FITTING AN EXPONENTIAL SOFTWARE RELIABILITY MODEL TO FIELD FAILURE DATA. (U) POLYTECHNIC INST OF NEW YORK BROOKLYN DEPT OF ELECTRICAL ENGI. R W SCHMIDT MAY 82
UNCLASSIFIED POLY-EE/CC-82-002 N00014-75-C-0858 F/G 12/1 NL
FITTING AN EXPONENTIAL SOFTWARE

RELIABILITY MODEL TO

FIELD FAILURE DATA

BY

R. W. SCHMIDT

Prepared for
Office of Naval Research
Contract N00014-75-C-0858

Report No. POLY EE/CS 82-002

Polytechnic Institute of New York
Department of Electrical Engineering
and Computer Science
333 Jay Street
Brooklyn, New York 11201
**Title**: Fitting An Exponential Software Reliability Model to Field Failure Data

**Authors**: R. W. Schmidt

**Department**: Dept. of Elec. Engineering & Computer Science, Polytechnic Institute of New York, 333 Jay Street, Brooklyn, New York 11201

**Contract or Grant Number**: N00014-75-C-0858

**Report Date**: May, 1982

**Number of Pages**: 51

**Distribution Statement**: Reproduction in whole or in part is permitted for any purpose of the United States Government.

**Abstract**: The quantitative prediction and measurement of software reliability is of vital importance in the development of high quality cost effective software. Many software reliability models have been postulated in the literature, however few have been applied to field data. A model based upon the assumption that the failure rate of the software is proportional to the number of residual software errors leads to a constant failure rate and an exponential reliability function. The model contains two constants: the proportionality constant $K$ and the initial (total) number of errors $T$. 

**Keywords**: software reliability, model, model construction, field failure data
The constants K and E_T can be estimated during early design by comparison of the present project with historical data. During the integration test phase, a more accurate determination of the model parameters can be obtained by using simulator test data as if it were operational failure data. The simulator data is collected at two different points in the integration test phase and the two parameters can be determined from moment estimator formulas. The more powerful maximum likelihood method can also be employed to obtain point and interval estimates. It is also possible to use least squares methods to obtain parameter estimates which is the simplest method and provides insight into the analysis of the data.

This report utilizes a set of software development and field data taken by John D. Muse as a vehicle to study the ease of calculation and the correspondence of the three methods of parameter estimation. The sensitivity of the reliability predictions to parameter changes are studied and compared with field results.

The results show if data is carefully collected, software reliability models are practical and yield useful results. These can serve as one measure to help in choosing among competitive designs and as a gauge of when to terminate the integration test phase.
FITTING AN EXPONENTIAL SOFTWARE

RELIABILITY MODEL TO

FIELD FAILURE DATA

BY

R. W. SCHMIDT

Prepared for
Office of Naval Research
Contract N00014-75-C-0858

Report No. POLY EE/CS 82-002

Polytechnic Institute of New York
Department of Electrical Engineering and Computer Science
333 Jay Street
Brooklyn, New York 11201
The quantitative prediction and measurement of software reliability is of vital importance in the development of high quality cost effective software. Many software reliability models have been postulated in the literature (Ref. 11), however few have been applied to field data. A model based upon the assumption that the failure rate of the software is proportional to the number of residual software errors leads to a constant failure rate and an exponential reliability function, (Ref. 1). The model contains two constants: the proportionality constant $K$ and the initial (total) number of errors $E_T$.

The constants $K$ and $E_T$ can be estimated during early design by comparison of the present project with historical data. During the integration test phase, a more accurate determination of the model parameters can be obtained by using simulator test data as if it were operational failure data. The simulator data is collected at two different points in the integration test phase and the two parameters can be determined from moment estimator formulas (Ref. 9). The more powerful maximum likelihood method can also be employed to obtain point and interval estimates (Ref. 3). It is also possible to use least squares methods to obtain parameter estimates which is the simplest method and provides insight into the analysis of the data (Ref. 12).
This thesis utilizes a set of software development and field data taken by John D. Musa (Ref. 10) as a vehicle to study the ease of calculation and the correspondence of the three methods of parameter estimation. The sensitivity of the reliability predictions to parameter changes are studied and compared with field results.

This thesis is based in part on a joint paper written by the author and Professor Martin L. Shooman, presented at the ORSA-TIMS Conference in Florida, January 1981. (14)

