

# SOFTWARE RELIABILITY AND COST MODELS

---

Christian Krog Madsen  
c973746@student.dtu.dk

August 27, 2000

04447 Software Reliability  
Institute of Mathematical Modelling  
Technical University of Denmark

# Contents

<b>1</b>	<b>Preface</b>	<b>3</b>
<b>2</b>	<b>Introduction to Software Reliability</b>	<b>3</b>
2.1	Faults, Failures and Metrics . . . . .	3
2.2	Tests for Reliability Growth . . . . .	4
2.2.1	Running Arithmetic Mean . . . . .	4
2.2.2	Laplace Test . . . . .	4
2.3	Analysing Predictive Accuracy . . . . .	5
2.3.1	The u-plot . . . . .	5
2.3.2	The y-plot . . . . .	5
2.3.3	Prequential Likelihood Ratio . . . . .	6
2.3.4	Model Noise . . . . .	6
<b>3</b>	<b>Software Reliability Growth Models</b>	<b>6</b>
3.1	Time Domain Models . . . . .	7
3.1.1	Jelinski-Moranda (JM) . . . . .	7
3.1.2	Littlewood-Verrall (LV) . . . . .	8
3.1.3	Musa-Okumoto (MO) . . . . .	8
3.2	Interval Domain Models . . . . .	9
3.2.1	Yamada S-shape (YS) . . . . .	9
3.2.2	Generalised Poisson (GP) . . . . .	10
3.2.3	Goel-Okumoto (GO) . . . . .	10
3.3	Supermodels . . . . .	11
3.3.1	SUPULC . . . . .	11
3.3.2	SUPRLC . . . . .	11
3.3.3	SUPDLC . . . . .	11
<b>4</b>	<b>Application of Models</b>	<b>11</b>
4.1	Data Set 136 . . . . .	11
4.1.1	Time Between Failures Approach . . . . .	11
4.1.2	Failure Count Approach . . . . .	19
4.2	Data Set 129 . . . . .	22
4.2.1	Time Between Failures Approach . . . . .	22
4.2.2	Failure Count Approach . . . . .	30
4.3	Data Set 278 . . . . .	33
4.3.1	Time Between Failures Approach . . . . .	33
4.3.2	Failure Count Approach . . . . .	41
4.4	Supermodels Versus Simple Models . . . . .	44
<b>5</b>	<b>Software Cost Estimation</b>	<b>47</b>
5.1	Software Cost Model . . . . .	47
5.2	Example Application . . . . .	48

<b>6 Conclusions</b>	<b>50</b>
<b>References</b>	<b>50</b>

# 1 Preface

This report gives an overview of Software Reliability metrics, models and analysis tools, and also presents some models for software development cost estimation and optimisation. The reliability models are applied to failure data sets using the CASRE tool [1].

Chapter 2 introduces the problem of reliability measurement and describes the metrics and analysis techniques used. Chapter 3 summarises the assumptions and formulae of the software reliability growth models used in the report. In chapter 4 the models are then applied to actual failure data sets, with graphical presentation of the results. The problem of cost estimation is the topic of chapter 5. Finally, in chapter 6 the results are discussed.

## 2 Introduction to Software Reliability

### 2.1 Faults, Failures and Metrics

In the following a distinction is made between the terms fault and failure. A fault is a deficiency in the source code that may or may not manifest itself as a failure. A failure is when the program does not behave as required, i.e. gives a wrong result, crashes, or exhibits poor response times.

Software reliability requires statistical methods because of two intrinsic uncertainties, namely the use of the software and the nature of fault fixes. Because we do not know how the software will be used in the field, we can not predict exactly when a fault will manifest itself. When we apply a fix to the software we can rarely be certain that the fix remedies the original fault, or indeed that it does not introduce a new fault – this is the second uncertainty.

There are two standard methods of measuring failures. In the Time Between Failures approach, the time between successive failures is recorded. The other way is to record the number of failures per unit of time (Failure Count).

There are several important software reliability metrics:

**ROCOF** ( $\lambda(t)$ ) Rate of occurrence of failures (at time  $t$ ).

**MTTF** Mean time to next failure.

**Reliability** ( $R(t)$ ) Probability that the software will run for a time  $t$  without failure.

To make predictions about the future reliability of a piece of software based on observed failures, a model needs to be formulated. In the literature there are many different software reliability models. Studies have shown that none of the models can be trusted to always give accurate results, however,

some models are more likely to fit a given data set than others. Furthermore, there is no way to select a suitable model *a priori*.

In this report models are ranked by their accuracy according to the method suggested by Nikora and Lyu [3]. This method first applies a goodness of fit test. If more than one model fits at a specified level, then models are ranked by their prequential likelihood, bias and trend in that order.

## 2.2 Tests for Reliability Growth

Before applying a software reliability growth model it is wise to test whether there is any sign of growth in the data. The two methods discussed here are the running arithmetic mean and the Laplace test.

### 2.2.1 Running Arithmetic Mean

The running arithmetic mean is an informal test for reliability growth. To perform the test, calculate the series  $\{\tau_j\}$ , which for time between failures data is given as

$$\tau_j = \frac{1}{j} \sum_{i=1}^j t_i, \quad i = 1, \dots, n$$

If the series is increasing it indicates reliability growth.

For failure count data the series is given as

$$\tau_j = \frac{1}{j} \sum_{i=1}^j n_i$$

where  $n_j$  is the number of failures in the  $j$ th interval. In this case the series will decrease if there is reliability growth. Note that the time intervals must be of equal length.

### 2.2.2 Laplace Test

The Laplace test is a formal statistical test using the Laplace trend factor. For time between failures data the factor is calculated as

$$u(k) = \frac{\frac{1}{k-1} \sum_{i=1}^k x_i - \frac{x_k}{2}}{x_k \sqrt{\frac{1}{12(k-1)}}}$$

while for failure count data it is given as

$$u(k) = \frac{\sum_{i=1}^k (i-1)n_i - \frac{k-1}{2} \sum_{i=1}^k n_i}{\sqrt{\frac{k^2-1}{12} \sum_{i=1}^k n_i}}$$

The Laplace factor is approximately normally distributed with zero mean and unit variance. The hypothesis of no trend in the data can be rejected in favor of reliability growth at the  $\alpha\%$  level if the Laplace factor is less than or equal to value at which the cumulative normal distribution function is  $\alpha/100$ .

Note that for failure count data, the intervals must be of equal length.

## 2.3 Analysing Predictive Accuracy

To remedy the problem of selecting models, several methods for analysing model accuracy have been developed.

### 2.3.1 The u-plot

The first analysis tool is the *u-plot* which can reveal consistent bias in the predictions.

Given a predicted cumulative distribution function  $\hat{F}_i(t)$  for the time to next failure and a realisation  $t_i$  of the true distribution  $F_i(t)$  the numbers

$$u_i = \hat{F}_i(t_i), \quad i = j + 1, \dots, n$$

would be uniformly distributed on  $(0, 1)$  if the predicted function were identical to the true function. The closer the  $u_i$ s are to being uniformly distributed, the more accurate are the predictions.

The u-plot is drawn by marking the  $u_i$ s on the first axis and drawing a step function that increases by  $1/(n + 1)$  for every  $u_i$ . The maximum vertical distance between the step function and the unit slope line, called the Kolmogorov-Smirnov distance, is used to determine whether the deviation is statistically significant. If the step function is above the unit slope line it indicates that the model is consistently optimistic in its predictions. Conversely, if it is below the line it indicates consistent pessimism.

In drawing the u-plot temporal information may be lost because the sequence of  $u_i$ s is permuted. This means, for example, that a model which is initially optimistic but later becomes pessimistic may not exhibit bias in the u-plot.

### 2.3.2 The y-plot

The *y-plot* is similar to the u-plot but preserves the temporal information. Whereas the u-plot shows consistent bias, the y-plot shows the bias trend in predictions.

The y-plot is constructed by transforming the  $u_i$ s in two steps

$$x_i = -\ln(1 - u_i)$$

$$y_i = \frac{\sum_{j=1}^i x_j}{\sum_{j=1}^n x_j}$$

The  $y_i$  are plotted using the same procedure as for the u-plot. The Kolmogorov-Smirnov distance is used to test for a bias trend.

### 2.3.3 Prequential Likelihood Ratio

The *prequential likelihood ratio* (PLR) is a method of comparing two prediction systems. It takes account of both bias and noise. The prequential likelihood is defined as

$$\text{PL}_n = \prod_{i=j+1}^{j+n} \hat{f}_i(t_i)$$

where  $\hat{f}_i(t)$  is the prediction for the  $i$ th failure time, and  $t_i$  is the observed value.

Given two prediction systems, A and B, the prequential likelihood ratio is

$$\text{PLR}_n = \frac{\text{PL}_n^A}{\text{PL}_n^B} = \frac{\prod_{i=j+1}^{j+n} \hat{f}_i^A(t_i)}{\prod_{i=j+1}^{j+n} \hat{f}_i^B(t_i)}$$

If system A provides more accurate predictions than does B, then the PLR will be an increasing function of  $n$ . A PLR above one means system A has given more accurate predictions *globally*. A positive slope of the PLR curve indicates that system A has given more accurate predictions *locally*.

### 2.3.4 Model Noise

A measure of model noise, i.e. unwarranted fluctuations in the predictions, is defined as

$$\text{Model noise} = \sum_i \left| \frac{\hat{t}_i - \hat{t}_{i-1}}{\hat{t}_{i-1}} \right|$$

where  $\hat{t}_i$  is the prediction for the time between failures ( $i - 1$ ) and  $i$ . A smooth model will have a low value of model noise in comparison with a noisy model.

## 3 Software Reliability Growth Models

Software reliability growth models operate on either time between failures data or failure count data. The first class predict the distribution of the next

time to failure, while the second class predict the distribution of the number of failures to be observed by a given time.

Another way to classify models is as either parametric models or non-homogeneous Poisson process models. For parametric models the ROCOF is a discontinuous function. Whenever a failure is discovered the ROCOF changes instantaneously. A non-homogeneous Poisson process model has a continuous ROCOF function.

All models assume the software is tested under conditions representative of field use.

### 3.1 Time Domain Models

In this section  $t$ s represent time between consecutive failures.

#### 3.1.1 Jelinski-Moranda (JM)

The assumptions of the parametric JM model are

- Failures occur independently and the time between successive failures is exponentially distributed.
- Faults are corrected instantaneously upon detection.
- Fault correction is perfect, i.e. does not introduce new faults.
- The failure rate is proportional to the number of remaining faults and is constant between failures.

The JM model treats only the first source of uncertainty, and tends to be overly optimistic because the assumption that all faults contribute evenly to the failure means it ignores the law of diminishing returns. Consequently, the JM model is rarely the best choice.

The model has two parameters,  $N$ , the initial number of faults and  $\phi$ , the contribution of each fault to the failure rate.

The probability density function is

$$f(t_i) = \lambda_i e^{-\lambda_i t_i}$$

where  $\lambda_i$  is the failure rate

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

The reliability function is

$$R_i(t) = e^{-(N-i+1)\phi t}$$

and the MTTF is

$$\text{MTTF}_i = \frac{1}{(N - i + 1)\phi}$$

### 3.1.2 Littlewood-Verrall (LV)

The assumption is

- Failures occur independently with exponentially distributed inter-failure times.

The LV model considers both types of uncertainty and can handle software reliability growth as well as decay. The model cannot estimate the number of remaining faults. The model gives very smooth predictions, that tend to be pessimistic.

The probability density function is

$$f_i(t) = \frac{\alpha}{\Psi(i, \beta)} \left( \frac{\Psi(i, \beta)}{\Psi(i, \beta) + t} \right)^{\alpha+1}$$

Different functions  $\Psi(i, \beta)$  may be used. The most common are

$$\Psi(i, \beta) = \beta_1 + \beta_2 i$$

$$\Psi(i, \beta) = \beta_1 + \beta_2 i^2$$

Models using these functions are referred to as linear LV and quadratic LV, respectively.

The reliability, failure rate and MTTF are

$$R_i(t) = \left( \frac{\Psi(i, \beta)}{\Psi(i, \beta) + t} \right)^\alpha$$

$$\lambda_i(t) = \frac{\alpha}{t + \Psi(i, \beta)}$$

$$\text{MTTF}_i = \frac{\Psi(i, \beta)}{\alpha - 1}$$

The parameters to be estimated are  $\alpha$ ,  $\beta_1$  and  $\beta_2$ . LV is a parametric model.

### 3.1.3 Musa-Okumoto (MO)

The assumptions of this model are

- Failures occur independently.
- The expected number of encountered failures is a logarithmic function of time.

- There are infinitely many faults.

The failure rate is

$$\lambda(t) = \frac{\varepsilon}{\beta + t}$$

with mean

$$m(t) = \varepsilon \ln \left( \frac{\beta + t}{\beta} \right)$$

Letting  $x_{i-1}$  be the total elapsed time until the  $(i-1)$ th failure, the reliability is given as

$$R_i(t | x_{i-1}) = e^{-\varepsilon \ln \left( 1 + \frac{t}{\beta + x_{i-1}} \right)}$$

The parameters of the MO model are  $\beta$  and  $\varepsilon$ . MO is a NHPP model.

### 3.2 Interval Domain Models

The  $ts$  in this section represent time from the start of testing.

#### 3.2.1 Yamada S-shape (YS)

The assumptions of YS are

- Failures occur independently.
- The probability of failure detection is proportional to the number of remaining faults.
- The time between failures  $(i-1)$  and  $i$  depends on the time to failure  $(i-1)$ .
- Immediate and perfect debugging.
- A finite expected number of faults.

The YS model considers the start of testing as a learning process, where the skill of testers increases.

