2006

Statistical modeling and assessment of software reliability

Louis Richard Camara
University of South Florida

Statistical Modeling and Assessment of Software Reliability

by

Louis Richard Camara

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosopy
Department of Mathematics
College of Arts and Sciences
University of South Florida

Major Professor: Chris P. Tsokos, Ph.D.
Kandethody Ramachandran, Ph.D.
O. Geoffrey Okogbaa, Ph.D.
Marcus McWaters, Ph.D.

Date of Approval:
June 21, 2006

Keywords: Logistic regression, Software Failure Data, Cumulative Number of Software Faults, Time Between Failure, Bayesian linear regression

Dedication

This dissertation is dedicated to my parents Victor Athanase Camara and Lucie Camara, to my wife Teresa, to my brothers and sisters: Antoinnette, Lucienne, Jerome, Vincent, Michelle, Therese, and Maurice, to my uncle Walwin and his family, to my uncle Francis and his family, to my late friend Clayt William Cuppy, to my nieces and nephews, to stepsons Deron and Devin, for their love, patience, understanding and support.

Acknowledgments

I would like to express my deep appreciation and indebtedness to my dissertation advisors, Dr. Chris P. Tsokos, and the late Dr. A. N. V. Rao, for their understanding, guidance, encouragement, support and advice during my graduate years. I wish to thank Dr. Marcus McWaters and to the entire faculty and staff of the University of South Florida's Mathematics Department for their encouragement and endless assistance. I wish to thank Dr. Dewey Rundus for his discussions. I wish to thank Dr. David Kelpard for his endless support.

Table of Contents

List of Tables

List of Figures

Statistical Modeling and Assessment of Software Reliability

Louis Richard Camara

ABSTRACT

The present study is concerned with developing some statistical models to evaluate and analyze software reliability.

We have developed the analytical structure of the logistic model to be used for testing and evaluating the reliability of a software package. The proposed model has been shown to be useful in the testing and debugging stages of the developmental process of a software package. It is important that prior to releasing a software package to marketing that we have achieved a target reliability with an acceptable degree of confidence.

The proposed model has been evaluated and compared with several existing statistical models that are commonly used. Real software failure data was used for the comparison of the proposed logistic model with the others. The proposed model gives better results or it is equally effective.

The logistic model was also used to model the mean time between failure of software packages. Real failure data was used to illustrate the usefulness of the proposed statistical procedures.

Using the logistic model to characterize software failures we proceed to develop Bayesian analysis of the subject model. This modeling was based on two different difference equations whose parameters were estimated with Bayesian regressions subject to specific prior and mean square loss function.

Chapter 1

Software Reliability

1.1 Introduction

We are depending more and more on computers, which are run by software systems that are built to be larger and larger and thus becoming more and more complex making it almost impossible for the software developers to thoroughly test them, remove all the software faults, release and warrant a highly reliable product for use.

Software systems, specially their reliability impact us directly since almost everything nowadays is run by computers: our finances through our Banks and automatic tellers , air traffic control, hotel reservations, interest due on a loan, incorrect billing, incorrect mail, security through Electronic warfare systems failing to identify a real threat or Electronic warfare systems identifying a threat when there is none, medication through incorrect doses and prescriptions, radiation therapy machines killing patients instead of helping them, and much more [33, 44].

Because of the complexity of the problem, the definition of Software Reliability varies from author to author. Some of the most commonly accepted definitions of software reliability, reported in [44], are the following:

The Institute of Electrical and Electronic Engineers (IEEE) defines software reliability as : *"the probability that software will not cause a system failure for a specified time under specified conditions. The probability is a function of the inputs to, and use of, the system as well as function of the existence of faults in the software. The inputs to the system determine whether existing faults,if any, are encountered."*

John Musa of AT and T Bell Laboratories defines software reliability as: *"the probability that a given software system operates from some time period without software error, on the machine for which it was designed, given that it is used within design limits."*

Dr. Martin Shooman of the New York Polytechnical University defines software reliability as: *"the*

probability of failure free operation of a computer program for a specified environment."
Since no one definition of software reliability nor one method of assessing or predicting software reliability is accepted as standard [44], a unique definition of software reliability and a standardized procedure for measuring the reliability of a software would be a major contribution to the industry.

The definition of software reliability that we will be using is the following:
"*Software reliability is the probability that a given software system in a given environment will operate correctly for a specified period of time.*", [33].

Software reliability is one of the most important topics in the subject area. Neufelder, [44], gives the following four major reasons why software reliability has become a very important issue in the last decade: Systems are becoming more software intensive than hardware intensive, many software-intensive systems are safety critical or mission critical or failure is extremely costly financially, customers are requiring more reliable softwares, software failures are not being tolerated by end users or by clients of end users, and the cost of developing software is increasing.

Some of the relevant papers in the subject area are: $[1, 5-18, 24, 26, 33, 44, 45]$, amomg others.

In the present study we will address various aspects of the subject area including Bayesian approach to software reliability.
Given below is a brief introduction on the problems we studied in each of the Chapters that encompass this dissertation.

1.2 Parameters Estimation for Software Reliability using Logistic Regression

In Chapter 2, we develop a realistic software reliability growth model that uses the mean square error as a criteria to providing a decision-making rule as to when to stop the testing and debugging phase and release the software or when to continue with the debugging process.
The proposed model allow us to estimate the proportion of the software mistakes are left in the system. Our proposed model fits very well the cumulative S-shaped software failure behavior, and offers the following advantages of being simple to implement and not assuming any prior distribution. After developing the necessary theory, we illustrate the effectiveness of our proposed model with other models in the subject area. Furthermore, seven real world experimental data have been

used not only to illustrate it applicability but also to use these real data in the comparison of the results that have been obtained using communally used statistical models.

1.3 An Early Estimation method for Software Reliability Assessment

In Chapter 3, using the main feature of our proposed Model in Chapter 2- its inflection point, we propose a software reliability growth model, which relatively early in the testing and debugging phase, provides accurate parameters estimation, gives a very good failure behavior prediction and enable software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software. Two real world experimental data previously analyzed in Chapter 2, have been used to compare our proposed Early Estimation Logistic Model effectiveness with several pre-existing models in the subject area.

1.4 Reliability Growth Model for Software Reliability Analysis

In Chapter 4, an accurate estimation of the $MTBF$ to predict the failure times of a given system is crucial when it comes to planning corrective strategies. Most of the existing models, [47], are based on the $NHPP$ with intensity function characterized by the inverse power law process. In Chapter 4, we develop a logistic model to forecast the mean time between failures of a software package, after the last correction has been implemented, to characterize the software reliability of the given package. After developing the model, two real world experimental data are used not only to illustrate it applicability but also to compare its effectiveness with some pre-existing models used in the subject area.

1.5 Discrete Logistic Models using Bayesian Procedures for Software Reliability

In Chapter 5, we develop two software reliability growth models that yield accurate parameter estimates, using Bayesian procedures. The models are based on two difference equations proposed by Morishita and Hirota and used by Satoh and Yamada, [1], each of which is a discrete analog of the logistic curve model. As far as we can determine this is the first time that software reliability growth models Bayesian in nature are derived from two difference equations discrete analogs of the logistic curve model proposed by Morishita and Hirota. After developing the models, two real world

experimental data are used not only to illustrate it applicability but also to compare its effectiveness with some pre-existing models used in the subject area.

1.6 Future Research

Finally, in Chapter 6, the last Chapter of this study, we extend our research findings in the present study to more realistic and useful analytical extensions.

In Chapter 2, we have developed of a simple, realistic, and easy to implement software reliability growth model that provides a decision rule as of when to stop the testing and debugging phase and release the software for use, for S-shaped cumulative software faults. We can use a Bayesian or Quasi-Bayesian procedure in the present development of the proposed Model.

Using the main feature of our proposed Model in Chapter 2- its inflection point, in Chapter 3, we have proposed an effective method for estimating the number of faults in the software, at an early stage of the testing and debugging phase. Our early estimation of the number of fault in the software enable the software developers to plan the software development process, manage their resources by avoiding cost due to over testing, make a software with higher reliability and decide when to ship it for use. We need to develop a cost reduction analysis associated with our Early Estimation proposed Model.

In Chapter 4, assuming a logistic model, we develop a procedure for predicting the mean time between failure, after the last correction, of a software package creating a connection between predicting the $MTBF$ and counting the cumulative failures experienced up to a given time. For the prediction of the $MTBF$, one possible extension is to introduce a Bayesian or Quasi-Bayesian procedure in the development of the proposed Model.

In Chapter 5, we have develop theoretical structure and given parameters estimation of two model using Bayesian procedures. Illustrating the Bayesian approach to reliability is very useful modeling in understanding the final evaluation of a software package. One of the key difficulties is identifying and justifying the choice of the prior. Thus, we propose to develop an empirical Bayes approach to software reliability and thus by-passing having to assume the choice of the prior. Furthermore, we extend to address the same problem from a nonparametric point of view by utilizing Kernel Density estimation procedures to characterize the behavior of the prior. We also believe that formulating nonparametric software reliability models using the kernel density approach to characterize software

failure would be of significant importance in cases where a classical distribution cannot be identify to statistically fit the software failures.

Chapter 2

Estimation of parameters for Software Reliability using Logistic Regression

2.1 Introduction

Embedded Computer Systems are growing immensely in size and in complexity, and especially in areas of applications. Every embedded computer system is characterized by its reliability, performance, maintainability and cost, among others. It has been rapidly being established in the literature [33, 44]among others, that reliability is the most important aspect of software quality, since any failure can be catastrophic. Consequently, the reliability of software must be accurately assessed during the testing and debugging phase before its actual release for use.

A software fault is defined as an unacceptable departure of program operation caused by a software fault remaining in the system, [1]. Software reliability is the probability that a software fault which causes deviation from the required output by more than the specified tolerances, in a specified environment, does not occur during a specified exposure period.

The main assumption that we make here is that once a software fault is found it is corrected for good and its correction does not introduce any new software faults. As a result, the reliability of the software increases, and we refer to such a model as the reliability growth model.
Finally, when assessing the reliability of a software, during the debugging and testing phase, any failures other then software faults could prevent its effectiveness.

The objective of this chapter is to develop a realistic software reliability growth model that uses the mean square error as a criteria to providing a decision-making rule as to when to stop the testing and debugging phase and release the software or when to continue with the debugging process.
The proposed model will allow us to estimate the proportion of the software mistakes are left in the system. That is, we claim that almost all the software faults have been found and corrected before

concluding their testing and debugging phase, with an acceptable level of confidence. Our proposed model fits very well the cumulative S-shaped software faults curves, and offers the following advantages:

(i) gives a very good failure behavior prediction

(ii) provides accurate parameters estimation

(iii) it is simple to implement

(iV) it does not assume any prior distribution

The remaining of this Chapter is organized as follows: In Section 2.2 we present an overview of a few models used for comparison: The conventional model reported by Satoh and Yamada [1], Satoh and Yamada's software reliability growth models that we will call S-Y models, [1], Ohba's models, and Huang-Kuo-Chen-Lo-Lyu's models. In Section 2.3 we present our proposed Logistic Like Model. Comparison of our proposed Logistic Like Model that we will refer to as LR-Model with several pre-existing models is made using data from seven real world problems in Section 2.4. Finally, our conclusions and recommendations are given in Section 2.5.

2.1.1 Statistical Abbreviations and Notations

For convenience, in the present statistical study, we shall introduce the following abbreviations and notations:

- S-Y Models will stands for Satoh and Yamada's Models
- $NHPP$ means Non-Homogeneous Poisson Process
- F-S Model will represent the Forman and Singpurwalla's Model
- C-Model will identify the Conventional Model
- H. G. D. M will be used for the Hyper Geometric Distribution Model
- J-M Model will stand for the Jelinski-Moranda Model.
- DDays means Debugging Days
- DTimesimes is used for Debugging Times
- CDT stands for Cumulative Debugging Times
- DFaults is used for Detected Faults
- CDFaults represents the Cumulated detected Faults
- MSE will stand for mean square error

- $ACNOF$ will represent the assumed cumulative number of failures.
- our proposed model, LR-Model
- G-O Model: Goel O. Model
- Cum. Faults: Cumulative number of faults
- Cum. Time : Cumulative Time

2.2 Preliminaries

In the present section we shall briefly introduce some methodology that is crucial in the present study for the convenience of the reader.

A logistic model is described by the following differential system:

$$\frac{dL(t)}{dt} = \frac{\alpha}{k} L(t)(k - L(t)),\ L(0) = \frac{K}{1+m},\ m \gg k$$

where L(t) is the cumulative number of software failures that have occurred up to testing time t and α ($\alpha > 0$) and k ($k > 0$) are unknown parameters to be statistically estimated from the given data. The parameter k is the total number of software faults in the software before the debugging phase. The solution of the above differential equation is given by:

$$L(t) = \frac{K}{1 + m\exp(-\alpha t)}$$

where m is the constant of integration.

A few models have been proposed for obtaining the behavior of $L(t)$ by various estimation procedures of its parameters α, k, and m. Since the Maximum Likelihood estimation is difficult to obtain, as shown by Satoh and Yamada [1], they use the least-squares estimation which gives the global optimum and is easy to implement. Satoh and Yamada developed two software reliability growth models (S-Y Models) and they compare their effectiveness with the conventional Model, [1]. We shall briefly discuss these models below since they will be compared with the proposed model developed in this study.

2.2.1 The Conventional Model

The differential equation,

$$\frac{dL(t)}{dt} = \frac{\alpha}{k} L(t)(k - L(t))$$

can be written as

$$\frac{1}{L(t)} \frac{dL(t)}{dt} = \alpha - \frac{\alpha}{k} L(t) \tag{2. 1}$$

Let

$$\begin{cases} t_n &= n\delta \\ L_n &= L(n\delta) \\ Y_n &= \frac{L_{n+1} - L_{n-1}}{2\delta L_n} \end{cases}$$

where δ is a small constant time interval and n a positive integer taking the values $1, 2, 3, 4, ...$ Discretizing equation 2. 1 as a difference equation; that is,

$$\frac{1}{L_n} \frac{L_{n+1} - L_{n-1}}{2\delta L_n} = -\frac{\alpha}{k} L_n + \alpha$$

and using the notation above, we can obtain the following regression line,

$$Y_n = A + BL_n$$

with

$$\begin{cases} A &= \alpha \\ B &= -\frac{\alpha}{k} \end{cases}$$

Estimating the regression coefficients $\hat{A}$ and $\hat{B}$ by the usual least squares method, we have

$$\begin{cases} \hat{k} &= -\frac{\hat{A}}{\hat{B}} \\ \hat{\alpha} &= \hat{A} \\ \hat{m} &= \frac{\sum_{n=1}^{N} (\hat{k} - L_n)}{\sum_{n=1}^{N} (L_n (\exp(-\hat{\alpha} t_n))} \end{cases}$$

Thus, for a given set of data one can obtain estimates of the three parameters and $\widehat{L(t)}$.

2.2.2 Satoh and Yamada Models

Satoh and Yamada described two software reliability growth models based on two discrete analogs of a logistic curve model proposed respectively by Morishita and Hirota, [1]. Satoh and Yamada reported that the difference equations have exact solutions, conserve the characteristics and tend to a differential equation on which the logistic curve model is defined when the time interval tends to zero. They have shown that the exact solutions of Morishita and Hirota difference equations also converges to the exact solution of the differential equation when the time interval tends to zero. According to Satoh and Yamada, their two software reliability growth models yield accurate parameter estimates in spite of a small amount of input data in an actual software testing making it possible to predict in the early development phase when software can be ready to be released, [1].

Discrete Logistic Curve Model: Morishita's difference equation

Equation 2. 1 is dicretized as follows:

$$L_{n+1} - L_n = \frac{\delta\alpha}{k} L_{n+1}(k - L_n) \tag{2. 2}$$

and is referred to as Morishita's difference equation [1]. The solution of the above difference equation is given by

$$L_n = \frac{k}{1 + m(1 - \delta\alpha)^{\frac{t_n}{\delta}}}$$
$$t_n = n\delta$$

From Equation 2. 2, we obtain

$$\frac{L_{n+1}}{L_n} - 1 = \delta\alpha \frac{L_{n+1}}{L_n} - \frac{\delta\alpha}{k} L_{n+1} \tag{2. 3}$$

$$\frac{L_{n+1}}{L_n}(1 - \delta\alpha) = 1 - \frac{\delta\alpha}{k} L_{n+1} \tag{2. 4}$$

$$\frac{L_{n+1}}{L_n}(1 - \delta\alpha) = 1 - \frac{\delta\alpha}{k} L_{n+1} \tag{2. 5}$$

$$\frac{L_{n+1}}{L_n} = \frac{-\delta\alpha}{k(1-\delta\alpha)}L_{n+1} + \frac{1}{1-\delta\alpha} \tag{2. 6}$$

and

$$y_n = BL_{n+1} + A$$

where,

$$\begin{cases} y_n = \frac{L_{n+1}}{L_n} \\ B = \frac{-\delta\alpha}{k(1-\delta\alpha)} \\ A = \frac{1}{1-\delta\alpha}. \end{cases} \tag{2. 7}$$

Using the Least square regression, the estimates $\hat{A}$ and $\hat{B}$ are obtained as follows:

$$\begin{cases} \hat{k} = \frac{1-\hat{A}}{\hat{B}} \\ \delta\hat{\alpha_{dm}} = 1 - \frac{1}{\hat{A}} \\ \hat{m} = \frac{\sum_{n=1}^{N}(\hat{k}-L_n)}{\sum_{n=1}^{N}(L_n(1-\delta\hat{\alpha_{dm}})^n)} \\ \widehat{\alpha_c} = -\frac{1}{\delta}\log(1-\delta\hat{\alpha_{dm}}) \end{cases}$$

Thus, using their results one can obtain and estimate of $\widehat{L(t)}$.

Discrete Logistic Curve Model: Hirota's difference equation

Hirota discretizes Equation 2. 1 as follows [1]:

$$L_{n+1} - L_n = \frac{\delta\alpha}{k}L_n(k - L_{n+1})$$

The solution of the above difference equation is given by

$$L_n = \frac{k}{1 + m(\frac{1}{1+\delta\alpha})^{\frac{t_n}{\delta}}}$$

where $t_n = n\delta$

Proceeding as in Equation 2. 2, we can obtain the following regression equation:

$$\frac{L_{n+1}}{L_n} = \frac{-\delta\alpha}{k} L_{n+1} + (\delta\alpha + 1)$$

and

$$y_n = BL_{n+1} + A$$

where:

$$\begin{cases} y_n &= \frac{L_{n+1}}{L_n} \\ B &= \frac{-\delta\alpha}{K} \\ A &= \delta\alpha + 1. \end{cases}$$

The regression estimates $\hat{A}$ and $\hat{B}$, are obtained, as follows:

$$\begin{cases} \hat{k} &= \frac{1-\hat{A}}{\hat{B}} \\ \delta\hat{\alpha_{dh}} &= \hat{A} - 1 \\ \hat{m} &= \frac{\sum_{n=1}^{N}(\hat{k}-L_n)}{\sum_{n=1}^{N}\left(L_n(\frac{1}{1+\delta\hat{\alpha_{dh}}})^n\right)} \\ \widehat{\alpha}_c &= \frac{1}{\delta}\log(1+\delta\hat{\alpha_{dh}}) \end{cases}$$

For a given set of data one can obtain an estimate of $L(t)$ using this model.

Satoh and Yamada's conclusions

According to Satoh and Yamada, [1], their two software reliability growth models yield accurate parameter estimates even with a small amounts of data. Their models give the same parameters estimates; Satoh and Yamada reported that although the conventional model uses a discrete equation as a regression equation, the model itself is a continuous time model, so it includes errors generated by discretization, however their models do not have this problem because they are themselves discrete

models and that they can analyze the software reliability without a continuous time model.
Satoh and Yamada also reported the following three advantages that their two discrete models have over the conventional model [1].

(i) The parameter estimation in the discrete logistic curve models reproduced the values of the parameters very accurately or perfectly, even when small data that do not include the inflection points are given. When the exact solution is used as the input data, the conventional model provides inaccurate parameter estimates with data that do not include the inflection point; accuracy were not so good even with sufficient data points.

(ii) The discrete models are independent of time scale:
The discrete models do not use the time scale in the regression equation. The same parameters estimates are obtained whatever value of the time scale we choose. When the conventional model is used we have to choose the time scale carefully, because the time scale needs to be used in the regression equation. As a result, the estimates depend on the choice of the time scale.

(iii) The discrete logistic curve models enable us to accurately estimate parameters in the early testing phase with real data.
The parameter estimates of the conventional model vary with the number of data points. The discrete models provide stable values of parameter estimates for various number of data points. This characteristic is very important for software reliability growth models.

2.2.3 Mitsuru Ohba's models

Ohba, in his research *Software reliability analysis models*, discusses improvement to conventional software reliability analysis models by making more realistic assumptions. He claimed that in contrast to the exponential growth in software reliability, S-shaped software reliability growth is more often observed in real systems [10].
Ohba reported that although it is quite practical to use the Gompertz model and the logistic curve, it is sometimes dangerous since these models may lead to a more optimistic assessment than other models.
According to Ohba results, there are many reasons why observed software reliability growth curves often become S-shaped. The S-shaped software reliability growth curves are typically caused by the definition of errors in testing a given system. He used the delayed S-shaped growth model, the

inflection S-shaped model, and the hypexponential model, [10].

2.2.4 Huang, Kuo, Chen, Lo, and Lyu's models

In their studies *Effort Index Based Software Reliability Growth Models and Performance Assessment*, Huang, Kuo, and Lyu after establishing that the logistic testing effort function is practically suitable for modeling software reliability growth and provides a reasonable description of resource consumption, developed a software reliability growth model (SRGM) with logistic testing-effort function and claimed that their SRGM model with logistic testing-effort function which is simple and compact, estimates the number of initial faults better than previous studies; They study the subject model, under perfect and imperfect debugging environments, using other testing effort functions.

SRGM with logistic testing-effort function

Huang et al. [5, 6, 7, 9] reported that usually, the test-effort during experimental phase and the time-dependent behavior of development effort in the software testing process is characterized by a weibull-type consumption curve. They used a logistic testing-effort function instead of the weibull-type testing effort consumption function as the test effort patterns during the software development process, since the weibull-type curve may not be suitable for modeling the test effort consumption curve.

The cumulative testing-effort consumption of logistic testing-effort function on the interval (0,t] is defined by

$$W(t) = \frac{N}{1 + A\exp[-\alpha t]} \tag{2. 8}$$

where N is the total amount of testing effort to be eventually consumed, α the consumption rate of testing-effort expenditures, and A a constant. As a result the current testing-effort expenditures at testing time t is given by

$$w(t) = \frac{NA\alpha\exp[-\alpha t]}{(1 + A\exp[-\alpha t])^2} \tag{2. 9}$$

where $w(t)$ is a smooth bell-shaped function with a left-tailed side, and reaches it maximum value at time

$$t_{max} = \frac{lnA}{\alpha} \tag{2. 10}$$

This SRGM model is based on the following assumtions:

(i) The fault removal process follows the Non-Homogeneous Poisson Process $(NHPP)$

(ii) The software system is subject to failures at random times caused by faults remaining in the system.

(iii) The mean number of faults detected in the time interval (t, t+ dt]by the current test-effort is proportional to the mean number of remaining faults in the system.

(iV) The proportionality is constant over time.

(V) The consumption curve of testing effort is modeled by a logistic testing-effort function.

(Vi) Each time a failure occurs, the fault that caused it is immediately removed, and no new faults are introduced.