The results show that if data is carefully collected, software reliability models are practical and yield useful results. These can serve as one measure to help in choosing among competitive designs and as a gauge of when to terminate the integration test phase.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td>Table of Contents</td>
<td>iv</td>
</tr>
<tr>
<td></td>
<td>Table of Figures</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Development of Error and Reliability Model</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>An Error Removal Model</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Development of the Exponential Reliability Model</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Meantime to Software Failure</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Musa's Model</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Estimation of Model Parameters</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>Moment Estimates</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Least Square Estimates</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Maximum Likelihood Estimates</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Musa's Data</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>20</td>
</tr>
<tr>
<td>4.2</td>
<td>Description of Raw Data</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>System Characteristics</td>
<td>23</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Estimation of Model Constants</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>Method of Moments</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>Least Squares Linear Regression</td>
<td>26</td>
</tr>
<tr>
<td>5.3</td>
<td>Maximum Likelihood Estimates</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>Model Parameter Accuracy and Sensitivity</td>
<td>35</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>6.2</td>
<td>Model Accuracy</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Model Sensitivity</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>43</td>
</tr>
<tr>
<td>FIGURE</td>
<td>TITLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>NORMALIZED ERROR RATE vs DEBUGGING TIME FOR THREE SUPERVISORY SYSTEMS</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>CUMULATIVE ERROR CURVE FOR SYSTEM A (FIG. 1)</td>
<td>5</td>
</tr>
<tr>
<td>2B</td>
<td>CUMULATIVE ERROR CURVE FOR SYSTEM B (FIG. 1)</td>
<td>6</td>
</tr>
<tr>
<td>2C</td>
<td>CUMULATIVE ERROR CURVE FOR SYSTEM C (FIG. 1)</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>VARIATION OF RELIABILITY FUNCTION R(t) WITH DEBUGGING TIME</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>COMPARISON OF MTTF vs DEBUGGING TIME FOR THE ERROR MODEL</td>
<td>9</td>
</tr>
<tr>
<td>4A</td>
<td>ERROR CORRECTION RATE MODELS</td>
<td>11</td>
</tr>
<tr>
<td>4B</td>
<td>GROWTH OF Eo(t) ASSOCIATED WITH ρ(t) OF FIG. 4A</td>
<td>11</td>
</tr>
<tr>
<td>4C</td>
<td>COMPARISON OF MTTF vs τ FOR VARYING ERROR CORRECTION RATE</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>GROWTH CURVE OF SOFTWARE MTTF (13)</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>MTTF vs TEST TIME (8)</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>SYSTEM 3 CUMULATIVE ERRORS vs FAILURE RATE</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>ESTIMATED θ vs TOTAL ERRORS $\hat{E}_T$ - SYSTEM 3</td>
<td>33</td>
</tr>
<tr>
<td>9</td>
<td>TOTAL ERRORS vs TESTING TIME - SYSTEM 3</td>
<td>38</td>
</tr>
<tr>
<td>10</td>
<td>CHANGE IN MTTF BY VARYING $\hat{E}_T$</td>
<td>39</td>
</tr>
<tr>
<td>11</td>
<td>EFFECT ON MTTF BY VARYING $\hat{E}_T$</td>
<td>40</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FAILURE INTERVALS-SYSTEM #3 TEST PHASE</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>SYSTEM #3 FAILURE RATE DATA</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>METHOD OF MOMENTS ESTIMATION OF ( \hat{K}, \hat{E}_T )</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>LEAST SQUARES REGRESSION ESTIMATION OF ( \hat{K}, \hat{E}_T )</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>MAXIMUM LIKELIHOOD ESTIMATION OF ( \hat{K}, \hat{E}_T )</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>COMPARISON OF MODEL PREDICTIONS WITH FIELD EXPERIENCE</td>
<td>42</td>
</tr>
</tbody>
</table>
1.0 Introduction

Software presently represents the highest cost item in the development of computer systems. There is a paucity of quantitative measures to judge the quality of the final software and use as a measure of progress during the test and debugging phase. The reliability and mean time to failure (MTTF) of the software is a most useful metric for both the above purposes.

An important class of software reliability models (see Refs. 1, 8) make the assumption that the operational software failure rate is proportional to the remaining number of errors. Thus the failure rate is dependent on development time $T$, but not on operating time $t$. This leads to a constant hazard and exponential reliability model, with two unknown parameters $K$ and $E_T$.

A major focus of this thesis is to investigate and provide insight into a number of issues related to the estimation of these two parameters:

1. The accuracy one can obtain by using historical data to determine $K$ and $E_T$ (Musa's);
2. A comparison of the accuracy obtained using three different methods of parameter estimation: the maximum likelihood, moments, least squares;
3. A comparison of predicted values of MTTF and observed MTTF values from field failure data;
4. Model parameter sensitivity; and,
5. Model accuracy/parameters related to the practical application by the software manager.
The conclusions reached in this study clearly indicate that parameter estimation during system development is a highly practical tool which can be used to successfully predict software MTTF and the 'debugging' time required to achieve that goal.
2.0 Development of Error and Reliability Models

2.1 An Error Removal Model

The reliability model used in this paper has been described in detail in a number of references (1, 2, 3). In brief, the model assumes that the program enters the integration test phase with \( E_T \) total errors remaining in the software. As integration testing proceeds, all detected errors are promptly corrected, and at any point in the development cycle (after \( T \) months of development time)*, a total of \( E_c(T) \) errors have been corrected, and the remaining number of errors, \( E_r \) is

\[
E_r(T) = E_T - E_c(T)
\]  

(1)

In a more advanced model (4) it is assumed in addition that new errors are generated during development. One can often normalize the above equation through division by the number of object code instructions \( I_T \) to yield

\[
E_r(T) = \frac{E_T - E_c(T)}{I_T}
\]

(2a)

\[
\varepsilon_r(T) = \frac{E_T - E_c(T)}{I_T}
\]

(2b)

Where \( \varepsilon_r = \frac{E_r}{I_T} \) and \( \varepsilon_c = \frac{E_c}{I_T} \)

Basically the error model used in this paper assumes that the total number of errors in the program is fixed and that if we record the

* In some cases the actual number of test hours is estimated and is used as the development time variable rather than the cruder calendar days.
cumulative number of errors corrected during debugging, then the difference represents the remaining errors. We can define error removal rates as:

\[ \frac{d\xi_r}{d\tau} = r_r(\tau) \] (3)

which can be normalized to yield

\[ \frac{d\xi_r(\tau)}{d\tau} = \rho_r(\tau) \] (4)

where

\[ \rho_r(\tau) = \frac{\text{errors removed}}{\text{total number of instructions}} \times \text{test hours} \] (5)

\[ \xi_r(\tau) = \int_{\infty}^{\tau} \rho(x)dx = \frac{\text{cumulative errors}}{\text{total number of instructions}}. \] (6)

In Reference 5 error data are reported for seven large supervisory programs and applications programs. In Fig. 1 the normalized error rate \( \rho(\tau) \) calculated from this data is plotted as a function of \( \tau \), the number of months of debugging after release for three of the seven systems. Although several curve shapes might be fitted to this data, one characteristic is common for all curves. The normalized error rate decreases over the entire curve or at least over the latter two-thirds or half of the curve; whereas initial behavior of \( \rho(\tau) \) differs from example to example.

