification to the Jelinski-Moranda software reliability growth model. Unpublished manuscript. The Centre for Software Reliability, The City University, London, U.K.

MARTIN, R.S., PETERS, G. and WILKINSON, J.H. (1965)
Symmetric decomposition of a positive definite matrix. Numerishe Mathematik, 7, pp.362-383.

MILLER, L.H. (1956)
Table of percentage points of Kolmogorov statistics. Amer. Stat. Assoc. Journal, March, pp.111-121.

MILLER, D.R. (1983)
Private communication.

MILLER, D.R. and SOFER, A. (1985)
Completely monotone regression estimates of software failure rates. Proc. of Eighth International Conference on Software Eng., IEEE Comp. Soc., Washington, D.C., pp.343-348.

MILLER, D.R. (1986)
Exponential order statistics models for software reliability growth. IEEE Trans. Software Eng, Vol.SE-12, pp.12-24.

MILLER, D.R. and SOFER,A. (1986a)
A non-parametric approach to software reliability, using complete monotonicity. Pergamon Infotech State of the Art Report on Software Reliability, Pergamon Infotech Ltd., pp.185-195.

MILLER, D.R. and SOFER, A. (1986b)
Least-squares regression under convexity and higher-order difference constraints with application to software reliability. Manuscript.

MOEK, G. (1982)
Maximum likelihood estimation of software reliability model parameter. Report available from National Aerospace Lab., Informatic Div., The Netherlands.

MOEK, G. (1983a)
Software reliability models on trial: selection, improved estimation and practical results. Report No. NLR MP 83059, National Aerospace Lab., The Netherlands. Also in IEEE Trans. Software Eng., 1985.

MOEK, G. (1983b)
Comparison of some software reliability models for simulated and real failure data. Proc. of the Fourth IASTED Symposium on modelling and simulation. Lugano, Switzerland. Also published as NLR MP-83032U, National Aerospace Lab., The Netherlands.

MUSA, J.D. (1975)
A theory of software reliability and its application. IEEE Trans. on Software Eng., Vol.SE-1, No.3, Sept., pp.312-327.

MUSA, J.D. (1980)
Software reliability data. Report available from Analysis Center for Software, Rome Air Development Center, N.Y., U.S.A.

MUSA, J.D. and OKUMOTO, K. (1984)
A logarithmic Poisson execution time model for software reliability measurement. Proc. Seventh International Conference on Software Eng., IEEE Comp. Soc. New York, pp.230-238.

NELDER, J.A. and MEAD, R. (1965)
A simplex method for function minimisation. Computer Journal, Vol.7, pp.308-313.

ROSENBLATT, M. (1952)
Remarks on a multivariate transformation. Ann. Math. Stat., Vol.23, pp.470-472.

STEWART, G.W. (1973)
Introduction to Matrix Computations. Academic Press, Inc., New York, U.S.A.

STOER, J. and BULIRSCH, R. (1980)
Introduction to Numerical Analysis. Springer-Verlag, New York, U.S.A.

WILKINSON, J.H. (1963)
Rounding errors in algebraic processes. Notes on Applied Science No. 32, Her Majesty's Stationary Office, London, U.K.