Thus, the differential equation is given by

$$\frac{dm(t)}{dt} = w(t)r[a - m(t)], a > 0, 0 < r < 1 \tag{2. 11}$$

describes analytically the testing-based-effort.

The solution of the above differential equation, under the boundary condition
$m(0) = 0$ is

$$m(t) = a(1 - e^{-r(W(t)-W(0))}) \tag{2. 12}$$

where $m(t)$ is the expected number of faults detected on the interval (0, t], w(t) the current testing-effort consumption at time t, a the expected number of initial faults, and r the error detection rate per unit testing-effort at testing time t that satisfies $r > 0$.

Yamada S-shaped model with logistic testing-effort function

Huang et al. reported, [5], that the Delayed S-shaped SRGM model proposed by Yamada et al. was a simple modification of the $NHPP$ to get an S-shaped growth curve for the cumulative number of failures detected. Since the testing phase contains a fault detection process and a fault isolation process, Huang et al. developed the following relationship between $m(t)$ and $w(t)$ to extend Yamada's S-shaped software reliability model: [5]

$$\frac{df(t)}{dt}\frac{1}{w(t)} = \omega[a - f(t)] \tag{2. 13}$$

and

$$\frac{dg(t)}{dt}\frac{1}{w(t)} = \epsilon[f(t) - g(t)] \tag{2. 14}$$

where $f(t)$ is the cumulative number of failures detected up to time t and $g(t)$ the cumulative number of failures isolated up to time t. Under boundary conditions $f(0) = g(0) = 0$ the solutions of the above differential equations are respectively,

$$f(t) = a(1 - e^{-\omega(W(t)-W(0))}) = a(1 - e^{-\omega W^*(t)}) \tag{2. 15}$$

and

$$g(t) = a[1 - \frac{1}{\omega - \epsilon}\left(\omega e^{-\epsilon W^*(t)} - \epsilon e^{-\omega W^*(t)}\right)] \tag{2. 16}$$

where ω and ϵ are the failure detection rate and the failure isolation rate, respectively.
Assuming fault detection rate parameter $\omega \cong \epsilon$, the NHPP model with a Delayed S-shaped growth curve of detected software faults is given by

$$m(t) = a(1 - (1 + \Psi(W(t) - W(0))e^{-\Psi(W(t)-W(0))}) = a(1 - (1 + \Psi W^*(t)e^{-\Psi W^*(t)}) \tag{2. 17}$$

where Ψ is the fault detection rate per unit testing-effort at testing time t satisfying $\Psi > 0$.

2.3 Development of The proposed Model

The objective of this section is to develop a software reliability growth model that

(i) does not assume any prior distribution

(ii) is free of convergence assumption

(iii) is simple

(iV) fits very well the cumulative number of software faults found and corrected

(V) estimates the remaining number of software faults in the software package

(Vi) yields accurate parameter estimates

Thus, allowing software developers to predict or decide when a software is ready to be released. Our proposed model which is not subject to errors generated by discretization, provides accurate parameter estimates when sufficient data points are available.

2.3.1 The Logistic Regression Model

The proposed logistic model is suitable for software reliability assessment. While in linear regression models the outcome variable is assumed to be continuous, in logistic regression models the outcome variable is binary or dichotomous. It is often the case that the outcome variable is discrete. In any regression problem, the key quantity when it exists is the conditional mean,
$E(y|x) = \beta_0 + \beta_1 x$, where y denotes the outcome variable and x denotes a value of the independent variable, which can take on any value, [4]. With dichotomous data the conditional mean must be greater than or equal to zero and less than or equal to one, $0 \leq [E(y|x)] \leq 1$. The change in $E(y|x)$ per-unit change in x becomes progressively smaller as the conditional mean gets closer to zero or one, and then, the curve is said to be S-shaped. When the logistic distribution is used, the logistic regression model we use is

$$E(y|x) = \pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

with a dichotomous outcome variable $y = \pi(x) + \epsilon$, the quantity ϵ is called the error that expresses an observation's deviation from the conditional mean [4]. In logistic regression the quantity ϵ may assume one of two possible values. If $y = 1$ then $\epsilon = 1 - \pi(x)$ with probability $\pi(x)$, and if $y = 0$

then $\epsilon = -\pi(x)$ with probability $1 - \pi(x)$. Thus, ϵ has a distribution with mean zero and variance equal to $\pi(x)[1 - \pi(x)]$. That is, the conditional distribution of the outcome variable follows a binomial distribution with probability given by the conditional mean, $\pi(x)$, [4].

In Summary, in regression analysis when the outcome variable is dichotomous, [4], we have

i) The conditional mean of the regression equation must be formulated to be bounded between zero and one. The above logistic regression model $\pi(x)$ satisfies this constraint.

ii) The binomial, not the normal, distribution describes the distribution of the errors and will be the statistical distribution upon which the analysis is based.

These characterizing properties of the logistic model makes it useful as a growth curve model.

2.3.2 Model Description

Let L(t) be the cumulative amount of software faults found and corrected in the debugging phase during exposure time interval [0,t]. The parameter k ($k > 0$) represents the initial total number of software faults in the software prior to the debugging phase as in [1].

The logistic curve model is described by the differential equation

$$\frac{dL(t)}{dt} = \frac{\alpha}{k} L(t)(k - L(t)), \qquad t \geq 0 \tag{2. 18}$$

where α ($\alpha > 0$) and k ($k > 0$) are constant parameters to be estimated by regression analysis [1].

Let

$$P(t) = \frac{L(t)}{k}$$

Then, from Equation 4. 8, we have

$$\frac{dP(t)}{dt} = \alpha P(t)(1 - P(t)), \qquad t \geq 0, \qquad P(0) = \frac{1}{1+m}, \qquad m \gg k$$

where

$$P(t) = \frac{L(t)}{k}$$

is the cumulative fraction number of software failures found and corrected in the debugging phase during exposure time interval [0,t].

The solution of the above differential equation is

$$P(t) = \frac{1}{1 + me^{-\alpha t}} \tag{2. 19}$$

where m is the constant of integration. Note that the graph of $P(t)$ is S-shaped and $0 \leq P(t) \leq 1$. A software fault found and corrected at each instant is a binary outcome variable. We assume that the binomial distribution describes the distribution of the errors and will be the statistical distribution upon which the analysis is based, [4].

Fewer software faults are found early in the testing and long into the testing and debugging phase when most of the faults are found and corrected and only a few faults are left in the software. For such cases we propose a Logistic Like Model to estimate the parameters k, m and α of $P(t)$.

2.3.3 Parameters Estimation

Our proposed model depends on an accurate estimate of the total number of faults k in a given software. Let t_i be the random detection time of the ith software fault . Let N denote the total number of available software faults found and corrected up to time t_N. We assume that once a software fault is found it is corrected for good. Using the N available data points collected during the testing and debugging phase and increasing N by 1 after each sequential logistic regression run, we run a series of logistic regressions models of the form

$$\widehat{P(t)} = \frac{e^{\beta_0 + \beta_1 t}}{1 + e^{\beta_0 + \beta_1 t}}$$

where β_0 and β_1 are the parameters of the logistic regression model, to estimate the true cumulative percentage number of software faults found and corrected in a given software up to time t of its testing and debugging process

$$P(t) = \frac{L(t)}{k} = \frac{1}{1 + me^{-\alpha t}}$$

That is, starting with the N available data value, and assuming that all the software faults were found during the testing and debugging phase, we run a logistic regression to obtain an estimate

$$\widehat{P(t)}_N = \frac{e^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}{1 + e^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}$$

of

$$P(t)_N = \frac{L(t)}{k} = \frac{1}{1 + me^{-\alpha t}}$$

Now, using

$$\widehat{L(t)}_N = (N)(\widehat{P(t)}_N) = \frac{Ne^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}{1 + e^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}$$

where $\hat{\beta}_{0N}$ and $\hat{\beta}_{1N}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, we obtain the predicted cumulative number of software failure assuming that there were N faults in the software which estimates the true cumulative number of software faults $L(t)$ in the given software. Now evaluating

$$\widehat{L(t)}_N = (N)(\widehat{P(t)}_N) = \frac{Ne^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}{1 + e^{\hat{\beta}_{0N} + \hat{\beta}_{1N} t}}$$

at each t_i, $i = 1, 2, ..., N$, the predicted cumulative failure behavior is obtained, for this run, assuming that there were N faults in the software. Finally, we can estimate the true mean square error MSE_N of the actual and predicted cumulative number of software faults by calculating

$$\widehat{MSE}_N = \frac{\sum_{i=1}^{N} (\widehat{L(t_i)}_N - L(t_i))^2}{N}$$

What if there is one more fault remaining in the software? For our second run, assuming that there are $N + 1$ software faults in the software of which N are found and corrected, we repeat the above procedure to obtain an estimate

$$\widehat{P(t)}_{N+1} = \frac{e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}{1 + e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}$$

of

$$P(t)_{N+1} = \frac{L(t)}{k} = \frac{1}{1 + me^{-\alpha t}}$$

and

$$\widehat{L(t)}_{N+1} = (N+1)(\widehat{P(t)}_{N+1}) = \frac{(N+1)e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}{1 + e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}$$

where $\hat{\beta}_{0N+1}$ and $\hat{\beta}_{1N+1}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1 evaluating

$$\widehat{L(t)}_{N+1} = (N+1)(\widehat{P(t)}_{N+1}) = \frac{(N+1)e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}{1 + e^{\hat{\beta}_{0N+1} + \hat{\beta}_{1N+1} t}}$$

at each t_i, $i = 1, 2, ..., N$, the predicted cumulative failure behavior is obtained, for the second run, assuming that there were $N + 1$ faults in the software. Then, we can estimate the true mean square error MSE_{N+1} of the actual and predicted cumulative number of software faults by calculating

$$\widehat{MSE}_{N+1} = \frac{\sum_{i=1}^{N}(\widehat{L(t_i)}_{N+1} - L(t_i))^2}{N}$$

We then plot the estimated $\widehat{MSE}_i$ as i takes the values $i = N, N + 1, N + 2, ...$
Now, we propose on using an estimate of k, $\widehat{k}$ where the minimum $\widehat{MSE}_i$ occurs, that is, $\widehat{k} = i_m$, where i_m is one of the positive integers $N, N + 1, N + 2, ...$ at which $\widehat{MSE}_i$ is minimal.

Having an estimate of k, $\widehat{k} = i_m$, we set up our best fit logistic like regression model in the minimal mean square error sense.
proposing

$$\widehat{P(t)} = \frac{e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}{1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}$$

to estimate the true cumulative percentage number of software failures found and corrected during debugging and testing phase up to testing and debugging time t, we obtain estimates of m and α needed in Equation 4. 9 as follows:

By identification method we obtain

$$\frac{1}{1 + me^{-\alpha t}} \equiv \frac{e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}{1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}} = \frac{1}{1 + \frac{1}{e^{\hat{\beta}_{0_{i_m}} e^{\hat{\beta}_{1_{i_m}} t}}}} = \frac{1}{1 + e^{-\hat{\beta}_{0_{i_m}}} e^{-\hat{\beta}_{1_{i_m}} t}} \tag{2. 20}$$

Now comparing the first and the last term of Equation 2. 20, we obtain the following estimates of m and α:

$$\begin{cases} \widehat{m} &= e^{-\hat{\beta}_{0_{i_m}}} \\ \widehat{\alpha} &= \hat{\beta}_{1_{i_m}} \end{cases}$$

where $\hat{\beta}_{0_{i_m}}$ and $\hat{\beta}_{1_{i_m}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, using $\widehat{k} = i_m$ and i_m the positive integer from $N, N + 1, N + 2, ...$ at which $\widehat{MSE}_i$ is minimal.
Thus, an estimate of $P(t)$ can be obtained using

$$\widehat{P(t)} = \frac{1}{1 + \widehat{m}e^{-\widehat{\alpha} t}} = \frac{1}{1 + e^{-\hat{\beta}_{0_{i_m}} e^{-\hat{\beta}_{1_{i_m}} t}}}$$

Note that, since $\hat{\beta}_{0_{i_m}} < 0$ and $\hat{\beta}_{1_{i_m}} > 0$, we obtain $-\hat{\beta}_{0_{i_m}} > 0$ and $-\hat{\beta}_{1_{i_m}} < 0$. As a result, $\widehat{P(t)}$ satisfies the conditions of a distribution function.

Finally the cumulative failure behavior of the proposed model for a given software is given by

$$\widehat{L(t)} = \frac{\hat{k}}{1 + \widehat{m}e^{-\widehat{\alpha}t}} = \frac{i_m}{1 + e^{-\hat{\beta}_{0_{i_m}} e^{-\hat{\beta}_{1_{i_m}} t}}}$$

It will be shown that our proposed estimate of $L(t)$ gives good results in comparison to the other models that are commonly used.

2.4 Comparisons of Models: Numerical Application to Software Failure Data

In this section we shall compare, using the mean square error, several frequently and pre-existing models to our Logistic model on seven sets of actual software failure data obtained from actual projects. It will be shown to that our proposed model provides not only competitive parameters estimates and also gives a very good modeling of the S-shaped cumulative number of software faults found and corrected curves during testing and debugging phase. The Kolmogorov-Smirnov test will be also used to assess the goodness-of-fit of our predicted versus the actual data.

2.4.1 PL/I software Failure Data

For application to actual software reliability failures data, and comparison of our proposed model with the previous models, we analyze a data set, PL/I application program test data given by Table 1, on page 23, reported and studied previously by Ohba [10] in 1984, and later analyzed by Huang, Kuo, and Lyu in 1997 and 2001 in their research [5, 6, 7, 9, 24]. Huang and al. reported that for the PL/I database application program, the size of the software is approximately 1,317,000 line of codes, the time axis is the execution time ,and that the cumulative number of faults found and detected after a long testing period was 358, value to be used as an addition comparison criterion, [7].

Table 1: PL/I software Failure Data

Time of observation (week)	Cumulative execution time	Cumulative number of failures
1	2.45	15
2	4.9	44
3	6.86	66
4	7.84	103
5	9.52	105
6	12.89	110
7	17.1	146
8	20.47	175
9	21.43	179
10	23.35	206
11	26.23	233
12	27.67	255
13	30.93	276
14	34.77	298
15	38.61	304
16	40.91	311
17	42.67	320
18	44.66	325
19	47.65	328

From Table 2, on page 24, we note that the mean square error is minimal when the assumed cumulative number of failures is 348. Thus, our estimate $\widehat{k} = 348$ and we estimate 20 remaining faults in the software after the last correction.

Table 2: MSE as ACNOF increases - PL/I software Failure Data

ACNOF	MSE	ACNOF	MSE	ACNOF	MSE
328	140.8832368	343	93.92986795	358	99.52222464
329	135.0976346	344	93.13746872	359	100.9549485
330	129.7745802	345	92.54189896	360	102.3613104
331	124.8980566	346	92.15235569	361	104.0786848
332	120.4513938	347	91.94171328	362	105.5920188
333	116.4264294	**348**	**91.92380892**	363	107.4049262
334	112.8021575	349	92.04011389	364	109.167073
335	109.3976453	350	92.31760154	365	111.255242
336	106.5299609	351	92.77149211	366	113.0563834
337	103.8659894	352	93.40226984	367	115.1903413
338	101.5493506	353	94.12959495	368	117.2057907
339	99.44696198	354	94.92442684	369	119.3523414
340	97.75058646	355	95.97719787	370	121.6077276
341	96.24790601	356	96.98046447		
342	94.95536474	357	98.23939824		

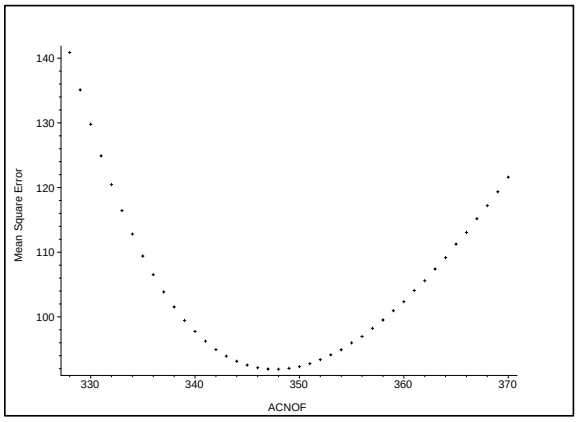

Figure 1: Plot of the MSE as ACNOF increases - PL/I software Failure Data

Table 3: Summary of models estimations: PL/I software Failure Data

Models	a or k	MSE	AE (percent)
Our model	348	91.92380892	2.79
Existing SRGMs			
HLM Model Group A, with Logistic function	394.076	118.29	10.06
HLM Model Group A, with Weibull function	565.35	122.09	57.91
HLM Model Group A, with Rayleigh function	459.08	268.42	28.23
HLM Model Group A, with Exponential	828.252	140.66	131.35
HLM Model Group B, with Logistic function	337.41	163.095	5.75
HLM Model Group B, with Weibull function	345.686	91.0226	3.43
HLM Model Group B, with Rayleigh function	371.438	158.918	3.75
HLM Model Group B, with Exponential	352.521	83.998	1.53
HLM Model Group C, with Logistic function	430.662	103.03	20.11
HLM Model Group C, with Weibull function	385.39	87.5831	7.65
HLM Model Group C, with Rayleigh function	379.947	406.71	6.13
HLM Model Group C, with Exponential	385.179	83.3452	7.69
HLM Model Group D, with Logistic function	582.538	96.9321	62.72
HLM Model Group D, with Weibull function	958.718	124.399	167.79
HLM Model Group D, with Rayleigh function	702.693	247.84	96.09
HLM Model Group D, with Exponential	1225.66	169.72	242.36
G-O Model	562.8	157.75	56.98
Inflection S-Shaped Model	389.1	133.53	8.69
Delayed S-Shaped Model	374.05	168.67	4.48
Exponential Model	455.371	206.93	27.09
HGDM	387.71	138.12	8.3
Logarithmic Poisson Model	NA	171.23	

• a or k is used to denote the total number of software fault, depending on the model

Table 3, on page 25, provides a comparative summary of our estimate $\widehat{k}$ along with several other estimates from some pre-existing models and the mean square error for each model. Table 3 also shows that by predicting 348 faults associated with a mean square error of 91.92380892 our proposed model, LR-Model, without any major assumptions, fitting well trough the K-S goodness-of-fit $(D = 0.1053, P = 1.00)$, demonstrates better performance than most of the pre-existing models.

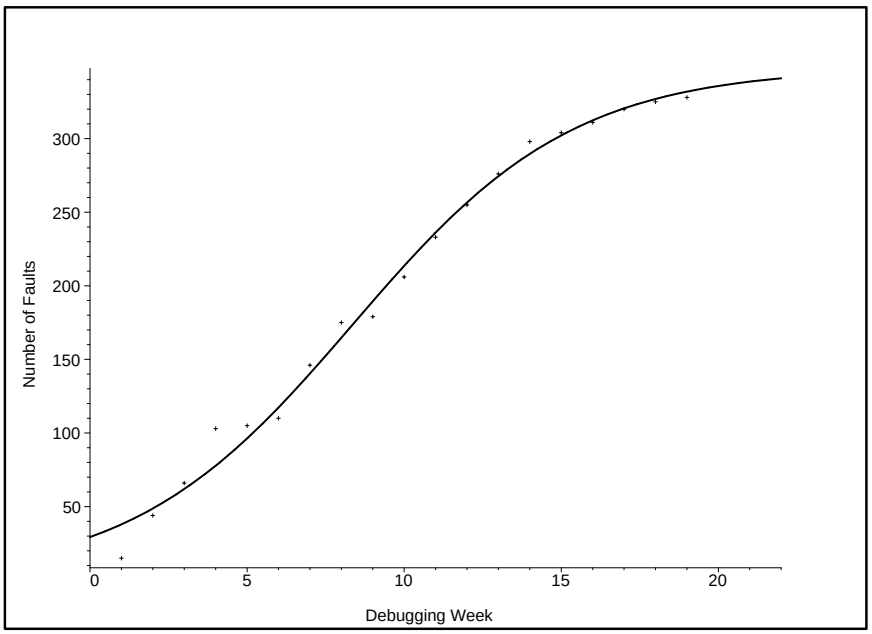

Figure 2: Comparison Of the PL/I Software Failure Data and our Predicted

Figure 2 shows that our LR-Model's estimated cumulative number of software faults found and corrected fits very well the actual cumulative software failure data.

2.4.2 Tohma's Software Failure Data

To evaluate and compare our proposed model with the others, we use a data set, the pattern of discovery of errors Table 4, on page 27, reported and studied by Tohma $[11]$ in 1989, and later analyzed by Huang, Kuo, and Lyu in 2001, in their studies $[5, 9]$. Thoma, in his paper *Hyper-Geometric Distribution Model to Estimate the number of Residual Software Faults*, reported that for pattern of discovery of errors program, in Table 4, debug times instead of the number of test workers, the number of detected faults and the cumulated detected faults were recorded day by day.

After twenty two days cumulating a total consumed debugging times of 93 CPU hours, the reported cumulative number of detected faults was 86.

Table 4: Tohma's Software Failure Data

Days	Times(hour)	Cumulative Time	Faults	Cumulative Faults
1	1	1	1	1
2	2	3	5	6
3	2	5	8	14
4	3	8	5	19
5	4	12	3	22
6	4	16	8	30
7	2	18	4	34
8	4	22	3	37
9	5	27	9	46
10	7	34	8	54
11	7	41	11	65
12	4	45	3	68
13	2	47	0	68
14	3	50	0	68
15	17	67	8	76
16	3	70	3	79
17	5	75	1	80
18	2	77	0	80
19	4	81	3	83
20	4	85	1	84
21	4	89	0	84
22	4	93	2	86

Table 5: MSE as ACNOF increases - Tohma's Software Failure Data

ACNOF	MSE
86	**7.402113064**
87	8.094494788
88	9.158227746
89	10.48334332
90	11.97355841
91	13.64098606
92	15.38582403
93	17.16645175
94	19.04502719
95	20.92112638

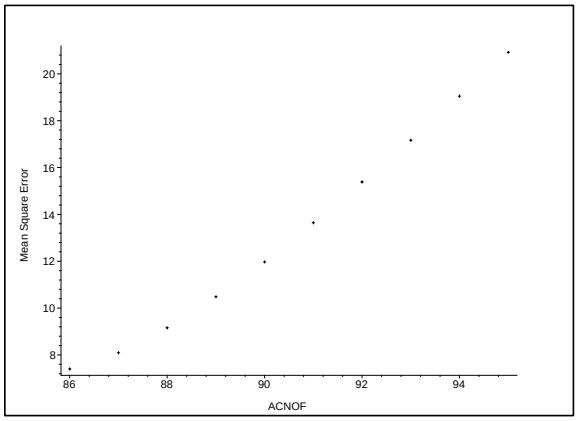

Figure 3: Plot of the MSE as ACNOF increases - Tohma's Software Failure Data

In Table 5 we observe that the mean square error is minimal when assumed cumulative number of failures is 86, which coincide here with the total the number of faults detected during debugging. Thus, our estimate is $\widehat{k}$ is 86 and we estimate 0 remaining faults in the software after the last correction.