The failure rate is

$$\lambda(t) = ab^2te^{-bt}$$

with mean

$$m(t) = a \left[ 1 - (1 + bt)e^{-bt} \right]$$

The reliability is

$$R_i(t | x_{i-1}) = e^{-a \left[ (1 + bx_{i-1})e^{-bx_{i-1}} - (1 + b(x_{i-1} + t))e^{-b(x_{i-1} + t)} \right]}$$

The parameters are  $a$  and  $b$ . The YS model is a NHPP model.

### 3.2.2 Generalised Poisson (GP)

The assumptions are

- The expected number of failures occurring in any time interval is proportional to the fault content at the time of testing, and to some function of the amount of time spent in failure testing.
- All failures are equally likely to occur and are independent of each other.
- Each failure is of the same order of severity as any other failure.
- Faults are corrected at the ends of the testing intervals, without introducing new faults.

The mean value function of this model is

$$m(t_i) = \phi(N - M_{i-1})t_i^\alpha$$

where  $M_{i-1}$  is the total number of faults removed up to the end of the  $(i-1)$ st debugging interval,  $\phi$  is a proportionality constant, and  $\alpha$  is a constant used to rescale  $t_i$ . In this paper only  $\alpha = 1$  is considered.

### 3.2.3 Goel-Okumoto (GO)

Model assumptions are

- At time 0, no failures have been discovered.
- The occurrence of a failure is independent of previous failures.
- At most one failure can occur in the time interval  $(t, t + dt)$ .
- Debugging is perfect and carried out immediately upon fault detection.

The GO model is an NHPP equivalent of the JM model.

The failure rate is

$$\lambda(t) = \mu\phi e^{-\phi t}$$

with the mean

$$m(t) = \mu \left(1 - e^{-\phi t}\right)$$

With  $x_{i-1}$  being the total elapsed time until the  $(i-1)$ th failure, the reliability is

$$R_i(t | x_{i-1}) = e^{-\mu e^{-\phi x_{i-1}} (1 - e^{-\phi t})}$$

The model parameters are  $\mu$ , indicating the total number of faults, and  $\phi$ .

### 3.3 Supermodels

#### 3.3.1 SUPULC

This model is formed by weighting the component models' predictions with static weights. The two component model chosen are the MO and the quadratic LV. The models are given different weights, making the supermodel an Unequally weighted Linear Combination (ULC). Informally, the model may be described as

$$\text{SUPULC} = \frac{2}{5}\text{MO} + \frac{3}{5}\text{LV}$$

#### 3.3.2 SUPRLC

This is a Result based Linear Combination (RLC) supermodel. At each calculation step the performance of the component models is evaluated and the models are ranked. The highest ranking model is given a weight of 6/10, the second highest is given a weight of 3/10 and the lowest ranked model is given a weight of 1/10. The evaluation of performance is done as an average over the previous five steps.

The three component models are JM, MO and quadratic LV.

#### 3.3.3 SUPDLC

The Dynamically weighted Linear Combination (DLC) supermodel assigns weights according to changes in prequential likelihood. The prequential likelihood is taken over the previous five steps, so the weight for prediction system  $r$ , when there are  $m$  models in all, is

$$w_n^r = \frac{\text{PL}_{5,n-1}^r}{\sum_{k=1}^m \text{PL}_{5,n-1}^k}$$

The component models used are JM, MO and quadratic LV.

## 4 Application of Models

### 4.1 Data Set 136

#### 4.1.1 Time Between Failures Approach

The running arithmetic mean in figure 1 clearly displays reliability growth in the data set, and this picture is repeated in the Laplace test in figure 2.

The time between failures plot in figure 3 shows that JM appears to be optimistic and LV pessimistic. It is also apparent that JM estimates that there are very few errors remaining in the software.

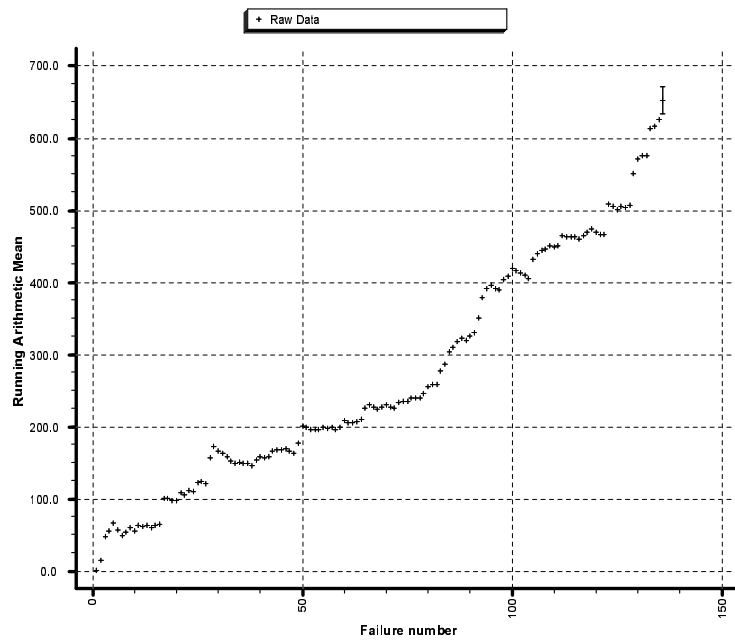


Figure 1: Running arithmetic mean for data set 136.

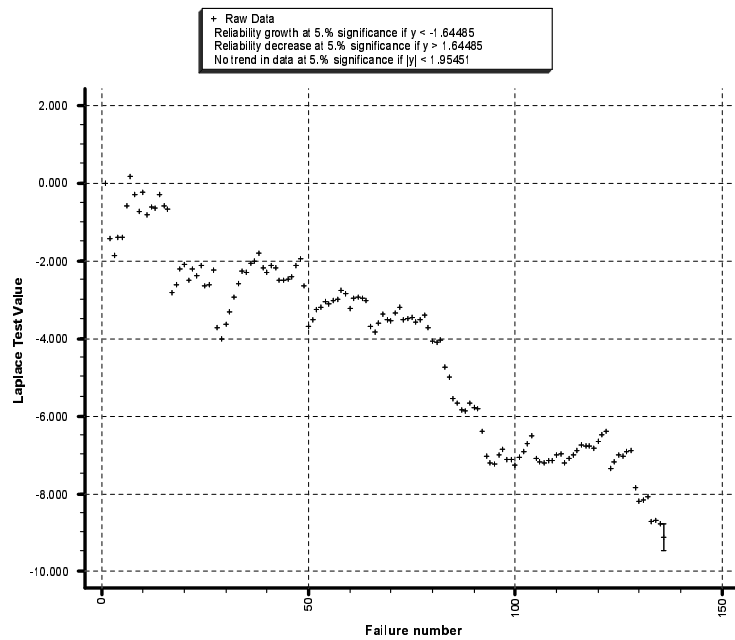


Figure 2: Laplace test for data set 136.

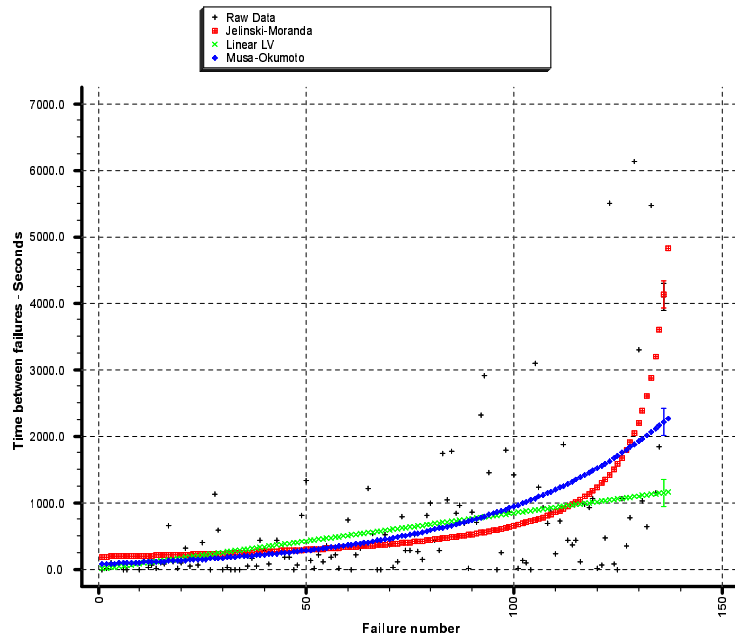


Figure 3: Time between failures for data set 136.

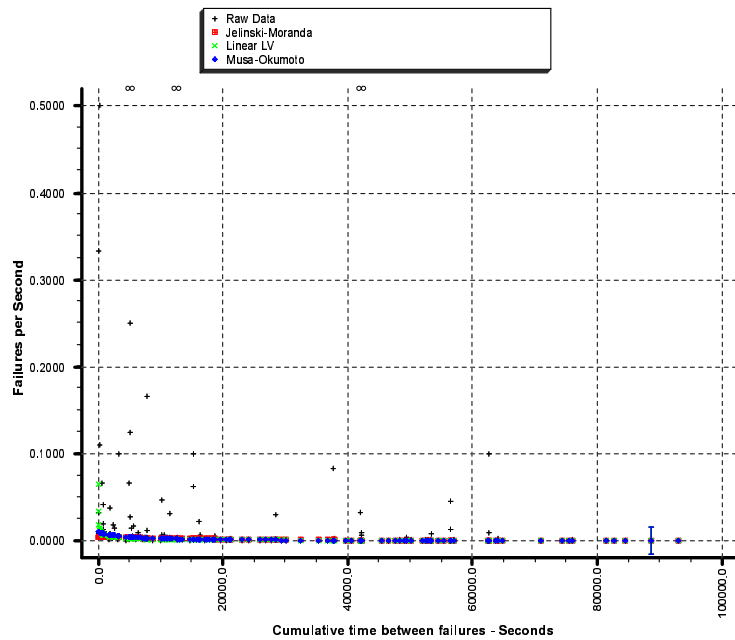


Figure 4: Failure intensity for data set 136.

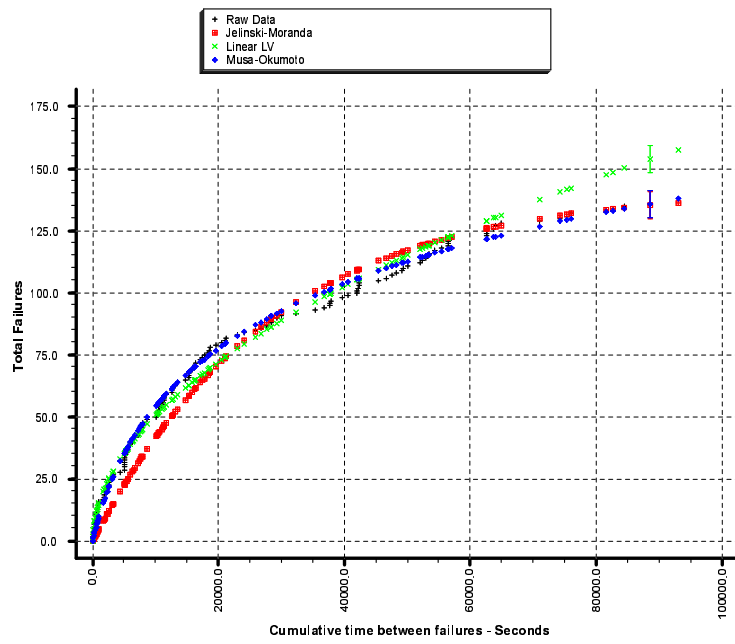


Figure 5: Cumulative failures for data set 136.

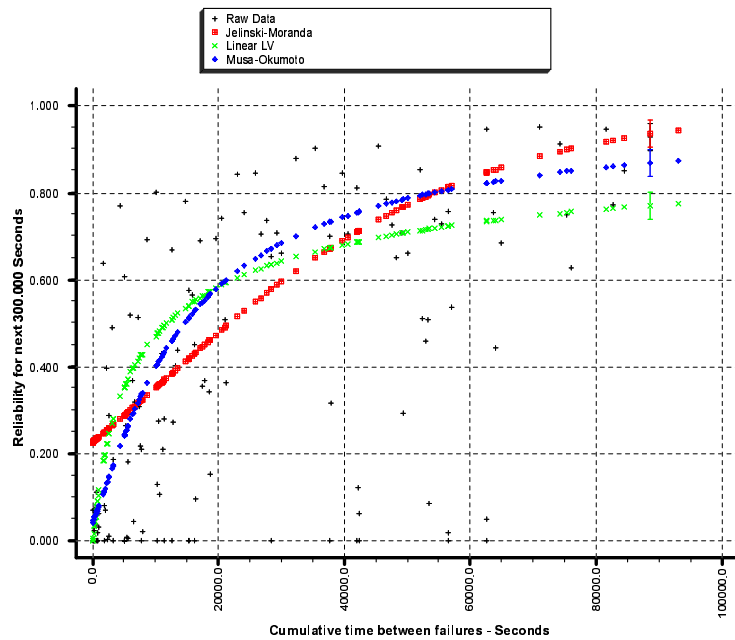


Figure 6: Reliability for five minutes for data set 136.

All models fit the cumulative failures graph in figure 5, although LV overestimates the number of expected failures towards the end of the interval.

The reliability graph in figure 6 again indicates that JM is optimistic and LV pessimistic.

