CHAPTER – 1

ELEMENTS OF SOFTWARE RELIABILITY

INTRODUCTION
This chapter presents an introduction to software reliability growth models. Section 1.1 describes elements of software reliability models. Section 1.2 describes statistical models. These models are essential background for subsequent chapters. All planning and data analysis for software reliability growth models are based on such models. Section 1.3 describes Non-homogenous Poisson process. It includes different types of Non-homogenous Poisson process software reliability models. Section 1.4 describes release policies which includes different criteria. Section 1.5 describes parameter estimation. It includes maximum likelihood estimation methods, which are basic for analyzing data. This section pertains only to constant stress tests and data with a single failure mode. This section also includes least square estimation & Bayesian Inference. This chapter presents basic definition and concepts of software reliability modeling.

1.1 Elements of Software Reliability Models
Reliability is probably the most important of the characteristics inherent in the concept “Software quality”. It is intimately connected with defects, and as Jones (1986) point out, defects represents the largest cost element in programming. Software reliability concerns itself with how well the software functions to meet the requirements of the customer. “Failure” means the program in its functioning has not met user requirements in some way. Not functioning to meet user requirements is really a very broad definition. In addition to its prominent importance, software
reliability has proved to be the most readily quantifiable at the attributes of software quality. Reliability represents a user-oriented view of software quality. Initial (and many present) approaches to measuring software quality were based on attempting to count the faults or defects found in a program. This approach is developer oriented.

1.1.1 Software reliability and hardware reliability

The division between hardware reliability and software reliability is somewhat artificial. Therefore, we may combine hardware and software component reliabilities to get system reliability. Both depend on the environment. The source of failures in software is design faults while the principal source in hardware has generally been physical deterioration. However, the concepts and theories developed for the software reliability could really be applied to any design activity, including hardware design. Once a software (design) defect is properly fixed, it is in general fixed for all time. Failure usually occurs only when a program (design) is exposed to an environment that it was not developed or tested for. Although manufacturing can affect the quality of physical components, the replication process for software (design) is trivial and can be performed to very high standards of quality. Since introduction and removal of design faults occur during this period. The “design reliability” concept has not been applied to hardware to any extent. It was possible to keep hardware was generally less complex logically than software. Hardware design failures had to be kept low because retrofitting of manufactured items in the field was very expensive. The emphasis in hardware reliability may be starting to change now, however. Awareness of the work that is going on in software reliability, plus a growing realization of the importance of design faults, may be having an effect. This growing
awareness is strengthened by the parallels that peoples that are starting to draw between software engineering and chip design.

A final characteristic of software reliability is that it tends to change continually during test periods. This happens either as new problems are introduced when new code is written or when repair action removes problems that exist in the code. Hardware reliability may change during certain periods, such as initial burn-in or the end of useful life. However, it has much greater tendency than software toward a constant value. Despite the foregoing differences, we can develop software reliability theory in way that is compatible with hardware reliability theory.

1.1.2 Failures and Faults

A failure is a departure of external results of program operation from program requirements on a run software failure means departure of the external results of programs operation from requirements. So our “failure” is something dynamic. The program has to be executing for a failure to occur. The term failure relates to the behavior of the program. Note that a failure is not the something as a “bug” or more properly, “fault”. This very general definition of failure is deliberate. It includes such things a deficiency in performance attributes and excessive response time.

A fault is the defect in the program that, when executed under particular conditions, causes a failure. There can be different sets of conditions that cause failures, or the conditions can be repeated. Hence a fault can be source of more than one failure. A fault is a property of the program rather a property of its execution or behavior. It is what we are really referring to in general when we use the term “bug”. A fault is created when a programmer makes an error. It’s very important to make the failure-fault distinction.
1.1.3 Software Reliability

The definition that we presented here for software reliability is one that is widely accepted throughout the field. It is the probability of failure free operation of a computer program for a specified time in a specified environment. Its means that the probability that given software operates failure free for a specified time on the machine for which it was designed, given that it was within design limits and that the last failure occurred at a given time. This definition implies that, for accurate reliability measurement during test, select runs randomly with the same probabilities expected to occur in operation. The definition also implies that the input space must be “well covered” for accurate reliability measurement. The term mean time to failure (MTTF) is used in the hardware reliability field and to decreasing extent in software reliability. It is the average value of the next failure interval. The use of MTTF is attractive, in that “larger” indicates “better”. However, there are many cases in software reliability in which MTTF is undefined. Failure intensity is preferred because it always exits. Also, failure intensities are simple to work with because they combine additively. In an approximate non-vigorous sense, the two are the inverse of each other. The term mean time between failures (MTBF) is used in the hardware reliability field when repair or replacement is occurring. It is the sum of MTTF and mean time to repair (MTTR). The earliest software development process model is called waterfall model (Royce (1970)). It views the software process as successive phases as follow:

1. Requirement analysis: The system service constraints and goals are established.

2. Specification: The translations of requirement into precise description of the external of the software system are specified.
3. Design: The creation of software system that is consistent with the specification and representing the functions of each software system in a manner that may readily be transformed into one or more computer programs.

4. Implementation and Unit Testing: The creation of software system which implements the design. Unit testing involves that each units meets its specifications.

5. System Testing: The individual programs are integrated and tested to determine whether implementation satisfies the requirements.

6. Maintenance: Involves the correlation of the faults which were not discovered during the previous stages and enhances the performance of the software system.

*Life Cycle Models:*

Many different software life cycles have been proposed. These have different motivations, strengths, and weaknesses. The life cycle modals generally require the same types of tasks to be carried out; they differ in the ordering of the tasks in time.

Different software development process (or life-cycle) models:

- Waterfall model
- Rapid prototyping
- Evolutionary development
- Component reuse
- V model
- Formal transformation

A major drawback of the waterfalls models is that the specification should be freeze at an early stage in the development process Summerville (1989) and therefore the model ignores that the role of
iterations in software development process. To overcome these problems, may software development model with enhance feature have been proposed and put into implementation such as Rapid Prototyping model Seewg (1982), Garmand (1985), Operational model Zave (1982) and Knowledge based model Balzer (1983).

**Figure:** Waterfall Life Cycle model
1.1.4 Availability

Software reliability is usually defined as the expected fraction of time during which a software component or system is functioning acceptably. It is the ratio of up time to the sum of up time plus down time, as the time interval over which the measurement is made approaches infinity. The down time is the product of the failure intensity and the mean time to repair (MTTR). Usually the failure intensity applied have is a figure computed for serious failures and not those involve only minor degradation of the system. It is generally not practical to hold up operation of the system while performing fault determination and correction in the field. Therefore, we ordinarily determine MTTR as the average time required to restore the data base for a program, reload the program, and resume execution. A Markov model was developed by Shooman and Trivedi (1976) that depicts the concepts of software reliability. Okumoto and Goel (1978a) extended this model by assuming that a failure may be imperfectly repaired. A more recent paper by Laprie (1984) deals the evaluation at availability during the operational phase.

1.1.5 Modeling

To model software reliability we first consider the principal factors that affect it: fault introduction, fault removal, and the environment. Fault introduction depends primarily on the characteristics of the developed process code (Code created or modified for the application) and development process characteristics. The most significant code characteristic is size. The models are distinguished from each other in general terms by the probability distribution of failures times or number of failures experienced and by the nature of the variation of the random process with time. A software reliability model specifies the general from of the dependence of the failure process on the factors mentioned. We
have assumed that it is, by definition, time based (this is not to say non-
time based model may not provide useful insights). The possibilities for
different mathematical forms to describe the failures process is almost
limit less. We have restricted ourselves to considering well-developed
model that have been applied fairly broadly with real data and have given
reasonable results. The specification form can be determined from the
general form by establishing the values of the parameters of the model
through either:

1. **Estimation:** Statistical inference procedures are applied to failure
data taken for the program, or

2. **Prediction:** Determination from properties of the software
product and the development process (this can be done before any
execution of the program).

In general, software reliability models are based on (although this is often
not explicitly) a stable program executing in a constant environment. This
means not neither the code nor the operational profile are changing. Thus
the models focus mainly on fault removal. Most models can account for
the effect of slow fault introduction, however, in general terms, a good
model enhances communication on a project and provides a common
framework of understanding for the software development process. It also
enhances visibility to management and other interested parties. Developing a particularly useful software reliability model involves
substantial theoretical work tool building and the accumulation of a body
of love from practical experience. This effort generally requires several
person years. In contract, the application of a model that is well
established in requires a very a small fraction of projects resources.
1.1.6 Uses

Software Reliability measures to evaluate software engineering technology quantitatively. New techniques are continually being proposed for improving the process of developing software, but unfortunately they have been exposed too little quantitative evaluation. A software reliability measure offers us that the possibility of evaluating development status during the test phases of project methods such as institution of designers or test team, per cent of tests completed, and successful execution of critical functional tests have been used to evaluate testing progress. None of these have been really satisfactory and some have been quite unsatisfactory. Reliability generally increases with the amount of testing. We can use a software reliability measure to monitor the operational performance of software and to control new features added design changes made to the software. The reliability of software usually decreases as a result of such changes. A quantitative understanding of software quality and the various factors influencing it and affected by it enriches my insight into the software product and the software development process.

1.1.7 Program

A program will be defined as a set of complete machine instructions (operations with operands specified) that executes within a single computer and accomplishes a specific function. More than one program can execute “simultaneously” on a single computer if the machine is multi-programmed. A program is generally assumed to be stable (not changing in size or content with time) for the purpose of software reliability modeling. A program may be designed (especially if written in assembly language) for a particular computer. Hence, a version for another computer is really a separate program in so far as reliability
attributes are concerned. Conversely we can design a program to be machine independent or highly portable. Then the reliability attributes are independent of the computer used, except for adjustment for instruction execution rate. A program can consist of one or more logical instruction streams. A logical instruction stream is a set of instructions that must be executed in a prescribed sequence. When there is more than one logical instruction stream, the program is said to be multi-tested. The computer on which the program is running may be multiprocessor, in which case there are multiple physical instruction streams executing simultaneously. The computer can be a nectar or array processor, in which case there are multiple data streams. Each instruction accesses one or more data elements.