A curve of the cumulative error data for the supervisory system A of Fig. 1 is shown in Fig. 2. Similar curves of \( \xi(\tau) \) drawn for the other examples of Fig. 1 all build up initially with a constant or increasing slope and then exhibit a decreasing curvature appearing to become asymptotic. The smoothing nature of integration makes all the \( \xi(\tau) \) curves look more alike than the \( \rho(\tau) \) curves do. (Figs. 2A & B). Both are needed for a detailed study.
FIGURE 1. NORMALIZED ERROR RATE vs DEBUGGING TIME FOR THREE SUPERVISORY PROGRAMS

FIGURE 2. Cumulative Error Curve for Supervisory System A Given in Figure 1.
FIGURE 2A
CUMULATIVE ERROR CURVE FOR SUPERVISORY SYSTEM B

FIGURE 2B
CUMULATIVE ERROR CURVE FOR SUPERVISORY SYSTEM C
If we assume that the total number of errors in the program \(E_T\) is constant and that the program contains \(I_T\) instructions, then the asymptote which the \(e(T)\) curves approach is \(E_T/I_T\). (Figs. 2, 2A, 2B).

The error model developed above will be used in the following section to formulate a reliability model.

2.2 Development of the Exponential Reliability Model

We assume that all operational software errors occur due to the occasional traversing of a portion of the program in which a hidden software error is lurking. We begin by writing an expression for the probability that an error is encountered in the time interval \(\Delta t\) after \(t\) successful hours of operation. We make the assumption that this probability is proportional to the number of errors remaining in the program. (See Ref. 6 for data substantiating this assumption).

From a study of basic probability and reliability theory we know that the probability of failure in time interval \(t\) to \(t + \Delta t\), given that no failures have occurred up till time \(\Delta t\), is proportional to the failure rate (hazard function) \(z(t)\). Thus, we obtain

\[
P(t < t_f < t + \Delta t \mid t_f > t) = z(t) \Delta t = K e_T(t) \Delta t
\]

where \(t_f\) = time to failure, (occurrence of a software error).

\[
P(t < t_f < t + \Delta t \mid t_f > t) = \text{probability of failure in interval } \Delta t,
\]
given no previous failure.

\(K\) = an arbitrary constant
From reliability theory (7) we can show that the probability of no system failures in the interval 0 to \( t \) is the reliability function which is related to the hazard, \( z(t) \), by:

\[
R(t) = e^{-\int_0^t z(x) \, dx}
\]  

(8)

If we substitute our expression for \( z(t) \) from Eq. 7 into Eq. 8 and assume \( K, K_x (T) \), are independent of operating time, we obtain

\[
R(t) = e^{-K_x (T) t} = e^{-\gamma t}
\]

(9)

Basically the above equation states that the probability of successful operation without software errors is an exponential function of operating time. When the system is first turned on, \( t = 0 \) and \( R(0) = 1 \). As operating time increases the reliability monotonically decreases as shown in Fig. 3. We depict the reliability function for three values of debugging time, \( T_0 < T_1 < T_2 \). From this curve we may make various predictions about the system reliability. For example, looking along the vertical line \( t = 1/\gamma \) we may state:

1. If we spend \( T_0 \) hours of debugging then \( R(1/\gamma) = 0.35 \)
2. If we spend \( T_1 \) hours of debugging then \( R(1/\gamma) = 0.50 \)
3. If we spend \( T_2 \) hours of debugging then \( R(1/\gamma) = 0.75 \)
NORMALIZED OPERATING TIME $t$

FIG. 3 Variation of Reliability Function $R(t)$ with Debugging Time $t$.

$\beta \times \text{MTTF} = \frac{1}{1 - \alpha t}$

Normalized Time $t$
- $0$
- $1/4 \alpha t$
- $1/2$
- $3/4$
- $1$

a. Constant Error Debug Rate

FIGURE 4. Comparison of MTTF vs Debug Time for the Error Model.
2.3 Meantime to Software Failure

A simpler way to summarize the results of the reliability model is to compute the mean time to (software) failure, MTTF, for the system.

\[ MTTF = \int_{0}^{\infty} \lambda(t) \, dt \]  

(10)

For purposes of illustration we let \( \lambda(t) \) be modeled by a constant rate of error correction \( \lambda_0 \). Solution of Eqs. 2 and 10 then yields

\[ MTTF = \frac{1}{\beta \left( \frac{E_T}{I_T} - \lambda_0 \tau \right) - \frac{1}{\beta (1 - \alpha \tau)}} \]  

(11)

where \( \beta = \frac{K E_T}{I_T} \) and \( \alpha = \frac{\lambda_0 I_T}{E_T} \).

This is depicted in Fig. 4 where \( \beta x \text{MTTF} \) is plotted vs \( \alpha \tau \), and we note that the greatest improvement in MTTF occurs during the last \( \tau \) of the debugging stage.

Reference 5 describes other error correction rate models, as illustrated in Fig. 4A. In order to compare the relative effects on system MTTF by assuming a varying \( \rho(T) \), it is essential that we integrate the latter over the same range. This is readily verified by noting that the total area under each curve is identical. Integration results are plotted in Fig. 4B for each model to yield the cumulative number of expected errors. The effect on system MTTF is depicted in Fig. 4C.
Figure 5. Growth Curve of Software Reliability Mean Time Between Software Errors. (From L. Miyamoto, Ref. 13).
Figure 6  MTBF vs. Test Time for Project 1.
(Replotted from the data of Fig. 3.
J. Musa, Ref. 8)
The rapid rise in MTTF toward the end of the integration phase is of considerable importance in software project management. Field data by others (see Figs. 5 & 6) confirms this basic shape. If a manager is pressed to release a system to the field at an early date, he may accept the current reliability and deliver the software. If we believe Figs. 4 thru 6, however, then a few more weeks could yield a big improvement in MTTF. The model predicts such behaviour, but if one only had test data for \( \hat{M} \), then it would be difficult to predict the sharp rise near \( \hat{M} = 1 \). The model is thus of great use in managing a project and setting its release date.