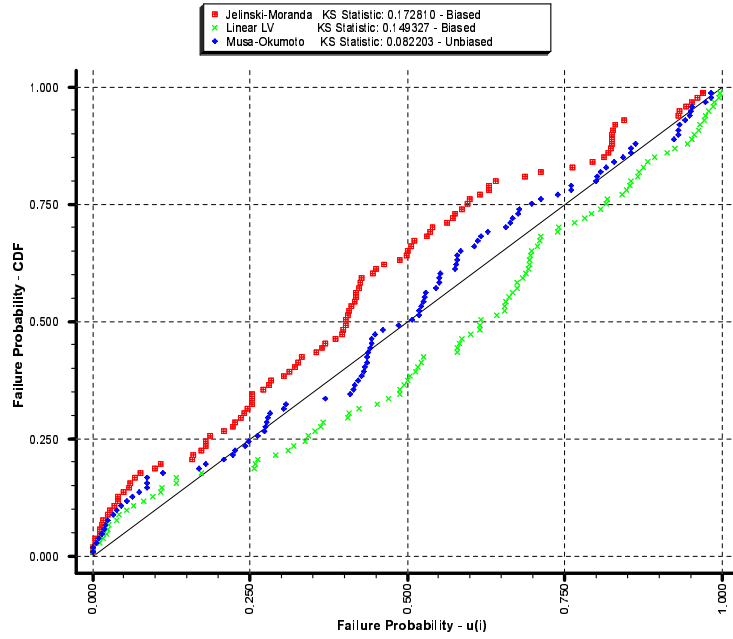


Figure 7: u-plot for data set 136.

The u-plot (figure 7) proves that JM is in fact highly optimistic and that LV is pessimistic. MO is the best performing model as far as bias is concerned.

None of the models show any bias trend in the y-plot (figure 8).

<i>Model</i>	<i>Noise</i>
JM	8.91
LV	2.89
MO	4.10

Table 1: Noise values for data set 136.

Figures 10 and 11 show that MO is the best model of the three for this particular data set, although between failures 128 and 136 it has not performed better than the JM model. MO is more noisy in its predictions than LV but far better than JM. All models pass the goodness of fit test at a 5% level.

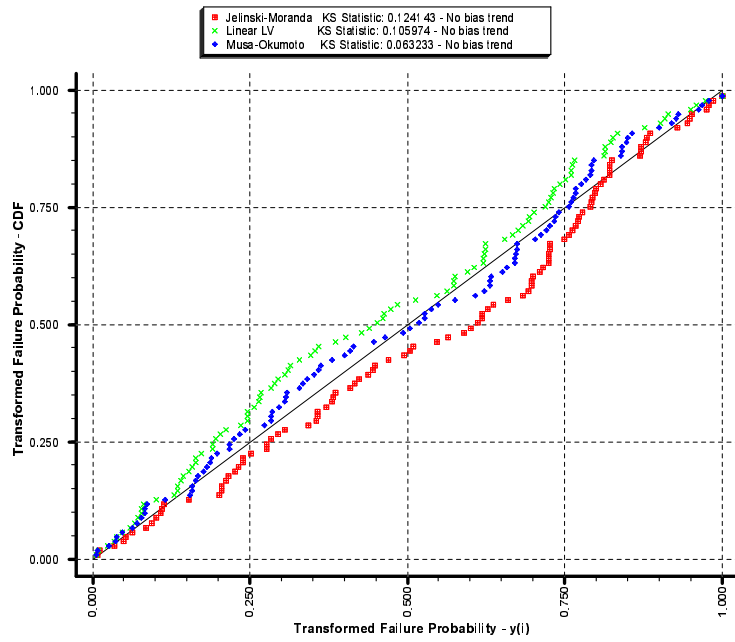


Figure 8: y-plot for data set 136.

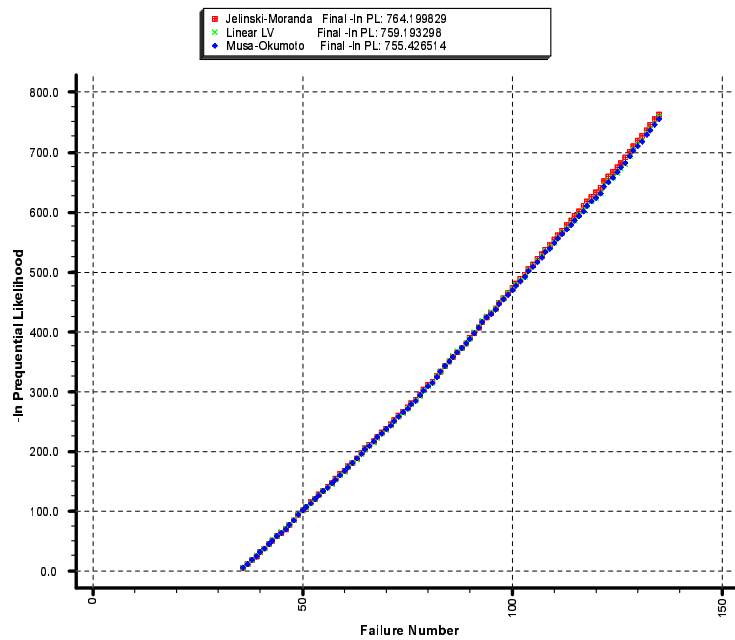


Figure 9: Prequential likelihood for data set 136.

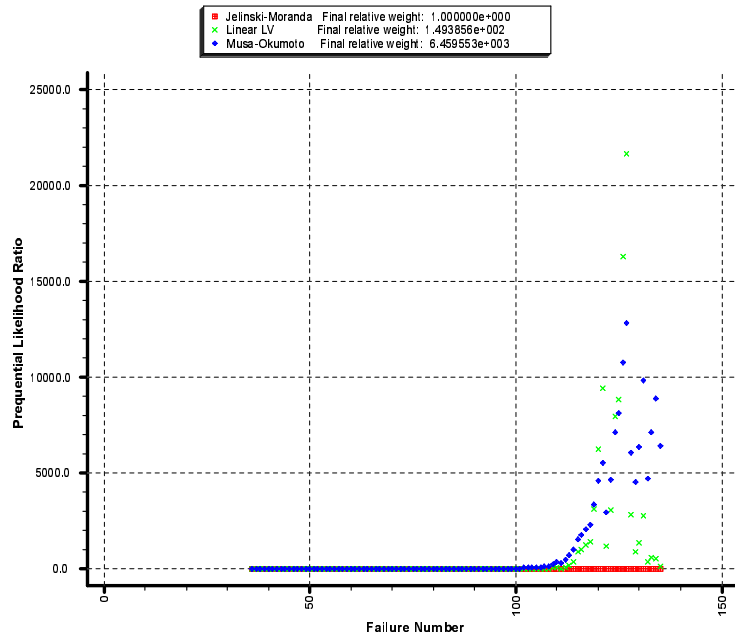


Figure 10: Prequential likelihood ratio for data set 136.

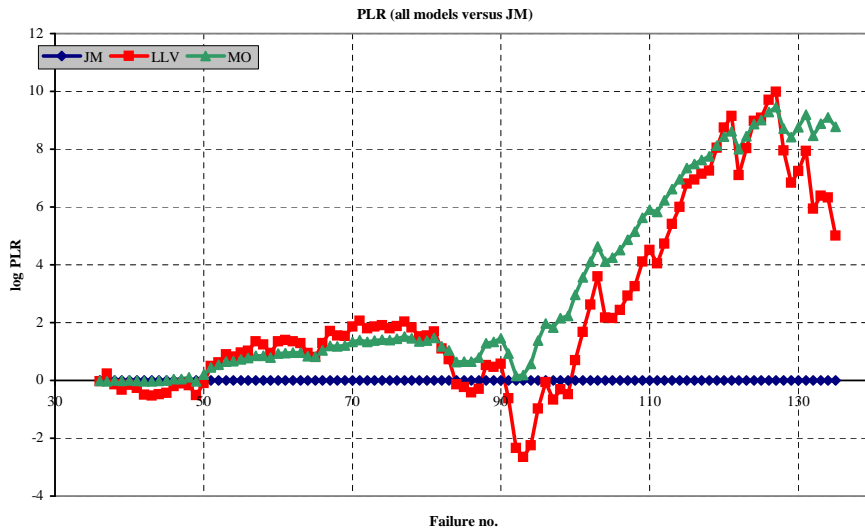


Figure 11: Logarithmic prequential likelihood ratio for data set 136.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
MO	1	0.870
LV	2	0.771
JM	3	0.936

Table 2: Model ranking for data set 136.

### 4.1.2 Failure Count Approach

The data set 136 is converted to the failure count format with 74 intervals of 1200 seconds each.

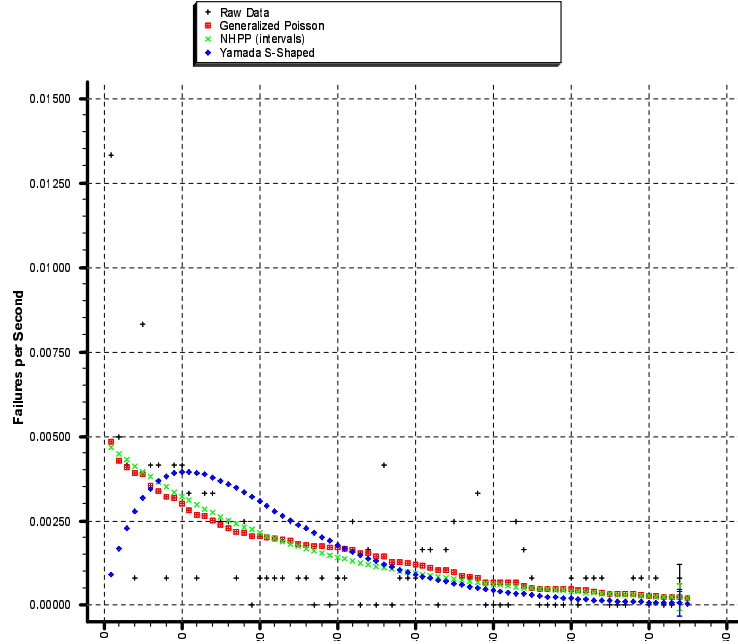


Figure 12: Failure intensity for data set 136.

The failure intensity graph (figure 12) shows the characteristic shape of the ROCOF for the Yamada S-shaped model. The GP and GO (NHPP) produce curves that are similar.

The cumulative failures graph (figure 13) does not fit the learning process assumption of the S-shaped model. Indeed there seems to be an inverse S-shape. Clearly the GP and GO models provide a better fit.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
GO	1	0.932
GP	2	0.931
YS	3	0.980

Table 3: Model ranking for data set 136.

The prequential likelihood ratio in figure 15 shows that the YS model is not a good choice for this data set. The GO and GP models are both considerably more accurate than YS. GO provides slightly more accurate predictions than GP.

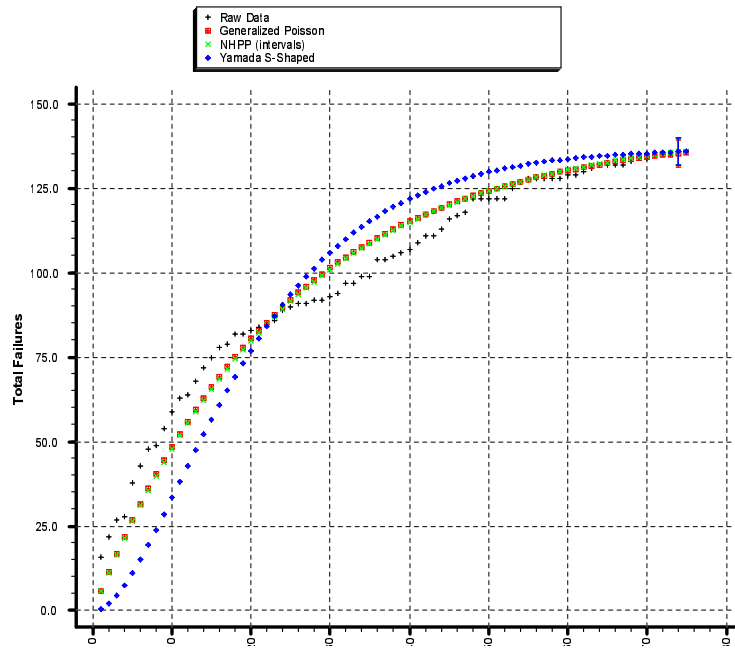


Figure 13: Cumulative failures for data set 136.

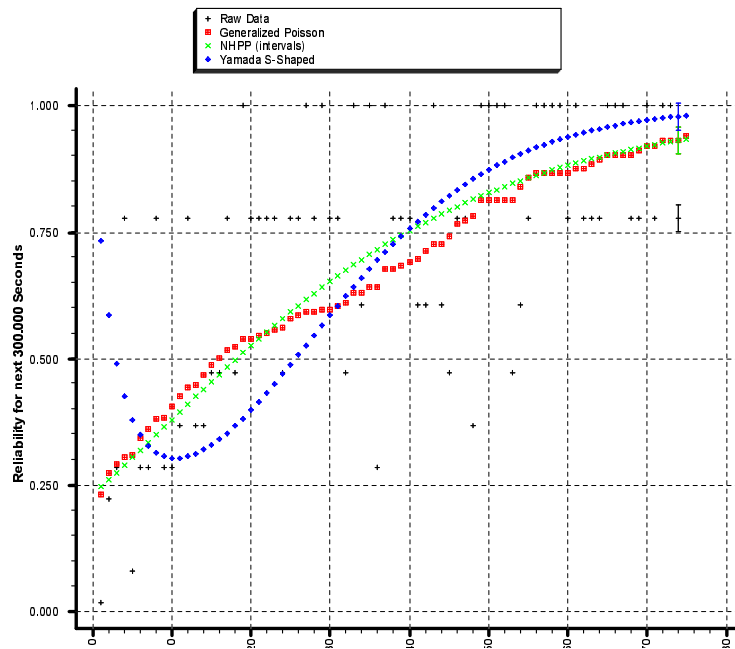


Figure 14: Reliability for data set 136.