1.1.8 Historical Development of Models

This sub-section presents the historical development of software reliability models. Our purpose is to indicate which concepts have been tried, which have been more successful and which less and which have been modified and adapted. This historical information should help me at a new problem in the field evaluated proposed solutions for their degree of promise.

There are a number of major themes in the historical development:

1. The creation at various models relating to time, failures experienced, and other variable.

2. The discovery that measurement with respect to execution time simplifies model reliability.

3. The classification of the distinction between “fault” and “failure” and the development of a rich conceptual base that amplified our understanding of just what we mean by “software reliability”.

4. A concern with how to estimate model parameters.
5. An interest in comparison of models, which led to the development of comparison criteria.
6. The classification of models.
7. An increasing concern with collecting better data.
8. The development of a “lore” of techniques in using and adapting models for the particular circumstances of various applications, and
9. The transition from use of meantime to failure in characterizing status to failure intensity.

This subsection will concentrate mainly on the first theme but it will touch on the others. The first study of software reliability appears to have been conducted by Hudson (1967). The chronological historical development discussed under chapter 2 (reviews of previous work).

1.1.9 Calendar Time

A calendar time component was developed for the model that related execution time to calendar time, allowing execution time predictions to be converted to dates. The calendar time components are based on the fact that available resources limit the amount of execution time that is practical each calendar day.

1.1.10 Model Classification Scheme

This sub-section presents a classification scheme developed by Musa and Okumoto (1983) for software reliability models. The scheme permits relationship to be derived for groups of models. It highlights relationship among the model and suggests new models where gap occur in the classification scheme. It reduces the task of model comparison.

Models are classified in terms of five attributes:

i) Time domain: Calendar time or execution (CPU or processor) time.

ii) Category: The number of failures that can be experienced in infinite time is finite or infinite.
iii) Type: The distribution of the number of failures experienced by time

iv) Class (finite failures category only): Functional form of the failure intensity in terms of time, and

v) Family (finite failures category only): Functional form of the failure intensity in terms of the expected number of failures experienced.

The classification approach was chosen to be different for the two different categories because of greater analytical simplicity and physical meaning.

1.2 Statistical Models and Methods

A statistical model for software reliability growth model is given in this section. Some of which are used in subsequent chapters. We also present their reliability and hazard rate.

1.2.1 Exponential Distribution

The exponential distribution is most widely known and used in reliability because of its great simplicity and applicability. The probability density function is:

\[ f(t) = \lambda e^{-\lambda t}, \quad \lambda > 0 \]  

1.1

The cumulative distribution function is:

\[ F(t) = 1 - e^{-\lambda t} \]  

1.2

Where,

\[ \lambda = \text{constant failure rate} \]

Then the reliability function is obtained from the equation:

\[ R(t) = 1 - F(t) \]  

1.3

\[ R(t) = e^{-\lambda t} \]

and the hazard rate is from equation

\[ z(t) = \frac{f(t)}{R(t)} \]
as \( z(t) = \lambda \)

Thus, the exponential distribution represents the constant hazard (or failures) rate.

### 1.2.2 Weibull Distribution

The Weibull distribution plays an important role in reliability because of its great adaptability. Depending on the values of the parameters in its reliability function it can be shaped to represent much distribution as well as shaped to fit many sets of data.

The probability density function is:

\[
f(t) = \alpha \beta t^{\alpha-1} e^{-\beta t^\alpha} \quad \alpha > 0, \quad \beta > 0
\]

Where, \( \alpha \) and \( \beta \) are the shape and scale parameters respectively.

The cumulative distribution function is:

\[
F(t) = 1 - e^{-\beta t^\alpha}
\]

The reliability function is obtained from the equation:

\[
R(t) = \int_0^t f(x)dx
\]

\[
R(t) = e^{-\beta t^\alpha}
\]

The hazard rate is:

\[
z(t) = \frac{f(t)}{R(t)} = \alpha \beta t^{\alpha-1}
\]

### 1.2.3 Rayleigh Distribution

This distribution is special of Weibull distribution. The probability density function of random variable \( T \) is given by:

\[
f(t) = \beta t e^{-\beta t^2 / 2} \quad t, \beta > 0
\]

The cumulative distribution function (cdf) is:

\[
F(t) = 1 - e^{-\beta t^2 / 2}
\]
The reliability function is:
\[ R(t) = e^{-\frac{t}{\lambda}} \]  
1.11

The hazard rate is:
\[ z(t) = \beta t \quad t > 0 \]  
1.12

### 1.2.4 Pareto Distribution

The probability density function of Pareto distribution is:
\[ f(t) = \alpha \beta (1 + \beta t)^{-\alpha - 1} \quad \alpha, \beta, t > 0 \]  
1.13

The cumulative distribution function (cdf) is:
\[ F(t) = 1 - (1 + \beta t)^{-\alpha} \]  
1.14

The reliability and hazards function are:
\[ R(t) = (1 + \beta t)^{-\alpha} \]  
1.15

and
\[ z(t) = \frac{\alpha \beta}{1 + \beta t} \] respectively.  
1.16

### 1.2.5 Burr Type III

The probability density function of Burr type III distribution:
\[ f(t) = m \delta \beta (\beta t)^{-\delta - 1} \left(1 + (\beta t)^{-\delta}\right)^{-\delta - 1} \quad m, \delta, \beta > 0, t > 0 \]  
1.17

The cumulative distribution function is:
\[ F(t) = \left[1 + (\beta t)^{-\delta}\right]^{-m} \]  
1.18

The reliability and hazards function are:
\[ R(t) = 1 - \left[1 + (\beta t)^{-\delta}\right]^{-m} \]  
1.19

and
\[ z(t) = \frac{m \delta \beta (\beta t)^{-\delta - 1} \left[1 + (\beta t)^{-\delta}\right]^{-\delta - 1}}{1 - \left[1 + (\beta t)^{-\delta}\right]^{-m}} \]  
1.20

### 1.2.6 Burr Type X

The probability density function of Burr type X is:
The reliability and hazards function are:

\[ F(t) = (1 - e^{-\beta t})^k \]  \hspace{1cm} 1.21

\[ f(t) = 2K\beta e^{\beta t} (1 - e^{-\beta t})^{k-1} \]  \hspace{1cm} 1.22

The cumulative distribution function is:

\[ R(t) = 1 - (1 - e^{-\beta t})^k \]  \hspace{1cm} 1.23

and

\[ z(t) = \frac{2K\beta e^{-\beta t} (1 - e^{-\beta t})^{k-1}}{1 - (1 - e^{-\beta t})^k} \]  \hspace{1cm} 1.24

1.3 Non-Homogeneous Poisson Process

The concept of Markov process is useful in modeling random behaviour of software in time such as faults remaining at time and failure experienced by time t. A Markov process has the property that the future of the process depends only on the present state and is independent of its history. The assumption may by reasonable for a software failure process, which is mainly dependent on faults remaining and the operational profile (the latter is not dependent on the past). Markov processes are, in general, characterized by the amount of time spent in a state and the transitions between stakes.

Poisson process provide a good approximation to the occurrence at many real world events such as telephone calls, order to a factory, breakdown of machinery, arrivals on a queue and insurance claims. In this section, we will study the software failure process using a non-homogeneous Poisson process (NHPP) with failure intensity \( \lambda(t) \). We will describe existing Poisson type models (of both finite and infinite failure category) as special cases of the underlying general NHPP.
1.3.1 Assumption

Let $M(t)$ denote failures experienced by time $t$. The $M(t)$ process for Poisson-type models based on the following assumptions.

i) There are no failure experienced at time $t=0$, that is $M(t)=0$ with probability 1.

ii) The process has independent increments. In other words, the number of failures experienced during $(t, t+\Delta t)$, that is, $M(t+\Delta t)$, is independent of the history. Note that this assumption implies the Markov property that the future $M(t + \Delta t)$ of the process depends only on the present state $M(t)$ and is independent of its past $M(x)$ for $x<t$.

iii) The probability that a failure intensity of the process during $(t, t+\Delta t)$ is $\lambda(t)\Delta t + O(\Delta t)$, where $\lambda(t)$ is the failure intensity of the process. Note that the function $O(\Delta t)$ is defined as:

$$\lim_{\Delta t \to 0} \frac{O(\Delta t)}{\Delta t R(t)} = 0$$  \hspace{1cm} 1.25

In practice, it implies that the second or the higher order effects of $\Delta t$ are negligible.

iv) The probability that more than one failure will occur during $(t, t+\Delta t)$ is $O(\Delta t)$.

1.3.2 Fault Reduction Factor for Poisson Type Models

Poisson type model also permits consideration of faults that can not be located, extra fault found through code inspection, and spawned faults. The fault removal process is characterized on an average basis by assuming that the fault correction rate proportional to the hazard rate. Musa (1975) called this proportionally constant a fault reduction factor, denoted by $B$. The “average basis” is reasonable because the useful application of the software theory is primarily macroscopic in nature.
For a given number of faults $W$, the total expected number of failures will be $v_0 = w_{o/1}$. Denote the cumulative distribution function of time to remove a fault by $G_a(t)$. Since the fault correction rate is the product of the fault reduction factor $B$ and the hazard rate $Z_a(t)$, we have:

$$G_a(t) = 1 - \exp\left[-B \int_0^t Z_a(x)dx\right]$$

1.26

The exact expression for the distribution of $M(t)$ is difficult to obtain, but it may approximated by the Poisson distribution:

$$P[M(t) = m] = \frac{[v_0 G_a(t)]^m \exp[-v_0 G_a(t)]}{m!}, \quad m=0,1,\ldots$$

1.27

The mean value intensity function is given by:

$$\mu(t) = v_0 G_a(t)$$

1.28

And the failure function is:

$$\lambda(t) = v_0 g_a(t)$$

1.29

Where $g_a(t)$ is the probability density function associated with $G_a(t)$. The program hazard rate is given by:

$$Z(t, t_{i-1}) = v_0 g_a(t_{i-1} + t')$$

1.30

Since the actual repair process occurs at times of failure, the Poisson type model would not tend to fit as well as the binomial-type. However, the effectiveness of the repair action in after imperfect and the degree of imperfection varies randomly from repair to repair. The Poisson-type model after the promise of approximating reality more closely because of its capability for handling imperfect repair.