Table 6: Summary of models estimations: Tohma's Software Failure Data

Models	a or k	MSE
Our model	86	7.40211306
Existing SRGMs		
HLM Model Group A, with Logistic function	88.8931	25.2279
HLM Model Group A, with Weibull function	87.0318	7.772
HLM Model Group A, with Rayleigh function	86.1616	3.91643
HLM Model Group B, with Logistic function	89.4528	14.06603
HLM Model Group B, with Weibull function	87.3126	18.956772
HLM Model Group B, with Rayleigh function	87.3472	20.4568
HLM Model Group C, with Logistic function	97.5332	7.354363
HLM Model Group C, with Weibull function	97.6841	6.5909
HLM Model Group C,with Rayleigh function	112.182	6.60318
HLM Model Group D, with Logistic function	106.1	7.33727
HLM Model Group D, with Weibull function	114.52	6.36531
HLM Model Group D, with Rayleigh function	112.183	6.60318
G-O Model	137.072	25.33
Delayed S-Shaped Model	88.6533	6.31268
HGDM	88.6533	6.31268
S-Y Models	73.11837775	196.7452124

• a or k is used to denote the total number of software fault, depending on the model

Table 6 provides a comparative summary of our estimate $\widehat{K}$ along with several other estimates from some pre-existing models and the mean square error for each model. Table 6 also shows that by predicting 86 faults associated with a mean square error of 7.40211306 our proposed model, LR-Model, without any major assumptions, fitting well trough the K-S goodness-of-fit $(D = 0.090, P = 1.00)$, demonstrates better performance than most of the other models.

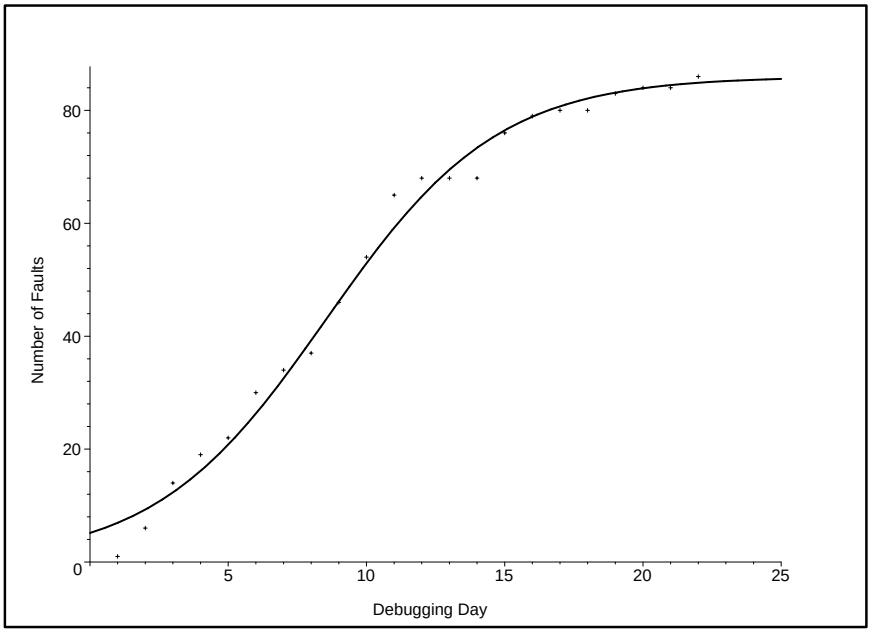

Figure 4: Comparison of Tohma's Software Failure Data and our Predicted

Figure 4 shows that our LR-Model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data. Thus, our proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

2.4.3 The F 11-D program test data

Here, we shall compare our model effectiveness with other commonly used models in the subject area. We analyze a set of software reliability field data of a data reduction program called the F 11-D program, in Table 7, on page 31, reported to be presented by Moranda 1975 and studied by Forman and Singpurwalla [13] in 1977 in their paper *An Empirical Stopping Rule for Debugging and Testing Computer Software*, later studied by Dale and Harris in 1982, in their paper *Approaches to software reliability prediction* [14], and by Tohma in 1989, [11]. The F 11-D program is reported to consist of approximately $3-4$ thousand Fortran statements [13]. As recorded in Table 7 after debugging the program for a total of 226.11 seconds of CPU time, a total of 107 software errors were detected.

Table 7: The F 11-D program test data

I.N.	Date	N.E.D.I.N	C.N.E, N	M.L.E of N
1	1/12	8	8	9
2	1/15	7	15	16
3	1/16	1	16	17
4	1/17	8	24	24
5	1/18	16	40	43
6	1/19	18	58	60
7	1/22	13	71	73
8	1/23	8	79	81
9	1/24	9	88	90
10	1/25	2	90	92
11	1/26	6	96	99
12	1/27	3	99	100
13	1/29	3	102	102
14	1/30	2	104	104
15	1/31	3	107	107

• I.N.: Interval Number • N.E.D.I.N.: Number of errors detected in Interval

Table 8: MSE as ACNOF increases - The F 11-D program test data

ACNOF	MSE
107	**9.009282277**
108	10.74342705
109	12.77994731
110	14.98197705
111	17.37213712

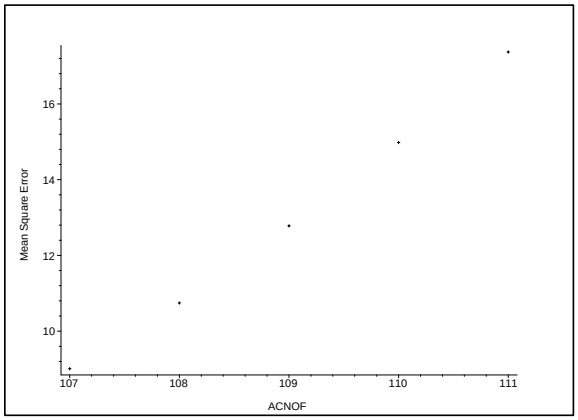

Figure 5: Plot of the MSE as ACNOF increases - The F 11-D program test data

Table 8, on page 31, shows that the mean square error is minimal when assumed cumulative number of failures is to 107 which agrees with the total the number of faults detected during actual debugging. Thus, our estimate $\widehat{k} = 107$ and we estimate 0 remaining faults in the software after the last modification.

Table 9: Summary of models estimations: The F 11-D program test data

Models	k or N	MSE
Our model	107	9.009282277
Existing SRGMs		
F-S Model	107	2.8
S-Y Models	107.4860452	8.955939003

Table 9 provides a comparative summary of our estimate $\widehat{k}$ along with a few other estimates from some pre-existing models and the mean square error for each model. Table 9 also shows that by predicting 107 faults associated with a mean square error of 9.009282277 our proposed model, LR-Model, fitting well trough the K-S goodness-of-fit $(D = 0.0667, P = 1.00)$, demonstrates a very good fit.

Our LR-Model, although does not demonstrates better performance than most of the pre-existing models in the mean square error sense, confirms Forman and Singpurwalla's empirical stopping rule

for debugging and testing Computer Software for this software package. Forman and Singpurwalla, [13], considering a simple model for describing the failures of computer software, outlined some difficulties associated with the application of the method of maximum likelihood for estimating the number of errors in a code. As pointed out in their conclusions, [13], their estimation procedure leading them to a stopping rule for debugging the software, calls for a critical examination of the actual likelihood function at each stage of the procedure. Forman and Singpurwalla, after demonstrating the usefulness of their technique on the the F 11-D program software reliability failure data, advise the software testing and debugging team to stop testing after 107 software faults were found and corrected [13]. Forman and Singpurwalla have also reported in [13] that since their technique is empirical in nature, a certain amount of subjectivity in executing it is inherently present. Dale reported that the maximum likelihood estimator can be highly misleading, and criticized Forman and Singpurwalla's maximum likelihood estimation of the remaining number of software faults: "*the estimated number of bugs remaining at the end of the twelfth debugging interval is the same as at the end of the third interval, despite the fact that* 83 *bugs have been corrected in the meantime*", [14].

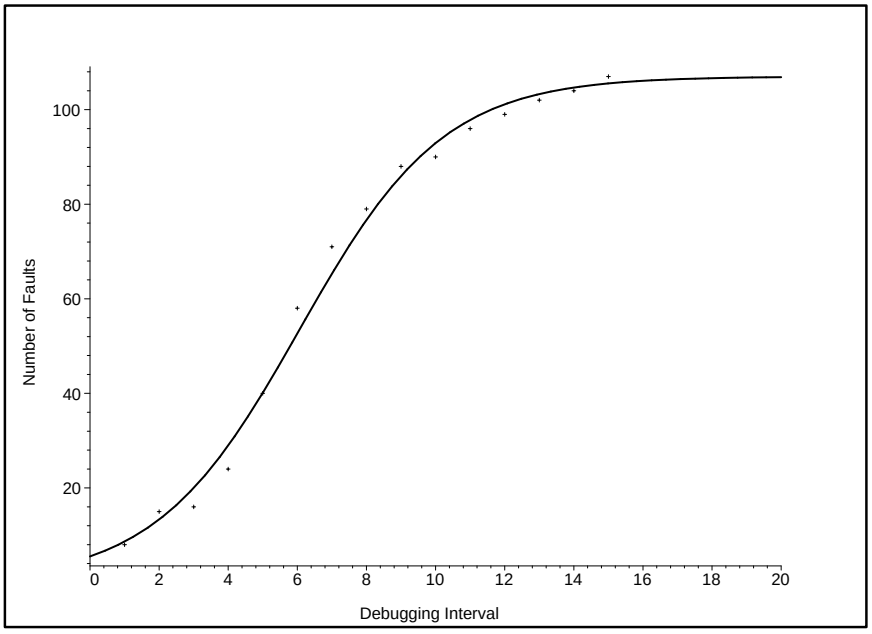

Figure 6: Comparison of The F 11-D program test data and our Predicted

The Figure 6, on page 33, shows that our LR-Model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data. Thus, our proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

2.4.4 Misra's Space Shuttle Software Failure Data

To illustrate the accuracy of our estimate over the pre-existing models a fourth data set, we study the pattern of discovery of errors in the software that supported the Space Shuttle flights STS2, STS3, STS4 at the Johnson Space Center. Table 10, on Page 35, show a weekly summary of the pattern of discovery of errors in the software that supported the Space Shuttle flights STS2, STS3, STS4 , reported and studied by Misra $[15]$ in 1983, and later analyzed by Huang, Kuo, and Chen in 1997 in their papers $[5, 6, 7, 9]$. Huang and al. reported that for the pattern od discovery of errors in the software that supported the Space Shuttle flights STS2, STS3, STS4, a weekly summary of software test hours and the errors of various severity discovered is given in $[15]$ provides a cumulative number of 231 discovered faults after 38 weeks.

Note that, for this actual data set, Table 11, on Page 36, shows that the mean square error is minimal when assumed cumulative number of failures is 237. Thus, our estimate $\widehat{k} = 237$ and we estimate 6 remaining faults in the software after the last correction.

Table 12, on Page 37, provides a comparative summary of our estimate $\widehat{k}$ along with a few other estimates from some pre-existing models and the mean square error for each model. Table 12 also shows that by predicting 237 faults associated with a mean square error of 38.38456266 our proposed model, LR-Model, fitting well trough the K-S goodness-of-fit $(D = 0.0526, P = 1.00)$, demonstrates better performance than most of the pre-existing models.

Figure 8, on Page 37, shows that our LR-Model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data. Thus, our proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

Table 10: Misra's Space Shuttle Software Failure Data

Week	Cr.E	Ma.E	Mi.E	C.E	Week	Cr.E	Ma.E	Mi.E	C.E
1	0	6	9	15	20	0	2	3	136
2	0	2	4	21	21	0	1	1	138
3	0	1	7	29	22	0	3	2	143
4	1	1	6	37	23	0	2	4	149
5	0	3	5	45	24	0	4	5	158
6	0	1	3	49	25	0	1	0	159
7	0	2	2	53	26	0	2	2	163
8	0	3	5	61	27	0	2	0	165
9	0	2	4	67	28	0	2	2	169
10	0	0	2	69	29	0	1	3	173
11	0	3	4	76	30	1	2	6	182
12	0	1	7	84	31	1	2	3	188
13	0	3	0	87	32	0	0	1	189
14	0	0	5	92	33	0	2	1	192
15	0	2	3	97	34	0	2	4	198
16	0	5	3	105	35	0	3	3	204
17	0	5	3	113	36	0	1	2	207
18	0	2	4	119	37	1	2	11	221
19	0	2	10	131	38	0	1	9	231

- Cr.E: Critical Errors
- Ma.E: Major Errors
- Mi.E: Minor Errors
- C.E.: Cumulative Errors.

Table 11: MSE as ACNOF increases - Misra's Space Shuttle Software Failure Data

ACNOF	MSE
231	39.24889498
232	38.96808896
233	38.73239432
234	38.55970054
235	38.4544495
236	38.39724536
237	**38.38456266**
238	38.4183368
239	38.49035903
240	38.6077261
241	38.75830898
242	38.94298088
243	39.16746369

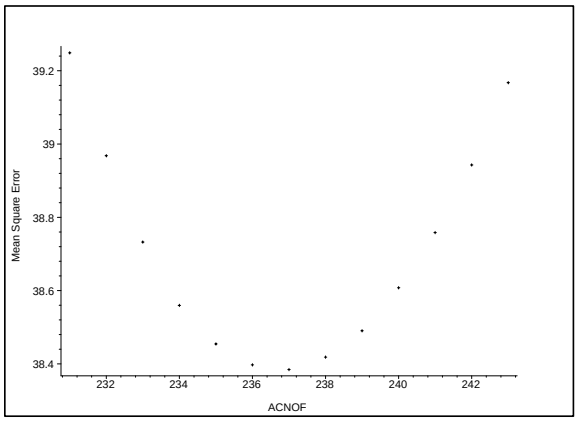

Figure 7: MSE as ACNOF increases - Misra's Space Shuttle Software Failure Data

Table 12: Summary of models estimations: Misra's Space Shuttle Software Failure Data

Models	a or K	MSE
Our model	237	38.38456266
G-O Model	597.887	78.87
S-Y Models	200.8486903	265.3335203

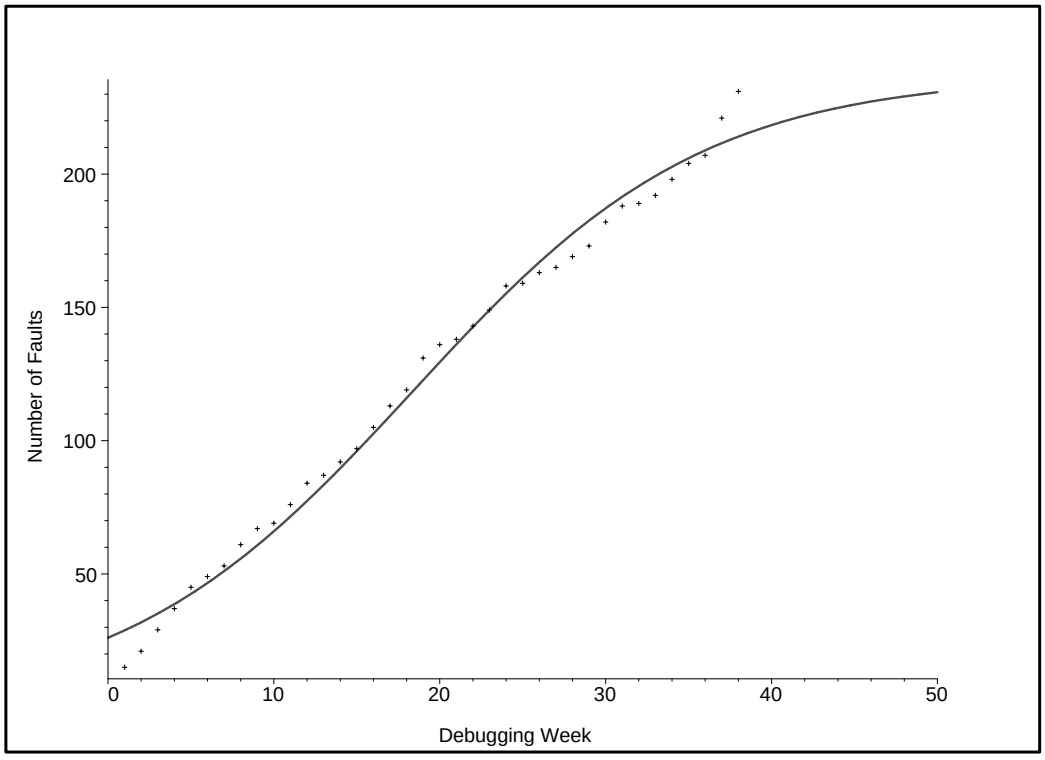

Figure 8: Comparison of Misra's Space Shuttle Software Failure Data and our Predicted

2.4.5 Musa's System T1 software Failure Data

To assess and compare our proposed model with the others pre-existing models, we study the System $T1$ of the Rome Air Development Center (RADC) projects Table 13, reported by Musa, [53], and later studied by Huang, Kuo, and Chen in 1997 in their papers [6, 7, 9]. The system $T1$ is reported to be used for a real-time command and control application; the size of the software is approximately $21,700$ object instructions. To complete the test, it took twenty-one weeks, nine programmers, about 25.3 CPU hours to remove 136 software errors.

Table 13: System T1

Interval of observation (interval length = 5 Days)	Cumulative number of failures
1	3
2	4
3	6
4	16
5	16
6	22
7	29
8	29
9	31
10	42
11	48
12	63
13	78
14	92
15	105
16	122
17	132
18	135
19	136

Table 14: MSE as ACNOF increases - System T1

ACNOF	MSE	ACNOF	MSE
136	59.73849038	164	20.74581198
137	56.45913482	165	20.43461528
138	53.51272497	166	20.16955946
139	50.74510692	167	19.93233847
140	48.05754858	168	19.73619002
141	45.68399539	169	19.56559153
142	43.41103347	170	19.42895933
143	41.32847537	171	19.3171211
144	39.34891981	172	19.2292499
145	37.56874349	173	19.16859798
146	35.97196455	174	19.1305098
147	34.38348791	**175**	**19.11480503**
148	32.98043088	176	19.11803102
149	31.58854155	177	19.14235957
150	30.37887505	178	19.18716939
151	29.25654819	179	19.24351235
152	28.22982752	180	19.31848929
153	27.28001268	181	19.41178997
154	26.36265956	182	19.52408042
155	25.52487927	183	19.63968954
156	24.77120468	184	19.77192253
157	24.09448225	185	19.91998986
158	23.44652547	186	20.08486024
159	22.90736243	187	20.24602725
160	22.35927548	188	20.4212037
161	21.90723966	189	20.61078576
162	21.45572338	190	20.81366771
163	21.0866153		

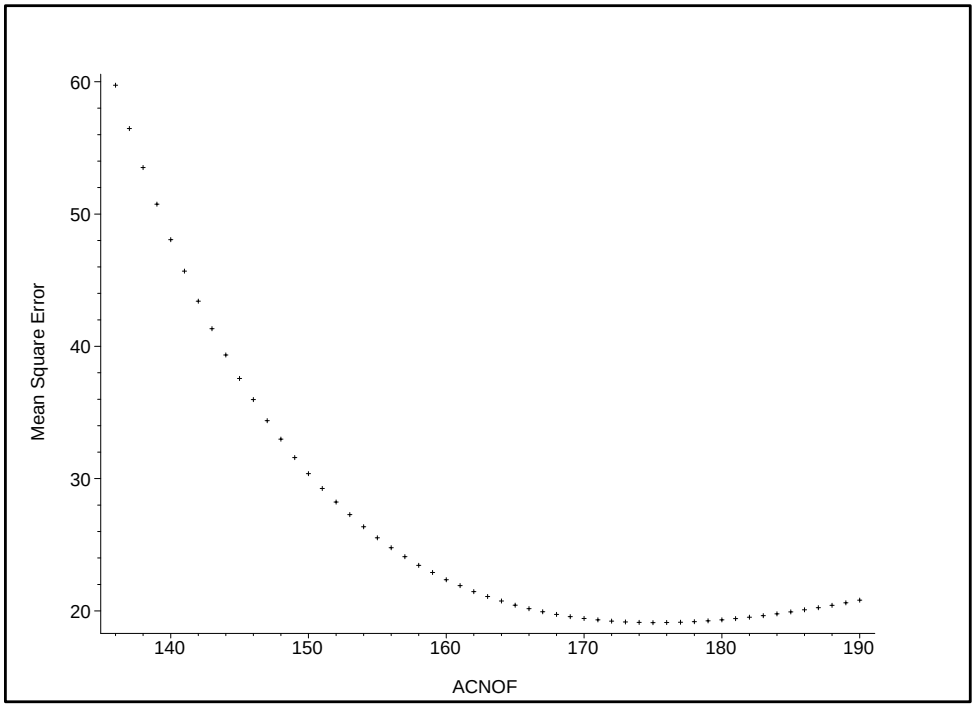

Figure 9: Plot of the MSE as ACNOF increases - System T1

Table 14, on page 39, shows that the Mean Square Error is minimal when assumed cumulative number of failures is to 175. Thus, our estimate $\widehat{k} = 175$ and we estimate 39 remaining faults in the software after the last correct.

Table 15, on page 41, provides a comparative summary of our estimate $\widehat{k}$ along with a few other estimates from some pre-existing models and the mean square error for each model. Table 15 also shows that by predicting 175 faults associated with a mean square error of 19.11480503 our proposed model, LR-Model, fitting well trough the K-S goodness-of-fit ($D = 0.1053, P = 1.00$), demonstrates better performance than most of the pre-existing models.

Table 15: Summary of models estimations: System T1

Models	a or K	MSE
Our model	175	19.11480503
Existing SRGMs		
HLM Model Group A, with Logistic function	138.026	62.41
HLM Model Group A, with Rayleigh function	866.94	89.24095
HLM Model Group B, with Logistic function	137.759	14.6442
HLM Model Group B, with Rayleigh function	150.047	12.137
HLM Model Group B, with Exponential function	187.537	19.73719
HLM Model Group C, with Logistic function	142.567	13.4266
HLM Model Group C, with Rayleigh function	156.715	10.9726
HLM Model Group C,with Exponential function	173.064	48.5971
HLM Model Group D, with Logistic function	164.106	38.121
HLM Model Group D, with Rayleigh function	1543.47	89.7666
Exponential Model	137.2	3019.66
G-O Model	142.32	2438.3
Delayed S-Shaped Model	237.196	245.246
S-Y Models	145.2562193	120.5350962

The Figure 10 below, shows that our LR-Model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data. Thus, our proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

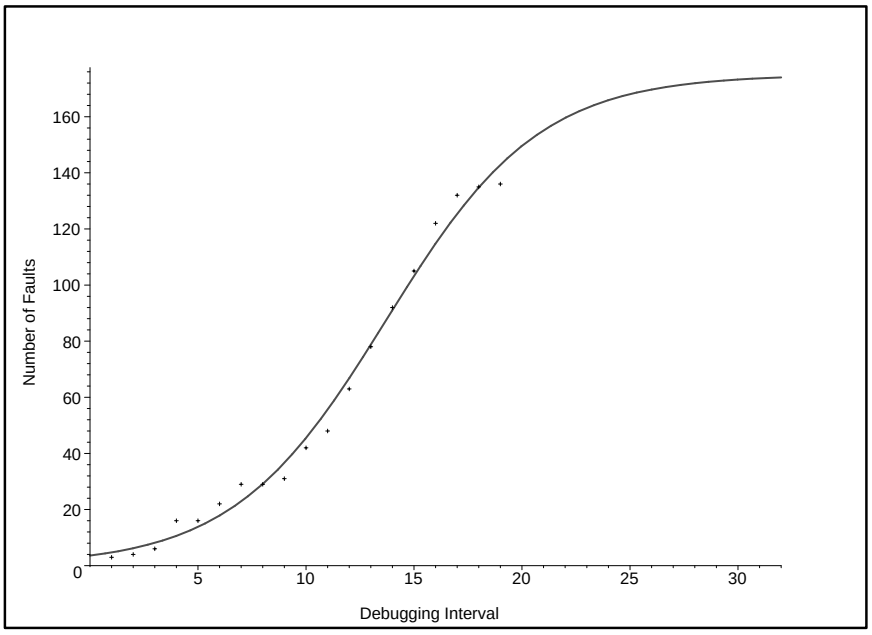

Figure 10: Comparison of the System T1 data set and our Predicted

2.4.6 Ohba's On-line data entry software test Data

For application to actual software reliability failures data,and comparison of our proposed model with the previous models, we analyze the sixth data set, Ohba's On-line data entry software package test data Table 16, on page 43, reported by Ohba in 1984 [10] and later analyzed by Tohma, Jacoby, Murata, and Yamamoto in 1989 in their papers [11]. The small on-line data entry control software package reported to have been available since 1980 in Japan, has an approximate size of 40000 lines of code; the testing time was measured on the basis of the number of shifts spent running test cases and analyzing the results. the number of persons on the test team was reported to be constant throughout the test period; During the twenty-one days of test, 46 software errors were removed [10]; the cumulative number of faults found and detected after a long testing period (three years) was reported to be 69, value to be used as an addition comparison criterion.