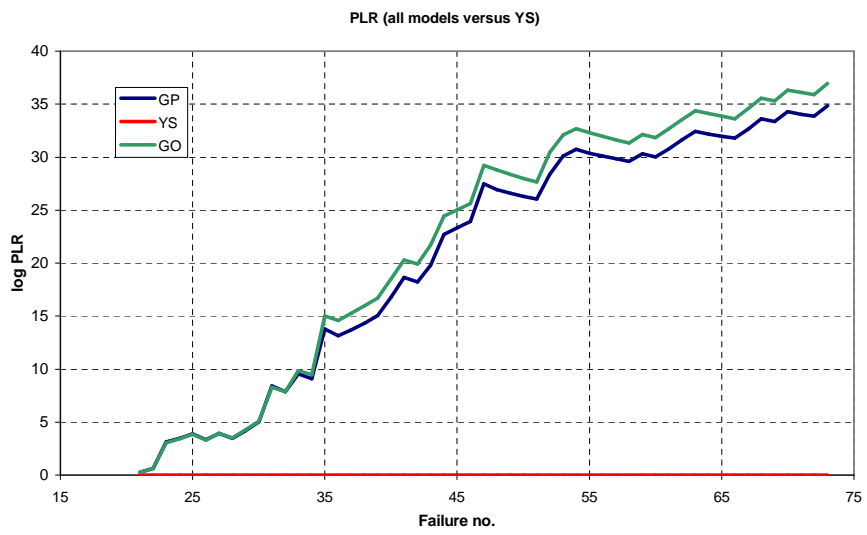


Figure 15: Logarithmic prequential likelihood ratio for data set 136.

## 4.2 Data Set 129

### 4.2.1 Time Between Failures Approach

This data set proved particularly troublesome since the linear LV model results were invalid using both least squares and maximum likelihood estimation. Because of this, the quadratic LV model is applied instead of linear LV.

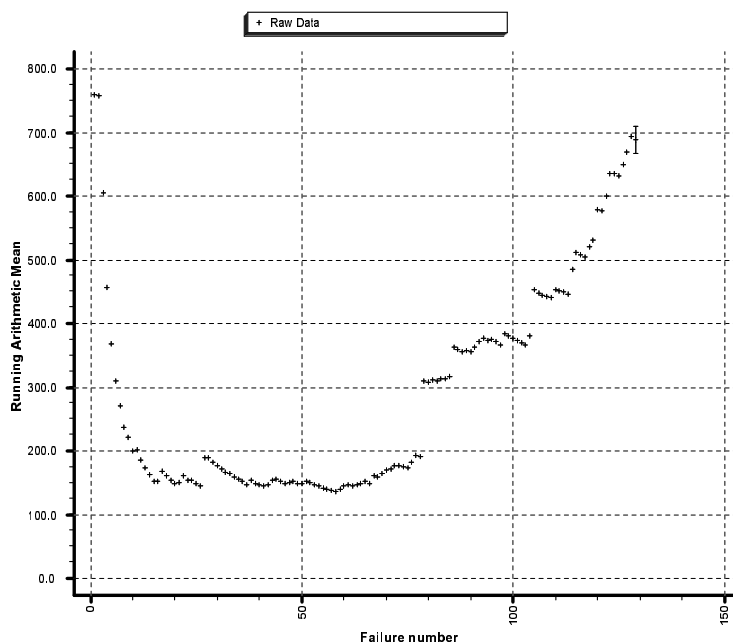


Figure 16: Running arithmetic mean for data set 129.

The running arithmetic mean (figure 16) and the Laplace test (figure 17) show an initial decrease in reliability followed by a period of constant reliability until around failure 80 where reliability begins to grow. In the final growth period the Laplace test indicates three intervals of local reliability decay. Looking at the raw data there are three very large time between failure observations and these probably explain the shape of the Laplace test plot.

Based on these trend tests, the reliability growth models should be applied only to the data after failure 80. However, the models fail to yield valid results when applied to this interval and therefore the entire data set is used.

The LV model grossly overestimates the time between failures (figure 18). For a good model about half of the observed values should be above the curve and about half should be below. When compared with the u-plot (figure 22), where LV has the lowest Kolmogorov-Smirnov distance, it is surprising that

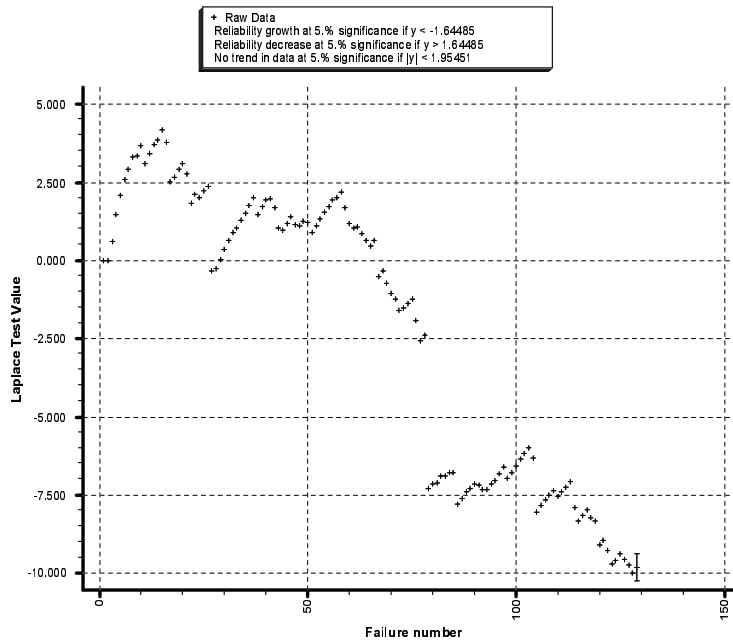


Figure 17: Laplace test for data set 129.

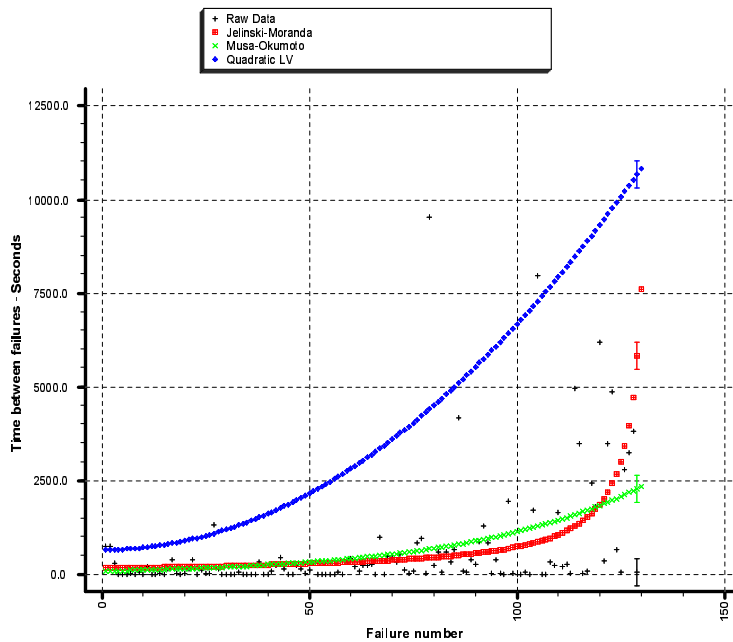


Figure 18: Time between failures for data set 129.

the time between failure estimates are so high. A possible explanation is that the estimated parameters change rapidly in the last few steps, and because CASRE uses only the final set of estimated parameters to draw the curve it does not fit the early data well.

The two other models fit the data fairly well. The JM model expects the software to be almost fault-free at the end of the data set.

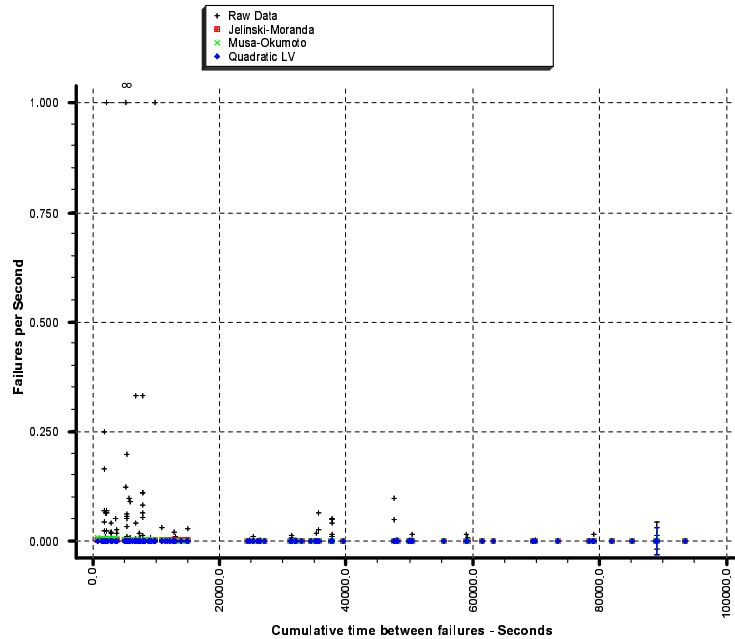


Figure 19: Failure intensity for data set 129.

The cumulative failures plot in figure 20 reveals three long jumps, corresponding to the three large values in the raw data, that also showed up in the Laplace test. MO and JM closely fit the data from about 25,000 seconds into the testing. Again the LV model stands out with its poor performance. It is contradictory that LV expects long times between failures, but in this plot predicts a very high expected number of failures.

The reliability graph is shown in figure 21. Compared with MO, JM is optimistic and LV is pessimistic.

As mentioned, the u-plot (figure 22) for LV does not correlate with the time between failures graph or the cumulative number of failures graph. All three models are biased according to the Kolmogorov-Smirnov test. JM exhibits a particularly high degree of pessimism.

The Kolmogorov-Smirnov test in the y-plot (figure 23) shows that JM has a bias trend. MO and LV show no trend.

The enigmatic behaviour of the quadratic LV model continues in the

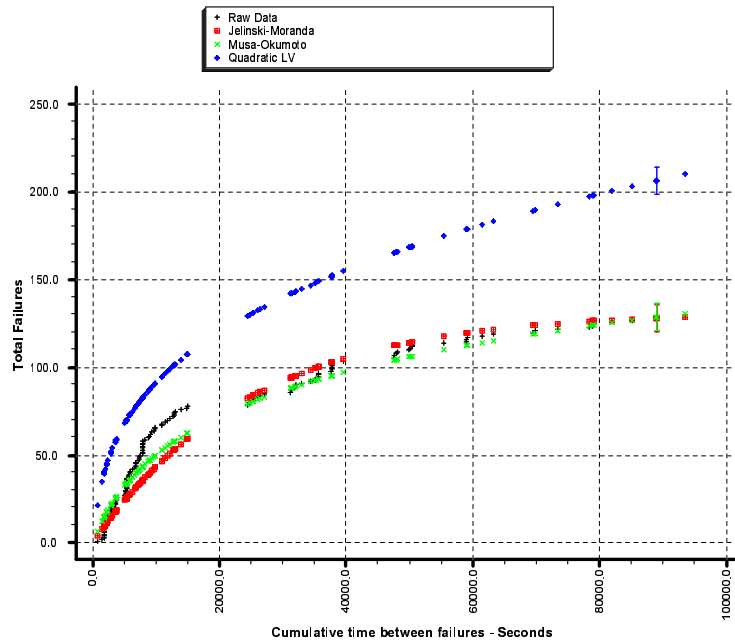


Figure 20: Cumulative failures for data set 129.

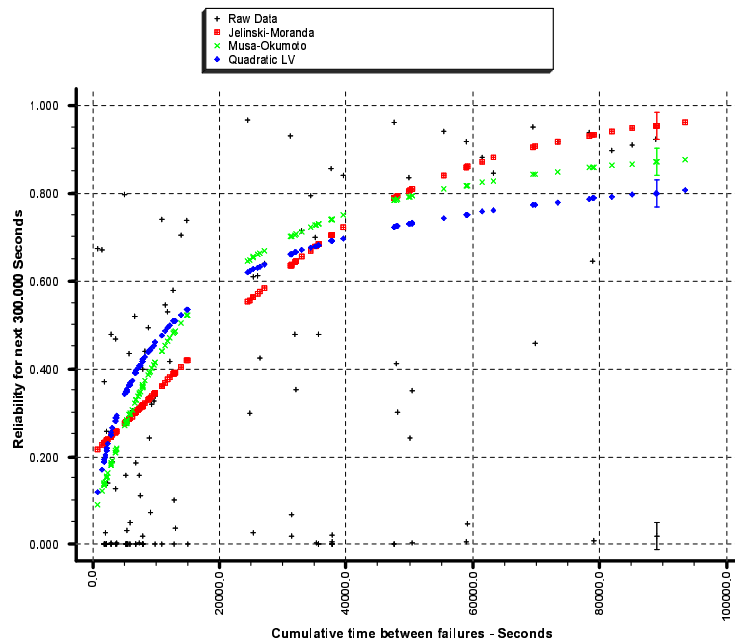


Figure 21: Reliability for five minutes for data set 129.

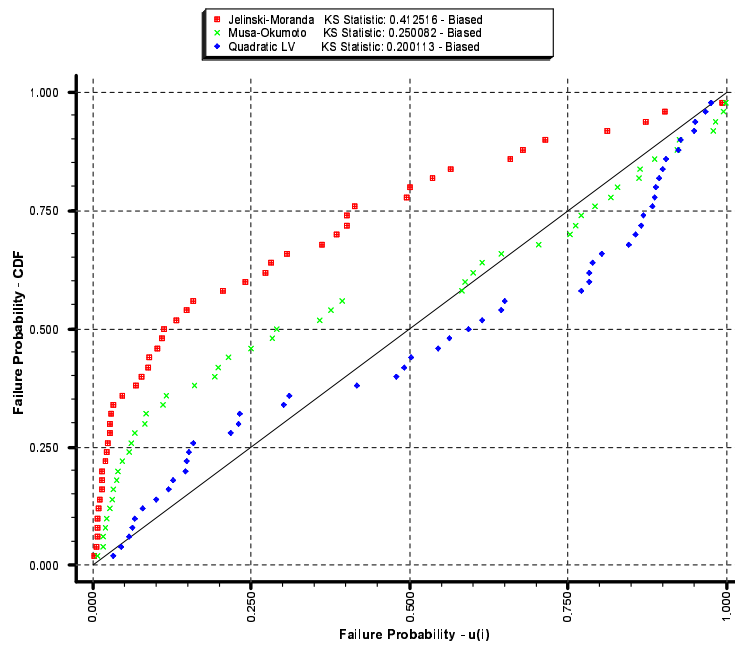


Figure 22: u-plot for data set 129.