### 1.3.3 Non-Homogeneous Poisson process Software Reliability Models

The software reliability assessment is important to evaluate and predict the reliability and performance of a software system. The models applicable to the assessment of software reliability are called software
reliability growth models. Software reliability growth models (SRGMs) also provide a plausible description of software occurrence phenomenon.

In this section we discuss some SRGM’s described by Non-Homogeneous Poisson Process (NHPP). SRGMs which use the testing the time as the unit of error detection period and SRGMs where the number of test runs or test cases instead on testing is the unit of error detection period are discussed.

SRGMs with enough flexibility called the generalized Erlang model with 3 stages and the extended with several stages are discussed and analyzed which can account for variability in the growth curve. Models have been validated on several software datasets obtained from different sources. Predictive validity of the model is also discussed. A SRGMs based on the software development project is also presented. Models have been validated using simulated data.

1.3.3.1 Jelinski-Moranda de-eutrophication model

**Model form:** From the overview of the model and the assumption, we can determine that if the time between-failure occurrences are \( X_i = T_i - T_{i-1}, \ i=1,\ldots,n, \) then the \( X_i \)'s are independent exponentially distributed random variables with mean \( = 1/\lambda(N-(i-1)) = 1/\lambda(X_i/T_{i-1}) \)

That is

\[
f(X_i/X_{i-1}) = z(X_i/X_{i-1}) \exp(-z(X_i/X_{i-1})X_i)
\]

\[
= \phi[N-(i-1)] \exp(-\phi[N-(i-1)]X_i) \quad 1.31
\]

Since this exponential model belongs to the binomial type, we have specifically:

\[
\mu(t) = N(1-\exp(-\phi t)) \quad 1.32
\]

and

\[
\lambda(t) = N\phi \exp(-\phi t) \quad 1.33
\]
for the mean value function and the failure intensity function. It is clearly a finite failures type model as
\[
\lim_{t \to \infty} \mu(t) = \lim_{t \to \infty} (N(1 - \exp(\theta))) = N
\]

### 1.3.3.2 Non-Homogeneous Poisson Process Model

**Model form:** From the assumptions it can be shown Goel and Okumoto (1979) that the mean value function must be of the form
\[
\mu(t) = N(1 - e^{-bt})
\]
for some constants \( b > 0 \) and \( N > 0 \). \( N \) is the expected total number of faults to be eventually detected. (Note: \( N \) is not required to be an integer since it is the expected number of faults that will eventually be detected). Since the failure intensity function is the derivative of \( \mu(t) \). We have, therefore
\[
\lambda(t) = Nbe^{-bt}
\]
Native that the failure intensity function is strictly decreasing for \( t > 0 \). Because it belongs to the exponential class, we have the distribution of a single individual fault, \( X \):
\[
f_X(x) = be^{-bx}
\]
We have for the failure intensity function
\[
\lambda(t) = Nbe^{-bt} = Nf_X(t)
\]
which shows the relationship between the failure intensity function and probability density function for a single fault.

From the assumptions, we also have that each \( f_i \), the fault count in the \( i^{th} \) interval, is an independent Poisson random variable with mean= \( \mu(t_i) - \mu(t_{i-1}) \). Therefore the joint density of the \( f_i, i = 1, ..., n \) is
\[
\prod_{i=1}^{n} \frac{[\mu(t_i) - \mu(t_{i-1})]^f_i \exp[\mu(t_i) - \mu(t_{i-1})]}{f_i!}
\]
for the mean value function and the failure intensity function. It is clearly a finite failures type model as

\[ \lim_{t \to \infty} \mu(t) = \lim_{t \to \infty} (N(1 - \exp(\theta))) = N \]

1.3.3.2 Non-Homogeneous Poisson Process Model

**Model form:** From the assumptions it can be shown Goel and Okumoto (1979) that the mean value function must be of the form

\[ \mu(t) = N(1 - e^{-\theta t}) \]

for some constants \( b > 0 \) and \( N > 0 \). \( N \) is the expected total number of faults to be eventually detected. (Note: \( N \) is not required to be an integer since it is the expected number of faults that will eventually be detected). Since the failure intensity function is the derivative of \( \mu(t) \). We have, therefore

\[ \lambda(t) = Nb e^{-\theta t} \]

Native that the failure intensity function is strictly decreasing for \( t > 0 \). Because it belongs to the exponential class, we have the distribution of a single individual fault, \( X \):

\[ f_X(x) = be^{-\theta x} \]

We have for the failure intensity function

\[ \lambda(t) = Nb e^{-\theta t} = N f_X(t) \]

which shows the relationship between the failure intensity function and probability density function for a single fault.

From the assumptions, we also have that each \( f_i \), the fault count in the \( i^{th} \) interval, is an independent Poisson random variable with mean= \( \mu(t_i) - \mu(t_{i-1}) \). Therefore the joint density of the \( f_i, s, i = 1, ..., n \) is

\[ \prod_{i=1}^{n} \left[ \mu(t_i) - \mu(t_{i-1}) \right]^k \exp\left[ \mu(t_i) - \mu(t_{i-1}) \right] / f_i! \]
1.3.3.3 Schneidewind’s Model

**Model form:** From the assumptions, the cumulative mean number of faults by the $i^{th}$ time period is

$$ D_i = \mu(t_i) = \frac{\alpha}{\beta} \left[ 1 - \exp(-\beta t_i) \right] $$

1.40

Thus the expected number of faults in the $i^{th}$ period is $f_i$,

$$ m_i = D_i - D_{i-1} = \mu(t_i) - \mu(t_{i-1}) = \frac{\alpha}{\beta} \left[ \exp(-\beta (i-1)) - \exp(-\beta i) \right] $$

1.41

Using the assumptions again pertaining to the $f_i$'s being independent non-homogeneous Poisson random variables and incorporating the concept of the different model types, we have the joint density

$$ \frac{M_{s-1} \exp(-M_{s-1})}{F_{s-1}} \prod_{j=1}^{s-1} m_j \exp m_j $$

1.42

where $s$ is some integer value chosen in the range 1 to $n$, $M_{s-1}$ is the cumulative mean number of faults in the intervals up to $s-1$, and $F_{s-1}$ is the cumulative number of faults detected up through interval $s-1$.

1.3.3.4 Musa’s basic execution time model

**Model form:** Since $\mu(t) = \beta(t)(1 - \exp(-\beta t))$, the failure intensity function for this model is

$$ \lambda(t) = \mu'(t) = \beta_0 \beta_t \exp(-\beta_0 t) $$

1.43

We notice that for large $\beta_0$ the failure intensity function will decrease rapidly, while for a small one it will decrease slowly. In either case, the function decreases exponentially to 0.

By making the correspondence that $\beta_i = B\phi$ and $\beta_0 = \nu_0$, where $B$ is defined as the fault reduction factor (the proportionality constant relating the fault correction rate to the hazard rate) and $\phi$ is the constant hazard rate per individual fault, the preceding formulation can be put into the
framework in which Musa originally introduced this model Musa et. al (1987).

The above expressions for the mean value and failure intensity function, we can show that the reliability function after \( (i-1) \) failures have occurred is

\[
R(\Delta t/t_{i-1}) = \exp(-[\beta_0 \exp(-\beta_{i-1}t)]\exp(-\beta_i \Delta t))
\]

for \( 0 \leq \Delta t \), and the conditional hazard rate is

\[
z(\Delta t/t_{i-1}) = \beta_0 \beta_i \exp(-\beta_{i-1}t)\exp(-\beta_i \Delta t)
\]

for \( 0 \leq \Delta t \)

For the development of the calendar time component of this model are referred to Musa et al. (1987).

1.3.3.4 Hyperexponential model

Model form: Notice that if \( K=1 \) we have the NHPP model of See 1.3.3.2. Also, \( \lim_{t \to \infty} \mu(t) = N \); so, as before, \( N \) represents the expected total number of faults to be eventually detected. (Note: \( N \) is not required to be an integer since it is the expected number of faults that will eventually be detected). For the \( i^{th} \) class, we also note that \( N_{P_i} \) is the expected number of faults within that class. Since the failure intensity function is the derivative of \( \mu(t) \), we therefore have

\[
\lambda(t) = N \sum_{i=1}^{K} P_i \beta_i \exp(-\beta_i t)
\]

1.46

Notice that the failure intensity function is strictly decreasing for \( t > 0 \).

1.3.3.6 Weibull model

Model form: Since this model belongs to the binomial type, and the cumulative distribution function for a Weibull, we have for the failure intensity function and the mean value function:

\[
\lambda(t) = N \alpha \beta \gamma(t) = N\alpha \beta t^{\alpha-1} \exp(-\beta t^\alpha)
\]

1.47
and
\[ \mu(t) = N F_a(t) = N \left( \exp \left( - \beta t^\alpha \right) \right) \]

Notice that \( \lim_{t \to \infty} \mu(t) = N \), the total number of faults in the system at the start. Also, from the assumptions we have that if \( \alpha = 1 \), the distribution \( f_a \) becomes the exponential, and if it equals 2 we have the Rayleigh distribution, another important failure model in hardware reliability theory. We also note for the case \( \alpha = 2 \) that this becomes the early model considered by Schick-Wolverton (1973). We can also see that if \( 0 < \alpha < 1 \), the per-fault hazard rate is decreasing with respect to time; if \( \alpha \) equals 1 (exponential) it is constant; and if \( \alpha > 1 \), it increases.