2.4 Musa's Model

Musa has developed a model similar to that given in Sec. 2.2. However, instead of basing his model on development time as the resource measure during integration, he utilized actual CPU time. Musa also used regular test data rather than simulator test data. His model is given by (8):

\[
R(T') = e^{-T'/T} \tag{12}
\]

\[
T = T_0 e^{-(G/T_0)} \tag{13}
\]
where

\[ T' = \text{hours of program operation} = t \]

\[ T = \text{mean time to failure in operating hours} = \text{MTTF} \]

\[ T_0 = \text{MTTF at the start of test} (t = 0) \]

\[ C = \text{ratio of equivalent operating time/test time} \]

\[ N_o = \text{number of failures which must occur to uncover all errors} = E_f \]

\[ \tau = \text{the CPU time in hours during testing} \]

Since we will be using Musa's data and some of his results in Sec. 5 of this thesis, we must carefully account for the different definitions of time between Musa's model and the exponential model during the analysis of his data.
3.0 Estimation of Model Parameters

3.1 Introduction

The exponential reliability model given in Eq. 9 contains the parameters $K$ and $E_r(t)$ which must be estimated. In many cases one wishes to use such a model to roughly predict MTTF during a proposal phase or early design of the project. In such a case the only available technique for determining values of the parameters is to use historical data. Presently Rome Air Development Center is developing a handbook and database on Software Reliability for just such a purpose.

3.2 Moment Estimates

In Ref. 9, a method is discussed for measurement of the two needed parameters based on simulation testing of the software. A program simulating the field environment is generally available for all real live computer programs. It is necessary that this program be run for a total of $H_1$ hours following $T_1$ months of development. It is assumed also that the testing will produce $r_1$ failures. Similarly after $T_2$ months of testing, the simulator is run and $H_2$ hours and $r_2$ failures are obtained. The MTTF for the data is given by the ratio $H_2/r_2$ and is equated to the MTTF expression obtained by substituting Eq. 9 into Eq. 10. The two equations (for $T_1$ and $T_2$) allow us to solve for the constants $K$ and $E_r$:

$$\frac{H_1}{r_1} = \frac{1}{K[E_T - E_r(T_1)]}$$  \hspace{1cm} (14)

$$\frac{H_2}{r_2} = \frac{1}{K'[E_T - E_r(T_2)]}$$  \hspace{1cm} (15)
Simultaneous solution of Eqs. 14 and 15 yields the desired values of $\hat{\varepsilon}_T$ and $\hat{k}$:

$$\hat{\varepsilon}_T = \frac{\left(\frac{\lambda_2}{\lambda_1} x \varepsilon_c(T)\right) - \varepsilon_c(T_2)}{\left(\frac{\lambda_2}{\lambda_1} - 1\right)}$$

(16)

$$\hat{k} = \frac{\lambda_1}{\hat{\varepsilon}_T - \varepsilon_c(T)}$$

(17)

Note that in deriving the above results we have assumed that the failure rate is constant and have used the common notation and well known results for constant failure rates:

$$s(t) = \lambda = \frac{x}{H} = \frac{1}{RTF}$$

3.3 Least Square Estimates

Another method of estimating model parameters is to rewrite equation 11 in the form

$$\lambda_i = \hat{k}\left[\hat{\varepsilon}_T - \varepsilon_c(T)\right]$$

(18)

for $\lambda = \text{failure rate}$

to yield $E_c(T_i) = E_T - \frac{1}{\hat{k}} \lambda_i$

(19)

This equation represents a straight line whose parameters can be determined from the slope and intercept of a least squares fit of the data, where
\begin{align*}
\hat{k} &= \frac{-1}{\text{slope}} \\
\hat{E}_T &= \text{intercept}
\end{align*}

Naturally, the larger and the more accurate the data set, the more precisely can the model parameters be predicted.

3.4 Maximum Likelihood Estimates

Another method which can also be used to estimate the values of \( \hat{k} \) and \( \hat{E}_T \) is known as the Maximum Likelihood Estimation technique (MLE). The likelihood function, \( L \) is the joint probability of occurrence for the observed set of test values. If during simulation testing we observe \( r_1 \) failure times \( (t_1, t_2, \ldots, t_{r_1}) \) and \( n_1-r_1 \) successful runs testing \( (T_1, T_2, \ldots, T_{n_1-r_1}) \) then the likelihood function for a single test after \( E_g(T_1) \) errors have been removed is given by

\[
L(K, E_T) = f(t_1)f(t_2)\ldots f(t_{r_1})R(t_1)R(T_2)\ldots R(T_{n_1-r_1})
\]

where

\[ f(t) = \text{the density function } K \xi(t) e^{-K \xi(t)/T}
\]

\[ R(T) = \text{the reliability function } e^{-\xi(T)/T}
\]

To maximize the likelihood function we take partial derivatives with respect to \( K \) and \( E_T \) and set them equal to zero. To solve for the two operations we need a second equation obtained from another likelihood equation based on a second set of test data at time \( T_2 \). Applying MLE
theory to two tests with \( r_1 \) and \( r_2 \) failures over \( H_1 \) and \( H_2 \) total hours,
we obtain (Appendix A, Ref. 3)

\[
\hat{K} = \frac{r_1 + r_2}{H_1 \left[ \frac{E - E_c}{c} (r_1) \right] + H_2 \left[ \frac{E - E_c}{c} (r_2) \right]}
\]  
(22)

\[
\hat{k} = \frac{1}{H_1 + H_2} \left[ \frac{r_1}{\frac{E - E_c}{c} (r_1)} + \frac{r_2}{\frac{E - E_c}{c} (r_2)} \right]
\]  
(23)

As is often the case with MLE, the above equations require numerical solution; however, most statisticians believe them to yield superior results to moment estimates. An iterative computer solution of Eqs. 22 and 23 is easily implemented, yet a graphical solution using a simple calculator suffices in most cases. The first step is to obtain starting values for \( \hat{E}_T \) and \( \hat{k} \) using some other method, such as Method of Moments. Values of \( \hat{E}_T \) above and below the starting value are substituted into Eqs. 22, 23 and the curves of \( K \) vs \( E_T \) plotted on the same axes. Their intersection determines the value of \( \hat{E}_T \) and \( \hat{k} \).
4.0 Musa's Data

4.1 Introduction

The software reliability data used in this paper was compiled by John D. Musa (10) over a period of time on a variety of large software systems, ranging from tens to hundreds of thousands of object code instructions. His objective was "to present in detail a substantial body of data that has been gathered in the application of the execution time theory of software reliability"; the end product is ideally suited for the purpose of this paper, as it presents a wealth of precise software failure data obtained under carefully controlled circumstances.