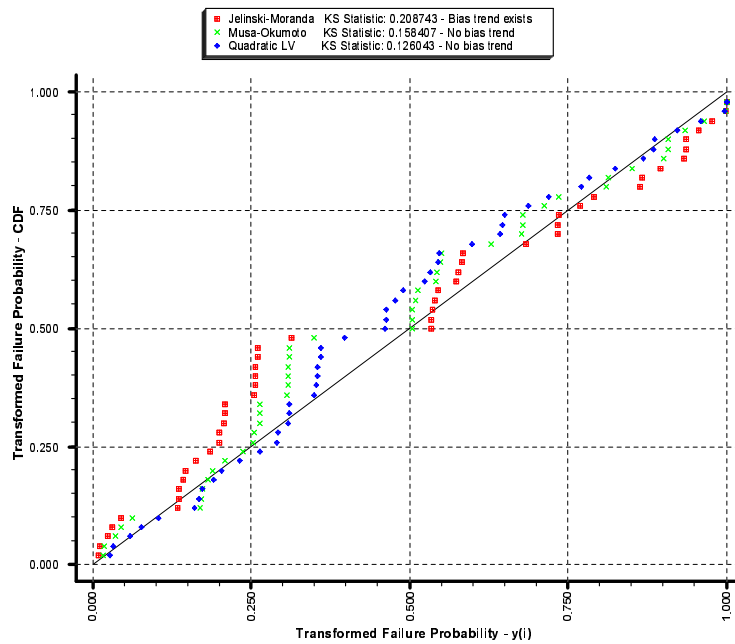


Figure 23: y-plot for data set 129.

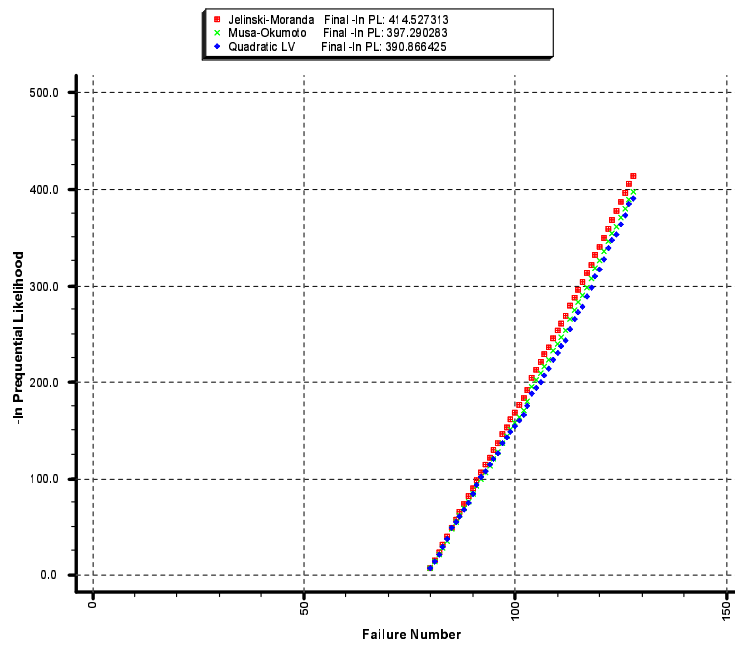


Figure 24: Prequential likelihood for data set 129.

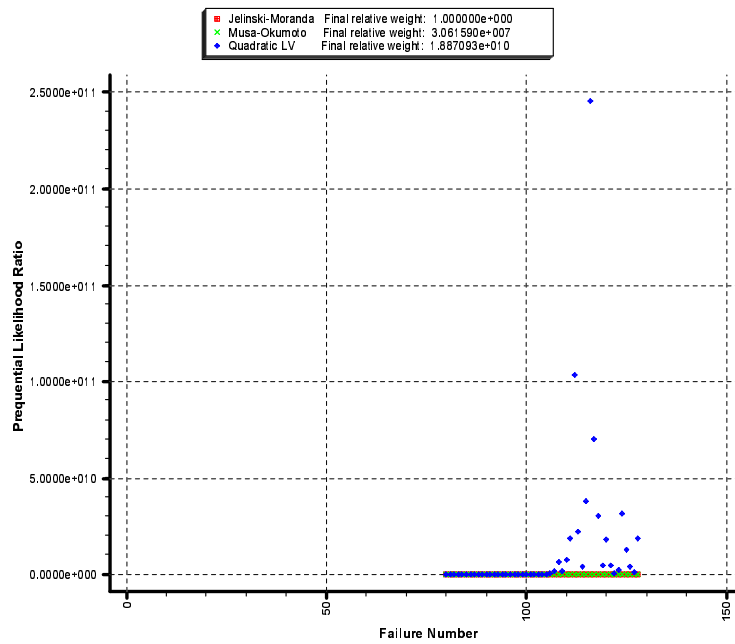


Figure 25: Prequential likelihood ratio for data set 129.

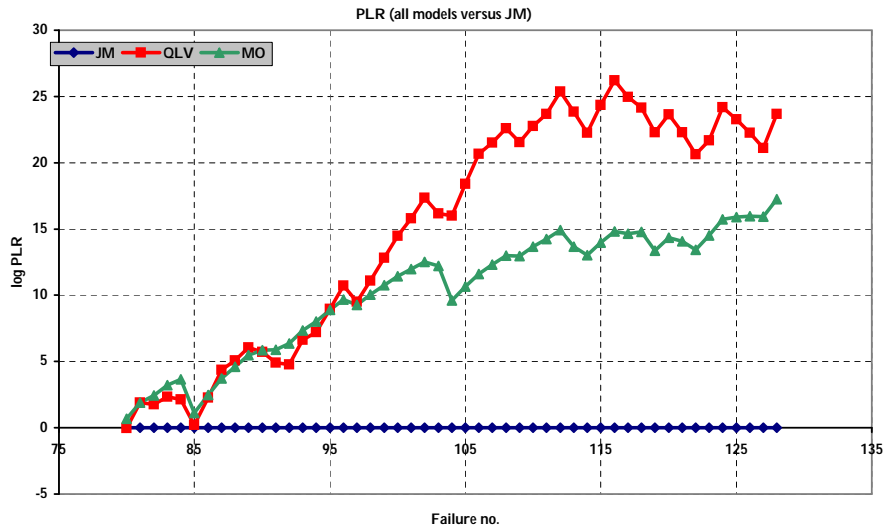


Figure 26: Logarithmic prequential likelihood ratio for data set 129.

<i>Model</i>	<i>Noise</i>
JM	7.12
LV	2.08
MO	2.02

Table 4: Noise values for data set 129.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
LV	1	0.802
MO	2	0.894
JM	3	0.956

Table 5: Model ranking for data set 129.

prequential likelihood ratio graphs (figures 25 and 26) where it is clearly the best model. This supports the conclusion from the u-plot.

It is apparent that LV is less affected than MO and JM by outliers in the data and this explains its good accuracy. Despite the strange time between failures graph and cumulative failures graphs the u-plot and PLR establish quadratic LV as the superior model for this data set. LV is also the only model to pass the goodness of fit test at a 5% level.

## 4.2.2 Failure Count Approach

The data set 129 is converted to failure count data with 75 intervals of 1200 seconds each.

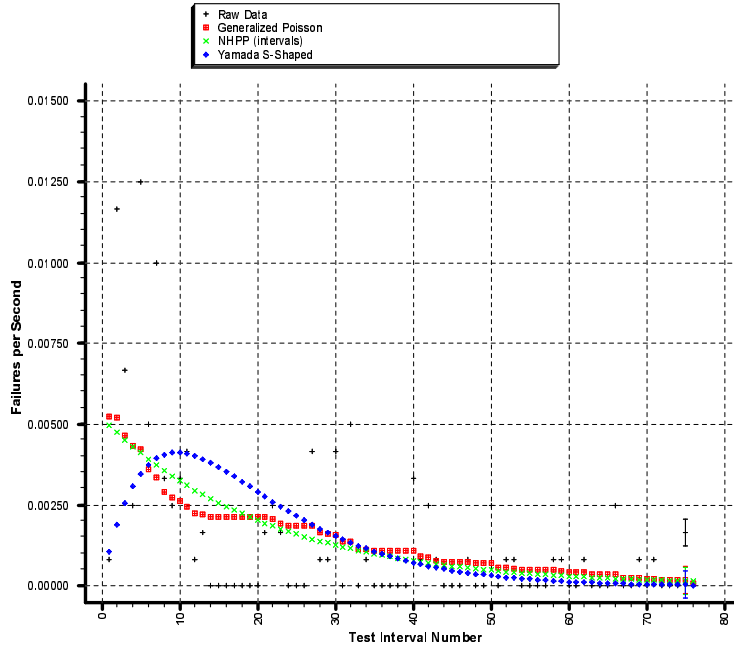


Figure 27: Failure intensity for data set 129.

Figure 27 closely resembles the corresponding graph for data set 136 (figure 12). The difference between GP and GO is somewhat greater in this plot.

As was the case for data set 136, the cumulative failures plot (figure 28) for this data set does not fit the assumptions of YS and this accounts for its poor performance. GP and GO give better results than YS, but the large hump on the curve between 6 and 20 cause them to overestimate the number of errors in the latter part of the data set.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
GO	1	0.956
GP	2	0.948
YS	3	0.989

Table 6: Model ranking for data set 129.

The PLR graph (figure 30) confirms that GP and in particular GO outperform YS. Notice how similar the curves are for GO and GP.

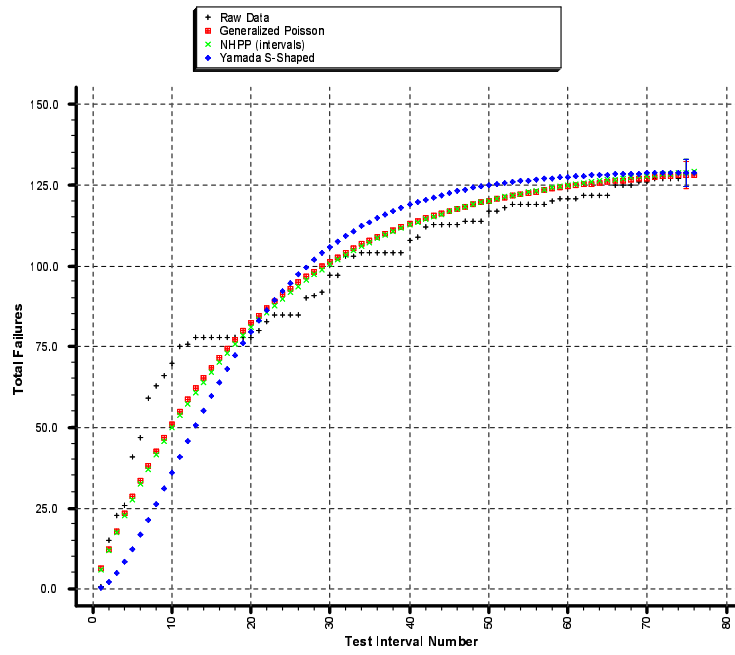


Figure 28: Cumulative failures for data set 129.

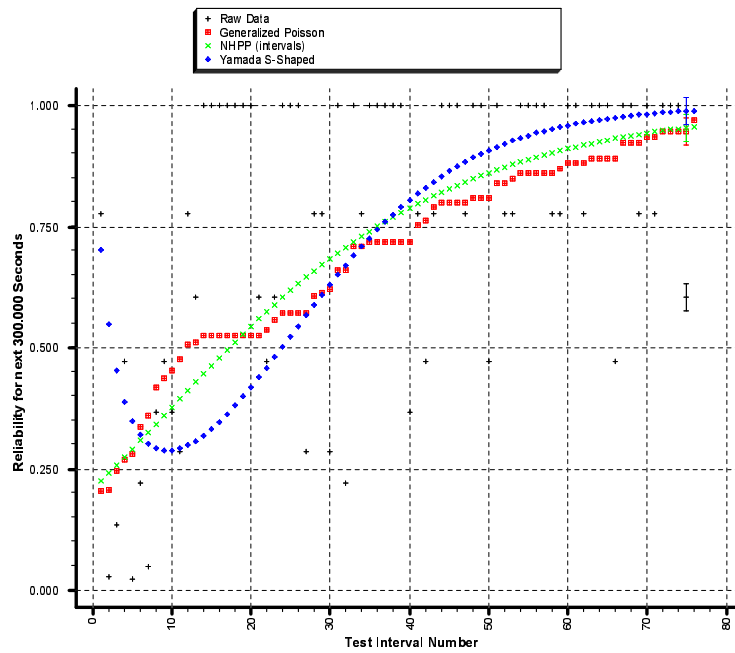


Figure 29: Reliability for data set 129.

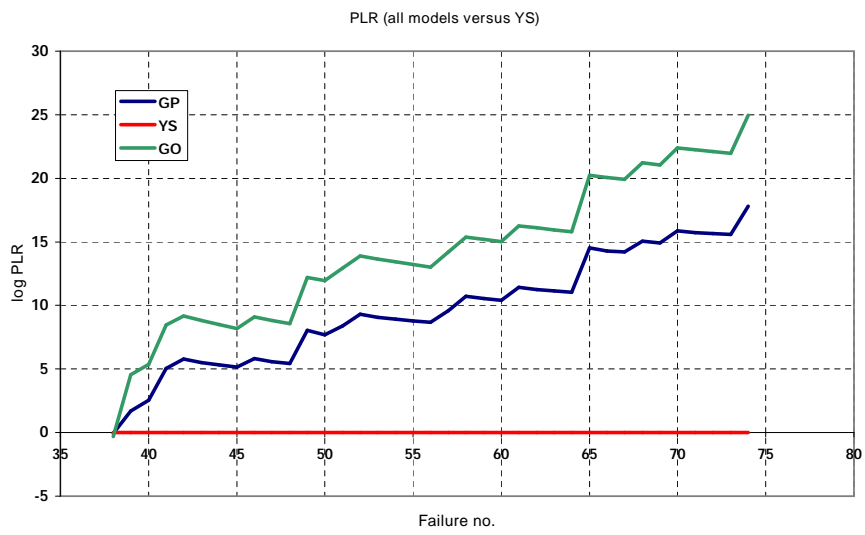


Figure 30: Logarithmic prequential likelihood ratio for data set 129.