The form of the conditional hazard rate is shown to be:
\[ z(t/t_{i-1}) = (N - i + 1) \alpha \beta (t + t_{i-1})^{\alpha - 1} \text{ for } t_{i-1} \leq t + t_{i-1} < t_i \]

This function is plotted for \( 0 < \alpha < 1 \) to contrast its behavior with the exponential class. For the Weibull distribution, the change occurs at fault detection, but the change is not constant. The effect on the hazard rate decreases with time because of the power function component.

The reliability function is obtained from the cumulative distribution function as \( R(t) = 1 - F(t) = \exp \left( - \beta t^\alpha \right) \) and, the MTTF is
\[ \text{MTTF} = \frac{1}{\beta} \Gamma \left( \frac{1}{\alpha} + 1 \right) \]

Where \( \Gamma(\bullet) \) is the gamma function.

1.3.3.7 S-shaped reliability growth model

**Model form:** Suppose we have a partition of the time interval over which the software is observed. This partition could represent the testing intervals of the software. Let \( T^* \) denote this partition, that is, \( t_0^* = 0 < t_1^* < ... < t_n^* \). Suppose \( f_1, f_2, ..., f_n \) are the number of software faults detected in each interval of the partition, that is, \( f_i \) is the number of faults
occurring in the interval of length \( t_i = t_i^* - t_{i-1}^* \). From the assumptions we have, each \( f_i \) is an independent Poisson random variable with mean

\[
\mu(t, j) - \mu(t, j - 1) = \alpha \left[ \left( 1 + \beta t_i \right) e^{-\beta t_i} \right] - \alpha \left[ \left( 1 + \beta t_{i-1} \right) e^{-\beta t_{i-1}} \right]
\]

\[
= \alpha \left[ \left( 1 + \beta t_i \right) e^{-\beta t_i} - \left( 1 + \beta t_{i-1} \right) e^{-\beta t_{i-1}} \right]
\]

Also, from the mean value function \( \mu(t) = \alpha \left[ \left( 1 + \beta t_i \right) e^{-\beta t_i} \right] \), we have the failure intensity function \( \lambda(t) = \mu(t) = \alpha \beta^{-\beta} e^{-\beta t} \). The model gets its S-shaped form because of the mean value function. Moreover, we can see that \( \lim_{t \to \infty} \mu(t) = \alpha < \infty \), so we indeed have a finite failures model with \( \alpha \) being the total number of faults in the system. If we plot the failure intensity function, we would see that it increases up to time \( t = 1/\beta \) and then begins to decrease asymptotically approaching the time axis. Since we have a Poisson type as well as a finite failures model, the per-fault time distribution between failures is \( f_\alpha(t) = \beta^2 e^{-\beta t} \) as \( \lambda(t) = \alpha f_\alpha(t) \). This is the gamma distribution.

Using the above relationships one can also establish the following reliability measures for this model:

The reliability of the function at time \( t_i + \Delta t \) given a failure at time \( t_i \):

\[
R(t_i + \Delta t | t_i) = \exp \left( -\alpha \left[ \left( 1 + \beta t_i \right) e^{-\beta t_i} - \left( 1 + \beta (t_i + \Delta t) \right) e^{-\beta (t_i + \Delta t)} \right] \right)
\]

The hazard rate function at time \( t_i + \Delta t \) given a failure at time \( t_i \):

\[
h(t_i + \Delta t | t_i) = \alpha \beta^2 (t_i + \Delta t) e^{-\beta (t_i + \Delta t)}
\]

The expected number of faults in the \( i \)th period of length

\[
l = \alpha \left[ \left( 1 + \beta \sum_{j=1}^{i-1} l_j \right) e^{-\beta \sum_{j=1}^{i-1} l_j} - \left( 1 + \beta \left( l + \sum_{j=1}^{i-1} l_j \right) \right) e^{-\beta \left( l + \sum_{j=1}^{i-1} l_j \right)} \right]
\]

1.3.3.8 Duane's model

**Model form:** From assumption 1 we have a Poisson process with a mean value function of \( \mu(t) = \alpha \beta^2 \). If \( T \) is the total time the software is observed, then we have
\[
\frac{\mu(t)}{T} = \alpha T^\beta = \text{expected number of failures by time } T \text{ over total testing time} \tag{1.56}
\]

So that if we take the natural log of both sides of the equations we have

\[
Y = \ln \left( \frac{\mu(t)}{T} \right) = \ln \left( \frac{\alpha T^\beta}{T} \right) = \ln(\alpha) + (\beta - 1)\ln(T) \tag{1.57}
\]

We can thus see if the first equation is plotted on ln-ln paper versus observed time \(T\), or the second equation is plotted on regular paper versus \(\ln(T)\) we will obtain a straight line. It is this form that is fitted to a given data set.

The failure intensity function is obtained by taking the derivative of the mean value function, that is, \(\lambda(t) = d\mu(t)/dt = \alpha \phi t^{\beta-1}\). From this function we see that the failure intensity function is strictly increasing for \(\beta > 1\), a constant for the case of a homogeneous Poisson process \((\beta = 1)\), and strictly decreasing for \(1 > \beta > 0\) only. For \(\beta > 1\), there can be no reliability growth!

### 1.3.3.9 Geometric model

**Model form:** We have the density for the time between failures of the \(i\)th and \((i-1)\)th is exponential of the form:

\[
f(X_i) = D \phi^{-i} \exp(-D \phi^{-i} X_i) = z(t_{i-1}) \exp(-z(t_{i-1})X_i) \tag{1.58}
\]

Thus the expected time between failures is

\[
E(X_i) = \frac{1}{z(t_{i-1})} = \frac{1}{D \phi^{-i}} \quad \text{for } i = 1, \ldots, n \tag{1.59}
\]

Using the fact that \(E(X_i) \approx \mu(t)\) and \(i = \mu(t)\) Musa et al. (1987), it follows that

\[
\mu(t) = \frac{1}{\beta} \ln \left( [D \beta \exp(\beta)]^t + 1 \right) \tag{1.60}
\]

and

\[
\lambda(t) = \frac{D \exp(\beta)}{[D \beta \exp(\beta)]^t + 1} \quad \text{where } \beta = -\ln(\phi) \quad \text{for } 0 < \phi < 1 \tag{1.61}
\]
Clearly, \( \lim_{t \to \infty} \mu(t) = \infty \), so we indeed have an infinite failures category model.

### 1.3.3.10 Musa-Okumoto logarithmic Poisson Model

**Model form:** The exponential type decay and the fact that the earlier encountered failures have a more dramatic impact than the later ones. The parameter \( \theta \) controls the shape of the curve.

A second expression of the logarithmic Poisson model to aid in obtaining the maximum likelihood estimates is through a re-parameterization of the model. We let \( \beta_0 = \theta^{-1} \) and \( \beta_i = \lambda_i \theta \). The intensity and mean value functions become in this case:

\[
\lambda(t) = \beta_0 \beta_i / (\beta_i t + 1)
\]

and

\[
\mu(t) = \beta_0 \ln(\beta_i t + 1)
\]

Musa (1987) derives the program reliability and the hazard rate functions after the \( (i-1) \)th failure, respectively, as

\[
R(\Delta t / t_{i-1}) = \left[ \frac{\beta_i t_{i-1} + 1}{\beta_i (t_{i-1} + \Delta t) + 1} \right]^{\beta_0} \text{ for } \Delta t \geq 0
\]

and

\[
z(\Delta t / t_{i-1}) = \beta_0 \beta_i / (\beta_i (t_{i-1} + \Delta t) + 1) \text{ for } \Delta t \geq 0
\]

### 1.3.3.11 Littlewood-Verral reliability growth model

**Model form:** To calculate the posterior distribution we first need the marginal distribution of the \( x_i \)'s. The prior distribution is of the form:

\[
g(x_i; \psi(i), \alpha) = \frac{\psi(i)^{x_i} \exp(-\psi(i))}{\Gamma(\alpha)} \quad \xi_i > 0
\]
Using this prior and the following conditional exponential distribution for the \( x_i \)'s: \( f(x_i \mid x_j, \xi_i) = \xi_i \exp(-\xi_i x_i) \) for \( x_i > 0 \), the marginal distribution of the \( x_i \), can be shown to be:

\[
f(x_i \mid \alpha, \psi(i)) = \frac{\alpha \psi(i)^{\alpha}}{[x_i + \psi(i)]^{a+1}} \quad \text{for} \quad x_i > 0
\]

that is, a Pareto distribution, so that the joint density is

\[
f(x_1, x_2, \ldots, x_n) = \frac{\alpha^n \prod_{i=1}^n \psi(i)^\alpha}{\prod_{i=1}^n [x_i + \psi(i)]^{a+1}} \quad \text{for} \quad x_i > 0, \ i = 1, \ldots, n
\]

Each \( \xi_i \) is an independent gamma distribution with parameters \( \alpha + 1 \) and \( 1/(x_i + \psi(i)) \). Therefore, if we use a quadratic loss function, the Bayesian estimate of \( \xi_i \) is the mean; namely, \( (\alpha + 1)/(x_i + \psi(i)) \). Therefore, if we use a quadratic loss function, the Bayesian estimate of \( \xi_i \) is the mean; namely, \( (\alpha + 1)/(x_i + \psi(i)) \).

Littlewood and Verrall suggest a linear and quadratic form for the \( \psi(i) \) function, that is, \( \psi(i) = \beta_0 + \beta_i i \) (the linear form) and \( \psi(i) = \beta_0 + \beta_i i^2 \) (the quadratic form).