Table 16: Ohba's On-line data entry software test Data

Time of observation (day)	Cumulative number of faults
1	2
2	3
3	4
4	5
6	9
7	11
8	12
9	19
10	21
11	22
12	24
13	26
14	30
15	31
16	37
17	38
18	41
19	42
20	45
21	46

Table 17: MSE as ACNOF increases - Ohba's On-line data entry software test Data

ACNOF	MSE
46	2.877909807
47	2.439920792
48	2.139457102
49	1.945403281
50	1.832603369
51	**1.784785998**
52	1.786831674
53	1.828631846
54	1.900772326
55	1.998562288
56	2.115459903
57	2.24590523
58	2.387993439
59	2.541360386
60	2.699959644

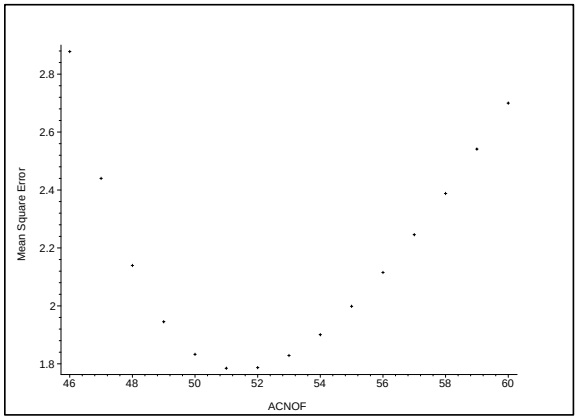

Figure 11: Plot of the MSE as ACNOF increases - Ohba's On-line data entry software test Data

Referring to Table 17, on page 44, we note that, for this real data set, the mean square error is minimal when assumed cumulative number of failures is to 51. Thus, our estimate $\widehat{k} = 51$ and we estimate 5 remaining faults in the software after the last correction.

Table 18 provides a comparative summary of our estimate $\widehat{k}$ along with a few other estimates from some pre-existing models and the mean square error for each model. Table 18 also shows that by predicting 51 faults associated with a mean square error of 1.784785998 our proposed model, LR-Model, fitting the actual data very well trough the K-S goodness-of-fit $(D = 0.1000, P = 1.00)$.

Table 18: Summary of models estimations: Ohba's On-line data entry software test data

Models	a or K	MSE
Our model	51	1.784785998
Delayed S-Shaped Model [10]	71.7	1.694955709

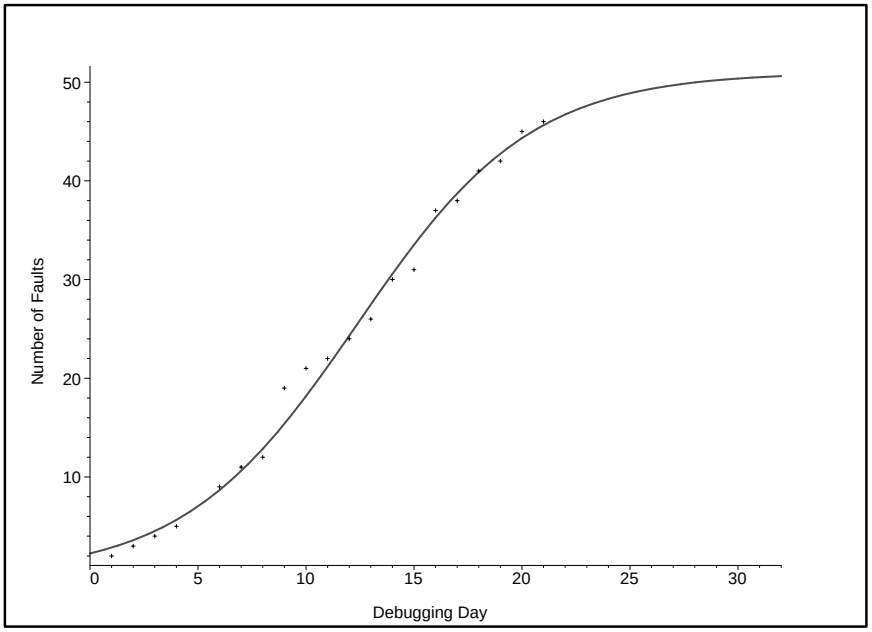

Figure 12: Comparison Of ohba's On-line data entry software test Data and our Predicted

The Figure 12, on page 45, shows that our LR-Model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data. Thus, our

proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

2.4.7 Tohma's software package test Data

For application to actual software reliability failures data, and comparison of our proposed model with the previous models, we analyze a last data set, the field report of the test of a software for monitoring and real-time control Table 19, on page 47, reported by Tohma, Tohunaga, and Nagase in 1989 $[11, 16, 18]$ and later analyzed by Doli, Wakana, Osaki, and Trivedi in there study, $[17]$. The software is reported to consist of about 200 modules, each of which is about 1000 lines in a high-level language like Fortran.

From Table 20, on page 48, we note that for this real data set, the mean square error is minimal when assumed cumulative number of failures is to 481. Thus, our estimate $\widehat{k} = 481$ and we estimate 0 remaining faults in this software package after the last correction.

Table 21, on page 49, provides a comparative summary of our estimate $\widehat{k}$ along with several other estimates from the Hypergeometric model $(HGDM)$ and the mean square error for each model. 413.9032446. Although our proposed model is associated with a relatively large mean square error of 413.9032446, respective to the Hypergeometric models, it fits the actual data very well trough the K-S goodness-of-fit $(D = 0.1712, P = 0.069)$.

The Figure 14, on page 50, shows that our proposed model's predictive cumulative number of software faults found and corrected fits very well the actual cumulative software failure data.

Table 19: Tohma's software package test Data

Days	Cum. F	Days	Cum. F	Days	Cum. F	Days	Cum. F	Days	Cum. F
1	5	1	234	1	417	67	465	90	475
2	10	2	236	2	425	68	466	91	475
3	15	3	240	3	430	69	467	92	475
4	20	4	243	4	431	70	467	93	475
5	26	5	252	5	433	71	467	94	475
6	34	6	254	6	435	72	468	95	475
7	36	7	259	7	437	73	469	96	476
8	43	31	263	8	444	74	469	97	476
9	47	32	264	9	446	75	469	98	476
10	49	33	268	10	446	76	469	99	476
11	80	34	271	11	448	77	470	100	477
12	84	35	277	12	451	78	472	101	477
13	108	36	290	13	453	79	472	102	477
14	157	37	309	14	460	80	473	103	478
15	171	38	324	61	463	81	473	104	478
16	183	39	331	62	463	82	473	105	478
17	191	40	346	63	464	83	473	106	479
18	200	41	367	64	464	84	473	107	479
19	204	42	375	65	465	85	473	108	479
20	211	43	381	66	465	86	473	109	480
21	217	44	401	67	465	87	475	110	480
22	226	45	411	68	466	88	475	111	481
23	230	46	414	69	467	89	475		

• Cum.F: Cumulative Faults

Table 20: MSE as ACNOF increases - Tohma's software package test Data

ACNOF	MSE
481	**413.9032446**
482	420.9482805
483	430.0166841
484	441.7300299
485	457.7926775
486	473.4347958
487	492.8117519
488	513.0814357
489	533.5721618
490	557.2539898

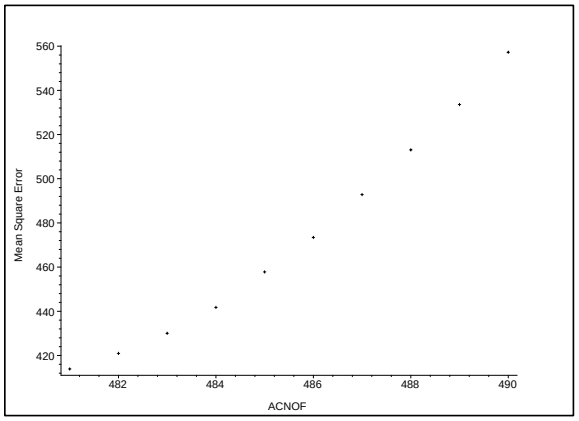

Figure 13: Plot of the MSE as ACNOF increases - Tohma's software package test Data

Table 21: Summary of models estimations: Tohma's software package test data

Models	a or K	MSE
Our model	481	413.9032446
HGDM (a) [17]	479	274.3
HGDM (b) [17]	497	242.4
HGDM (c) [17]	479	273.5
HGDM (d) [17]	497	242.5
HGDM (e) [17]	531	298.3
HGDM (f) [17]	476	566.3
HGDM (g) [17]	546	977.1
HGDM (h) [17]	497	364.3
HGDM (i) [17]	496	353.6
HGDM (j) [17]	498	371.8
HGDM (k) [17]	482	253.2
HGDM (l) [17]	486	328.7
S-Y Models	444.1965400	3843.008443

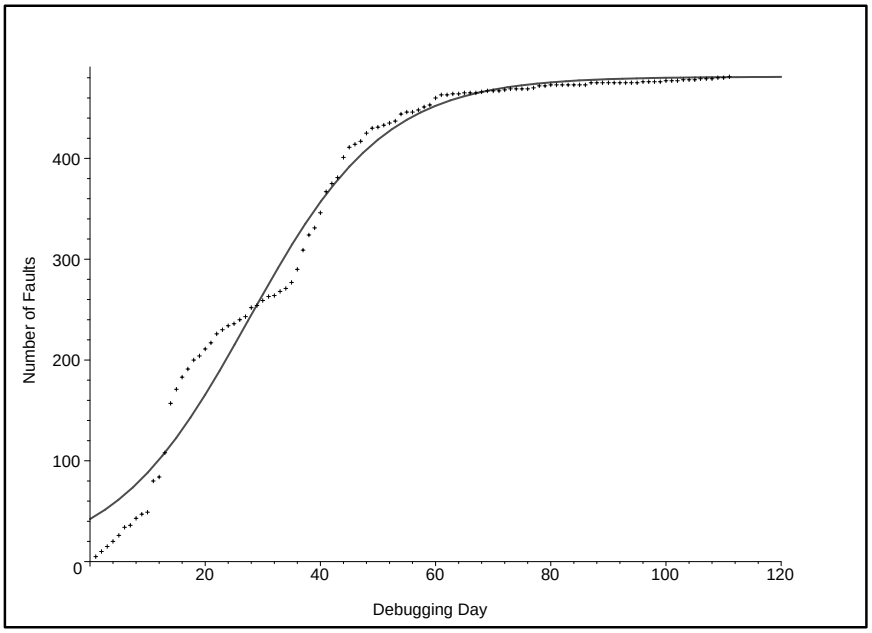

Figure 14: Comparison of Tohma's software package test Data set and our Predicted

2.5 Conclusions

In the present study we have introduced a logistic software reliability growth model for the purpose of estimating the cumulative number of faults in a given software. The analytical form of this new model is given by

$$\widehat{L(t)} = \frac{\hat{k}}{1 + \widehat{m}e^{-\widehat{\alpha}t}} = \frac{i_m}{1 + e^{-\hat{\beta}_{0_{i_m}} e^{-\hat{\beta}_{1_{i_m}} t}}}$$

with

$$\begin{cases} \widehat{m} &= e^{-\hat{\beta}_{0_{i_m}}} \\ \widehat{\alpha} &= \hat{\beta}_{1_{i_m}} \end{cases}$$

where $\hat{\beta}_{0_{i_m}}$ and $\hat{\beta}_{1_{i_m}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, using $\widehat{k} = i_m$ and i_m the positive integer from $N, N+1, N+2, ...$ at which $\widehat{MSE_i}$ is minimal.
This model not only gives very good predictions of the cumulative number of software faults, but it is easy to apply and it is free of assumptions.
The new model was compared with the following commonly used models to in the subject area
i) Exponential Model
ii) G-O Model
iii) Delayed S-Shaped Model
iV) S-Y Models
V) C-Model
Vi) HGDM Hypergeometric Model
Vii) Huang, Kuo, Chen, and Lyu's Models [5, 6, 7, 9]

We used seven different sets of actual data, namely,
i) the PL/I Database Application test data by Ohba,[10]
ii) Tohma's software failure data, [11]
iii) the F 11-D program test data by Forman and Singpurwalla, [13]
iV) the pattern of discovery of errors STS2, STS3, STS4 by Misra [15]
V) the System T1 by Musa, [53]

Vi) the On-line data entry software test Data by Ohba,[10]

Vii) Tohma's software package test Data, [11, 16, 18]

The mean square error criteria was used to compare the results of our proposed model with other models stated above. The results presented in tables and graphical forms support the fact that the new model is more effective in estimating the number of faults found during the testing and debugging phase of a given software.

Chapter 3

Logistic Regression Approach to Software Reliability Assessment

3.1 Introduction

The increasing dependency on computers along with an increasing need for safety, efficiency and cost reduction, causes not only for a higher demand for software and make software testing a crucial phase of the development of any software to produce highly reliable packages. It is clear that an air traffic control software will demand a very accurate failure behavior prediction, then a software designed for a fun game. A crucial decision any software developer has to make, in software testing and debugging phase, is the following: given all the restrictions, can we stop the testing and debugging phase and release a software package for use, or should we continue testing?

Since the occurrence of software failures are random, if the testing and debugging team hits a long random run where no software mistake is found; they are likely to stop testing and falsely conclude that the software is error free. This leads to a premature end of the testing and debugging phase that affect the reliability of the software to be released for marketing. Satoh and Yamada's software reliability growth models do not yield accurate reliability estimates in spite of a small amount of failure data causing their procedure and justification do not lead to a good estimation at the early stage of the testing and debugging phase. In this chapter, we propose a more realistic model that is more effective and eliminates many of the difficulties that the S-Y models require. Our proposed software reliability growth model is simple to implement and does not assume any prior distribution.

Relatively early in the testing and debugging phase, it provides accurate parameters estimation, gives a very good failure behavior prediction and it is possible for software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software.

The remaining of this Chapter is organized as follows: in Section 3.2 we summarize Satoh and Yamada's models, [1]. In Section 3.3, we present our proposed Logistic Early Estimation Model.

Comparison of the Logistic Early Estimation Model, LR-EE-Model, with several pre-existing models is made using real data from seven actual world problems in Section 3.4. Finally, our conclusions and recommendations are given in Section 3.5.

3.1.1 Statistical Abbreviations and Notations

- LR-EE-Model: our Logistic Regression Early Estimation Model
- S-Y Models: Satoh and Yamada's Software reliability growth models

3.2 preliminaries

Here, we shall give a brief description of Satoh and Yamada statistical models.

3.2.1 Satoh and Yamada's models and conclusions

Satoh and Yamada reported that, their two software reliability growth models yield accurate parameter estimates even with a small amount of data, [1]. Their proposed models give the same parameters estimates. According to Satoh and Yamada, although the conventional model uses a discrete equation as a regression model, the model itself is a continuous time model, thus, it includes errors generated by discretization, however, their models do not have this problem because they are themselves discrete models and that they can analyze the software reliability without a continuous time model.

The Satoh and Yamada discrete models were reported to have three advantage over the conventional model [1].

(1) The parameter estimation in the discrete logistic curve models reproduced the values of the parameters very accurately, even with small data that do not include the inflection point. When the exact solution is used as the input data, the conventional model provides inaccurate parameter estimates with data that do not include the inflection point.

(2) The discrete models are independent of time scale. The same parameters estimates are obtained no matter what value of the time scale they choose. When the conventional model is used we have to choose the time scale carefully, because the time scale needs to be used in the regression equation. As a result, the estimates depend on the choice of the time scale.

(3) The discrete logistic curve models enable them to accurately estimate the parameters in the early testing phase with real data.

The parameter estimates of the conventional model vary with the number of data points. The discrete models provide stable values of parameter estimates for various number of data points. This characteristic is very important for software reliability growth models.

3.2.2 Remarks

we believe that Satoh and Yamada's conclusion results from not simulating the data properly; they use the exact solution is used as the input data not adding any noise. As a result, the problem is purely deterministic. Since there are three parameters k, m, and α, to estimate in $L(t) = \frac{k}{1+m\exp(-\alpha t)}$, if no noise is added to an exact solution when preparing the data, as a Saturated model one only needs as many data points as there are parameters to estimate the parameters. While Satoh and Yamada's approach perfectly reproduces the values of the parameters, when the data set satisfy an exact solution with no noise, in early stage of the debugging and testing phase (three data points), their parameter estimation with too small data points does not give accurate parameter estimates on prepared data with noise and on actual data.

3.3 Development of The Proposed Model: Early Estimation

The crucial decision any software developer have to make, in software testing and debugging phase is wether to stop, conclude the testing and the debugging phase of a software or continue testing and debugging it. Having to take into consideration factors like reliability of the software being developed and the cost of its development process, it is of gold for the software developers to have a stopping rule, at the early stage of the testing and debugging phase, associated with a high confidence level that testing and debugging phase must be concluded at some time t^*, to reduce cost of over testing the software or that testing must proceed a bit longer to avoid a premature ending of the debugging process to produce more reliable or highly reliable softwares.

The objective of this section is to develop a software reliability growth model that

(i) does not assume any prior distribution

(ii) is free of any major convergence assumptions

(iii) is simple

(iV) fits very well the cumulative number of software faults found and corrected

(V) provides accurate parameter estimates relatively early in the testing phase once the available data points during testing and debugging phase include a bit more then the inflection point

(Vi) Our proposed model provides stable values of parameter estimates once the available data includes a bit more then the inflection point; this characteristic is very important for software reliability growth models.

(Vii) provides an additional decision-making rule to the software developers respective to when to conclude the testing phase, manage their resources, and schedule the release time of the software package to be marketed for use.

(Viii) not subject to errors generated by discretization.

(iX) our LR-EE-Model, by providing a stopping and release rule, saves time and cuts cost in preventing over-testing the software to be marketed, and produces a more reliable software in preventing an early release of the software to be market, specially when the testing and debugging team hits a long run where not software fault is found.

3.3.1 Model Description

The description of our model is as proposed in Chapter 2, Section 2. The proposed model is dependent on an accurate estimated number of software faults in the software $\widehat{k}$, before complete modeling of the cumulative failure behavior is obtained. In this chapter, we propose a new way of estimating the parameter k, as an increasing number of available failure data are recorded, while the testing and debugging phase proceeds. Our new estimate of the parameter k is based on an accurate method to locating the inflection point of a gradually increasing amount of actual failure data points. Once the inflection point is located, a prediction of the total number of software faults $\widehat{k}$ is obtained; then estimation of the remaining parameters m and α is obtained, as proposed in Chapter 2, by identification.

3.3.2 Parameters Estimation

Clearly, if k the initial number of software faults in the software prior the debugging phase or the total number of software faults found and corrected in debugging phase during an infinitely long exposure time interval, when a Cumulative logistic curve model is fitted, at the inflection point $\frac{k}{2}$

software faults have been found and corrected. In this section, we propose a method for locating the inflection point; then an estimate of k is found via estimating $\frac{k}{2}$.

Let $T_1 < T_2 < T_3 < ... < T_n$ be the first n successive random failure times, as software ages, with their corresponding cumulative number of software faults found and corrected:

$L(T_1) = 1$, $L(T_2) = 2$, $L(T_3) = 3$, ..., $L(T_n) = n$.

We run sequential linear regression on two kind of successive overlapping chains of size N ordered pairs (T_i, $L(T_i) = i$) that we will refer to as Chain 1 or Chain 2.

Description of overlapping chain I

Our chain I consists of overlapping chains of size $N \geq 2$ constructed as follows:Our first run consist of finding the equation of the regression on the first N ordered pairs (T_i , $L(T_i) = i$), $i = 1, ..., N$ and recording its slope; then for any other successive kth run $k \geq 2$, we run a linear regression line on (T_i , $L(T_i) = i$), where $i = (k-1)(N-1), i+2, i+3, ..., i+N$.

Description of overlapping chain II

Our chain II consists of overlapping chains of size $N \geq 4$ and N is even, constructed as follows: our kth run consists of finding the equation of successive regressions lines on the ordered pairs (T_i , $L(T_i) = i$), where $i = (k-1)(\frac{N}{2}) + 1, i+1, i+2, ..., i+(N-1)$.

Locating the inflection point

For both successive overlapping chains of size N ordered pairs (T_i , $L(T_i) = i$), the slopes of the successive regression lines are recorded and their patten is observed.

To locate the inflection point, we look for the chain on which the successive slope behavior is maximal then consistently decreasing. Once the chain including the inflection point is identified, averaging is used to estimate k.

(1) Select using the maximum slope criteria an ith chain

(2) In the selected ith chain, two times the actual cumulative faults is a candidate for $\hat{k}$

(3) Test each candidate for $\hat{k}$ in the selected ith chain, using the minimum mean square error criteria:

Let N, M, and three candidates from the selected ith chain such the $N < M < L$ If $MSE_N >$

MSE_M, discard candidate N and proceed with comparing the mean square errors $MSE_M and MSE_L$ ...

If $MSE_N < MSE_N$, select candidate N as the estimate of the parameter k.

After estimating k, we are ready to set up our Logistic Like Software Reliability Growth Model proposed in Chapter 2.

Having an estimate of k, we name it $\widehat{k} = i_{early}$, we set up our best fit logistic like regression model in the minimal mean square error sense.

proposing

$$\widehat{P(t)} = \frac{e^{\hat{\beta}_{0_{i_{early}}} + \hat{\beta}_{1_{i_{early}}} t}}{1 + e^{\hat{\beta}_{0_{i_{early}}} + \hat{\beta}_{1_{i_{early}}} t}}$$

to estimate the true cumulative percentage number of software failures found and corrected during debugging and testing phase up to testing and debugging time t, we obtain estimates of m and α needed in Equation 4. 9 as follows:

By identification method we obtain

$$\frac{1}{1 + me^{-\alpha t}} \equiv \frac{e^{\hat{\beta}_{0_{i_{early}}} + \hat{\beta}_{1_{i_{early}}} t}}{1 + e^{\hat{\beta}_{0_{i_{early}}} + \hat{\beta}_{1_{i_{early}}} t}} = \frac{1}{1 + \frac{1}{e^{\hat{\beta}_{0_{i_{early}}} e^{\hat{\beta}_{1_{i_{early}}} t}}}} = \frac{1}{1 + e^{-\hat{\beta}_{0_{i_{early}}} e^{-\hat{\beta}_{1_{i_{early}}} t}}} \quad (3.\ 1)$$

Now comparing the first and the last term of Equation 3. 1, we obtain the following estimates of m and α:

$$\begin{cases} \widehat{m} &= e^{-\hat{\beta}_{0_{i_{early}}}} \\ \widehat{\alpha} &= \hat{\beta}_{1_{i_{early}}} \end{cases}$$

where $\hat{\beta}_{0_{i_{early}}}$ and $\hat{\beta}_{1_{i_{early}}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, using $\widehat{k} = i_{early}$ and i_{early} the positive integer from $N, M, L, ...$ at which slope is maximal and mean square error is minimal .