4.2 Description of Raw Data

Software failure interval data was presented in the following format:

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Failure interval</th>
<th>Day of failure</th>
</tr>
</thead>
</table>

Failure interval was measured in seconds, and represents either running clock time, (operating time on the computer), or, in one case, actual CPU time. 'Day of Failure' is the working day count:1 from the start of project on which the failure occurred. (See Table 1).
### Table 1

**Failure Intervals - System 3 Test Phase**

<table>
<thead>
<tr>
<th>Failure Number</th>
<th>Failure Interval</th>
<th>Day of Failure</th>
<th>Failure Number</th>
<th>Failure Interval</th>
<th>Day of Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>115</td>
<td>1</td>
<td>21</td>
<td>15</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>22</td>
<td>390</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>83</td>
<td>3</td>
<td>23</td>
<td>1863</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>178</td>
<td>3</td>
<td>24</td>
<td>1337</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>194</td>
<td>3</td>
<td>25</td>
<td>4508</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>136</td>
<td>3</td>
<td>26</td>
<td>834</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>1077</td>
<td>3</td>
<td>27</td>
<td>3400</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>3</td>
<td>28</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>3</td>
<td>29</td>
<td>4561</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>92</td>
<td>3</td>
<td>30</td>
<td>3186</td>
<td>44</td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>3</td>
<td>31</td>
<td>10571</td>
<td>47</td>
</tr>
<tr>
<td>12</td>
<td>71</td>
<td>3</td>
<td>32</td>
<td>563</td>
<td>47</td>
</tr>
<tr>
<td>13</td>
<td>606</td>
<td>6</td>
<td>33</td>
<td>2770</td>
<td>47</td>
</tr>
<tr>
<td>14</td>
<td>1189</td>
<td>8</td>
<td>34</td>
<td>652</td>
<td>48</td>
</tr>
<tr>
<td>15</td>
<td>40</td>
<td>8</td>
<td>35</td>
<td>5593</td>
<td>50</td>
</tr>
<tr>
<td>16</td>
<td>788</td>
<td>18</td>
<td>36</td>
<td>11696</td>
<td>54</td>
</tr>
<tr>
<td>17</td>
<td>222</td>
<td>18</td>
<td>37</td>
<td>6724</td>
<td>54</td>
</tr>
<tr>
<td>18</td>
<td>72</td>
<td>18</td>
<td>38</td>
<td>2546</td>
<td>55</td>
</tr>
<tr>
<td>19</td>
<td>615</td>
<td>18</td>
<td>39</td>
<td>10175*</td>
<td>56</td>
</tr>
<tr>
<td>20</td>
<td>589</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: If the last interval is followed by an asterisk, there was no failure at the end of the period and the time represents the interval between the last failure and the end of the period. (10).*
For the purpose of this paper, failures occurring on the same working day were summed together to form one statistical data point, and were tabulated under the following format: (See Table 2)

| S | WD | E | Ec | s | T | Tc |

where:

S = sequential serial number assigned to each statistical data point
WD = working day on which failure(s) occurred
E = number of errors occurring that working day
Ec = cumulative errors to date
T = total operating time (failure interval time) for that working day
Tc = cumulative failure interval time to date
s = failure rate (E/T) for that working day

Presentation of Musa's data in this format had a two-fold purpose:

a) reduce the sheer bulk of the raw data without affecting its statistical significance by grouping the occurrence of software failures by working day; and,
b) tabulate the data in a format more suitable for subsequent calculations.
4.3 System Characteristics

Systems 1-4 studied by Musa are real-time command and control software packages consisting of 21,700 to 33,500 object instructions and a failure sample size of 38 to 136.

System 5 is a realtime commercial application, consisting of 2,445,000 object instructions and a failure sample size of 831. Musa notes that for system 5, design changes involving 21% of the source code were introduced after approximately 30% of total testing time had elapsed.
### Table 2

**System #3 Failure Rate Data**

<table>
<thead>
<tr>
<th>S</th>
<th>WD</th>
<th>E</th>
<th>Ec</th>
<th>z</th>
<th>T</th>
<th>Tc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>.01739</td>
<td>115</td>
<td>115</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>10</td>
<td>12</td>
<td>.00523</td>
<td>1911</td>
<td>2026</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>1</td>
<td>13</td>
<td>.00165</td>
<td>606</td>
<td>2632</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>2</td>
<td>15</td>
<td>.00163</td>
<td>1229</td>
<td>3961</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>4</td>
<td>19</td>
<td>.00236</td>
<td>1697</td>
<td>5558</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>3</td>
<td>22</td>
<td>.00302</td>
<td>994</td>
<td>6552</td>
</tr>
<tr>
<td>7</td>
<td>27</td>
<td>1</td>
<td>23</td>
<td>.00054</td>
<td>1863</td>
<td>9415</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>1</td>
<td>24</td>
<td>.00075</td>
<td>1337</td>
<td>9752</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>1</td>
<td>25</td>
<td>.00022</td>
<td>4508</td>
<td>14,280</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1</td>
<td>26</td>
<td>.00119</td>
<td>834</td>
<td>15,094</td>
</tr>
<tr>
<td>11</td>
<td>40</td>
<td>2</td>
<td>28</td>
<td>.00059</td>
<td>3406</td>
<td>18,500</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>1</td>
<td>29</td>
<td>.00022</td>
<td>4561</td>
<td>23,061</td>
</tr>
<tr>
<td>13</td>
<td>44</td>
<td>1</td>
<td>30</td>
<td>.00031</td>
<td>3126</td>
<td>26,247</td>
</tr>
<tr>
<td>14</td>
<td>47</td>
<td>3</td>
<td>33</td>
<td>.00022</td>
<td>13,904</td>
<td>40,151</td>
</tr>
<tr>
<td>15</td>
<td>48</td>
<td>1</td>
<td>34</td>
<td>.00153</td>
<td>652</td>
<td>40,003</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>1</td>
<td>35</td>
<td>.00018</td>
<td>5593</td>
<td>46,396</td>
</tr>
<tr>
<td>17</td>
<td>54</td>
<td>2</td>
<td>37</td>
<td>.00011</td>
<td>18,420</td>
<td>64,316</td>
</tr>
<tr>
<td>18</td>
<td>55</td>
<td>1</td>
<td>38</td>
<td>.00039</td>
<td>2546</td>
<td>67,762</td>
</tr>
</tbody>
</table>
5.0 Estimation of Model Constants

In the following section, we will be estimating the model constants \( k \) and \( E_c \) of several systems, using system #3 as a working example to demonstrate the calculations used for Least Squares, Method of Moments, and Maximum Likelihood Estimates. These constants will then be used to estimate system MTTF using equation 11.