The failure intensity functions for the linear and quadratic forms can be shown Musa (1987) to be

\[
\lambda_{\text{linear}}(t) = \frac{\alpha - 1}{\sqrt{\beta_0^2 + 2\beta_i(t-1)}}
\]

and

\[
\lambda_{\text{quadratic}}(t) = \frac{v_1}{\sqrt{v_1^2 + v_2}} \left[ \left( t + \left( v_2^2 + v_2 \right)^{\frac{1}{2}} \right)^{\frac{1}{2}} - \left( t + v_2 \right)^{\frac{1}{2}} \right]^{\frac{1}{2}}
\]

where
\[ v_1 = \frac{(a-1)^{1/3}}{(18\beta_1)^{1/3}} \text{ and } v_2 = 4\beta_2 \frac{(a-1)^{2/3}}{9(\beta_1)} \]

1.3.3.12 Exponential SRGM

Let the rate of the number of failure or faults removed, \( m'(t) \), be given

\[ m'(t) = b[a - m(t)] \]

Under the initial condition \( m(t) = 0 \), we have

\[ m(t) = a(1 - \exp^{-bt}) \]

Assuming that the fault causing a failure is immediately removed, \( m(t) \) gives the number of failures or faults removed by time \( t \). This model is known as exponential SRGM. This model was developed by Goel and Okumoto (1979) and has been widely used.

1.3.3.13 Modified-Exponential SRGM

In this model, it is assumed that the software system contains two types of errors namely type I (which are easy to detect) and type II (which are difficult to detect). It is assumed that the errors detected early in testing are different from those detected later on.

Let \( p_i \), for \( i = 1,2 \), denote the proportion of type \( i \) errors in the software system \( (p_1 + p_2 = 1) \) and \( h_i \) be the error detection rate of type \( i \) errors per error per unit time, \( m_i(t) \) can be written as follows:

\[ m_i(t) = p_i a(1 - e^{-h_i t}) \]

and

\[ m(t) = \sum_{i=1}^{2} m_i(t) = \sum_{i=1}^{2} i = ap_1 (1 - e^{-h_1 t}) \]

with \( 0 < h_2 < h_1 < 1 \). This model was discussed by Yamada and Osaki (1985).
1.3.3.14 SRGM with Testing Effort

It has been assumed in all the SRGMs discussed so far that as the testing time increases, testing effort also increases. If testing time becomes quite large, testing effort also becomes quite large. In reality, no software developer may spend infinite resources on testing effort expenditure. Let \( w(t) \) be the testing effort expenditure at time \( t \). Then the total testing effort expenditure in \((0, t)\) is expressed as:

\[
W(t) = \int_0^t w(x)dx
\]

Under the assumption that the number of errors detected in \((t, t + \Delta t)\) per unit testing effort expenditure is proportional to the remaining errors, we have:

\[
\frac{m'(t)}{w(t)} = b(a - m(t))
\]

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

\(w(t)\) has been defined in the literature by an exponential or Rayleigh curve, i.e.

\[
W_e(t) = \alpha\left(1 - e^{-\beta t}\right) \quad \text{or} \quad W_r(t) = \alpha\left(1 - e^{-\beta e^t/\alpha^2}\right)
\]

This model is due to Yamada et al. (1986).

1.3.3.15 Discrete Exponential SRGM

Assuming that the errors detected between the \(n^{th}\) and \((n + 1)^{th}\) test run are proportional to the number of errors remaining after execution of the \(n^{th}\) test run, we may write the following difference equation

\[
m(n+1) - m(n) = b(a - m(n))
\]

Solving we get

\[
m(n) = a\left(1 - (1 - b)^r\right)
\]
1.3.3.16 Discrete SRGM with Testing Effort

Let \( w(n) \) be the testing resources spent on the \( n^{th} \) test run for the software system. Assuming \( w(n) \) is described by a discrete Rayleigh curve, we may write

\[
\begin{align*}
  w(n+1) &= W(n+1) - W(n) = \beta(n+1)[\alpha - W(n)] \\
  W(n) &= a\left(1 - \prod_{i=0}^{n-1}(1 - i\beta)\right)
\end{align*}
\]

where \( \alpha \) and \( \beta \) are the parameters of the testing effort function and \( W(n) \) is the total testing resources spent on a test runs. We have

\[
W(n) = a\left(1 - \prod_{i=0}^{n-1}(1 - i\beta)\right)
\]

and hence

\[
w(n) = a\beta n\prod_{i=0}^{n-1}(1 - i\beta)
\]

Assuming that the number of errors removed between \( n^{th} \) and \( (n + 1)^{th} \) test run per unit testing effort expenditure is proportional to the remaining errors, we may write:

\[
\frac{m(n+1) - m(n)}{w(n)} = b(a - m(n))
\]

Solving, we get

\[
m(n) = a\left[1 - \prod_{i=0}^{n}(1 - bw(i))\right]
\]

This model is due to Kapur et al. (1994).

1.3.3.17 Discrete SRGM with Leading and Dependent Errors

Assuming that the number of leading errors removed between \( (n + 1)^{th} \) and \( n^{th} \) test run are proportional to the remaining leading errors and the number of dependent errors removed between \( n^{th} \) and \( (n + 1)^{th} \) test run proportional to the remaining dependent errors and the ratio of leading errors removed to the total error content with a lag, we have:

\[
m_i(n+1) - m_i(n) = b(a_i - m_i(n))
\]

and
\[ m_2(n+1) - m_2(n) = c(a_2 - m_2(n)) \frac{m_1(n+1-N)}{a} \]  

where \[ a = a_1 + a_2 \]

Solving we get

\[ m_1(n) = a_1 \left(1 - (1-b)^n\right) \]  

and

\[ m_2(n) = a_2 \left[1 - \prod_{i=0}^{n-1} \left(1 - \frac{ca_1}{a} \left(1 - (1-b)^i\right)\right)\right] \]

The total number of error removed by \( n^{th} \) test run is given by

\[ m(n) = m_1(n) + m_2(n) = a - a_1(1-b)^n - a_2 \prod_{i=0}^{n-1} \left(1 - \frac{ca_1}{a} \left(1 - (1-b)^i\right)\right) \]

This model is due to Kapur et al. (1995).

### 1.4 Release Policies

Besides, developing software reliability growth models, it is also of interest to know when to stop testing and the software for use. If the release of the software is unduly delayed, the manufacturer (Software developer) may suffer in terms of and revenue loss, while a premature release may cost heavily in terms of fixes (removals) to be done after release and may even harm manufacturer's reputation. Software release time problems have been classified in different ways. One is, when to release a software so that the cost incurred during the life cycle (consisting of the development and operational phases) of the software is minimized or the reliability is maximized Okumoto and Goel (1980). This problem can also be defined alternatively in terms of maximizing gain Bai and Yun (1988). Gain is defined as the difference in cost incurred when all the faults are removed during the operational phase as against the cost when some faults are removed during the testing phase and others are removed during the operational phase. It can be proved that maximizing gain is same as minimizing cost other approach that is also used for a release time problem is, when to stop testing so that the
failure intensity/reliability reaches a desired level irrespective of the cost
curred. Stopping rules in this regard have been studied by Forman and
Singpurwala (1977), Ross (1985), and Xie (1991). There is yet another
criterion which is to minimize cost during software life cycle subject to
achieving a desired level of reliability, Yamada and Osaki (1987).
Alternatively, the problem can also be proposed as, maximize reliability
subject to the software cost meeting the budgetary constraint. Thus, based
on the type of the project, one of the above two policies may be used.

In this section, release policies for several software reliability growth
models (SRGMs) based on Non-Homogenous Poisson Process (NHPP)
were discussed. Release policies have been discussed for SRGMs both in
continuous and discrete time. Release policies are based on cost and
reliability criteria and their variants. We also discuss a new release policy
called the bicriterian release policy, optimizing two conflicting objectives
namely software cost subject to budget and reliability constraints.

1.4.1 Release Policies based on different Criteria

For an exponential SRGM in continuous time, the mean value function
m(t) (number of failures/faults removed) is defined as:

\[ m(t) = \alpha (1 - e^{-b t}) \]  \hspace{1cm} 1.93

where \( \alpha \) is the total expected error content in the software and \( b \) is the
error detection rate per remaining error. The failure intensity is:

\[ \lambda(t) = m'(t) = ab e^{-bt} \]  \hspace{1cm} 1.94

It may be observed that \( \lambda(t) \) is a decreasing function in \( t \) with \( \lambda(0) = ab \)
and \( \lambda(\infty) = 0 \). The software cost incurred by the manufacturer during
software life cycle (time till the software is supported by the
manufacturer) is given as (assuming \( T \) to be the release time of the
software)

\[ C(T) = C_1 m(T) + C_2 (m(T) - m(T)) + C_3 T \]  \hspace{1cm} 1.95
where $C_1$ and $C_2$ are the cost of fixing a fault before (after) releasing the software, $C_3$ is the testing cost per unit time and $T_i$ is the software life cycle length ($> T$). $C_2$ is assumed to be greater than $C_1$.

The expected software reliability $R(x/T)$ the probability that a software failure does not occur in $(T, T + x]$, given that the last failure occurred in $T \geq 0$ ($x \geq 0$) is defined as:

$$R(x/T) = e^{-m[T+x]-m[T]}$$

### 1.4.1.1 Cost Criterion

Here the objective is to find a release time $T$, such that the total expected software cost during software life cycle is minimized.

Differentiating the cost function $C(T)$ in with respect to $T$, we get:

$$C'(T) = -(C_2 - C_1)m'(T) + C_3$$

$$C'(T) = 0 \text{ if } m'(T) = C_3/(C_2 - C_1)$$

If $ab \leq C_3/(C_2 - C_1)$, it is clear that $C'(T) > 0$ for $T > 0$ and hence $C(T)$ is minimum for $T=0$.

If $ab > C_3/(C_2 - C_1)$, there exists a finite $T = T_0 (> 0)$, satisfy $C'(T) = 0$. Thus $C(T)$ is decreasing for $T < T_0$ and increasing for $T > T_0$. $C(T)$ is minimum for $T = T_0$. We, then have following theorem.

**Theorem 1.** Assume $T_i > T_0$

If $ab \leq C_3/(C_2 - C_1)$, optimum release time $T^*$ is $0$. If $ab > C_3/(C_2 - C_1)$, optimum release time is $T_0$.