### 4.3 Data Set 278

#### 4.3.1 Time Between Failures Approach

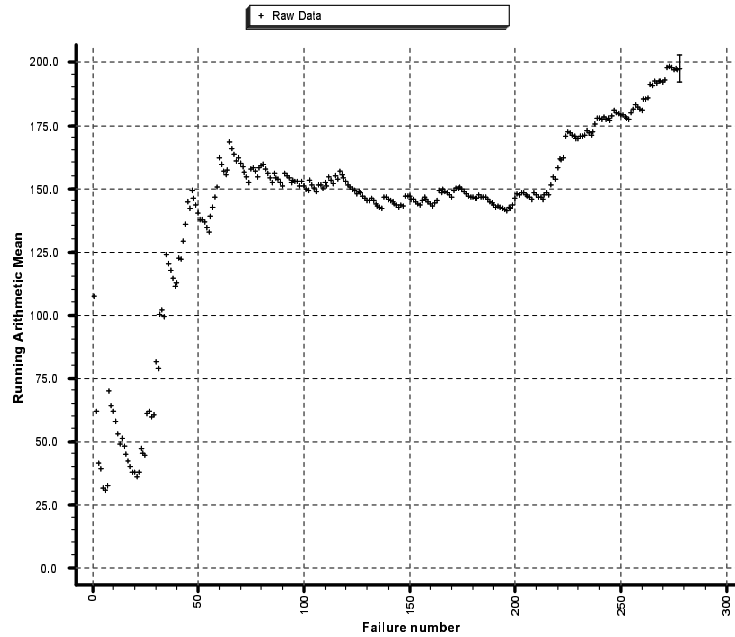


Figure 31: Running arithmetic mean for data set 278.

The running arithmetic mean graph in figure 31 shows an increase in reliability between failures 25 and 65 followed by a slight decrease and then from failure 220 onwards the reliability grows again.

The Laplace test in figure 32 shows the same trend as the running arithmetic mean. After some initial fluctuations there is global reliability growth, but local reliability decay, between failures 30 and 80. From failure 80 until 220 the data reveals no significant trend. Finally, after failure 220 there is both global and local reliability growth.

The two trend tests show that the reliability growth occurs between failures 220 and 278. To get the most accurate results, the software reliability growth models should be applied only to this interval. Unfortunately, CASRE cannot calculate valid estimates for the model parameters when this interval is used. Therefore the models are applied to the whole interval, but the data from 1 to 220 is only used to estimate the initial model parameters.

With maximum likelihood estimation the linear LV model predicts *negative* times between failures and failure intensities. This appears to be a problem with the numerical procedure used by CASRE for computing pa-

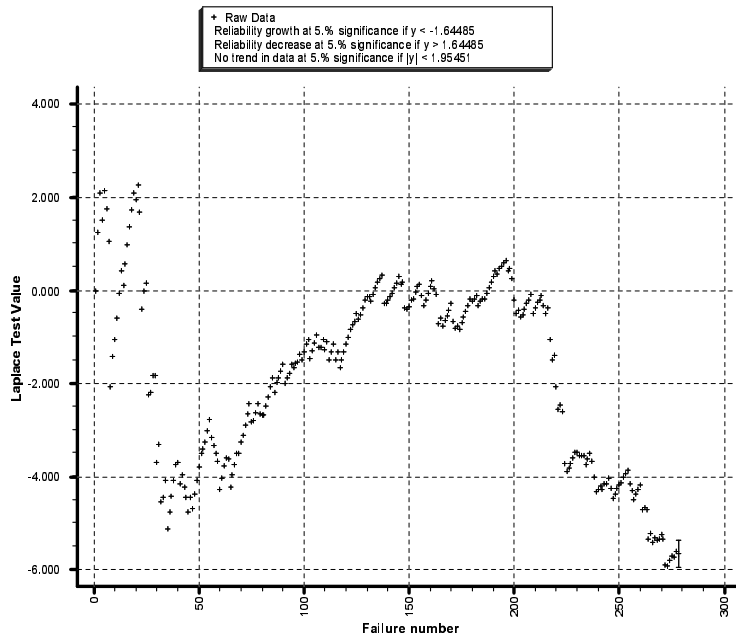


Figure 32: Laplace test for data set 278.

parameter estimates. In figures 33 through 36 least squares estimates are used, while the remaining figures use maximum likelihood, because they cannot be drawn with least squares estimates.

Figure 33 shows the time between failures as estimated by the the models JM, Linear LV and MO. Towards the end the JM model is clearly the most optimistic as was expected.

Figure 34 shows the failure intensities. This plot does not give much information.

The cumulative failures plot in figure 35 confirms the picture from the trend tests – until failure 220 the number of failures discovered is approximately proportional to the execution time, indicating no improvement in the reliability, and after number 220 the time between failures gradually increases, indicating reliability growth. Both MO and JM fit the data reasonably well while linear LV grossly overestimates the number of failures encountered after a given time.

The reliability plot in figure 36 displays considerable disagreement between the models. JM is optimistic, LV pessimistic and MO neutral. The area in which the three curves cross coincides with the 220th failure.

In the u-plot in figure 37 and the following figures, the LV model is disregarded because of the aforementioned problems with maximum likelihood estimation.

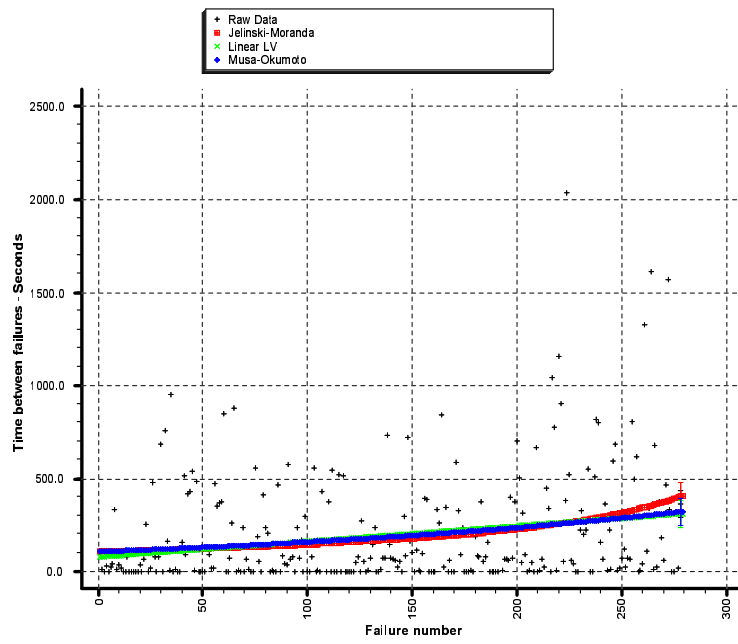


Figure 33: Time between failures for data set 278.

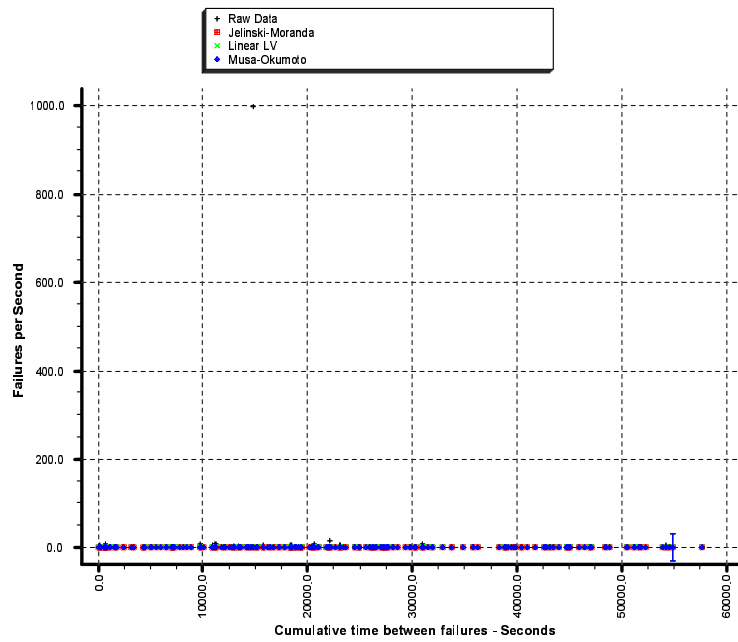


Figure 34: Failure intensity for data set 278.

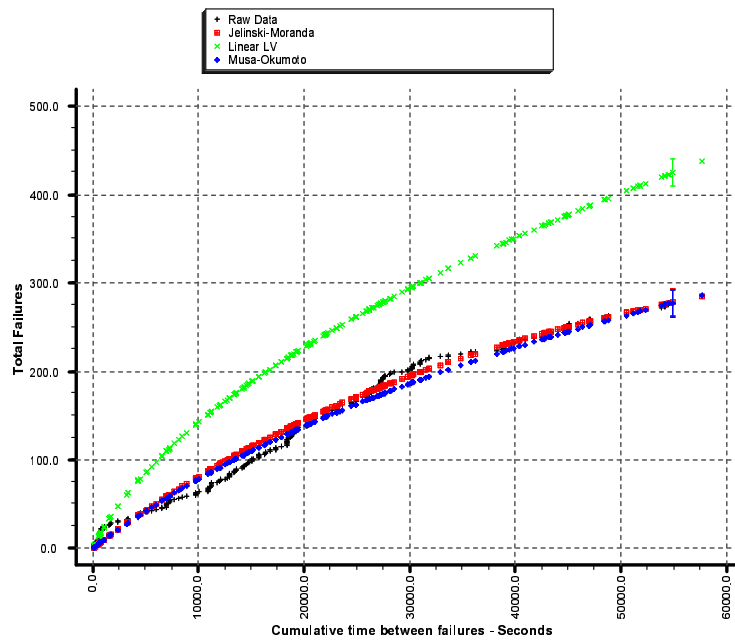


Figure 35: Cumulative failures for data set 278.

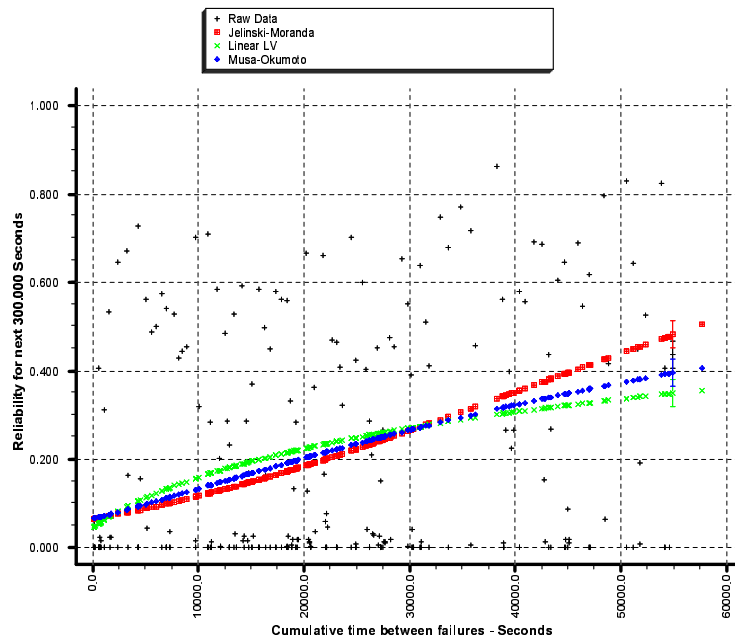


Figure 36: Reliability for five minutes for data set 278.

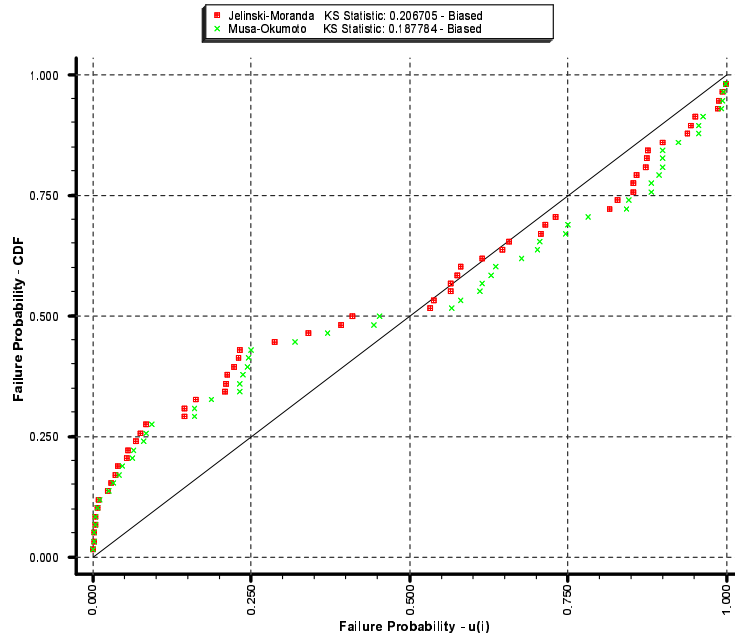


Figure 37: u-plot for data set 278.