Thus, an estimate of $P(t)$ can be obtained using

$$\widehat{P(t)} = \frac{1}{1 + \widehat{m}e^{-\widehat{\alpha} t}} = \frac{1}{1 + e^{-\hat{\beta}_{0_{i_{early}}} e^{-\hat{\beta}_{1_{i_{early}}} t}}}$$

Finally the cumulative failure behavior of the proposed model for a given software is given by

$$\widehat{L(t)} = \frac{\hat{k}}{1 + \widehat{m}e^{-\widehat{\alpha}t}} = \frac{i_{early}}{1 + e^{-\hat{\beta}_{0_{i_{early}}} e^{-\hat{\beta}_{1_{i_{early}}} t}}}$$

It will be shown that our proposed estimate of $L(t)$ gives good results in comparison to the other models that are commonly used.

3.4 Comparisons of Models: Numerical Application to Software Failure Data

In this section we shall illustrate our approach and compare the performance of our Early Estimation Logistic Model, using overlapping chains, and using the mean square error, with several frequently and pre-existing models on an actual software failure data already analyzed in
Chapter 2: the PL/I application program test data Table 1, on page 23 , reported and studied by Ohba $[10]$ in 1984.

It will be shown that our proposed model yields accurate parameters estimates early into the testing and debugging phase, gives a very good modeling of the S-shaped cumulative number of software faults found and corrected curves during testing and debugging phase giving the software developers an early protocol as far as when to stop the testing and debugging phase and release the software for use, cutting cost on maintenance, and due to over testing.

Our Early Estimation on the PL/I Software Failure Data

From Tables 22, Table 23, Table 24, and Table 25, respectively on pages 60, 61, 61, and 62, we estimated a cumulative number of software fault of 350. Thus, our estimate $\widehat{K} = 350$.

Table 26, on page 63, provides a comparative summary of our estimate $\widehat{K}$ along with several other estimates from some pre-existing models and the mean square error for each model. Table 26 also shows that by predicting 350 faults associated with an overall mean square error of 92.31760154 our proposed model, LR-Model, without any major assumptions, fitting well trough the K-S goodness-of-fit, demonstrates better performance than most of the pre-existing models when data points a bit after the inflection points are available.

Creating Overlapping Chains: PL/I Database Application Software

Table 22: Overlapping Chain 1 of Size $N = 2$ -PL/I Database Application Software

RUNS	[Ti, T(i+1)] [Ni, N(i+1)]	slope of RL	r
RUN 1	[1,3]; [15,66]	25.5	1
RUN 2	[2,4]; [44,103]	29.5	1
RUN 3	[3, 5]; [66,105]	19.5	1
RUN 4	[4, 6]; [103,110]	3.5	1
RUN 5	[5,7]; [105,146]	20.5	1
RUN 6	[6, 8]; [110, 175]	32.5	1
RUN 7	[7,9]; [146,179]	16.5	1

Note that only the data points up to the $9th$ week are use to predict the total number of faults in the software.

Table 23: Overlapping Chain 1 of Size $N = 2$ -PL/I Database Application Software

N	MSE	Pair number
N= 88	6.61E-09	2
N=132	15.30759212	3
N=206	16.17930627	4
N=220	179.7683445	6
N=292	179.2839406	7
N = 350	**165.1157652**	**8**
N=351	165.259913	
N=358	183.1532424	9
N=412	189.5550172	10
N=466	189.8825614	11

Table 24: Overlapping Chain 1 of Size $N = 3$-PL/I Database Application Software

RUNS	[Ti, T(i+1)] [Ni, N(i+1)]	slope of RL	r
RUN 1	[1,2,3]; [15,44,66]	25.5	r := .9968748929
RUN 2	[2,4 , 5]; [44,103,105]	21.64285714	r := .9539628928
RUN 3	[4, 6 ,7]; [103,110,146]	12.78571429	r := .8464894644
RUN 4	[6, 8,9]; [110,175,179]	24.35714286	r := .9605519477
RUN 5	[8,10,11]; [175,206,233]	18.78571429	r := .9887218045

Note that only the data points up to the $11th$ week are use to predict the total number of faults in the software.

Table 25: Overlapping Chain 1 of Size $N = 5$-PL/I Database Application Software

RUNS	[Ti, T(i+1)] [Ni, N(i+1)]	slope of RL	r
RUN 1	[1,2,3,4,5] ; [15,44,66,103,105]	23.9	r := .9778998350
RUN 2	[4,6,7,8,9] ; [103,110,146,175,179]	17.33783784	r := .9416620919
RUN 3	[8,10,11,12,13]; [175,206,233,255,276]	20.60810811	r := .9952196625

Note that only the data points up to the $13th$ week are use to predict the total number of faults in the software.

Table 26: Summary of models estimations: PL/I Database Application Software

Models	a or K	MSE	AE
Our early estimation model	350	92.31760154	2.23
Our LR-Model	348	91.92380892	2.79
HLM Model Group A, with Logistic function	394.076	118.29	10.06
HLM Model Group A, with Weibull function	565.35	122.09	57.91
HLM Model Group A, with Rayleigh function	459.08	268.42	28.23
HLM Model Group A, with Exponential	828.252	140.66	131.35
HLM Model Group B, with Logistic function	337.41	163.095	5.75
HLM Model Group B, with Weibull function	345.686	91.0226	3.43
HLM Model Group B, with Rayleigh function	371.438	158.918	3.75
HLM Model Group B, with Exponential	352.521	83.998	1.53
HLM Model Group C, with Logistic function	430.662	103.03	20.11
HLM Model Group C, with Weibull function	385.39	87.5831	7.65
HLM Model Group C, with Rayleigh function	379.947	406.71	6.13
HLM Model Group C, with Exponential	385.179	83.3452	7.69
HLM Model Group D, with Logistic function	582.538	96.9321	62.72
HLM Model Group D, with Weibull function	958.718	124.399	167.79
HLM Model Group D, with Rayleigh function	702.693	247.84	96.09
HLM Model Group D, with Exponential	1225.66	169.72	242.36
G-O Model	562.8	157.75	56.98
Inflection S-Shaped Model	389.1	133.53	8.69
Delayed S-Shaped Model	374.05	168.67	4.48
Exponential Model	455.371	206.93	27.09
HGDM	387.71	138.12	8.3
Logarithmic Poisson Model	NA	171.23	

3.5 Conclusions

Modeling accurately the cumulative number of software faults k in a software package is crucial to software developers. In the present study we have introduced a logistic software reliability growth model for the purpose of early estimation of the cumulative number of faults in a given software. The analytical form of this new model is given by

$$\widehat{L(t)} = \frac{\hat{k}}{1 + \widehat{m}e^{-\widehat{\alpha}t}} = \frac{i_{early}}{1 + e^{-\hat{\beta}_{0_{i_{early}}} e^{-\hat{\beta}_{1_{i_{early}}} t}}}$$

with

$$\begin{cases} \widehat{m} &= e^{-\hat{\beta}_{0_{i_{early}}}} \\ \widehat{\alpha} &= \hat{\beta}_{1_{i_{early}}} \end{cases}$$

where $\hat{\beta}_{0_{i_{early}}}$ and $\hat{\beta}_{1_{i_{early}}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, using $\widehat{k} = i_{early}$ and i_{early} the positive integer candidate at which slope is maximum and at which $\widehat{MSE_i}$ is minimal as the number of pairs increases.

This model not only gives very good predictions of the cumulative number of software faults, but it is easy to apply and it is free of any major assumptions.

The new model was compared with the following commonly used models to in the subject area

i) Exponential Model

ii) G-O Model

iii) Delayed S-Shaped Model

iV) S-Y Models

V) C-Model

Vi) Hypergeometric Model

Vii) Huang, Kuo, Chen, and Lyu's Models $[5, 6, 7, 9]$

For a first application, we used one set of actual data, namely, the PL/I Database Application test data by Ohba, $[10]$

Later we will apply our early estimation to the remaining sets analyzed in Chapter 2: i) the pattern of discovery of errors test Data by Tohma, $[11]$

ii) the F 11-D program test data by Forman and Singpurwalla, [13]

iii) the pattern of discovery of errors STS2, STS3, STS4 by Misra [15]

iV) Musa's System T1 V) the On-line data entry software package test Data by Ohba,[10]

Vi) A field report test Data by Tohma, [11, 16, 18]

The mean square error criteria was used to compare the results of our proposed model with other models stated above. The results presented in tables form support the fact that the new model is more effective in estimating the number of faults found during the testing and debugging phase of a given software.

Chapter 4

Reliability Growth Model For Software Reliability

4.1 Introduction

There are usually four ways of characterizing failures that occurs when evaluating software packages, [52]:

1. time of failure

2. time interval between failures

3. cumulative failures experienced up to a given time

4. failures experienced in a time interval.

It is further understood that each of the above phenomena is being characterized by a random variable.

In this Chapter we shall focus on predicting the time interval between failures for a given software package. The Mean Time Between Failure ($MTBF$) is the time difference between the expected next failure time and current failure time. When working with softwares packages during their development process, the mean time between failure ($MTBF$) is a very important concept when assessing the reliability of a given package after each modification. As a result, an accurate estimation of the $MTBF$ to predict the failure times of a given system is crucial when it comes to planning corrective strategies.

As software faults are found and corrected during the testing and debugging phase, with the assumptions discussed in Chapter 2, the reliability of the given software package increases meaning that the time between failure of the software is expected to be relatively increasing in the long run. Thus, the larger the time between failure (TBF) the more reliable the software becomes. Most of the existing models assume that the time between failures are exponentially distributed which make them independent of time , [47]. When a dependency on time is exhibited, authors proposed the

nonhomogeneous Poisson process $(NHPP)$ to measure the reliability growth of the software and make prediction about its failure behavior, [47]. Suresh and Rao developed a software reliability growth model based on the NHPP with intensity function characterized by the inverse power law process which uses an unbiased estimate of failure rate for prediction. The objective of this study is, to develop a logistic model, to characterize the software reliability of a given package. A proposed model to forecast the mean time between failures of a software package, after the last correction has been implemented. Our proposed software reliability growth model offers the following advantages:

(i) gives a very good mean time to next failure behavior prediction

(ii) provides accurate parameters estimation

(iii) it is simple to implement

(iV) the estimates are in closed form

(V) it is free of any major statistical assumptions

(Vi) tracks the failure rate with respect to increasing, decreasing or remaining constant.

In Section 4.2 we present an overview of a few models that are commonly used: Horigome-Singpurwalla model, Mazucchi-Soyer model, Suresh-Rao model [47], and Quiao-Tsokos model, [45]. In Section 4.3, we develop our logistic model to characterize behavior of the failures growth. Comparison of our proposed model that we will refer to as LR-MTBF-Model with the commonly used models is made using data from two real world problems in Section 4.4. Finally, our conclusions and recommendations are given in Section 4.5.

4.1.1 Statistical Abbreviations and Notations

- TBF: Time Between Failure
  - Horigome-Singpurwalla model: H-M model
  - Mazucchi-Soyer model: M-S model
  - Suresh-Rao SRGM model: S-R SRGM model
  - LR-MTBF-Model: our proposed model
  - BEBE model: Mazzuchi and Soyer's model
  - SRGM: Software Reliability Growth Model

4.2 Preliminaries

Let T_i, $i = 1, 2, 3, ..., n$ denote the age of the software as software faults are found and Y_i, $i = 1, 2, 3, ..., n$ denote the life length of the software at the ith stage of testing following a modification attempting to remove an error, [47]

Several models have been proposed for modeling the $MTBF$ and estimating software failure times when they can be characterized by the Power Law Process.

4.2.1 Bayes, Empirical-Bayes Model

Mazucchi and Soyer developed a Bayes Empirical-Bayes model using similar assumption as the Littlewood-Verrall model, [38, 47].

Bayesian setting

Given $Y_1, Y_2, Y_3, Y_4, ..., Y_n$.

1. The probability density function (pdf) of Y_i is given by:

$$f(y_i|\lambda) = \lambda_i e^{-\lambda_i y_i}, \quad y_i > 0 \tag{4. 1}$$

with failure rates depending on the stage of testing.

2. Given the failure rates at each stage of testing, the life-lengths of the software at each stage are statistically independent.

3. The failure rates at each stage of testing are random variables. The pdf of λ_i is given by:

$$g(\lambda_i|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} {\lambda_i}^{\alpha-1} e^{-\beta\lambda_i}, \quad \lambda_i > 0, \quad \alpha > 0, \quad \beta > 0 \tag{4. 2}$$

4. Given the parameters α and β, the failure rates at each stage of testing are statistically independent.

5. Given the background information H, uncertainty of α and β is expressed via the prior

marginal pdf's:

$$\Pi(\alpha|H) = \frac{1}{u} \quad 0 < \alpha < u \tag{4. 3}$$

and

$$\Pi(\beta|H) = \frac{b^a}{\Gamma(a)}\beta^{a-1}e^{-b\beta}, \quad \beta > 0 \tag{4. 4}$$

where $u > 0$, $a > 0$, and $b > 0$ are known quantities.

6. Given background, the information H, α and β are statistically independent.

7. Given α, β, and $(\lambda_1, \lambda_2, ..., \lambda_n)$, the Y_i's are statistically independent with each Y_i statistically independent of α, β, and all λ's other than λ_i.

Computation of the posterior failures rates leads to integrals which cannot be expressed in closed form, Lindley's approximation, [47] is used to approximate the integrals. Mazucchi and Soyer's model was reported to be an improvement over the littlewood/Verrall model, after applying this model to some actual software failure data first reported in [47].

4.2.2 Suresh-Rao SRGM

Suresh and Rao [47], extending the Nonhomogeneous Poisson Process to more appropriately describe the software failure process, developed a software reliability growth model with intensity function characterized by the inverse power law (Weibull process). This intensity, characterized as a function of the time , the number of errors found, and the time of occurrence of the last error, is reported to allow them to consider increasing and decreasing software failure rate that tends to change continually during test periods. Suresh and Rao's model, was reported to give a significantly better prediction of software reliability than Mazucchi and Soyer's model and time series model ,[47].

Starting in the testing phase of the development of a software, the failure intensity is given by:

$$\nu(t) = \lambda\delta t^{\delta-1}, \quad t > 0, \quad \lambda > 0, \quad \delta > 0 \tag{4. 5}$$

Letting $Y_i = T_i - T_{i-1}, i = 1, 2, ...,$ be the times between successive failures of the software. The time to first failure is the Weibull distribution with failure rate as above [47]. Then for $\delta < 1$, the software is improving with time. The conditional distribution of T_k given T_{k-1} , if n_{t_{k-1},t_k} is the

number of failure in the interval between t_{k-1} and t_k, is given by:

$$\Pr(T_k > t_k | T_{k-1} = t_{k-1}) = Pr(n_{t_{k-1},t_k}) = \exp(-\int_{t_{k-1}}^{t_k} \nu(t)dt) = \exp(-\lambda t_k^{\delta} + \lambda t_{k-1}^{\delta}), \quad t_k \geq t_{k-1}$$

The joint density of $T_1, T_2, ..., T_n$ is given by:

$$f(t_1, t_2, ..., t_n) = (\lambda\delta)^n \exp[-\lambda {t_n}^{\delta}] \prod_{i=0}^{n} t_i^{\delta-1}$$

Statistical inference procedures for the parameters λ and δ have been obtained by Crow, Flinkelstein, Lee and Lee, and by Engelhardt and Bain, [47]. From the likelihood function:

$$L(\lambda, \delta; t_1, t_2, ..., t_n) = (\lambda\delta)^n \exp[-\lambda t_n^{\delta}] \prod_{i=0}^{n} t_i^{\delta-1}$$

The maximum Likelihood estimators of the parameters λ, and δ using $T_1, T_2, ..., T_n$ are given by:

$$\widehat{\delta} = \frac{n}{\sum_{i=1}^{n} \log(\frac{T_n}{T_i})} \tag{4. 6}$$

$$\widehat{\lambda} = \frac{n}{T_n^{\widehat{\delta}}} \tag{4. 7}$$

The unbiased estimate of δ is derived to be:

$$\overline{\delta} = \tfrac{n-1}{n}\widehat{\delta}$$

Suresh obtained the unbiased estimate of ν_n:

$$\overline{\nu_n} = \frac{n\overline{\delta}}{T_n}$$

where $\overline{\delta}$ is not defined for $n = 0$ and $\overline{\delta} = 0$ for $n = 1$.

reported the estimator of mean time to next failure at T_n to be:

$$\frac{1}{\overline{\nu_n}}$$

.

4.2.3 Quiao-Tsokos Models

Quiao and Tsokos have formulated the basic structure of estimating the $MTBF$ and the intensity function for the Weibull process and derived bounds for estimating the expected time between the nth and $(n+1)th$ failures given the time of the nth failure, [45].
Quiao and Tsokos developed a procedure for obtaining the best efficient estimate of a set of efficient estimates for the Weibull Process. Then employed this statistical procedure to obtain the best efficient estimate to the shape parameter of the Weibull process. Additionnally, the Linearly derived best efficient estimates of the intensity function and its reciprocal when the Weibull process is being used in reliability analysis. Quiao and Tsokos demonstrated the effectiveness of their statistical procedure analytically and numerically with several examples, using the concept of relative efficiency, [45].

4.3 Development of The proposed Model

The objective of this section is to develop a software reliability growth model modeling the $MTBF$ to estimate software failure times, when the un-grouped cumulative number of software faults is S-shaped, without deriving an intensity function required for the Power Law, that

(i) does not assume any prior distribution

(ii) gives a very good $MTBF$ prediction in the MSE sense, providing better performance than the pre-existing models

(iii) is simple and easy to implement

(iV) does not require any form of approximation

The very important of our proposed model allows software developers to connect time interval between failures and cumulative failures experienced up to a given testing and debugging time, giving them more details about the software failure data under study, to predict when a software is ready to be released for use.

4.3.1 The Logistic Model

Let $L(t)$ be the cumulative amount of software faults found and corrected in the debugging phase during exposure time interval $[0, t]$. The parameter k ($k > 0$) represents the initial total number of software faults in the software prior to the debugging phase as in Chapter 2 and in [1].
The logistic curve model is described by the differential equation

$$\frac{dL(t)}{dt} = \frac{\alpha}{k} L(t)(k - L(t)), \qquad t \geq 0 \tag{4. 8}$$

where α ($\alpha > 0$) and k ($k > 0$) are constant parameters to be estimated by regression analysis, [1].
Let

$$P(t) = \frac{L(t)}{k}$$

Then, from Equation 4. 8, we have

$$\frac{dP(t)}{dt} = \alpha P(t)(1 - P(t)), \qquad t \geq 0, \qquad P(0) = \frac{1}{1+m}, \qquad m \gg k$$

where

$$P(t) = \frac{L(t)}{k}$$

is the cumulative fraction number of software failures found and corrected in the debugging phase during exposure time interval $[0, t]$.
The solution of the above differential equation was noted to be

$$P(t) = \frac{1}{1 + me^{-\alpha t}} \tag{4. 9}$$

where m is the constant of integration. Note that the graph of $P(t)$ is S-shaped and $0 \leq P(t) \leq 1$.
As developed in Chapter 2, we propose a Logistic Model to estimate the parameters k, m and α of $P(t)$.

4.3.2 Parameter Estimates

Following the proposed procedure in section 2.3 and plotting the estimated $\widehat{MSE}_i$ as i takes the values $i = N, N+1, N+2, ...$, we propose on using an estimate of k, $\widehat{k}$ where the minimum $\widehat{MSE}_i$ occurs, that is, $\widehat{k} = i_m$, where i_m is one of the positive integers $N, N+1, N+2, ...$ at which $\widehat{MSE}_i$ is minimal.

Having the estimate of k, $\widehat{k} = i_m$, we propose

$$\widehat{P(t)} = \frac{e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}{1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}$$

where $\hat{\beta}_{0_{i_m}}$ and $\hat{\beta}_{1_{i_m}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1 as describe in chapter 2; $\widehat{P(t)}$ given above estimates the true cumulative percentage number of software failures found and corrected during debugging and testing phase up to testing and debugging time t. Thus, an estimate of $P(t)$ can be obtained using

$$\widehat{P(t)} = \frac{1}{1 + \widehat{m} e^{-\widehat{\alpha} t}} = \frac{1}{1 + e^{-\hat{\beta}_{0_{i_m}}} e^{-\hat{\beta}_{1_{i_m}} t}}$$

Finally the cumulative failure behavior of the proposed model for a given software was derive to be:

$$\widehat{L(t)} = \frac{\hat{k}}{1 + \widehat{m} e^{-\widehat{\alpha} t}} = \frac{i_m}{1 + e^{-\hat{\beta}_{0_{i_m}}} e^{-\hat{\beta}_{1_{i_m}} t}}$$

4.3.3 Derivation of our proposed $LR - MTBF$ Model

Starting with

$$L(t + h) \approx L(t) + L'(t)h \tag{4. 10}$$

and dividing both sides by k, we obtain the following approximation:

$$\frac{L(t + h)}{K} \approx \frac{L(t)}{k} + \frac{L'(t)}{k} h \tag{4. 11}$$

which is equivalent to

$$P(t + h) \approx P(t) + P'(t)h \tag{4. 12}$$

from which we obtain

$$P(t+h) - P(t) \approx P'(t)h \tag{4. 13}$$

On the other hand, the probability that one software fault is found between t and $t+h$ is given by:

$$P(t+h) - P(t) = \frac{1}{k} \tag{4. 14}$$

Combining the above equations gives

$$\frac{1}{k} = P(t+h) - P(t) \approx P'(t)h \tag{4. 15}$$

from which we derive the following estimates:

$$\frac{1}{\hat{k}} = \widehat{P(t+h)} - \widehat{P(t)} \approx \widehat{P'(t)}h \tag{4. 16}$$

and obtain

$$h \approx \frac{1}{\hat{k}\widehat{P'(t)}} \tag{4. 17}$$

Note that since

$$\widehat{P(t)} = \frac{e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}{1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}$$

we differentiate $\widehat{P(t)}$ to obtain

$$\widehat{P(t)}' = \frac{\hat{\beta}_{1_{i_m}} e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}}{\left(1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} t}\right)^2}$$

Finally, our proposed estimated (time between the nth and the $n+1th$ failure) mean time to next failure at time T_n is given by:

$$LR - MTBF(T_n) = \frac{1}{\hat{k}\widehat{P'(T_n)}} = \frac{1}{\hat{k}\widehat{P(T_n)}'} = \frac{\left(1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} T_n}\right)^2}{i_m \hat{\beta}_{1_{i_m}} e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} T_n}} \tag{4. 18}$$

Note that our $MTBT$ prediction depends on our estimate $\hat{k}$ of the total number of faults in the Software. When at the end of a testing debugging phase we estimate that all the k faults in the software were found and corrected, the $MTBF$ get significantly large; in this case stop the $MTBF$ prediction at $LR - MTBF(T_{k-2})$. It will be shown that our proposed estimate of the mean times between failure gives good results in comparison to the other models that are commonly used.