### 1.4.1.2 Reliability Criterion

The objective is to find a release time $T$, satisfying $R(x/T) \geq R$, where $(0 < R < 1)$ is the required level of reliability. From $R(x/0) = e^{-m(x)}$ and $R(x/\infty) = 1$. Differentiating $R(x/T)$ with respect to $T$, we have:

$$R'(x/T) = e^{-m[T+x]-m[T]} \left( ab e^{-m(T)} \right)$$

$$= e^{-m[T+x]-m[T]} \left( ab e^{-m(T)} (1 - e^{-3h}) \right)$$

32
Since \( R'(x/T) > 0 \) for all \( T \geq 0 \), \( R(x/T) \) is increasing for all \( T > 0 \). Thus, if \( R(x/T) < R_0 \), there exists \( T = T_i(>0) \) such that \( R(x/T_i) = R_0 \). We have the following theorem.

**Theorem 2.** Assume \( T_i > T \), if \( R(x/0) \geq R_0 \) then \( T' \geq 0 \), but \( < T_i \) if \( R(x/0) < R_0 \) then \( T' \geq T_i \), but \( < T_i \).

### 1.4.1.3 Cost and Reliability Criteria

The objective can be either to minimize cost subject to reliability not less than a predefined reliability level or to reliability subject to cost not exceeding a predefined finite budget. The objective is, therefore, either minimize \( C(T) \) subject to \( R(x/T) \geq R_0 \) or maximize \( R(x/T) \) subject to \( C(T) \geq C_B \), where \( C_B \) is the predefined budget level.

From the above, we know that if \( ab < C_j/(C_2 - C_1) \), \( C(T) \) is minimum at \( T = 0 \), and if \( ab > C_j/(C_2 - C_1) \), \( C(T) \) is minimum at \( T = T_0 \). Such that

\[
C'(T)/T = T_0 = 0,
\]

more over,

1. if \( ab \leq C_j/(C_2 - C_1) \) and
   (a) if \( C(0) > C_B \), budget constraint is not met any \( T \geq 0 \).
   (b) if \( C(0) = C_B \), budget constraint is met any \( T = 0 \)
   (c) if \( C(0) = C_B \), budget constraint is met for all \( T(0 \leq T \leq T_0) \), where

\[
C(T)_{T_0} = C_B
\]

2. If \( ab > C_j/(C_2 - C_1) \) and \( C(T_0) > C_B \), budget constraint is not met for any \( T \geq 0 \).
3. If \( C(T_0) = C_B \), budget constraint is met for any \( T = T_0 \).
4. If \( C(0) < C_B \) and \( C(T_0) < C_B \), budget constraint is met for all \( T(0 \leq T \leq T_0) \) where \( C(T)/T = T_{T_0} \).
5. If \( C(0) > C_B \) and \( C(T_0) < C_B \), budget constraint is met for all \( T(0 \leq T \leq T_0) \) where \( C(T)/T = T_{T_0} \).

\[
C_B = C_B,
\]

and \( C(T)/T = T_{T_0} = C_B \) and \( C(T)/T = T_{T_0} = C_B \).
We know that if $R(x/0) < R_o$, there exists $T = T_i$ such that $R(x/T)/_{T-T_i} = R_o$.

Combining the cost and reliability requirement, we have the following two theorems:

Case (i) Minimize $C(T)$ subject to $R(x/T) \geq R_o$.

Theorem 3. Assume $T_i > T_o$ and $T > T_i$.

1. if $ab \leq C_i/(C_2 - C_1)$ and $R(x/0) \geq R_o$, then $T^* = 0$.
2. if $ab \leq C_i/(C_2 - C_1)$ and $R(x/0) < R_o$, then $T^* = T_i$.
3. if $ab > C_i/(C_2 - C_1)$ and $R(x/0) \geq R_o$, then $T^* = T_o$.
4. if $ab > C_i/(C_2 - C_1)$ and $R(x/0) < R_o$, then $T^* = \max(T_o, T_i)$.

Case (ii) Maximize $R(x/T)$ subject to $C(T) \leq C_B$.

Theorem 4. Assume $T > T_o$, $T_i > T_o$, and $T > T_i$.

1. if $ab \leq C_i/(C_2 - C_1)$ and $C(0) > C_B$ or if
2. $ab > C_i/(C_2 - C_1)$ and $C(T_o) > C_B$, more budget is required in order to releases the software to met the above objectives.
   (a) if $ab \leq C_i/(C_2 - C_1)$ and $C(0) = C_B$, $T^* = 0$.
   (b) if $ab \leq C_i/(C_2 - C_1)$ and $C(T_o) = C_B$, $T^* = T_o$.
   (c) if $ab > C_i/(C_2 - C_1)$ and $C(T_o) = C_B$, $T^* = T_o$.
   (d) if $ab > C_i/(C_2 - C_1)$ and $C(T_o) = C_B$, $T^* = T_o$.

1.4.1.4 Penalty-cost criterion

Here we introduce the concept of delivery time (time at which software is supposed to be released for use). If the manufacturer tails to release the software at the scheduled delivery time, he has to pay a price termed as penalty cost. Let $T_s$ (the scheduled delivery time) be a random variable with cumulative distribution function $G(t)$ with finite probability density function $g(t)$. If $P_c(t)$ is the penalty cost in $(0,1]$ due to delay in software
release, then the expected penalty cost in \((T,T)\) is expressed as Yamada el al. (1984).

\[
\int_0^T p_i(T - t) dG(t)
\]

The total expected software cost during software life cycle can be written as:

\[
C(T) = C(T_{/T_1}) = C_i m(T) + C_i (m(T) - m(T)) + C_i T + \int_0^T p_i (T - t) dG(t)
\]

We assume expected penalty cost to be an increasing function in \(t\).

Differentiating \(C(T)\) with respect to \(T\) we get

\[
C'(T) = (C_i - C_i) m'(T) - \int_0^T \frac{d}{dT} p_i (T - t) dG(t)
\]

Equating \(C'(T) = 0\), we get

\[
(C_i - C_i) m'(T) - \int_0^T p_i (T - t) dG(t) = C_i
\]

Assuming the release policy is based on minimizing \(C(T)\) subject to \(R(x/T) \geq R_0\) where \(T \geq T_1\),

We consider the following two cases.

Case (i) \(T_1\) is deterministic

Then,

\[
G(t) = \begin{cases} 1, & t \geq T_1 \\ 0, & t < T_1 \end{cases}
\]

Where \(T_1\) is known. From (1.100), we get

\[
Q(T_1) \equiv (C_i - C_i) m'(T_1) - \frac{d}{dt} p_i (T - T_1) = C_i
\]

The left hand side of (1.102) is a decreasing function in \(T(T \geq T_1)\), where \(Q(T_1) \equiv (C_i - C_i) m'(T_1) > 0\) and \(Q(\infty) > 0\).
Therefore, if \( Q(T_0) > C_1 \), there exits a finite and unique \( T(T_0) > T \) satisfying (1.102). \( C(T) < 0 \) for \( T < T_0 \) and \( C'(T) > 0 \) for \( T_0 < T < \infty \). Thus \( C(T) \) is minimum for \( T = T_0 \). If \( Q(T_0) \leq C_1 \), \( C(T) > 0 \) for all \( T > T_0 \). Thus \( T = T_0 \) minimizes \( C(T) \).

We know that \( R(x/T) \) is increasing for all \( T(T_0 < T < \infty) \). Thus if \( R(x/T) < R_0 \), there exists a finite and unique \( T' > T_0 \) satisfying \( R(x/T)/\gamma = T = R_0 \).

We, thus have the following theorem.

Theorem 5. Assume \( T_i > T_0, T_i > T, T_i > T_0 \).

1. if \( Q(T) \leq C_3 \) and \( R(x/T) \geq R_0 \), then \( T' = T \).
2. if \( Q(T) \leq C_3 \) and \( R(x/T) < R_0 \), then \( T' = T_0 \).
3. if \( Q(T) > C_3 \) and \( R(x/T) \geq R_0 \), then \( T' = T_0 \).
4. if \( Q(T) > C_3 \) and \( R(x/T) < R_0 \), then \( T' = T \max(T_0, T) \).

Case (ii) \( T_i \) has an arbitrary distribution

Let \( T_i \) have an arbitrary \( G(t) \) distribution with finite mean \( \mu \), then from (1.100), we have

\[
P(T) = (C_2 - C_1)\mu(T) - \int_0^T \frac{d}{dT} p_i(T - t)dG(t) = C_3
\]

Note that \( P(T) \) is a decreasing function in \( T \), with \( P(0) = (C_2 - C_1) \), \( P(0) > 0 \) and \( P(\infty) < 0 \). If \( P(0) > C_3 \), there exists a finite and unique \( T(T_0) \).

Thus, \( T = T \) minimizes \( C(T) \).

If \( P(0) \leq C_3 \), \( T=0 \) minimizes \( C(T) \). Combining the cost and reliability requirements as in case (i) above, we have the following theorem.

Theorem 6. Assume \( T_i > T_0 \) and \( T_i > T_0 \).

1. if \( P(0) \leq C_3 \) and \( R(x/0) \geq R_0 \), then \( T' = 0 \).
2. if \( P(0) \leq C_3 \) and \( R(x/0) < R_0 \), then \( T' = T_0 \).
3. if \( P(0) > C \) and \( R(z/0) \geq R_0 \), then \( T' = T_0 \).

4. if \( P(0) > C \) and \( R(z/0) < R_0 \), then \( T' = T \max(T_0, T_1) \).