Although both models are biased according to the Kolmogorov-Smirnov test at a 5% significance level, the fact that the curves of both models cross the line of unit slope at about 0.5 shows that this is not the case. Instead the models tend to either wildly overestimate or wildly underestimate, but not consistently one of the two.

The y-plot in figure 38 shows no significant bias trend in either model.

Figure 40 shows the prequential likelihood ratio, where the smallest prequential likelihood at each step is the denominator. The JM model slightly outperforms MO.

<i>Model</i>	<i>Noise</i>
JM	1.57
LV	1.60
MO	1.04

Table 7: Noise values for data set 278.

The log of the PLR is shown in figure 41. Here the JM model is the reference model for all steps. As before JM is seen to outperform MO slightly. LV is included in this final plot and with the outrageous predictions (*negative* times between failures) it is no surprise that it lags far behind the two other

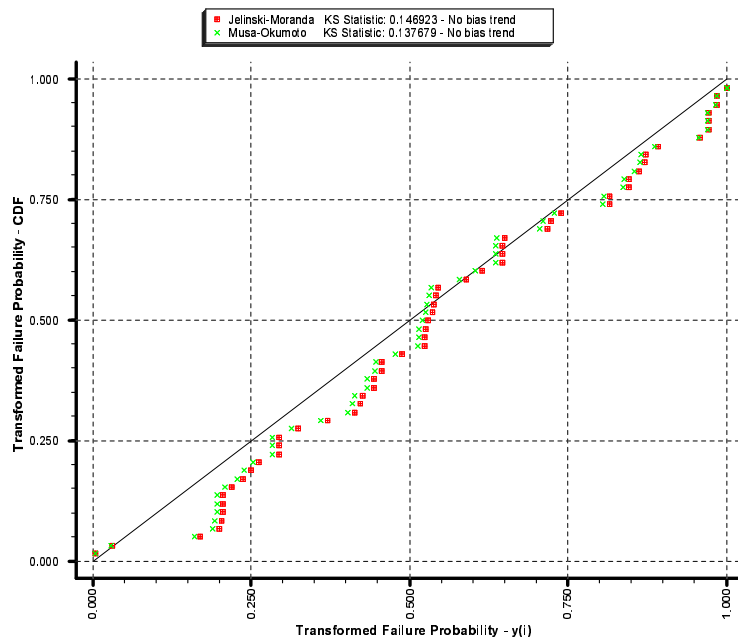


Figure 38: y-plot for data set 278.

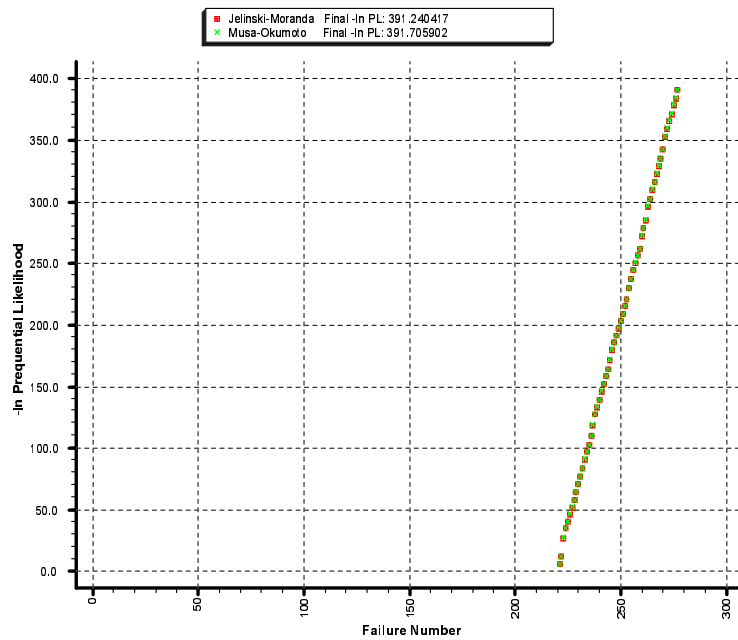


Figure 39: Prequential likelihood for data set 278.

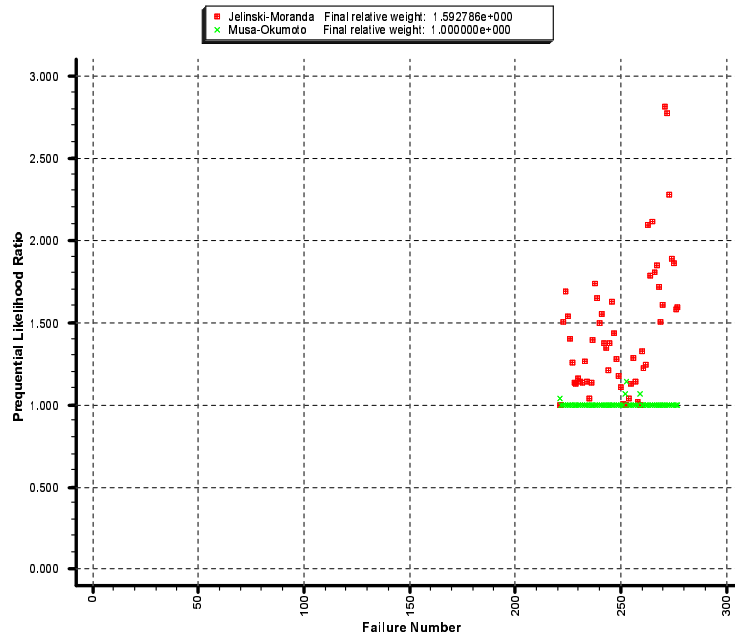


Figure 40: Prequential likelihood ratio for data set 278.

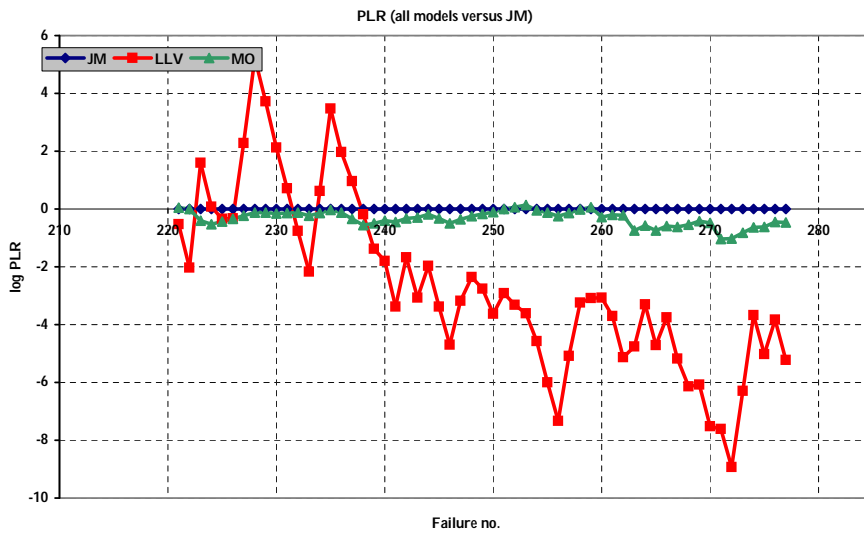


Figure 41: Logarithmic prequential likelihood ratio for data set 278.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
JM	1	0.457
MO	2	0.396
LV	3	0.412

Table 8: Model ranking for data set 278.

models with regards to accuracy. JM has a very low noise value for this data set. No models pass the goodness of fit test.

### 4.3.2 Failure Count Approach

The 278 inter-failure times are collected into 92 intervals of 600 seconds each.

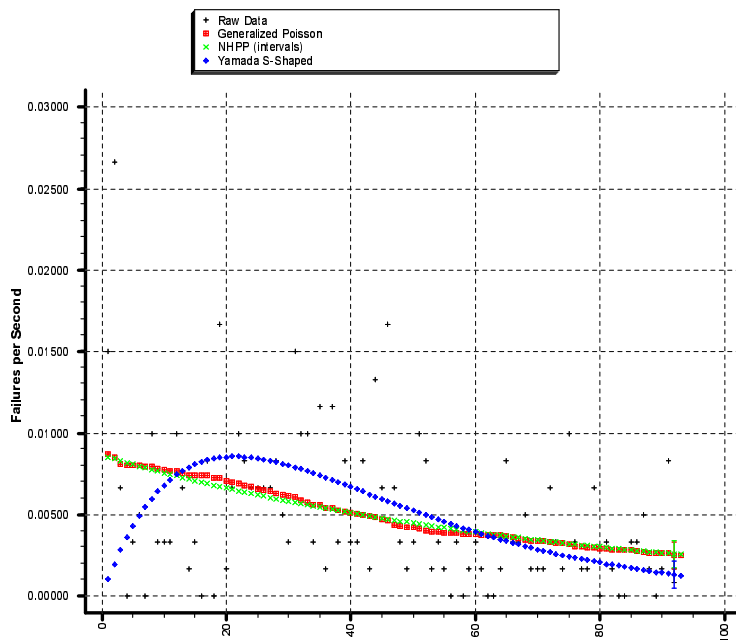


Figure 42: Failure intensity for data set 278.

Figure 34 again shows the characteristic shape of the S-shaped model and again the GP and GO have very similar curves.

The cumulative failures curve appears to fit the learning process assumption of the YS model. From interval 70 the GO and GP model provide a better fit of the raw data.

<i>Model</i>	<i>Rank</i>	<i>Reliability 300s</i>
GP	1	0.467
GO	2	0.459
YS	3	0.667

Table 9: Model ranking for data set 278.

The PLR graph supports the cumulative failures curve, because it shows that up to interval 70 the YS model is the most accurate but is then surpassed by GO and GP.

The best model for the next few predictions is GP.

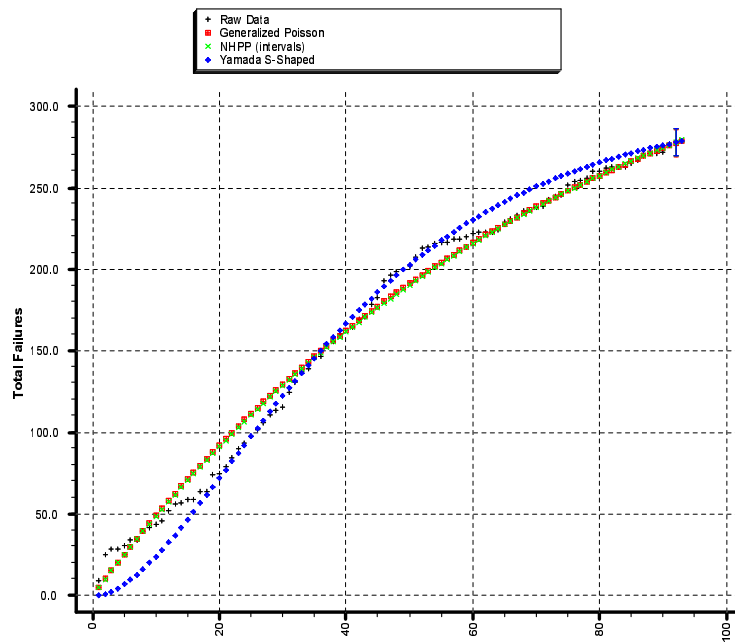


Figure 43: Cumulative failures for data set 278.

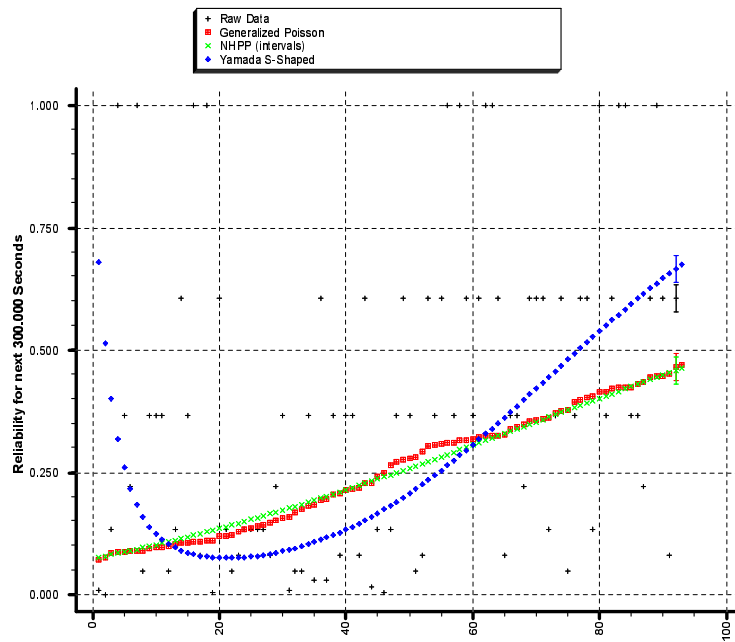


Figure 44: Reliability for data set 278.

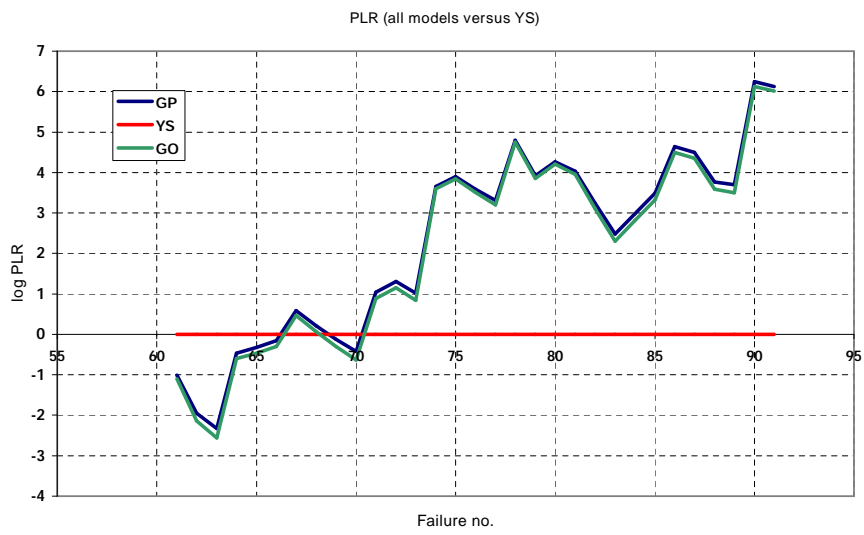


Figure 45: Logarithmic prequential likelihood ratio for data set 278.