4.4 Comparisons of Models: Numerical Applications to Software Failure Data

After analyzing two actual failure data sets to compare their model with some pre-existing models in terms of the mean square error, Suresh and Rao reported better performance of their model than the Mazucchi-Soyer model (BEBE model), the Littlewood-Verral model, The Singpurwalla Horigome model.

In this section we shall compare, using the mean square error, the Mazucchi-Soyer (BEBE), and Suresh-Rao model to our proposed model on two sets of actual software failure data obtained from actual projects. It will be shown to that our proposed model provides a very good modeling of the mean time between failure, when un-grouped cumulative number of software faults found and corrected during testing and debugging phase is S-shaped.

4.4.1 Apollo 8 software failure data: Comparison of the models

For application to actual software reliability failures data, and comparison of our proposed model with the previous models, we analyze a data set, the Apollo 8 software test data given by Table 27, on page 76 , that was first reported in Jelenski and Moranda in 1972, then studied by Verrall and Littlewood in 1973, by Mazuchi and Soyer in 1988 , by Suresh $[47]$ in 1992, and re-visited by Roberts in 2000, $[26]$.

From Table 28, on page 77 ,after applying our model proposed in Chapter 2, we note that the mean square error is minimal when the assumed cumulative number of failures is 28. Thus, our estimate $\widehat{K} = 28$. Table 28 also shows that by predicting 28 faults associated with a mean square error of 0.269812501 along with the following estimates $\hat{\beta}_{0_{im}} = -3.1653$and $\hat{\beta}_{1_{im}} = 0.0377$

From the results summarized in Table 29, on page 79 , we note that with a $MSE = 10.32$ our proposed model, LR-MTBF-Model, without any assumptions, demonstrates equal performance when compared to the Suresh-Rao Model associated with a $MSE = 10.34$; and compared the the BEBE Model with a $MSE = 11.41$ versus a $MSE = 19.486$ our proposed model demonstrates better performance then the BEBE model. The advantage of our proposed model is that it not only provides very good $MTBF$ prediction when un-grouped cumulative failure behavior is S-shaped Figure 16, on page 78, but it also give a very good cumulative software failure prediction Figure 15, on page 78, and estimates the total number of fault in the software allowing the testing and debugging team to decide when to release the software for use. Thus, our proposed model which is easier

to use and free of any major assumption gives very good if not better results then other models used in industry.

Table 27: Apollo 8 Failure Data

Failure Number	Actual TBF	Actual Failure Time
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	7	123
23	2	125
24	1	126

Table 28: MSE as ACNOF increases - Apollo 8 data

ACNOF	Mean Square Error
24	0.638528862
25	0.444854629
26	0.336156119
27	0.283498583
28	**0.269812501**
29	0.28087377
30	0.308914428
31	0.345979846
32	0.391210652
33	0.441151908
34	0.492927989
35	0.548889331
36	0.601757717
37	0.656688524
38	0.710669697
39	0.761844255
40	0.812992029
41	0.862548918
42	0.91197982

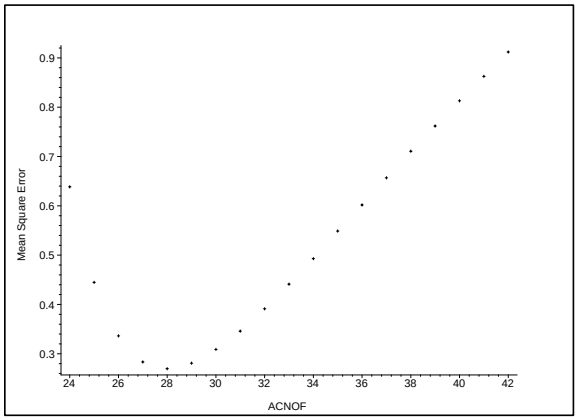

Figure 15: Plot of the MSE as ACNOF increases - Apollo 8 Failure data

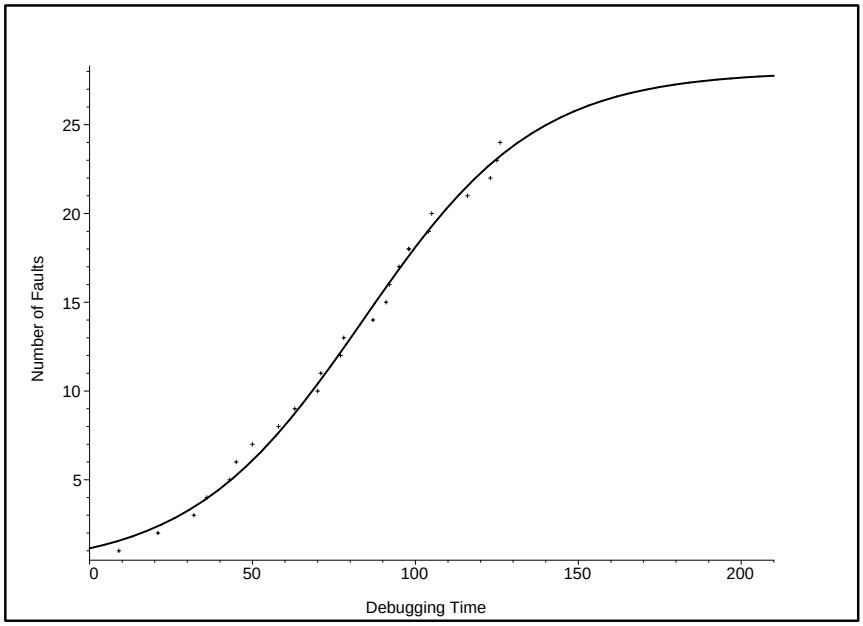

Figure 16: Comparison Of the Apollo 8 data set and our Predicted

Table 29: Actual and Predicted MTBF on the Apollo 8 data

Failure Number	Actual TBF	BEBE	SRGM	LRCRL
1	9	.....	.....	.....
2	12	10.53	.....	17.94
3	11	11.84	8.89	12.15
4	4	11.79	9.01	8.75
5	7	9.64	6.12	7.83
6	2	9.15	5.92	6.53
7	5	7.85	4.47	6.23
8	8	7.44	4.3	5.57
9	5	7.55	4.81	4.77
10	7	7.27	4.64	4.41
11	1	7.27	4.87	4.06
12	6	6.66	4.13	4.02
13	1	6.62	4.26	3.85
14	9	6.16	3.72	3.84
15	4	6.39	4.24	3.8
16	1	6.23	4.11	3.86
17	3	5.89	3.7	3.88
18	3	5.71	3.55	3.96
19	6	5.56	3.43	4.06
20	1	5.59	3.58	4.36
21	11	5.35	3.3	4.42
22	7	6.84	3.85	5.35
23	2	9.73	4.04	6.24
24	1	9.39	3.84	6.55
Mean Square Error		19.486		11.41
Mean Square Error		20.27	10.33	10.32

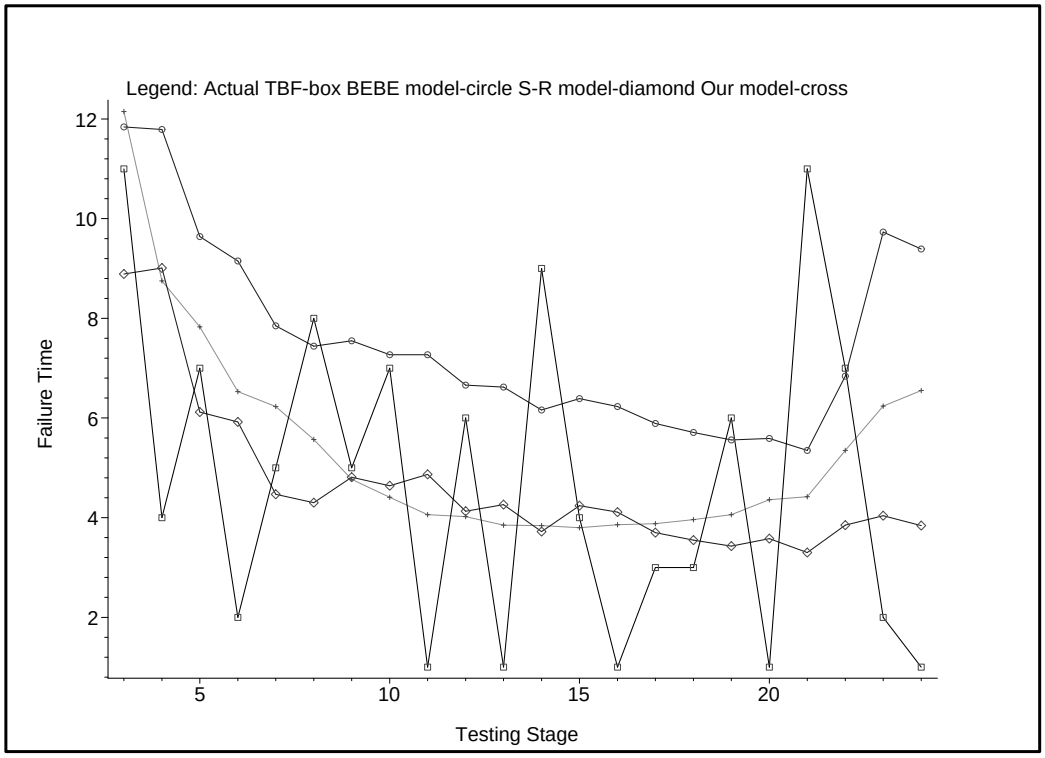

Figure 17: Comparison Of the Apollo 8 actual and Predicted MTBF

4.4.2 Musa's Project 14C software failure Data: Comparison of models

Here, we shall compare our model effectiveness with the Suresh-Rao model that was reported to perform better than the BEBE model that was an improvement over the Littlewood/Verall model. We analyze a set of software reliability field data call project $14C$ test data given by Table 30, on page 82, collected by John D. Musa of Bell Telephone Laboratories [52], to assist software managers in monitoring test status and predicting schedules and to assist software researchers in validating software reliability models. The project $14C$, an application of Real time command and control, is one of sixteen projects collected throughout the mid 1970s for which careful controls were used during data collection to ensure that the data would be of high quality, [53]; Its size is reported to be of hundreds of thousands delivered object code instructions. During the testing and debugging phase 36 failures were found.

From Table 31, on page 83 , after applying our model proposed in Chapter 2, we note that the mean square error is minimal when the assumed cumulative number of failures is 36. Thus, our estimate $\widehat{K} = 36$ which agrees with the total the number of faults detected during actual debugging. Table 31 also shows that by predicting 36 faults associated with a mean square error of 3.094882808 along with the following estimates $\hat{\beta}_{0_{i_m}} = -2.9125$ and $\hat{\beta}_{1_{i_m}} = 4.829.10^{-7}$.

From the results summarized in Table 32, on page 85, we note that with a $MSE = 2.79596.10^{11}$ our proposed model, LR-MTBF-Model, without any assumptions, demonstrates better performance when compared to the Suresh-Rao Model associated with a $MSE = 6.68654.10^{11}$. Once again, our proposed model not only provides very good $MTBF$ prediction when un-grouped cumulative failure behavior is S-shaped Figure 20, on page 86 , but it also give a very good cumulative software failure prediction Figure 19, on page 84, and estimates the total number of fault in the software allowing the testing and debugging team to decide when to release the software for use. Thus, our proposed model which is easier to use and free of any major assumption gives very good if not better results then other models used in industry.

Table 30: Musa's Project 14C Data

F.N.	F. I. L.	Cum. F. T.	F.N.	F. I. L.	Cum. F. T.
1	191520	191520	19	228315	5631060
2	2078820	2270340	20	51480	5682540
3	514560	2784900	21	44820	5727360
4	1140	2786040	22	850080	6577440
5	3120	2789160	23	361860	6939300
6	327480	3116640	24	39300	6978600
7	15420	3132060	25	545280	7523880
8	60000	3192060	26	256980	7780860
9	140160	3332220	27	396780	8177640
10	937620	4269840	28	91260	8268900
11	72240	4342080	29	1225620	9494520
12	737700	5079780	30	120	9494640
13	250680	5330460	31	1563300	11057940
14	2965	5333425	32	513000	11570940
15	196	5333621	33	177660	11748600
16	65173	5398794	34	2469000	14217600
17	2370	5401164	35	1678260	15895860
18	1581	5402745	36	170760	16066620

- F.N.: Failure Number
  - F. I. L.: Failure Interval Lenght
  - Cum. F. T.: Cumulative Failure Time

Table 31: MSE as ACNOF increases - Musa's Project 14C Data

ACNOF	Mean Square Error
36	**3.094882808**
37	3.959599606
38	4.928868157
39	5.931329159
40	6.942521785
41	7.92560927
42	8.878038857
43	9.791517534
44	10.66027927
45	11.47428189
46	12.25633147

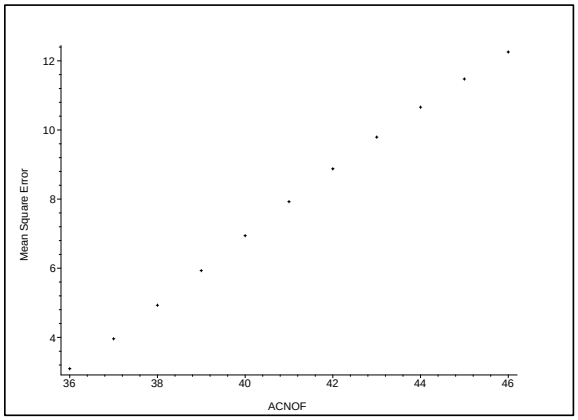

Figure 18: Plot of the MSE as ACNOF increases - Musa's Project 14C Data

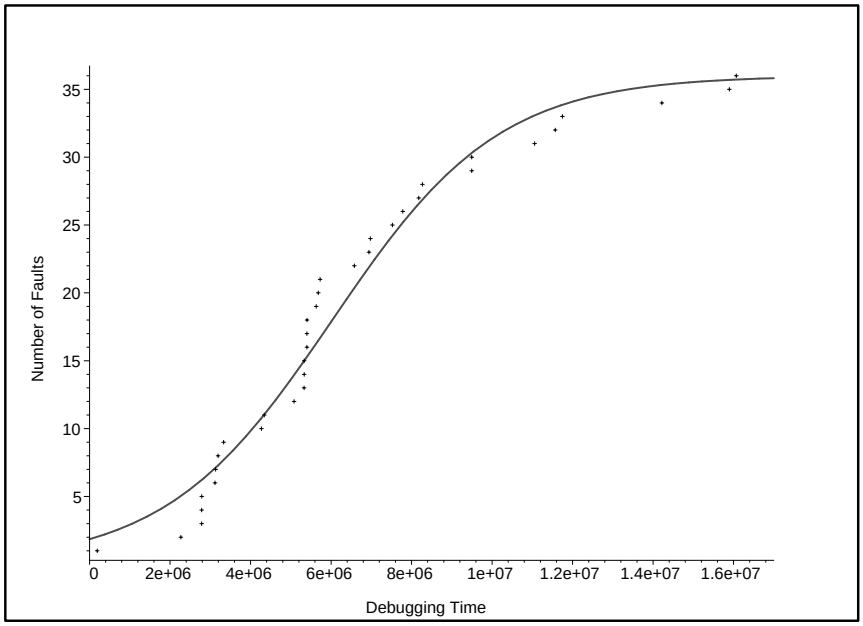

Figure 19: Comparison Of Musa's Project 14C Data and our Predicted

Table 32: Actual and Predicted MTBF on Musa's Project 14C Data

F. N.	TBF	S-R SRGM	LR-MTBF	F. N.	TBF	S-R SRGM	LR-MTBF
1	191520			19	228315	237626.3298	235431.213
2	2078820			20	51480	233864.8431	232246.5118
3	514560	1332150014	478061.3697	21	44820	214988.3137	231726.4976
4	1140	1.61403E+11	402889.9732	22	850080	198190.1049	231332.4906
5	3120	1.597E+11	402744.7631	23	361860	248290.2288	234115.9895
6	327480	5605186548	402347.7938	24	39300	255329.8335	241329.7101
7	15420	1.21607E+11	364141.6983	25	545280	237020.0068	242340.8523
8	60000	91508318446	362503.4189	26	256980	257738.5758	261292.4014
9	140160	46698576008	356258.5331	27	396780	256089.2429	273653.5196
10	937620	3.54226E+11	342450.7742	28	91260	264275.4747	297622.1508
11	72240	40820287309	274280.3111	29	1225620	251414.6349	304049.2101
12	737700	2.18242E+11	270535.9194	30	120	314021.1506	432154.5279
13	250680	67724173.19	242450.5302	31	1563300	293094.1114	432171.6521
14	2965	54651909080	236742.4777	32	513000	373699.4164	771808.0491
15	196	55927419308	236685.7869	33	177660	382993.8163	953889.7591
16	65173	29415353580	236682.0481	34	2469000	370730.9139	1028352.278
17	2370	54349168455	235499.0811	35	1678260	502013.8426	3113131.758
18	1581	54698599781	235458.3178	36	170760	579874.5015	.....
			S-R SRGM				LR-LRT
		MSE	6.68654E+11				2.79596E+11

- TBF: is the actual Time between Failure
- LR-MTBF: stands for our $MTBF$ prediction
- S-R SRGM: stands for the Suresh-Rao $MTBF$ prediction
- $E + 11$: 10^{11}

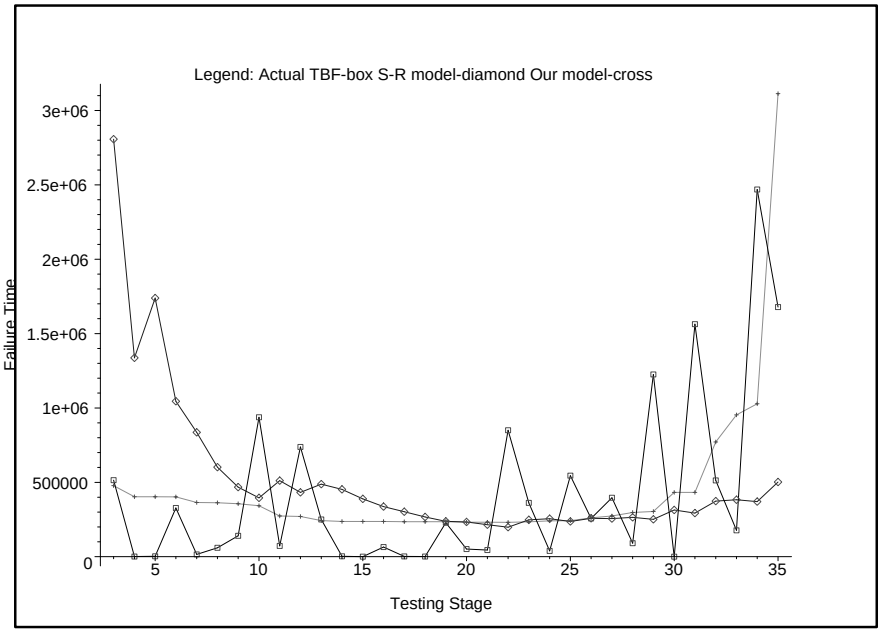

Figure 20: Comparison Of the Musa's Project 14C actual and Predicted MTBF

4.5 Conclusion

In the present study we have introduced a logistic software reliability growth model for the purpose of estimating the mean time between failure in a given software. The analytical form of this new model is given by

$$MTBF_{L.R.S} = \frac{1}{\hat{k}\widehat{P'(T_n)}} = \frac{1}{\hat{k}\widehat{P(T_n)}'} = \frac{\left(1 + e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} T_n}\right)^2}{i_m \hat{\beta}_{1_{i_m}} e^{\hat{\beta}_{0_{i_m}} + \hat{\beta}_{1_{i_m}} T_n}} \tag{4. 19}$$

where $\hat{\beta}_{0_{i_m}}$ and $\hat{\beta}_{1_{i_m}}$ are the Maximum Likelihood Estimators of the parameters β_0 and β_1, using $\widehat{k} = i_m$ and i_m the positive integer from $N, N+1, N+2, ...$ at which $\widehat{MSE}_i$ is minimal, as presented in Chapter 2

This model not only gives very good predictions of the mean time to next failure, but it is easy to apply and it is free of assumptions.

The new model was compared with the following commonly used models to in the subject area

i) Mazzuchi and Soyer will be referred to as the Bayes Empirical-Bayes Exponential (BEBE) model.

ii) Suresh-Rao SRGM

We used two different sets of actual data, namely,

i) the Apollo 8 Software Failure Data

ii) Musa's Project 14C software failure Data

The mean square error criteria was used to compare the results of our proposed model with other models stated above. The results presented in tables and graphical forms support the fact that the new model is more effective in estimating the mean time between failure during the testing and debugging phase of a given software.

Chapter 5

Logistic Models using Bayesian procedures for Software Reliability

5.1 Introduction

Softwares a getting larger and larger thus becoming more and more complex. Although software developers are not always able to test every software well enough to make them highly reliable, highly reliable software are required to prevent catastrophes in several crucial areas like medical, financial, defense, air traffic control, ...ect. Under the assumptions that once a software fault is found it is corrected for good and that its correction does not introduce any new software faults. As a result, we shall develop in this chapter two software reliability growth models, using Bayesian procedures. The models are based on two difference equations proposed by Morishita and Hirota, each of which is a discrete analog of the logistic curve model. As far as we can determine this is the first time that these software reliability growth models Bayesian in nature are derived from two difference equations discrete analogs of a logistic curve model proposed by Morishita and Hirota and used by Satoh and Yamada, [1]. Section 5.2 provides the answer to the question: why do we go Bayesian? and gives our underlying model.In Section 5.3, properties and advantages of the underlying model are presented. In Section 5.4, we implement a Bayesian procedure and derive the Bayesian estimates for the parameters of the underlying model. Application of the general Bayesian regression model is applied to Morishita's difference equation in Section 5.5. In Section 5.6, application of the general Bayesian regression model is applied to Hirota's difference equation. In section 5.7, we summarize the quantities depending on the actual data that are involved in the Bayesian estimates for six of the data sets previously studied in Chapter 2. Finally, our conclusions and recommendations are given in Section 5.8.

5.2 Logistic Curve Model and Bayesian Regression Models

A logistic curve model is described as:

$$\frac{dL(t)}{dt} = \frac{\alpha}{k} L(t)(k - L(t))$$

where $L(t)$ is the cumulative number of software failures that occurred up to testing time t; α ($\alpha > 0$) and k, ($k > 0$) are constant parameters to be estimated by regression analysis. Recall that the parameter k is the total number of software faults in the software before the testing and debugging phase.

In this Chapter, we assume that parameter k is a constant and that parameter α behave like a random variable with prior distribution $\pi(\alpha)$. α is directly proportional to the per capital growth rate at which the software faults are found. The rate at which the mistakes are found depends on the expertise of the tester, and *the difficulty in modeling realistic testing processes* (specially when multiple testers are used) which introduce randomness enter the structure and cause the parameter α to behave like a random variable.

Starting with Model I, as proposed by Satoh and Yamada,

$$y_i = A + Bx_i + \epsilon_i \qquad i = (1, ..., n) \qquad (ModelI)$$

In order to generalize Model I to a Bayesian Regression Model, assume that parameters A and B behave like random variables with joint prior distribution $\pi(A, B)$ and that $\epsilon_i \sim N(0, \sigma^2)$.