### 1.4.1.5 Testing Effort Criterion

In the release policies discussed so far, the testing cost is increasing in \( T \). If \( T \) becomes infinitely large, so does the testing cost? In reality, no software developer will spend infinite resources on testing the software. Testing resources include CPU time, man power etc. Testing effort curves have been used in the literature to measure testing resources (Yamada and Othera (1990)). The most commonly used testing effort curves are either exponential. In this section, we discuss release policy for an exponential SRGM under the added assumption that testing resources are described by an exponential testing effort curve. For an exponential testing effort curve, the instantaneous testing resources may be expressed as

\[
w(t) = \alpha \beta e^{-\beta t}
\]

where \( \alpha \) and \( \beta \) are the parameters of the testing effort curve. The testing effort expenditure in time \( t \) is given as

\[
W(t) = \int_0^t w(x) dx = \alpha (1 - e^{-\beta t})
\]

It may be noted that the total testing effort expenditure never exceeds \( \alpha \) even if the software is tested for an infinitely large time before release. Assuming that the s-expected number of errors detected in \( (t, t + \Delta t) \) to the current testing effort expenditure is proportional to the s-expressed errors remaining in the software at time \( t \), we have

\[
\frac{m'(t)}{w(t)} = b(a - m(t))
\]

where \( a \) is the total error content and \( b \) is the constant of proportionality. Integrating we get
\[ m(t) = a\left[1 - e^{-at}\right] \]  

It may be seen that even if the software is tested for an infinitely large span of time, all the faults in the software are not detected. At \( t = \infty \),

\[ m(\infty) = a\left[1 - e^{-at}\right] \]  

Assuming that the release policy is based on minimizing cost subject to reliability not less than a predefined reliability objective \( R_0 \), we have

\[ C(T) = C_1 m(T) + C_2 (m(T) - m(T)) + C_3 W(T) \]  

When \( C_3 \) is the cost per unit testing effort expenditure. Differentiating with respect to \( t \), we get

\[ C'(T) = w(T) \left[ -(C_2 - C_1) be^{-at} + C_3 \right] \]  

Since \( w(T) > 0 \) for \( 0 < T \), \( C'(T) = 0 \) if

\[ (C_2 - C_1) ab e^{-at} = C_3 \]  

The left hand side of (1.109) is decreasing in \( T \). It is easily seen that if

\[ (C_2 - C_1) ab > C_3 ab e^{-at} = C_3 \]  

There exits a finite and unique \( T(T_0) > 0 \) satisfy (1.109). Thus, \( T = T_0 \) minimizes \( C(T) \). If \( (C_2 - C_1) ab \leq C_3 \), (1.109) has no positive solution \( > 0 \) i.e., \( C(T) \) is minimum for \( T=0 \). If \( (C_2 - C_1) ab e^{-at} \geq C_3 \), \( C(T) < 0 \) for all \( T(0 \leq T < \infty) \).

The software reliability \( R(x/T) \) is given by

\[ R(x/T) = \exp\left\{e^{-am(T)} - e^{-am(T+c)}\right\} \]

It can be easily verified that \( R(x/T) \) is increasing in \( T \) with

\[ R(x/0) = e^{-m(x)} \] and \( R(x/\infty) = 1 \).

If \( R(x/0) < R_0 (0 < R < 1) \), there exists a finite and unique \( T = T_i \) satisfying

\[ R(x/T)_{T=T_i} = R_0. \]

Combining the cost and reliability requirements, we have the following theorem.

Theorem 7. Assume \( T_i > T_0 \) and \( T_i > T \).
1. if \((C_2 - C_1)ab > C_3\) and \((C_2 - C_1)ab^{-0.5} < C_3\)

\[ T' = \max(T_0, T_1) \text{ for } R(x/0) < R_0 < 1, \text{ or } \]

\[ T' = T_0 \text{ for } R(x/0) \geq R_0 \]

2. if \((C_2 - C_1)ab \leq C_3\)

\[ T' = T_1 \text{ for } R(x/0) < R_0 < 1, \text{ or } \]

\[ T' = 0 \text{ for } R(x/0) \geq R_0 \]

3. \((C_2 - C_1)ab^{-0.5} \geq C_3\)

\[ T' = T_1 \text{ for } R(x/0) < R_0 < 1, \text{ or } \]

\[ T' = 0 \text{ for } R(x/0) \geq R_0 \]

It may be noted in the 3rd case that though (theoretically) optimum release time is infinite, software may be released, if desired, once the required reliability level is achieved through testing. When the testing effort curve is not exponential but Rayleigh or some other curve released policies can be similarly discussed.

**1.5 Parameter Estimation**

Binomial and Poisson type models, occupy an important position in software reliability work. An entire section will be devoted to a discussion of basic inference procedures for them. Although the results in the following section are specific to the above model types, the discussion illustrates several general points about inference for all models in general. Parametric point and internal estimation will be the primary topics considered. Three popular methods will be presented. The first, maximum likelihood estimation has in our experience been the best choice and is presented at great length in subsection 1.5.1. The second method is that of least squares. It provides an excellent alternative to maximum likelihood and is described in subsection 1.5.2. Finally, the Bayesian approach to estimation is briefly covered in sub-section 1.5.3.
1.5.1 Maximum Likelihood Estimation (MLE)

The most important and widely used formal estimation technique is the method of maximum likelihood. Estimation by maximum likelihood is a general technique that may be applied which the underlying distributions of the data are specified or know. This sub-section deals with a great many results for binomial and Poisson type’s models using failure time and grouped data.

The foundation of the maximum likelihood method is the likelihood function. The function is defined as the joint density of the observed data, \( L(\beta; Y_o) \). This in turn, is considered to be a function of the unknown set of \( w+1 \) parameters \( \beta \). Here \( Y_o \) represents a set of observations.

For the notion, we have the definition of maximum likelihood estimators. For each data set, let \( \hat{\beta} \) be the values of the parameters that make \((\beta; Y_o)\)as large as possible.

These maximizing values will, of course, be functions of the data. The functions themselves are called the maximum likelihood estimators. The values these functions take on are known as the maximum likelihood estimates.

Maximum likelihood estimates can be obtained by solving the simultaneous equations

\[
\frac{\partial L(\beta; Y_o)}{\partial \beta_k} = 0 \quad k = 0, \ldots, w
\]

In practice it is customary and often more convenient to work with \( \ln L(\beta; Y_o) \) instead of \( L(\beta; Y_o) \), the derivatives of both vanishing together. Thus we will find our estimates by solving the simultaneous equations (which are called the maximum likelihood equations).

\[
\frac{\partial \ln L(\beta; Y_o)}{\partial \beta_k} = 0 \quad k = 0, \ldots, w
\]
Generally the maximum likelihood equations are highly complicated and a numerical solution will be possible only with a computer.

Maximum likelihood estimators possess many desirable optimum properties such as consistency, efficiency and asymptotic normality Kendall and Stuart (1961). An estimator is said to be consistent if its variance tends to zero and if its expectation tends to the true population parameter as the sample size tends to infinity. If two different estimators have the same expectation, than the one with the smaller variance is said to be more efficient. Furthermore, an estimator is called asymptotically normal if its distribution is almost normal for sufficiently large sample size. These properties "say" that for large sample size the maximum likelihood estimators are as good a set of estimators as there is by a large sample size, we mean one in which the information content approaches infinity. For small to medium sample size, other estimators (for example, least squares estimators) may be better. That is, if other estimators, such as the least squares estimators discussed in sub-section 1.5.2, might be just as good but not better. The asymptotic normality property of maximum likelihood estimators is important because it will often be the only alternative available to establish confidence internals for the unknown parameters.

Finally maximum likelihood estimators possess an important property that is after called the invariance property. This property means that the MLE for any one-to-one function of P. say Q, is given by a straightforward substitution as $Q(\hat{\beta})$, where $\hat{\beta}$ is the maximum likelihood estimators for $\beta$. Thus, the maximum likelihood technique provides a simple way of finding estimators having "good" properties.

Point estimates are useful, yet it is often desirable to have them accompanied by some measures of the possible error of the estimate. For
example, a point estimate might be accompanied by some interval in which the true value of the parameter lies with some measure of confidence. The likelihood function and the asymptotic normality property (Mood et al. (1974) and Kendall and Stuart (1961)) enjoyed by maximum likelihood estimation can be used to establish these interval estimates under the appropriate conditions.

Let first our one parameter example. It can be shown by Kendall and Stuart (1961) that the maximum likelihood estimator of $\beta_1$ is asymptotically normally distributed with mean $\beta_1$ and variance $1/I(\beta_1)$, where $I(\beta_1)$ is the expected, or Fisher information given by:

$$I(\beta_1) = E\left[ -\frac{\partial^2 \ln L(\beta_1; Y_j)}{\partial \beta_1^2} \right]$$

Thus we can write

$$\frac{\hat{\beta}_1 - \beta_1}{\sqrt{I(\hat{\beta}_1)}} \sim N(0,1)$$

where $N(0,1)$ denote a normal distribution with zero mean and unit variance. From equation (1.114) it can be shown that the upper and lower limit of an approximate $100(1-\lambda)$ per cent confidence interval for $\beta_1$ are given by:

$$\hat{\beta}_1 \pm \frac{K_{1-\alpha/2}}{\sqrt{I(\hat{\beta}_1)}}$$

where $K_{1-\alpha}$ is the appropriate normal deviate 95 per cent confidence interval for $\beta_1$. Such an approximate confidence interval may include “impossible” value for $\beta_1$ outside the range of permissible values. This problem, a frequent occurrence for small sample sizes, can be avoided by applying the asymptotic normal distribution to a transformation of $\beta_1$ for
which the range is unrestricted. For example, if $\beta_i$ is restricted to positive values then $\ln \beta_i$ will have an unrestricted range.

An extension of this discussion on confidence internals to more than one parameter is straightforward. The result is that $\hat{\beta}$ is distributed approximately as the multivariate normal distribution with mean equal to $\beta$ and covariance matrix that is the inverse of the Fisher information matrix whose entries are given as

\[ I_{kl}(\beta) = E \left[ -\frac{\partial^2 \ln L(\beta; Y_0)}{\partial \beta_k \partial \beta_l} \right], \quad k, l = 0, \ldots, w \tag{1.116} \]

The diagonal elements of the covariance matrix give the variance of the respective parameters. The off-diagonal elements give the covariance between the corresponding elements.

Let $E$ denote the event on which we are conditioning. The conditional likelihood can be obtained by dividing. The unconditional likelihood functions by the marginal probability of $E$ equaling its observed value (denoted by $e$) for a discrete event or by the marginal density function of $E$ for a continuous event. Thus, in the discrete case

\[ L(\beta / E; Y_0) = \frac{L(\beta; Y_0)}{P(E = e; \beta)} \tag{1.117} \]

To be of any gain over the use of unconditional likelihood function equation (1.117) must depend on fewer parameters, the missing parameters being estimated (if desired) in a different fashion.