As indicated in Sec. 4.2, Musa's raw data was condensed to form one statistical data point per working day during which one or more errors were uncovered.

In the following section, this condensed data was reduced even further to allow 2, 3, 4, 6, 8... points to represent an entire system as required. For example, if it were desired to represent system 3 (see Table 2) consisting of 18 entries, by two points (Fig. 7), the first nine entries were averaged statistically to yield the first point and the last 9 entries to yield the second point.

5.1 Method of Moments

The data reported by Musa for system #3 has been processed and grouped by working day as described in Sec. 4.2 (see Table 2). Using data points for the 9th and 18th group of failures, \((E_c 9, T_c 9)\), \((E_c 18, T_c 18)\), we obtain the average failure rate for the interval 0-9:

\[
\lambda_1 = \frac{E_c 9 - E_c 9}{T_c 9} = \frac{25 \text{ failures}}{14,260 \text{ sec.}} = 0.00175 \text{ failures/sec.} = 6.3 \text{ failures/hr.}
\]

and similarly the average failure rate for the interval 10-18 is:

\[
\lambda_2 = \frac{E_c 18 - E_c 9}{T_c 18 - T_c 9} = \frac{38 - 25}{67,262 - 14,260} = 2.448 \times 10^{-4} \text{ failures/sec.} = 0.881 \text{ failures/hr.}
\]
\[ \lambda_2 / \lambda_1 = 0.1396 \]

\[ \therefore \hat{\lambda}_T = \frac{\lambda_2 / \lambda_1 \cdot \text{Ec9} - \text{Ec19}}{(\lambda_2 / \lambda_1 - 1)} = \frac{(0.1396 \cdot 25) - 38}{(0.1396 - 1)} = 40.110 \]

and \[ K = \frac{\lambda_1}{\hat{\lambda}_T - \text{Ec9}} = \frac{0.00175}{40.110 - 25} = 1.160 \times 10^{-4} \]

See Table 3 for a summary of all systems.

5.2 Least Squares Linear Regression

In this method, we plot the failure rate \( z \) vs cumulative errors Ec. Theoretically the best fit y-intercept yields \( \hat{\lambda}_T \) and the negative reciprocal of the slope equals \( K \).

The primary equations used in linear regression are:

\[
\text{slope } m = \frac{\Sigma XY - \frac{\Sigma X \Sigma Y}{N}}{\Sigma X^2 - \frac{(\Sigma X)^2}{N}} \tag{24}
\]

\[
\text{intercept } y' = \frac{\Sigma Y - m \Sigma X}{N} \tag{25}
\]

for the straight line given by:

\[ y = mx + y' \tag{26} \]

Applying the above to equation 19, and referring back to Table 2,
### TABLE 3

**METHOD OF MOMENTS ESTIMATION OF \( \hat{E}_T, \hat{K} \)**

<table>
<thead>
<tr>
<th>System</th>
<th>#Pts</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Ec = 136 )</td>
<td>( \hat{E}_T )</td>
<td>151.6</td>
<td>153.6</td>
<td>126.4</td>
</tr>
<tr>
<td></td>
<td>( \hat{K} )</td>
<td>5.514</td>
<td>4.576</td>
<td>13.56</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Ec = 54 )</td>
<td>( \hat{E}_T )</td>
<td>61.6</td>
<td>52.6</td>
<td>38.7</td>
</tr>
<tr>
<td></td>
<td>( \hat{K} )</td>
<td>3.909</td>
<td>5.688</td>
<td>7.731</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Ec = 38 )</td>
<td>( \hat{E}_T )</td>
<td>40.1</td>
<td>38.2</td>
<td>28.3</td>
</tr>
<tr>
<td></td>
<td>( \hat{K} )</td>
<td>11.60</td>
<td>21.40</td>
<td>19.61</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Ec = 53 )</td>
<td>( \hat{E}_T )</td>
<td>54.9</td>
<td>54.9</td>
<td>22.2</td>
</tr>
<tr>
<td></td>
<td>( \hat{K} )</td>
<td>19.88</td>
<td>13.40</td>
<td>26.20</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Ec = 74 )</td>
<td>( \hat{E}_T )</td>
<td>111.7</td>
<td>117.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \hat{K} )</td>
<td>28.345</td>
<td>27.639</td>
<td></td>
</tr>
</tbody>
</table>

*N.B. \( \hat{K} \) values x 10^{-5}*
\[ E_{c9} = 25 \quad E_{c18} = 38 \]

\[ T_{c9} = 14,260 \quad T_{c18} = 67,362 \]

We plot

\[ X_1 = \frac{25}{14,260} = 1.75 \times 10^{-4} \]

\[ X_2 = \frac{38-25}{67,362-14,260} = 2.45 \times 10^{-4} \]

\[ Y_1 = E_{c9} = 25 \]

\[ Y_2 = E_{c18} = 38 \]

to yield

\[ \hat{X}_T = 40.1 \]

\[ \hat{X} = 1.1603 \times 10^{-4} \]

The above calculations are depicted in Fig. 7 for system 3 and summarized for all systems in Table 4.