#### 4.4 Supermodels Versus Simple Models

This section compares the results of JM, quadratic LV and MO against those of the supermodels SUPULC, SUPRLC and SUPDLC described in section 3.3. The data set 129 is used for the comparison.

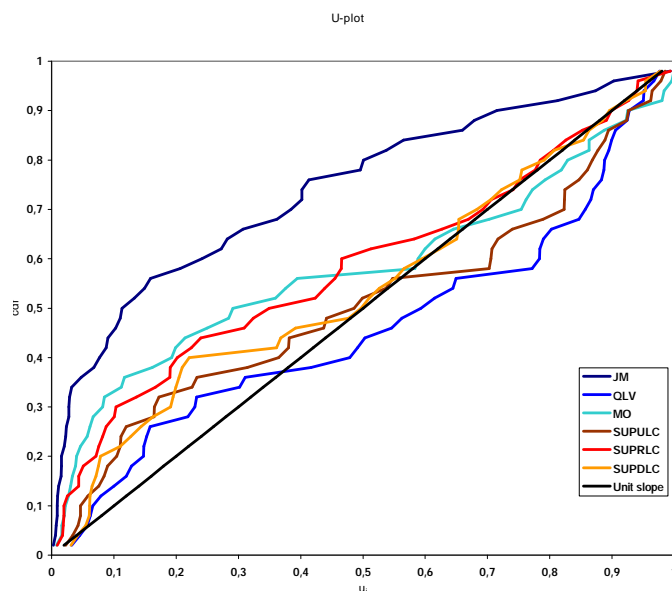


Figure 46: u-plot.

<i>Model</i>	<i>KS distance</i>	<i>Test 5% level</i>
JM	0.412518	Biased
LV	0.200113	Biased
MO	0.250082	Biased
SUPULC	0.153848	Not biased
SUPRLC	0.209018	Biased
SUPDLC	0.187147	Not biased

Table 10: Kolmogorov-Smirnov test statistics for u-plot.

The u-plot in figure 46 and the Kolmogorov-Smirnov distances show that all the supermodels perform well. Both SUPULC and SUPDLC are better than all of the component models and show no bias on a 5% level.

In the PLR the good performance of the supermodels is also apparent. All supermodels outperform MO and JM and are very close to LV.

The values of model noise in table 11 show that the supermodels are not as smooth as LV and MO. The SUPDLC model, where weights are assigned

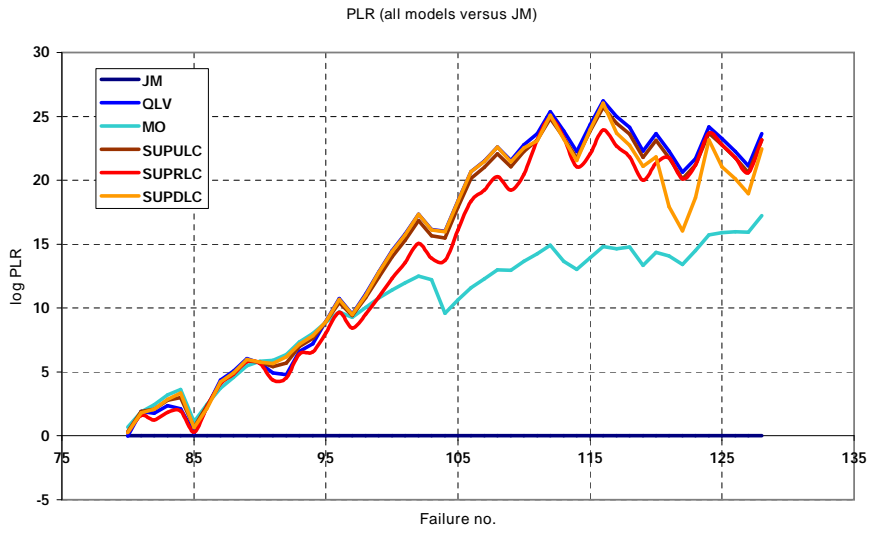


Figure 47: Logarithmic prequential likelihood ratio.

<i>Model</i>	<i>Noise</i>
JM	7.12
LV	2.08
MO	2.02
SUPULC	6.52
SUPRLC	6.21
SUPDLC	8.75

Table 11: Noise values.

based on changes in prequential likelihood, is the most noisy of all the models. This indicates that the weights change rapidly, which in term indicates that the model cannot decide which component model it should “trust” to be the most accurate.

<i>Model</i>	<i>Rank</i>	<i>Reliability</i>
LV	1	0.941
SUPULC	2	0.951
SUPRLC	3	0.977
SUPDLC	4	0.960
MO	5	0.966
JM	6	0.989

Table 12: Model ranking.

Although the supermodels perform well, LV is the only model to pass the goodness of fit test at the 5% level. LV also has the highest prequential likelihood. Therefore quadratic LV is still the best model for data set 129 as shown in the ranking in table 12.

## 5 Software Cost Estimation

### 5.1 Software Cost Model

One of the questions facing software development managers is when to release the software under given constraints of cost, time and reliability. This chapter presents and applies one model designed to answer such questions. The model is described in [2].

The notation used is

$m(T)$	expected number of errors to be detected by time $T$
$a$	total number of software errors to be eventually detected
$b$	exponential index
$\lambda(T)$	fault detection rate per unit time or intensity function
$x$	mission time
$R(x T)$	reliability function of software by time $T$ for a mission time $x$
$T$	software release time
$T^*$	optimum release time
$C_1$	software test cost per unit time
$C_2$	cost of removing each error per unit time during testing
$C_3$	cost due to software failure
$E(T)$	expected total cost of testing, debugging and possible failure of a software system by time $T$
$Y$	time to remove an error during testing phase
$\mu_y$	expected time to remove an error during testing phase

The general assumptions of the cost model are

- The cost of testing is proportional to the testing time.
- The cost of removing an error during testing is proportional to the total time spent removing errors detected by the end of testing.
- There is a risk cost related to the reliability at each release time point.
- The time required to remove an error during testing follows a truncated exponential distribution.

Under these assumptions, the expected cost of the software is

$$E(T) = C_1T + C_2m(T)\mu_y + C_3 [1 - R(x | T)]$$

Here  $C_1T$  is the cost of testing for time  $T$ . The cost of the debugging effort is  $C_2m(T)\mu_y$ . The expected cost of a potential failure within the mission time is  $C_3 [1 - R(x | T)]$ .

## 5.2 Example Application

For this example the data set 136 will be used in the failure count format, but this time with 25 time intervals. The model used is GO for which the mean value function is  $m(T) = a(1 - e^{-bT})$ . The cost of testing is assumed to be  $C_1 = 5$ , the cost of debugging during testing is assumed to be  $C_2 = 4$  and the cost of a failure is assumed to be  $C_3 = 1000$ . The mission time will be  $x = 1$  and the time to remove an error is  $\mu_y = 0.1$ .

First, the model is fitted to the data. Using the values  $a = 142.4$  and  $b = 0.12$  the mean value curve in figure 48 was obtained. The corresponding cost and reliability curves are shown in figures 49 and 50. The optimum time to release the software is the point at which the total cost is minimised. This point is shown in the cost graph and the value is  $T^* = 48.10$ . The total cost may then be calculated by inserting the mean value function and reliability function of the GO model into the expression for the expected cost. This calculation yields  $E(T^*) = 346.2$ .

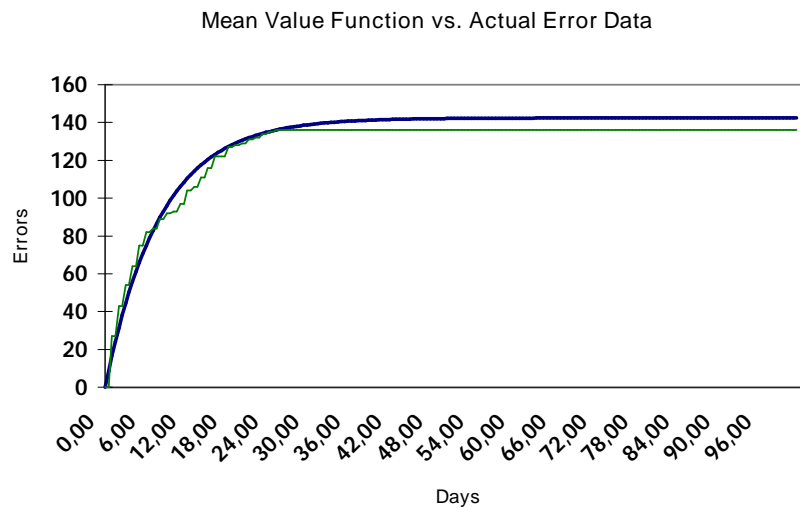


Figure 48: Mean value function.

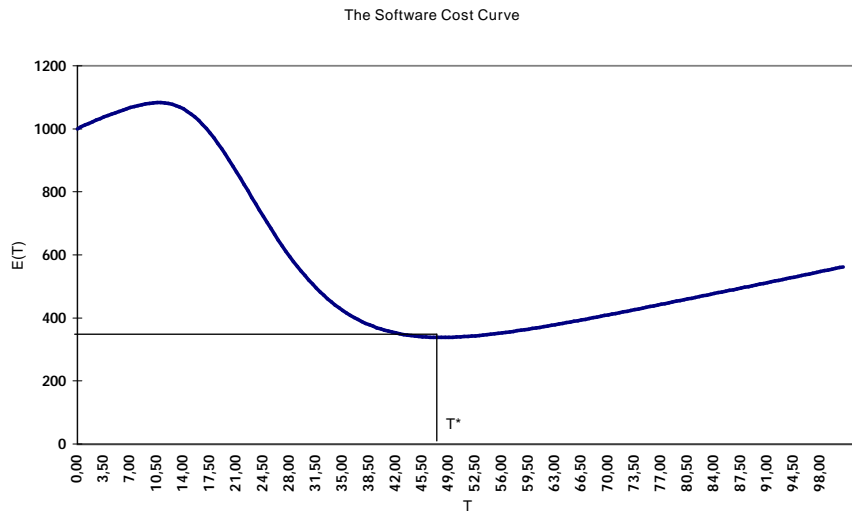


Figure 49: Expected cost function.

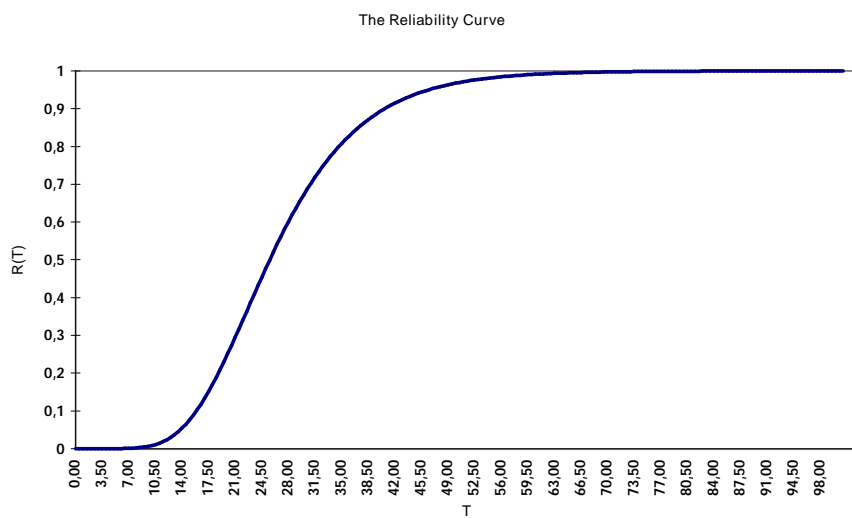


Figure 50: Reliability function.

## 6 Conclusions

This study has underlined that no one software reliability model is universally good. This is vividly illustrated by the fact that on the three time between failures data sets three different models were ranked number one.

The study has also confirmed that there is no way to select a good model a priori. The data set 278 which features periods of both reliability growth and decay should favour the Littlewood-Verrall model, because it does not assume perfect fixes, but in the ranking the Littlewood-Verrall model is last. Surprisingly the Jelinski-Moranda model is the best for this data set.

Among the interval domain models, the Yamada S-shaped model was the least accurate on all three data sets. The model assumes the initial testing phase involves a learning process, where testers gradually improve their skills. In fact all three data sets seem to indicate the opposite situation: there is a temporary drop in the efficiency of failure detection after the first part of the testing phase, but before the law of diminishing returns sets in. This is seen on the cumulative failures graphs.

The potential of supermodels has been demonstrated. The three supermodels examined performed well in comparison with the component models, although none of them attained the accuracy of the quadratic Littlewood-Verrall model. Even though the supermodels were not the best, two of them were unbiased while all the component models were biased.

To make a good supermodel requires component models that give very different predictions. For example a pessimistic model and an optimistic model may be combined to form an unbiased supermodel. If all the component models tend to make the same mistakes then the supermodels are not going to be any better.

The cost estimation models are undoubtedly a promising tool for software developers. However, the difficulty of estimating the many parameters is a problem that needs to be addressed if they are to gain widespread use in the industry.

## References

- [1] Allen P. Nikora, *Computer Aided Software Reliability Estimation User's Guide Version 3.0*. 1999.
- [2] Hoang Pham, *Software Reliability*. Springer Verlag 2000.
- [3] Allen P. Nikora and Michael R. Lyu, *An Experiment in Determining Software Model Applicability*. Proceedings of the Sixth International Symposium on Software Reliability Engineering, Toulouse, France 1995.