Rewriting Model I, as follows, gives:

$$y_i = (A + B\overline{x}) + B(x_i - \overline{x}) + \epsilon_i \qquad (5.\ 1)$$

$$y_i = \beta_0 + \beta_1 z_i + \epsilon_i \qquad (ModelII) \qquad (5.\ 2)$$

with:

$$\begin{cases} \beta_0 = A + B\overline{x} \\ \beta_1 = B \\ z_i = x_i - \overline{x} \end{cases} \qquad (5.\ 3)$$

5.3 Properties of Model II

Property 5.1

$$\widehat{\beta_0} = \overline{y} \tag{5. 4}$$

$$\widehat{\beta_1} = \frac{\sum_1^n y_i z_i}{\sum_1^n z_i^2} = \frac{\sum_1^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_1^n (x_i - \overline{x})^2} \tag{5. 5}$$

Proof. Let $SRS(\beta_0, \beta_1) = \sum_1^n (y_i - \beta_0 - \beta_1 z_i)^2$ to be minimized to find $\widehat{\beta_0}$ and $\widehat{\beta_1}$.

Differentiating $SRS(\beta_0, \beta_1)$ with respect to β_0 and setting equal to zero, gives:

$2\sum_1^n (y_i - \beta_0 - \beta_1 z_i)(-1) = 0$

$\Leftrightarrow \sum_1^n y_i - n\beta_0 - \beta_1 \sum_1^n z_i = 0$

since $\sum_1^n z_i = 0$, we get $\widehat{\beta_0} = \overline{y}$

Differentiating $SRS(\beta_0, \beta_1)$ with respect to β_1 and setting equal to zero, gives:

$2\sum_1^n (y_i - \beta_0 - \beta_1 z_i)(-z_i) = 0$

$\sum_1^n (y_i z_i - \beta_0 z_i - \beta_1 z_i^2) = 0 \Leftrightarrow$

$\sum_1^n y_i z_i - \beta_0 \sum_1^n z_i - \beta_1 \sum_1^n z_i^2 = 0$

which gives: $\widehat{\beta_1} = \frac{\sum_1^n y_i z_i}{\sum_1^n z_i^2}$

To prove second part of equation (ii) note that:

$\sum_1^n (x_i - \overline{x})(y_i - \overline{y}) = \sum_1^n y_i z_i$

□

Property 5.2 *$\widehat{\beta_0}$ and $\widehat{\beta_1}$ are independent*

Proof. Recall the theorem: Let $y_1, y_2, ..., y_n$ be i.i.d $N(\mu, \sigma^2)$ random variables then $\overline{y}$ and $(y_1 - \overline{y}, y_2 - \overline{y}, ..., y_n - \overline{y})$ are independent. From Property 1,

we have shown $\widehat{\beta_0} = \overline{y}$ and $\widehat{\beta_1} = \frac{\sum_1^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_1^n (x_i - \overline{x})^2}$

Since $\widehat{\beta_1}$ is a linear function of $(y_i - \overline{y})$ the conclusion follows. □

Property 5.3 *$\widehat{A}$ and $\widehat{B}$ are jointly sufficient for A and B*

β_0 and β_1 are jointly sufficient for $\widehat{A}$ and $\widehat{B}$

Thus, β_0 and β_1 are jointly sufficient for A and B.

Property 5.4

$$\widehat{\beta_0} \sim N(\beta_0, \frac{\sigma^2}{n}) \tag{5. 6}$$

$$\widehat{\beta_1} \sim N(\beta_1, \frac{\sigma^2}{\sum_1^n Z_i^2}) \tag{5. 7}$$

Proof. Let $y_1, y_2, ..., y_n$ be i.i.d $N(\mu, \sigma^2)$ random variables.

$\widehat{\beta_0}$ and $\widehat{\beta_1}$ are normally distributed, since they are linear combinations of independent normal random variables.

(i) $y_i = \beta_0 + \beta_1 z_i + \epsilon_i$

$y_i \sim N(\beta_0 + \beta_1 z_i, \sigma^2)$. Note that $E(y_i) = \beta_0 + \beta_1 z_i$

$E(\widehat{\beta_0}) = E(\overline{y}) = \frac{E(\sum_1^n y_i)}{n} = \frac{\sum_1^n E(y_i)}{n} = \frac{\sum_1^n (\beta_0 + \beta_1 z_i)}{n} = \frac{n\beta_0 + \beta_1 \sum_1^n z_i}{n} = \frac{n\beta_0}{n} = \beta_0,$

since $\sum_1^n z_i = 0$.

$Var(\overline{y}) = \frac{Var(\sum_1^n y_i)}{n^2} = \frac{1}{n^2} \sum_1^n Var(y_i) = \frac{1}{n^2} \sum_1^n \sigma^2 = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}.$

(ii) $E(\widehat{\beta_1}) = E(\frac{\sum_1^n y_i z_i}{\sum_1^n z_i^2}) = \frac{1}{\sum_1^n z_i^2} E(\sum_1^n y_i z_i) = \frac{1}{\sum_1^n z_i^2} \sum_1^n z_i E(y_i)$

$= \frac{1}{\sum_1^n z_i^2} \sum_1^n z_i(\beta_0 + \beta_1 z_i) = \frac{1}{\sum_1^n z_i^2} \sum_1^n (z_i \beta_0 + \beta_1 z_i^2)$

$= \frac{1}{\sum_1^n z_i^2} (\beta_0 \sum_1^n z_i + \beta_1 \sum_1^n z_i^2) = \frac{1}{\sum_1^n z_i^2} \beta_1 \sum_1^n z_i^2 = \beta_1.$

$Var(\widehat{\beta_1}) = Var(\frac{\sum_1^n y_i z_i}{\sum_1^n z_i^2}) = \frac{1}{(\sum_1^n z_i^2)^2} \sum_1^n z_i^2 Var(y_i) = \frac{1}{(\sum_1^n z_i^2)^2} \sum_1^n z_i^2 \sigma^2 =$

$\frac{\sigma^2}{(\sum_1^n z_i^2)^2} (\sum_1^n z_i^2) = \frac{\sigma^2}{\sum_1^n z_i^2}.$

□

Property 5.5 *$\widehat{\beta_0}$ and $\widehat{\beta_1}$ are respectively unbiased estimator of β_0 and β_1.*

Proof. In the proof of property 5.4, since both $E(\widehat{\beta_0}) = \beta_0$ and $E(\widehat{\beta_1}) = \beta_1$ were established, the conclusion follows. □

Property 5.6 *The unbiased Least Square estimate of σ^2 is s^2 given by:*

$$s^2 = \frac{\sum_1^n (y_i - \widehat{y_i})^2}{n-2} = \frac{\sum_1^n (y_i - \widehat{\beta_0} - \widehat{\beta_1} z_i)^2}{n-2} = (\frac{1}{n-2}) \left(\sum_1^n y_i^2 - \widehat{\beta_0} \sum_1^n y_i - \widehat{\beta_1} \sum_1^n z_i y_i \right)$$

Proof. By definition:

$$s^2 = \frac{\sum_1^n (y_i - \widehat{y_i})^2}{n-2}$$

Starting with our underlying model:

$$y_i = \beta_0 + \beta_1 z_i + \epsilon_i$$

since $\epsilon_i \sim N(0, \sigma^2)$ and $E(y_i) = \beta_0 + \beta_1 z_i$ we obtain:

$\widehat{y_i} = \widehat{\beta_0} + \widehat{\beta_1} z_i$

and

$$\sum_1^n (y_i - \widehat{y_i})^2 = \sum_1^n (y_i - (\widehat{\beta_0} + \widehat{\beta_1} z_i))^2 = \sum_1^n (y_i - \widehat{\beta_0} - \widehat{\beta_1} z_i))^2$$

which gives:

$$\sum_1^n (y_i - \widehat{y_i})^2 = \sum_1^n y_i^2 - \widehat{\beta_0} \sum_1^n y_i - \widehat{\beta_1} \sum_1^n z_i y_i$$

Finally, we obtain:

$$s^2 = \frac{\left(\sum_1^n y_i^2 - \widehat{\beta_0} \sum_1^n y_i - \widehat{\beta_1} \sum_1^n z_i y_i \right)}{n-2}$$

which complete the derivation.

□

5.4 Implementing Bayesian procedures

5.4.1 Bayesian procedures on Model I

Using Model I, we have:

$f(A, B|\overrightarrow{y}) = f(A, B|\widehat{A}, \widehat{B})$

$f(A, B|\widehat{A}, \widehat{B}) = f(A, B|\widehat{\beta_0}, \widehat{\beta_1}) = f(\widehat{\beta_0}, \widehat{\beta_1}|A, B)\pi(A, B) = f(\widehat{\beta_0}|A, B) f(\widehat{\beta_1}|A, B)\pi(A, B)$

where $\pi(A, B)$ is the joint prior of parameters A and B that need not be independent which is a disadvantage when using Model I. To avoid this problem, we propose working with Model II

5.4.2 Bayesian procedures on Model II

The posterior density is defined as:

$$f(\beta_0, \beta_1|\overrightarrow{y}) = f(\beta_0, \beta_1|\widehat{\beta_0}, \widehat{\beta_1}) = \frac{f(\widehat{\beta_0}, \widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0, \beta_1)}{\int f(\widehat{\beta_0}, \widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0, \beta_1) d\beta_0 d\beta_1}$$

Now using Model II and its properties

starting with:

$f(\widehat{\beta_0}, \widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_0, \beta_1)$

Since $\widehat{\beta_0}$ and $\widehat{\beta_1}$ are independent it is reasonable to assume that β_0 and β_1 are independent and write:

$$f(\widehat{\beta_0}, \widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0, \beta_1) = f(\widehat{\beta_0}|\beta_0, \beta_1)f(\widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0).\pi(\beta_1)$$

Thus, the posterior density using Model II is:

$$f(\beta_0, \beta_1|\overrightarrow{y}) = f(\beta_0, \beta_1|\widehat{\beta_0}, \widehat{\beta_1}) = \frac{f(\widehat{\beta_0}|\beta_0, \beta_1)f(\widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0).\pi(\beta_1)}{\int\int f(\widehat{\beta_0}|\beta_0, \beta_1)f(\widehat{\beta_1}|\beta_0, \beta_1).\pi(\beta_0).\pi(\beta_1)d\beta_0 d\beta_1}$$

5.4.3 Derivation of the Bayesian estimates

With our underlying model

$y_i = \beta_0 + \beta_1 z_i + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$

and using the Natural Conjugate prior approach, let's assume that

$\pi(\beta_0) \sim N(\mu_0, \tau_0^2)$ and $\pi(\beta_1) \sim N(\mu_1, \tau_1^2)$

where μ_0, μ_1, τ_0^2, and τ_1^2 are provided by experts.

To obtain a close form for the Bayesian estimates, we work with the square error loss function. Let's denote the Bayesian estimates of β_0 and β_1 by $\widehat{\beta}_{0B}$ and $\widehat{\beta}_{1B}$ respectively, under the square error loss function, we calculate the mean of the posterior density as follows:

$$\widehat{\beta}_{0B} = E(\beta_0|Data) = E(\beta_0|\widehat{\beta_0}, \widehat{\beta_1}) = \frac{\int\int \beta_0 f(\widehat{\beta_0}|\beta_0, \beta_1)f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_0)\pi(\beta_1)d\beta_0 d\beta_1}{\int\int f(\widehat{\beta_0}|\beta_0, \beta_1)f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_0)\pi(\beta_1)d\beta_0 d\beta_1}$$

which is equivalent to:

$$\widehat{\beta}_{0B} = E(\beta_0|Data) == \frac{\int \beta_0 f(\widehat{\beta_0}|\beta_0, \beta_1)\pi(\beta_0)d\beta_0 \int f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_1)d\beta_1}{\int f(\widehat{\beta_0}|\beta_0, \beta_1)\pi(\beta_0)d\beta_0 \int f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_1)d\beta_1}$$

and similarly we obtain:

$$\widehat{\beta}_{1B} = E(\beta_1|Data) == \frac{\int \beta_1 f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_1)d\beta_1 \int f(\widehat{\beta_0}|\beta_0, \beta_1)\pi(\beta_0)d\beta_0}{\int f(\widehat{\beta_0}|\beta_0, \beta_1)\pi(\beta_0)d\beta_0 \int f(\widehat{\beta_1}|\beta_0, \beta_1)\pi(\beta_1)d\beta_1}$$

with:

$$\widehat{\beta_0} \sim N(\beta_0, \frac{\sigma^2}{n})$$

$$\widehat{\beta_1} \sim N(\beta_1, \frac{\sigma^2}{\sum_1^n Z_i^2})$$

$$\beta_0 \sim N(\mu_0, \tau_0)$$

$$\beta_1 \sim N(\mu_1, \tau_1)$$

and our underlying model:

$$y_i = \beta_0 + \beta_1 z_i + \epsilon_i \tag{5. 8}$$

is such that:

$$y_i \sim N(\beta_0 + \beta_1 z_i, \sigma^2)$$

and

$$\epsilon_i \sim N(0, \sigma^2)$$

We obtain:

$$\widehat{\beta}_{0B} = \frac{\frac{1}{(2\pi)^{4/2}} \frac{1}{\tau_0 \tau_1 \frac{\sigma}{\sqrt{n}} \frac{\sigma}{\sqrt{\sum_i z_i^2}}} \int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\left(\frac{(\beta_0-\hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0-\mu_0)^2}{\tau_0^2}\right)} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_1-\hat{\beta}_1)^2}{\frac{\sigma^2}{\sum_i z_i^2}} + \frac{(\beta_1-\mu_1)^2}{\tau_1^2}\right)} d\beta_1}{\frac{1}{(2\pi)^{4/2}} \frac{1}{\tau_0 \tau_1 \frac{\sigma}{\sqrt{n}} \frac{\sigma}{\sqrt{\sum_i z_i^2}}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_0-\hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0-\mu_0)^2}{\tau_0^2}\right)} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_1-\hat{\beta}_1)^2}{\frac{\sigma^2}{\sum_i z_i^2}} + \frac{(\beta_1-\mu_1)^2}{\tau_1^2}\right)} d\beta_1}$$

which simplify to:

$$\widehat{\beta}_{0B} = \frac{\int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\left(\frac{(\beta_0-\hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0-\mu_0)^2}{\tau_0^2}\right)} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_1-\hat{\beta}_1)^2}{\frac{\sigma^2}{\sum_i z_i^2}} + \frac{(\beta_1-\mu_1)^2}{\tau_1^2}\right)} d\beta_1}{\int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_0-\hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0-\mu_0)^2}{\tau_0^2}\right)} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(\beta_1-\hat{\beta}_1)^2}{\frac{\sigma^2}{\sum_i z_i^2}} + \frac{(\beta_1-\mu_1)^2}{\tau_1^2}\right)} d\beta_1}$$

we note that

$$\frac{1}{2}\left(\frac{(\beta_0-\hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0-\mu_0)^2}{\tau_0^2}\right) = \frac{1}{2}\left((\frac{1}{\tau^2} + \frac{1}{\tau_0^2})\beta_0^2 - 2(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\beta_0 + (\frac{\hat{\beta}_0^2}{\tau^2} + \frac{\mu_0^2}{\tau_0^2})\right) \tag{5. 9}$$

Defining

$$\rho_1 = \frac{1}{\tau^2} + \frac{1}{\tau_0^2} \tag{5. 10}$$

and

$$\tau^2 = \frac{\sigma^2}{n} \tag{5. 11}$$

we obtain:

$$\frac{1}{2}\left[\frac{(\beta_0 - \hat{\beta}_0)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_0 - \mu_0)^2}{\tau_0^2}\right]$$

$$= \frac{1}{2}\rho_1\left[\beta_0^2 - \frac{2}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\beta_0\right] + \frac{1}{2}(\frac{\hat{\beta}_0^2}{\tau^2} + \frac{\mu_0^2}{\tau_0^2})$$

$$= \frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2 - \frac{1}{2\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2}) + \frac{1}{2}(\frac{\hat{\beta}_0^2}{\tau^2} + \frac{\mu_0^2}{\tau_0^2})$$

$$= \frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2 + \frac{(\hat{\beta}_0 - \mu_0)^2}{2(\tau_0^2 + \tau^2)}$$

Similarly, note that:

$$\frac{1}{2}\left(\frac{(\beta_1 - \hat{\beta}_1)^2}{\frac{\sigma^2}{\sum_i z_i^2}} + \frac{(\beta_1 - \mu_1)^2}{\tau_1^2}\right) = \frac{1}{2}\left((\frac{1}{\tau^2} + \frac{1}{\tau_1^2})\beta_1^2 - 2(\frac{\hat{\beta}_1}{\tau^2} + \frac{\mu_1}{\tau_1^2})\beta_1 + (\frac{\hat{\beta}_1^2}{\tau^2} + \frac{\mu_1^2}{\tau_1^2})\right) \tag{5. 12}$$

Defining

$$\rho_2 = \frac{1}{\nu^2} + \frac{1}{\tau_1^2} \tag{5. 13}$$

and

$$\nu^2 = \frac{\sigma^2}{\sum_i z_i^2} \tag{5. 14}$$

we obtain:

$$\frac{1}{2}\left[\frac{(\beta_1 - \hat{\beta}_1)^2}{\frac{\sigma^2}{n}} + \frac{(\beta_1 - \mu_1)^2}{\tau_1^2}\right]$$

$$= \frac{1}{2}\rho_2\left[\beta_1^2 - \frac{2}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\beta_1\right] + \frac{1}{2}(\frac{\hat{\beta}_1^2}{\nu^2} + \frac{\mu_1^2}{\tau_1^2})$$

$$= \frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2 - \frac{1}{2\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2}) + \frac{1}{2}(\frac{\hat{\beta}_1^2}{\nu^2} + \frac{\mu_1^2}{\tau_1^2})$$

$$= \frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2 + \frac{(\hat{\beta}_1 - \mu_1)^2}{2(\tau_1^2 + \nu^2)}$$

Thus,

$$\widehat{\beta}_{0B} = \frac{\int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2 - \frac{(\hat{\beta}_0 - \mu_0)^2}{2(\tau_0^2 + \tau^2)}} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2 - \frac{(\hat{\beta}_1 - \mu_1)^2}{2(\tau_1^2 + \nu^2)}} d\beta_1}{\int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2 - \frac{(\hat{\beta}_0 - \mu_0)^2}{2(\tau_0^2 + \tau^2)}} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2 - \frac{(\hat{\beta}_1 - \mu_1)^2}{2(\tau_1^2 + \nu^2)}} d\beta_1} \tag{5. 15}$$

Which simplify to:

$$\widehat{\beta}_{0B} = \frac{\int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2} d\beta_1}{\int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\nu^2} + \frac{\mu_0}{\tau_0^2})\right]^2} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2} d\beta_1} \tag{5. 16}$$

equivalently, we obtain

$$\widehat{\beta}_{0B} = \frac{\int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2} d\beta_0 \int_{-\infty}^{\infty} e^{-\frac{1}{2}\rho_2\left[\beta_1 - \frac{1}{\rho_2}(\frac{\hat{\beta}_1}{\nu^2} + \frac{\mu_1}{\tau_1^2})\right]^2} d\beta_1}{\frac{2\pi}{\sqrt{\rho_1\rho_2}}} \tag{5. 17}$$

then,

$$\widehat{\beta}_{0B} = \frac{\frac{\sqrt{2\pi}\sqrt{\rho_2^{-1}}}{\sqrt{2\pi}\sqrt{\rho_2^{-1}}} \int_{-\infty}^{\infty} \beta_0 e^{-\frac{1}{2}\rho_1\left[\beta_0 - \frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right]^2} d\beta_0 . \sqrt{2\pi}\sqrt{\rho_1^{-1}}}{\frac{2\pi}{\sqrt{\rho_1\rho_2}}} \tag{5. 18}$$

then,

$$\widehat{\beta}_{0B} = \frac{\sqrt{2\pi}\sqrt{\rho_2^{-1}}\left[\frac{1}{\rho_1}(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2})\right] . \sqrt{2\pi}\sqrt{\rho_1^{-1}}}{\frac{2\pi}{\sqrt{\rho_1\rho_2}}} \tag{5. 19}$$

which equate:

$\widehat{\beta}_{0B}$ of β_0:

$$\widehat{\beta}_{0B} = \frac{1}{\rho_1}\left(\frac{\hat{\beta}_0}{\tau^2} + \frac{\mu_0}{\tau_0^2}\right) \tag{5. 20}$$

Recalling that:

$$\rho_1 = \frac{1}{\tau^2} + \frac{1}{\tau_0^2} = \frac{\tau_0^2 + \tau^2}{\tau_0^2\tau^2} \tag{5. 21}$$

and

$$\tau^2 = \frac{\sigma^2}{n} \tag{5. 22}$$

Finally, after simplification, we obtain the following Bayesian estimate $\widehat{\beta}_{0B}$ of β_0:

$$\widehat{\beta}_{0B} = \frac{\tau_0^2}{\tau_0^2 + \frac{\sigma^2}{n}}\hat{\beta}_0 + \frac{\frac{\sigma^2}{n}}{\tau_0^2 + \frac{\sigma^2}{n}}\mu_0$$

Now, similarly to the derivation of $\widehat{\beta}_{0B}$, we derive $\widehat{\beta}_{1B}$ to be :

$$\widehat{\beta}_{1B} = \frac{\tau_1^2}{\tau_1^2 + \frac{\sigma^2}{\sum_i z_i^2}}\hat{\beta}_1 + \frac{\frac{\sigma^2}{\sum_i z_i^2}}{\tau_1^2 + \frac{\sigma^2}{\sum_i z_i^2}}\mu_1$$

Clearly, the Bayesian estimates of the parameters β_0 and β_1 are highly dependent on the choice of the parameters of the prior distributions $\pi(\beta_0)$ and $\pi(\beta_1)$. To obtain suitable value for the parameters μ_0, τ_0^2, μ_1, and τ_1^2, we propose the following procedure:

5.5 Our Bayesian procedure on Morishita difference equation

$$L_{n+1} - L_n = \frac{\delta\alpha}{k} L_{n+1}(k - L_n)$$

Rewriting as Model I:

$$\frac{L_{n+1}}{L_n} = \frac{-\delta\alpha}{k(1-\delta\alpha)} L_{n+1} + \frac{1}{1-\delta\alpha} \tag{5. 23}$$

$$\tag{5. 24}$$

$$y_n = BL_{n+1} + A \tag{5. 25}$$

where:

$$\begin{cases} y_n = \frac{L_{n+1}}{L_n} \\ B = \frac{-\delta\alpha}{k(1-\delta\alpha)} \\ A = \frac{1}{1-\delta\alpha}. \end{cases} \tag{5. 26}$$

Rewriting as Model II

$$\frac{L_{n+1}}{L_n} = \left(\frac{1}{1-\delta\alpha} - \frac{\delta\alpha}{k(1-\delta\alpha)}\overline{L_{n+1}}\right) - \frac{\delta\alpha}{k(1-\delta\alpha)}\left(L_{n+1} - \overline{L_{n+1}}\right) + \epsilon_n \quad (5.\ 27)$$

$$(5.\ 28)$$

$$y_n = \beta_0 + \beta_1 z_n + \epsilon_n \quad (5.\ 29)$$

where:

$$\begin{cases} \beta_0 = \frac{1}{1-\delta\alpha} - \frac{\delta\alpha}{k(1-\delta\alpha)}\overline{L_{n+1}} \\ \beta_1 = -\frac{\delta\alpha}{k(1-\delta\alpha)} \\ z_n = L_{n+1} - \overline{L_{n+1}} \end{cases} \quad (5.\ 30)$$