5.3 Maximum Likelihood Estimates (MLE)

The primary equations used (see Sec. 3.4) are:

\[ \hat{X}_1 = \frac{r_1 + r_2}{H_1(E_T-E_{c1})+H_2(E_T-E_{c2})} \quad (22) \]

\[ \hat{X}_2 = \frac{1}{H_1+H_2} \left[ \frac{r_1}{E_T-E_{c1}} + \frac{r_2}{E_T-E_{c2}} \right] \quad (23) \]
FIGURE 7
CUMULATIVE ERRORS vs FAILURE RATE
SYSTEM 3
TABLE 4

LEAST SQUARES REGRESSION ESTIMATES OF $\hat{E}_T$, $\hat{k}$

<table>
<thead>
<tr>
<th>System</th>
<th>#Pts</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>24*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\hat{E}_T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>151.6</td>
<td>140.1</td>
<td>103.3</td>
<td>74.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.515</td>
<td>5.245</td>
<td>17.669</td>
<td>543.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\hat{k}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>61.6</td>
<td>51.7</td>
<td>45.5</td>
<td>44.9</td>
<td>42.9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40.1</td>
<td>34.9</td>
<td>32.8</td>
<td>30.6</td>
<td>23.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.603</td>
<td>25.490</td>
<td>23.525</td>
<td>27.942</td>
<td>55.593</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>54.9</td>
<td>56.8</td>
<td>49.8</td>
<td>42.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19.885</td>
<td>14.029</td>
<td>20.460</td>
<td>40.601</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2390</td>
<td>998.9</td>
<td>805.2</td>
<td>742.9</td>
<td>497.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.089</td>
<td>30.675</td>
<td>44.983</td>
<td>51.475</td>
<td>94.518</td>
<td></td>
</tr>
</tbody>
</table>

N.B. All $\hat{k}$ values x 10^{-5}

* Number of points used represents total system data without interval grouping.
A numerical solution of the above equations can be readily obtained, since we have two equations in two unknowns, \( \hat{K} \) and \( \hat{E}_T \). A simpler approach assuming we only have a few estimates to make, is to effect a graphical solution:

Step 1: obtain a starting value for \( E_T \) by any method of your choice: Method of Moments, Least Squares Regression, Non-deterministic (i.e. guess), ...

Step 2: substitute a value \( E_T \) obtained in Step 1 into equations 22 and 23. Try other values of \( E_T \) above and below this value and repeat the calculations, plotting them on the same axis of \( \hat{K} \) vs \( \hat{E}_T \) for the two equations; and,

Step 3: the intersection of these two curves yields the required parameters.

The method is illustrated using system 3 as an example:

Step 1: \( E_T = 40.1 \) using the starting value obtained by the Method of Moments (MOM) Table 3.

Step 2: For \( E_{T1} = 39 \), \( K_1 = 1.504 \times 10^{-4} \) \( (\text{from 22}) \)
\( K_2 = 2.195 \times 10^{-4} \) \( (\text{from 23}) \)

For \( E_{T2} = 41 \), \( K_1 = 8.355 \times 10^{-5} \) \( (\text{from 22}) \)
\( K_2 = 7.008 \times 10^{-5} \) \( (\text{from 23}) \)

Step 3: The above results are plotted on Fig. 8 to yield
\( E_T = 40.1 \)
\( \hat{K} = 11.6 \times 10^{-5} \)

Once an approximate value is obtained from the curves' intersection, iterative methods can be used to obtain the required degree
of accuracy.

See Table 5 for a summary of the MLE estimates for all four systems.
FIGURE 8
ESTIMATED \( \hat{k} \) vs TOTAL ERRORS \( \hat{E}_T \)
SYSTEM 3

\[ \hat{k} = 11.66 \times 10^{-5} \]
\[ \hat{E}_T = 40.109 \]
### Table 5

**Maximum Likelihood Estimates of \( \hat{E}_T, \hat{k} \)**

<table>
<thead>
<tr>
<th>System</th>
<th>( \hat{E}_T )</th>
<th>( \hat{k} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>151.59</td>
<td>( 5.141 \times 10^{-5} )</td>
</tr>
<tr>
<td>2</td>
<td>61.61</td>
<td>( 3.909 \times 10^{-5} )</td>
</tr>
<tr>
<td>3</td>
<td>40.11</td>
<td>( 1.160 \times 10^{-4} )</td>
</tr>
<tr>
<td>4</td>
<td>54.80</td>
<td>( 2.249 \times 10^{-4} )</td>
</tr>
</tbody>
</table>
6.0 Model Parameter Accuracy and Sensitivity

6.1 Introduction

Several additional factors must be considered when using this exponential model - or any other model for that matter - in predicting software system performance.

Two aspects considered in this thesis are:

1. Model Accuracy: The more data one has on a particular system, the more confident one can be that the particular model predictions will approximate reality;

2. Model Sensitivity: When software field failure data is available to compare with model prediction estimates, it seems - initially that slight changes in model parameters are not proportionally reflected in model predictions for system MTTF;

6.2 Model Accuracy

Intuitively, one feels that the more data one has about a particular system, the more accurate will be our predictions about system performance. To a certain extent this is true. This implies, however, that model prediction accuracy only approaches reality as we near the final debugging stage, whereas the 'software manager' is quite interested in estimating software reliability during early testing stages to determine future trends. As noted in Section 2.3, this is strongly implied by Fig. 4 which indicates that the maximum return for
one's efforts is obtained only during the latter 25% of the debugging phase.

Hindsight will usually enable us to look back at a particular software development project and to announce that 'x' time rather than 'y' time should have been spent on development and testing.

The onset of the release point can be determined in several ways, two of which are to:

a) closely monitor the slope of our MTTF curve plotted versus time spent on debugging and to watch for a sharp increase as it nears a vertical asymptote (Fig. 4, 4C) or,

b) since the system MTTF is directly related to the number of software errors remaining \((E_T - E_c)\), look for the point in time when model predictions of \(\hat{E}_T\) applied to the system in question approach a horizontal asymptote. This aspect is depicted in Fig. 9 by the shape of expected value and 2σ confidence bands around \(E_T\) as testing nears completion.