Given the estimates of the regression coefficients, under our proposed Bayesian procedure, $\widehat{\beta}_{0B}$ and $\widehat{\beta}_{1B}$ of the parameters β_0 and β_1 respectively,we can write:

$$\begin{cases} \hat{\beta}_{0_B} = \frac{1}{1-\delta\hat{\alpha}} - \frac{\delta\hat{\alpha}}{\hat{k}(1-\delta\hat{\alpha})}\overline{L_{n+1}} \\ \hat{\beta}_{1_B} = -\frac{\delta\hat{\alpha}}{\hat{k}(1-\delta\hat{\alpha})} \end{cases} \quad (5.\ 31)$$

where, $\hat{\alpha}$ and $\hat{k}$, respectively, our proposed estimates for the parameters α and k can be derive as follows:

Starting with

$$\hat{\beta}_{0_B} = \frac{1}{1-\delta\hat{\alpha}} + \hat{\beta}_{1_B}\overline{L_{n+1}} \quad (5.\ 32)$$

we derive our estimate $\hat{\alpha}$ of the parameter α as follows:

$$\frac{1}{1-\delta\hat{\alpha}} = \hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}} \quad (5.\ 33)$$

$$\frac{1}{\hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}}} = 1 - \delta\hat{\alpha} \quad (5.\ 34)$$

$$\delta\hat{\alpha} = 1 - \frac{1}{\hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}}} \quad (5.\ 35)$$

$$\hat{\alpha} = \frac{1}{\delta}[1 - \frac{1}{\hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}}}] \quad (5.\ 36)$$

Knowing $\hat{\alpha}$ and using

$$\hat{\beta}_{1_B} = -\frac{\delta\hat{\alpha}}{\hat{k}(1-\delta\hat{\alpha})} \tag{5. 37}$$

we obtain our estimate $\hat{k}$ of the parameter k

$$\hat{k} = -\frac{\delta\hat{\alpha}}{\hat{\beta}_{1_B}(1-\delta\hat{\alpha})} \tag{5. 38}$$

We use, under Morishita, the following estimates of m and α_c (alpha continous) as in [1]:

$$\hat{m} = \frac{\sum_{n=1}^{N}(\hat{k} - L_n)}{\sum_{n=1}^{N}(L_n(1-\delta\hat{\alpha})^n)} \tag{5. 39}$$

$$\widehat{\alpha}_c = -\frac{1}{\delta}\log(1-\delta\hat{\alpha_{dm}}) \tag{5. 40}$$

5.6 Our Bayesian procedure on Hirota difference equation

$$L_{n+1} - L_n = \frac{\delta\alpha}{k}L_n(k - L_{n+1})$$

Rewriting as Model I:

$$\frac{L_{n+1}}{L_n} = \frac{-\delta\alpha}{k}L_{n+1} + (\delta\alpha + 1) \tag{5. 41}$$

$$y_n = BL_{n+1} + A \tag{5. 42}$$

where:

$$\begin{cases} y_n &= \frac{L_{n+1}}{L_n} \\ B &= \frac{-\delta\alpha}{k} \\ A &= \delta\alpha + 1 \end{cases} \tag{5. 43}$$

Rewriting as Model II

$$\frac{L_{n+1}}{L_n} = \left((\delta\alpha+1) - \frac{\delta\alpha}{k}\overline{L_{n+1}}\right) - \frac{\delta\alpha}{k}\left(L_{n+1} - \overline{L_{n+1}}\right) + \epsilon_n \tag{5. 44}$$

$$y_n = \beta_0 + \beta_1 z_n + \epsilon_n \tag{5. 45}$$

where:

$$\begin{cases} \beta_0 &= \delta\alpha + 1 - \frac{\delta\alpha}{k}\overline{L_{n+1}} \\ \beta_1 &= -\frac{\delta\alpha}{k} \\ z_n &= L_{n+1} - \overline{L_{n+1}} \end{cases} \tag{5. 46}$$

Given the estimates of the regression coefficients, under our proposed Bayesian procedure, $\widehat{\beta}_{0B}$ and $\widehat{\beta}_{1B}$ of the parameters β_0 and β_1 respectively,we can write:

$$\begin{cases} \hat{\beta}_{0_B} &= \delta\hat{\alpha} + 1 - \frac{\delta\hat{\alpha}}{\hat{k}}\overline{L_{n+1}} \\ \hat{\beta}_{1_B} &= -\frac{\delta\hat{\alpha}}{\hat{k}} \end{cases} \tag{5. 47}$$

where, $\hat{\alpha}$ and $\hat{k}$, respectively, our proposed estimates for the parameters α and k can be derive as follows:

Starting with

$$\hat{\beta}_{0_B} = \delta\hat{\alpha} + 1 + \hat{\beta}_{1_B}\overline{L_{n+1}} \tag{5. 48}$$

we derive our estimate $\hat{\alpha}$ of the parameter α as follows:

$$\delta\hat{\alpha} = \hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}} - 1 \tag{5. 49}$$

$$\hat{\alpha} = \frac{1}{\delta}[\hat{\beta}_{0_B} - \hat{\beta}_{1_B}\overline{L_{n+1}} - 1] \tag{5. 50}$$

Knowing $\hat{\alpha}$ and using

$$\hat{\beta}_{1_B} = -\frac{\delta\hat{\alpha}}{\hat{k}} \tag{5. 51}$$

we obtain our estimate $\hat{k}$ of the parameter k

$$\hat{k} = -\frac{\delta\hat{\alpha}}{\hat{\beta}_{1_B}} \tag{5. 52}$$

We use, under Hirota, the following estimates of m and α_c (alpha continous) as in [1]:

$$\hat{m} = \frac{\sum_{n=1}^{N}(\hat{k} - L_n)}{\sum_{n=1}^{N}\left(L_n(\frac{1}{1+\delta\hat{\alpha}})^n\right)} \tag{5. 53}$$

$$\widehat{\alpha}_c = \frac{1}{\delta}\log(1 + \delta\hat{\alpha_{dh}}) \tag{5. 54}$$

5.7 Numerical Application: Summary of the data dependent quantities

In this section we shall prepare six sets of actual software failure data obtained from actual projects, already analyzed in Chapter 2, by evaluating all the actual data dependent quantities involved in the proposed Bayesian estimates.

i) PL/I software Failure Data by Ohba,[10]

ii) Tohma's Software Failure Data, [11]

iii) the F 11-D program test data by Forman and Singpurwalla, [13]

iV) Misra's Space Shuttle Software Failure Data [15]

V) Musa's System T1 software Failure Data

Vii) Tohma's field report test Data [11, 16, 18]

we have listed the values needed for the calculation of our Bayesian estimates

values to be used in the Bayesian estimates for Data set 1

$$n = 18$$

$$\sum_i z_i^2 = 161920.98$$

$$\hat{\beta}_0 = 1.235837821$$

$$\hat{\beta}_1 = -0.001422003$$

$$s^2 .... > \sigma^2 = 0.199206243$$

$$\overline{L_{n+1}} = 199.9473684$$

values to be used in the Bayesian estimates for Data set 2

$$n = 21$$

$$\sum_i z_i^2 = 14335.3204$$

$$\hat{\beta}_0 = 1.397320307$$

$$\hat{\beta}_1 = -0.018307372$$

$$s^2.... > \sigma^2 = 1.006263898$$

$$\overline{L_{n+1}} = 53.81818182$$

values to be used in the Bayesian estimates for Data set 3

$$n = 14$$

$$\sum_i z_i^2 = 15210.9926$$

$$\hat{\beta}_0 = 1.229176596$$

$$\hat{\beta}_1 = -0.001399806$$

$$s^2.... > \sigma^2 = 0.082157217$$

$$\overline{L_{n+1}} = 66.46666667$$

values to be used in the Bayesian estimates for Data set 4

$$n = 37$$

$$\sum_i z_i^2 = 129179.97$$

$$\hat{\beta}_0 = 1.080216539$$

$$\hat{\beta}_1 = -0.000178343$$

$$s^2.... > \sigma^2 = 0.008896242$$

$$\overline{L_{n+1}} = 124.0789474$$

values to be used in the Bayesian estimates for Data set 5

$$n = 18$$

$$\sum_i z_i^2 = 38542.5842$$

$$\hat{\beta}_0 = 1.272513145$$

$$\hat{\beta}_1 = -0.001410044$$

$$s^2 .... > \sigma^2 = 0.147405441$$

$$\overline{L_{n+1}} = 58.36842105$$

values to be used in the Bayesian estimates for Data set 7

$$n = 110$$

$$\sum_i z_i^2 = 2349985.9$$

$$\hat{\beta}_0 = 1.049033124$$

$$\hat{\beta}_1 = -0.000436099$$

$$s^2 .... > \sigma^2 = 0.015019537$$

$$\overline{L_{n+1}} = 358.7387387$$

5.8 Conclusion

In the present study we have introduced two software reliability growth models that yield accurate parameter estimates, using Bayesian procedures. The models are based on two difference equations proposed by Morishita and Hirota, each of which is a discrete analog of the logistic curve model [1], for the purpose of estimating the cumulative number of faults in a given software. The analytical form of these two new models have been derived.

Working with the square error loss function, the Bayesian estimates derived are very sensitive to the choice of the priors. As a result, after a careful investigation of the sensitivity of the obtained

Bayesian estimates with respect to the choice of the priors, we hope to be able to illustrate our approach and test its effectiveness.

Chapter 6

Future Work and Extensions

The present study enable us to identify several potential directions to extend and apply our results. In this chapter, we shall briefly discuss some possible extensions for our research findings to more realistic and useful analytical extensions.

In Chapter 2, we have developed of a simple, realistic, and easy to implement software reliability growth model that provides a decision rule as of when to stop the testing and debugging phase and release the software for use, for S-shaped cumulative software faults. We can use a Bayesian or Quasi-Bayesian procedure in the present development of the proposed Model.

In Chapter 3, using the main feature of our proposed Model in Chapter 2- its inflection point, we have proposed an effective method for estimating the number of faults in the software, at an early stage of the testing and debugging phase. Our early estimation of the number of fault in the software enable the software developers to plan the software development process, manage their resources by avoiding cost due to over testing, make a software with higher reliability and decide when to ship it for use. We need to develop a cost reduction analysis associated with our Early Estimation proposed Model.

In Chapter 4, assuming a logistic model, we develop a procedure for predicting the mean time between failure, after the last correction, of a software package creating a connection between predicting the $MTBF$ and counting the cumulative failures experienced up to a given time. For the prediction of the $MTBF$, one possible extension is to introduce a Bayesian or Quasi-Bayesian procedure in the development of the proposed Model.

In Chapter 5, Using Bayesian procedures, we have developed two discrete software reliability growth models based on two difference equations discrete analog of the logistic curve model respectively proposed by Morishita and Hirota. we shall illustrated that the Bayesian Approach to reliability is very useful modeling in understanding the final evaluation of a software package. One of the key difficulties is to identifying and justifying the choice of the prior. Thus, we propose to

develop an Empirical Bayes approach to Software Reliability and thus bypassing having to assume the choice of the prior. Furthermore, we extend to address the same problem from a nonparametric point of view by utilizing Kernel Density estimation procedures to characterize the behavior of the prior.

We also believe that formulating nonparametric Software Reliability models using the kernel density approach to characterize software failure would be of significant importance in cases where a classical distribution cannot be identify to statistically fit the software failures.

References

[1] D. Satoh and S.Yamada, "Parameter Estimation of Discrete Logistic Curves Models for Sofware Reliability Assessement," Japan J. Indust. Appl. Math., $19(2002), 39 - 53$

[2] F. Morishita, "The fitting of the logistic equation to the rate of increase of population density," Res. Popul. Ecol. $VII(1965), 52 - 55$

[3] "Computer science and statistics : proceedings of the Sixteenth Symposium on the Interface," Atlanta, 1984.

[4] D. W. Hosmer and S. Lemeshow, "Applied logistic regression," John Wiley and Sons, NY, 1989.

[5] C. Y., J. H. Lo, and S.Y. Kuo, "Pragmatic study of Parametric Decomposition Models for Estimating Software Reliability Growth," Proc. $9th$ Int'l. Symp. Software Reliability Engineering (ISSRE'98), 1998, pp. $111 - 123$.

[6] C. Y. Huang, S. Y. Kuo, and I. Y. Chen, "Analysis of a Software Reliability Growth Model with Logistic Testing-Effort Function," Proc. $8th$ Int'l. Symp. Software Reliability Engineering (ISSRE'97), 1997, pp. $378 - 388$.

[7] C. Y. Huang, S. Y. Kuo, and M. R. Lyu, "Effort-Index-Based Software Reliability Growth Models and Performance Assessment," Proc. $24th$ Ann. Int'l. Computer Software and Applications Conf. (COMPSAC 2000), 2000.

[8] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Applying Various Learning Curves to Hyper-Geometric Distribution Software Reliability Growth Model," Proc. $5th$ Int'l. Symp. Software Reliability Engineering, 1994, pp. $7 - 16$.

[9] S. Y. Kuo, C. Y. Huang, and M. R. Lyu, "Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates," IEEE TRANSACTIONS ON RELIABILITY, VOL. 50, NO. 3, SPTEMBER 2001.

[10] M. Ohba, "Software reliability analysis models," IBM J. Res. and Development, vol. 28, no. 4, pp. $428 - 443$, Jul. 1984.

[11] Y. Tohma , R. Jacoby, Y. Murata, and M. Yamamoto, "Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults," Proc. COMPsac-89, Orlando, pp. $610 - 617$, September 1989.

[12] Y. Tohma, H. Yamamoto and R. Jacoby, "Parameters Estimation of the Hyper-Geometric Distribution for Real/Test Data," Proc. of the $18th$ International Symposium on Fault Tolerant Computing, pp. $148 - 153$, 1988, Japan.

[13] E. H. Forman and N. D. Singpurwalla, "An Empirical Stopping Rule for Debugging and Testing Computer Software," J. American Statistical Association, Vol. 72, December 1977, pp. $750 - 757$.

[14] C. J. Dale and L. N. Harris: "Approach to Software Reliability Prediction," Proc. Ann. Reliability and Maintainability Symposium, pp. $167 - 175$, 1982.

[15] P. N. Misra: "Software reliability analysis," IBM Systems Journal, Vol 22, NO 3, 1983.

[16] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The Estimation of Parameters of the Hypergeometric Distribution and Its Application to the Software Reliability Growth Model," IEEE, 1991.

[17] T. Doli, N. Wakana, S. Osaki, and K. S. Trivedi, "Analysis of Hypergeometric Distribution Software Reliability Model," IEEE, 2001.

[18] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution," IEEE, 1989.

[19] M. A. Vouk: "Software Reliability Engineering," 2000 Annual Reliability and Maintainability Symposium.

[20] C. M. Bishop: "Bayesian Regression and Classification," Advances in learning Theory: Methods, Models and Applications, J. A. K. Suykens et al. (Editors), IOS Press, NATO Science Series III: Computer and Systems Sciences, volume "190".

[21] S. Yamada, J. Hishitani, and S. Osaki, "Software-Reliability Growth with a Weibull Test-Effort: A Model and Application," IEEE, 1993.

[22] El-Aroui: "A Bayes Nonparametric Framework for Software-reliability Analysis," IEEE, 1996.

[23] Crow: "An Extended Reliability Growth Model For Managing And Assessing Corrective Actions," IEEE, 2004.

[24] C. Y. Huang, C. T. Lin, S. Y. Kuo, M. R. Lyu, and C. C. Sue, "Software Reliability Growth Models Incorporating Fault Dependency with Various Debugging Times Lags," IEEE, 2004.

[25] "Computer science and statistics : proceedings of the Seventeenth Symposium on the Interface," Lexing, 1985.

[26] Roberts, H., "Predicting the Performance of Software Systems via the Power Law Process," Ph.D. Dissertation, University of South Florida, December 2000.

[27] Kjell A. Doksum, "Mathematical and statistical methods in reliability," Bo H. Lindqvist, 2003.

[28] Isaac Elishakoff, "Safety factors and reliability : friends or foes?" Kluwer Academic Publishers, Boston, 2004.

[29] J. D. Musa, "Software reliability data," report and database available from: Data and Analysis Center for Software; Rome Air Development Center (RADC), Rome, NY, USA.

[30] Bowyer, K.W., "Ethics and Computing: Living Responsibly in a Computrized World," Los Alamitos, CA: IEEE Computer Society Press, 1996.

[31] "proceedings annual Reliability and Maintainability Symposium," Los Angeles, Ca., USA, 1982.

[32] Benington, H. D., "Production of Large Computer Programs", in Proceedings, ONR Symposium on Advanced Programming Methods for Digital Computers, June 1956, pp. $15 - 27$.

Reprinted in Proceedings, $9th$ International Conference on Software Engineering, Monterey, CA, USA, March 30-April 2, 1987, pp. $299 - 310$.

[33] Scott H. Dick, "Computational intelligence in software quality assurance," Ph.D. Dissertation, University of South Florida, 2002.

[34] Bowyer, K. W., "Ethics and Computing: Living Responsability in a Computerized World", Los Alamitos, CA: IEEE Computer Society Press, 1996.

[35] Cai, K. Y., Wen, C.Y.; Zhang, M. L., "A critical review on software reliability modeling," Reliability Engineering and System Safety, vol. 32 no.3, 1991, pp. $357 - 371$.

[36] J. Higgins, C.P. Tsokos, "A quasi-bayes estimate of the failure intensity of a reliability growth model," IEEE Trans. Reliability, vol R-30, No.5, 1981, pp. $471 - 475$.

[37] M. Horigome, N.D. Singpurwalla, R. Soyer, "A Bayes Empirical Bayes Approach For (Software) Reliability growth," Computer Science and Statistics, The Interface, 1985, pp. $47 - 55$; North Holland.

[38] B. Littlewood, J.L. Verrall, "A Bayesian reliability growth model for computer software,"Applied Statistics vol 22, 1973, pp. $332 - 346$.

[39] G.J. Schick, R.W. Wolverton, "Assessment of software reliability," Proceedings of Operations Research, Physica- Verlag, Wurzburg-Wien, 1973, pp. $395 - 422$.

[40] Shimi, I. N. and Tsokos, C.P. (1977). "The Bayesian and nonparametric approach to reliability studies: a survey of recent work", in " Proc. Of the Conf. On the Theory and Applications of Relaibility With Emphasis on Bayesian and Nonparametric Methods," vol.1, Acad. Press, New York, pp. $5 - 47$.

[41] Tsokos, C., and Rao, A. (1994). "Estimation of Failure Intensity for the Weibull Process," Reliability Engineering and System Safety, 45 pp. $271 - 275$.

[42] Berger, J. (1985). "Statistical Decision Theory and Bayesian Analysis," $2nd$ Edition, Springer-Verlag, New York, Berlin, and Heidelberg.

[43] Canavos, G.C. (1972). "Bayesian approach to parameter and reliability estimation in the Poisson distribution," IEEE Trans. Reliab.R-21, No. 1, 52-56.

[44] Ann Marie Neufelder, "Ensuring software reliability", Marcel Dekker,Inc., NY, 1993.

[45] Qiao, H.; Tsokos, C.P., "Estimation of the three parameter Weibull probability distribution," Mathematics and Computers in Simulation, vol. 39, 1995, pp. $173 - 185$.

[46] Tsokos, C.P. et al. (1973). "Bayesian estimation of life parameters in the Weibull distribution," J. of Operations Research 21, No.3, pp. $755 - 763$.

[47] Nalina Suesh, "Modeling and analysis of software reliability", Ph.D. Dissertation, University of South Florida, 1992.

[48] F.W. Brefogle, "Statistical Methods for Testing, Development and Manufacturing," John Wiley and Sons, New York, NY (1991).

[49] H.F. Martz and R. A. Waller, "Bayesian Reliability Analysis," John Wiley and Sons, New York, NY (1982).

[50] V.K. Rohatgi, "An Introduction to Probability Theory and Mathematical Statistics," Wiley Eastern Limited (1990).

[51] Fellenburg, B. and Pilz, J. (1985). "On the choice of prior distribution for Bayesian reliability analysis,"Freiberger Forsch D-170, $49 - 68$.

[52] John D. Musa, Anthony Iannino, Kazuh, "Software reliability : measurement, prediction, application," 1990.

[53] DACS, "DACS Software Reliability Dataset,"
http:// www.dacs.dtic.mil/datavases/sled/swel.shtml. Data and Analysis Center for Software .

[54] Everett, W.W.; Musa, J.D., "A software engineering practice," IEEE Computer, vol. 26 no. 3 March 1993, pp. $77 - 79$.

[55] Farr, W., "Software Reliability Modeling Survey," Handbook of Software Reliability Engineering, New York: McGraw- Hill, 1996, pp. $71 - 115$.

[56] Friedman, M.A.; Voas, J. M., "Software Assessment: Reliability Safety, Testability," New York: John Wiley and Sons, Inc. 1995.

[57] Goel, A.L., "Software reliability models: assumptions, limitations, and applicability," IEEE Transactions on Software Engineering, vol. 11 no. 12, Dec. 1985, pp. $1411 - 1423$.

[58] Goel, A.L.; Okumoto, K., "Time -dependent error-detection rate model for software reliability and other performance measures," IEEE Transactions on Reliability, vol. 28 no. 3, August 1979, pp. $206 - 211$.

[59] Jelinski, Z.; Moranda, P.B., "Software reliability research," in Proceedings, Statistical Computer Performance Evaluation, November $22 - 23$, 1971, Providence, RI, USA, pp. $465 - 484$.

[60] Lewis, E.E., "Introduction to Reliability Engineering," $2nd$ Ed., New York: John Wiley and Sons , Inc., 1996.

[61] Lyu, M.R., Ed., "Handbook of Software Reliability Engineering," New York: McGraw-Hill, 1996.

[62] Mazzuchi,T.A.; Soyer, R., "A Bayes-empirical Bayes model for software reliability," IEEE Transactions on Reliability, vol. 37 no.2, June 1988, pp. $248 - 254$.

[63] Wallace R. Blischke, D.N. Prabhakar Murthy, "Reliability : modeling, prediction, and optimization," 2000.

[64] Ross, S.M., "Software reliability: the stopping rule problem," IEEE Transactions on Software Engineering, vol. 11 no. 12, Dec. 1985, pp. $1472 - 1476$.

[65] Singpurwalla, N.D.; Soyer, R., "Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications," IEEE Transactions on Software Engineering, vol. 11 no. 12, Dec. 1985, pp. $1456 - 1464$.

[66] Yamada, S.; Ohba, M.; Osaki, S., "S-shaped reliability growth modeling for software error detection," IEEE Transactions on Reliability, vol. 32 no. 5, Dec. 1983, pp. $475 - 478$.

[67] Yamada, S.; Osaki, S., "Software reliability growth modeling: models and assumptions," IEEE Transactions on Software Engineering, vol. 11 no. 12, Dec. 1985, pp. $1431 - 1437$.

About the Author

Louis Richard Camara was born in Dakar, Senegal, a french speaking country on the West coast of Africa, on February 23, 1967. He came to the United States of America in September 1996 to pursue Mathematics studies at the University of South Florida. Louis, after completion of an Accelerated BA/MA Program, received both his Bachelor's degree (Magna Cum laude) in Mathematics and his Master's degree in Mathematics from the University of South Florida in August 2001. He has been a Ph.D student in the Department of Mathematics at the University of South Florida since the Fall of 2001. Louis is a member of The Golden Key and Phi Kappa Phi honor societies.