6.3 Model Sensitivity

The un-normalized equation used to calculate system MTTF is

\[
MTTF = \frac{1}{K(E_T - E_c(T))} \tag{27}
\]
and the derivative with respect to \( \hat{K} \) is

\[
\frac{\partial \text{MTTF}}{\partial \hat{K}} = \frac{-1}{K^2(E_T - E_c(\tau)))} = -\frac{\text{MTTF}}{K}
\]

This indicates that variations in \( \hat{K} \) result in a quasi-linear effect on MTTF. For small variations in \( \hat{K} \) this variation can be linearized as shown in Fig. 10 where the percentage change in \( \hat{K} \) is plotted vs resulting percentage change in MTTF.

Slight variations of \( \hat{E}_T \), however, result in dramatic changes in MTTF, especially as the former approaches the value of \( E_c(\tau) \). Calculating \( \frac{\partial \text{MTTF}}{\partial E_T} \) yields:

\[
\frac{\partial \text{MTTF}}{\partial E_T} = \frac{-1}{K(E_T - E_c(\tau))^2} = -\frac{\text{MTTF}}{(E_T - E_c(\tau))}
\]

Thus, as \( E_c(\tau) \to E_T \), the sensitivity becomes quite large. It can be seen from Fig. 11 that a 10% change in \( \hat{E}_T \) results in 70% variation of system MTTF. This factor alone emphasizes the requirement of high-quality failure data if accurate model predictions are to ensue.
FIGURE 9

TOTAL ERRORS vs TESTING TIME

SYSTEM 3
FIGURE 11
EFFECT ON MTTF BY VARYING $E_0$

% CHANGE IN $E_0$

LEGEND
* SYSTEM #2
- SYSTEM #3
7.0 Conclusion

The primary aim of this thesis was to present a model of software system parameter estimation, to subsequently compare predicted system characteristics with actual field data, and to guide the software manager in the practical application thereof to software systems under his development.

Table 6 summarizes model parameters by system and method of calculation. It can be seen that MTTF prediction accuracy varied from 2% to 67%, depending on the system, with an overall average of 45%.

Surprisingly enough, there was less than ½% difference in prediction accuracy between the three different methods of model parameter calculation: Method of Moments, Least Squares, and Maximum Likelihood Estimates. Although most statisticians consider MLE to yield superior results to other methods, the findings of this thesis would indicate that parameter estimation using Least Squares is preferable for the software manager due to the less cumbersome calculations required.

Model parameter sensitivity was explored and indicated that a variation in \( \hat{K} \) resulted in a linear variation in estimated MTTF. This was not the case with variations in total system estimated errors (\( \hat{E}_T \)), however, as minute variations in the latter resulted in drastic changes in predicted MTTF.

Once model parameters and system MTTF have been calculated, it is of great interest to the software manager to use these calculations to determine/predict the optimal release date for the system under development. Sections 2.3 and 6.2 suggest several ways of doing so.
<table>
<thead>
<tr>
<th>Syst. No.</th>
<th>K×10⁻⁵</th>
<th>$E_T$</th>
<th>$E_G$</th>
<th>$MTTF_T$</th>
<th>$C$</th>
<th>$MTTF_P$ (C/MTF₀)</th>
<th>$MTTF_E$</th>
<th>$%A₁$</th>
<th>$MTTF_E$ (MCM)</th>
<th>$%A₂$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>McN</td>
<td>5.515</td>
<td>142</td>
<td>136</td>
<td>.223</td>
<td>15.1</td>
<td>4.9</td>
<td>14.6</td>
<td>66.6</td>
<td>99.7</td>
</tr>
<tr>
<td></td>
<td>LS</td>
<td>5.515</td>
<td>131</td>
<td>132</td>
<td>.223</td>
<td>4.9</td>
<td>5.2</td>
<td>66.6</td>
<td>64.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLE</td>
<td>5.515</td>
<td>152</td>
<td>136</td>
<td>.223</td>
<td>4.9</td>
<td>5.2</td>
<td>66.6</td>
<td>64.2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3.999</td>
<td>62</td>
<td>94</td>
<td>.933</td>
<td>13.6</td>
<td>12.7</td>
<td>31.4</td>
<td>59.5</td>
<td>43.5</td>
<td>38.5</td>
</tr>
<tr>
<td></td>
<td>3.999</td>
<td>62</td>
<td>94</td>
<td>.933</td>
<td>12.7</td>
<td>12.4</td>
<td>31.4</td>
<td>59.5</td>
<td>43.5</td>
<td>38.5</td>
</tr>
<tr>
<td></td>
<td>3.999</td>
<td>62</td>
<td>94</td>
<td>.933</td>
<td>12.7</td>
<td>12.4</td>
<td>31.4</td>
<td>59.5</td>
<td>43.5</td>
<td>38.5</td>
</tr>
<tr>
<td>3</td>
<td>11.60</td>
<td>40</td>
<td>38</td>
<td>1.14</td>
<td>13.2</td>
<td>15.0</td>
<td>30.3</td>
<td>50.4</td>
<td>30.4</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>11.60</td>
<td>40</td>
<td>38</td>
<td>1.14</td>
<td>15.0</td>
<td>15.0</td>
<td>30.3</td>
<td>50.4</td>
<td>30.4</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>11.60</td>
<td>40</td>
<td>38</td>
<td>1.14</td>
<td>15.0</td>
<td>15.0</td>
<td>30.3</td>
<td>50.4</td>
<td>30.4</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>19.88</td>
<td>55</td>
<td>53</td>
<td>.722</td>
<td>9.63</td>
<td>9.17</td>
<td>5.1</td>
<td>14.5</td>
<td>58.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22.49</td>
<td>55</td>
<td>53</td>
<td>.888</td>
<td>8.98</td>
<td>2.0</td>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LEGEND**

- $MTTF_T$ - Predicted MTBF based on testing time
- $MTTF₀$ - Predicted operational (field) MTBF
- $MTTF_E$ - Actual operational (field) MTBF
- $\%A₁$ - % difference between actual and predicted MTBF (exponential model)
- $\%A₂$ - % difference between actual and predicted MTBF (MCM'S model)
- McN - Method of Moments
- LS - Least squares
- MLE - Maximum Likelihood Estimate
REFERENCES


REFERENCES


REFERENCES