The efficiency, consistency, asymptotic normality, and invariance properties of unconditional maximum likelihood estimators apply in this case. The adequacy of the normal approximation for generating confidence interval must, of course, still be checked for each particular application. If it is inadequate, an appropriate transformation or the likelihood ratio should be used.
1.5.1.1 Failure Time Data

In the following sub-sections we will derive the necessary formulas to do point and internal estimation based on failure time data of both binomial and Poisson type models. We treat first the subject of point estimation for model parameters, followed by interval estimation for these same parameters. We conclude with a discussion on estimation for derived quantities, such as the present failure intensity.

1.5.1.1.1 Point estimation of model parameters

We suppose that estimation is to be performed at a specified time $t_e$, not necessarily corresponding to a failure, and with a total of $m_e$ failures being experienced at times $t_1, ..., t_m$. The unconditional likelihood function in this case is, in complete generality,

$$ L(\beta; t_1, ..., t_{m_e}) = f(t_1, ..., t_{m_e}) \cdot P[T_{m_e} + 1 > t_e / T_i = t_1, ..., t_{m_e} = t_{m_e}^*] $$  

or equivalently (though less commonly expressed this way)

$$ L(\beta; t_1, ..., t_{m_e}) = \left[ \prod_{i=1}^{m_e} f(t_1, ..., t_{m_e}) \right] \cdot P[T_{m_e} + 1 > t_e / T_i = t_1, ..., t_{m_e} = t_{m_e}^*] $$  

where $f(t_1, ..., t_{m_e})$ is the joint density function of $(T_1, ..., T_m)$, and $T_i$ the random variable denoting the time of the $i^{th}$ failure. Note that $f(t_i / t_0) = f(t_i)$.

A property of both binomial and Poisson-type models that simplifies Equation (1.119) to some extent is that the conditional density function of $T_i$ depends only on the previous failure time $T_{i-1}$ that is:

$$ f(t_i / t_1, ..., t_{i-1}) = f(t_i / t_{i-1}) $$  

Thus Equation (1.119) can be expressed as

$$ L(\beta) = \left[ \prod_{i=1}^{m_e} f(t_i / t_1, ..., t_{i-1}) \right] \cdot P[T_{m_e} + 1 > t_e / T_i = t_{m_e}] $$  

where, for simplicity, we have suppressed the explicit notation of the dependency of $L$ on $(t_1, ..., t_{m_e})$. 

44
As we mentioned in the introduction to Section 1.5.1, estimation can also be based on a conditioning event. Such a definition will depend on whether the testing process is observed for a pre-determined time or the testing period is taken up to a pre-assigned number of failures. In practice, we do not predetermine the total test time or the number of failures to be observed. Rather estimation is performed at various times (as needed or on a regular basis) throughout testing. Cost and schedule may, in fact, influence the available time for system testing for some projects. The material presented here pertains solely to the former type of conditioning.

By selecting the time $t_e$ as our basis of observation, the number of failures experienced in the interval $(0, t_e]$ will be a (discrete) random variable. The realization of this random variable is $m_e$ and will be used as the conditioning event. Thus, the conditional likelihood function given by equation:

$$L(\beta | m_e) = \frac{L(\beta)}{P[M(t_e) = m_e; \beta]}$$

1.122

It is worthwhile at this point to discuss an interesting result that is obtained when using Equations (1.121) and (1.122) for binomial-and Poisson-type models. The result is this: Provided $\mu(t_e) = m_e$, unconditional and conditional estimation yields identical point and interval estimates for either type of model. Furthermore the estimates for both model types will also be the same. That is, the two types of models are indistinguishable from each other using either type of estimation. The models are of course, distinguishable from their other properties.

This condition is intuitively appealing since $m_e$ failures have been observed by time $t_e$. In addition, for scaled mean value functions, $\mu(t_e) = m_e$ is the solution of the unconditional maximum likelihood
equation for \( \beta_o \) for Poisson-type models. It is approximately the solution obtained using the unconditional maximum likelihood equation for \( \beta_o \) for binomial-type.

**Binomial-type Models**

This models are of the finite failures category and as such are described with \( \beta_o = \mu_o \), and \( \mu_o(t; \beta_A) = F_u(t; \beta_A) \). For the sake of convenience, the notation \( F_u(t) \), \( f_u(t; \beta_u) \) and \( z_u(t) \) will be adopted. In addition, because of the extreme importance of the parameter \( \mu_o \), the dependency of the likelihood function on this parameter and \( \beta_i \) will be noted explicitly. That is, \( L(\mu_o, \beta_i) \) will be used instead of \( L(\beta) \).

First we consider the unconditional likelihood approach. To particularize equation (1.121) we must evaluate the two factors on the right side of the equation. Let us stand with the first factor and one of its typical terms, that is, \( f(t_i/t_{i-1}) \). Using

\[
\frac{d}{dt} P[T_i > t_i/t_{i-1} = t_{i-1}]
\]

The first factor of equation (1.121) becomes

\[
\prod_{i=1}^{\nu} f(t_i/t_{i-1}) = \left[1 - F_u(t_i)\right]^{\nu_i - \nu} \prod_{i=1}^{\nu} (u_o - i + 1) f_u(t_i)
\]

The likelihood function after the appropriate substitutions and algebra is:

\[
L(\mu_o, \beta_i) = \prod_{i=1}^{\nu} f_u(t_i)\left[1 - F_u(t_i)\right]^{\nu - \nu_i} \prod_{i=1}^{\nu} (u_o - i + 1) f_u(t_i)
\]

**Poisson-type Models**

Proceeding in a fashion similar to what we did for binomial-type models, we have the following results:

\[
f(t_i/t_{i-1}) = f(t_i) \exp\left[\mu(t_i) - \mu(t_{i-1})\right]
\]

and on substitution of those expressions into equation (1.121).

\[
L(\beta) = \prod_{i=1}^{\nu} \lambda(t_i) \exp[-\mu(t_i)]
\]
The probability of observing $m_e$ failures by time $t_e$

$$P[M(t_e) = m_e] = \frac{[\mu(t_e)]^{m_e}}{m_e!} \exp[-\mu(t_e)]$$  \hspace{1cm} 1.128

and thus after the appropriate substitutions the conditional likelihood is:

$$L(\beta_e/m_e) = \frac{m_e! \prod_{l=1}^{m_e} \delta(t_l)}{[\mu(t_e)]^{m_e}}$$  \hspace{1cm} 1.129

This result is identical to the one for binomial-type models. The conditional likelihood function cannot be used to estimate $\beta_0$ (which is interpreted as $v_0$ for finite failures models). However, if we use $\mu(t_e; \hat{\beta}) = m_e$ to estimate $\beta_0$, the conditional likelihood estimates are equipment to the unconditional likelihood estimates.

### 1.5.1.2 Grouped Data

Estimation based on grouped failure data parallels that based on failure time data. The only difference between the two will be in the formulas regarding $\beta$. For one parameter scaled mean value functions $\beta_e$ is, once again, the basis for all inferences. Results from the previous sections regarding $\beta_0$ and derived quantities remain unchanged except that $m_e$ should be replaced with $y_p$ and $t_e$ with $x_p$. Hence, the following section will tend to be brief and will give the necessary formulas for the estimation of $\beta$.

#### 1.5.1.2.1 Point Estimation of Model Parameters

Let $y_l (l = 1, ..., P)$ be the number of failures in $(0, x_l]$. Then the likelihood function is:

$$L(\beta; y_1, y_2, ..., y_P) = \prod_{l=1}^{P} P[M(x_l) = y_l]$$

$$= \prod_{l=1}^{P} P[M(x_l) = y_l, M(x_{l-1}) = y_{l-1}, P[M(0) = 0]$$  \hspace{1cm} 1.130

Where $x_0 = y_0 = 0$, and by assumption, $P[M(0) = 0] = 1$
It is easy to see that

\[ \left[ \kappa = (1-x)\nu - \nu \right] \prod_{d} = \left[ \frac{1}{1-x} - \kappa = (1-x)\nu - \nu \right] \prod_{d} = \left( \frac{\kappa}{\nu} \right) \gamma \]

Since the Poisson process has independent increments may be written as:

**Poisson-type Models**

\[ \kappa = \left( x \right) \nu \]

represents the number of failures in \( X \). That is, \( \kappa \)

where

\[ \kappa =\begin{cases} \left( x \right) \nu - \nu \end{cases} \]

made. Thus equation (1.130) becomes:

\[ \nu = \begin{cases} \left( x \right) \nu - \nu \end{cases} \]

Each factor of the unconditional likelihood is given by the obvious

**Binomial-type Models**

maximum likelihood equation for \( q \). For scaled mean value functions, the results of the

be nearly satisfied. From this we see that \( \kappa = \left( x \right) \nu \) is the results of the

never be satisfied. However, if the model is just the condition will

require the more restrictive \( \kappa = \left( x \right) \nu \). This will in practice

unconditional estimation. For binomial-type models, equivalence hence

equivalent form in estimation point of view. The only exception is

and the two types of models (Poisson and binomial) are

II

If the two forms of estimation given by equation (1.130) and

\[ \left[ \frac{\kappa}{\nu} = \left( \frac{\nu}{\nu} \right) \right] \prod_{d} = \left( \frac{\kappa}{\nu} \right) \gamma \]

The conditional likelihood when grouped data is used is given as:
\[
P[M(x_i) - M(x_{i-1}) = y_i] = \frac{(\Delta \mu_i)^y}{y!} \exp(-\Delta \mu_i)
\]

and
\[
L(\beta) = \left\{ \prod_{i=1}^{n} \frac{(\Delta \mu_i)^y}{y!} \right\} \exp[-\mu(x_i)]
\]

1.5.2 Least Squares Estimation (LSE)

The second method of statistical inference describes the least squares approach. We have already discussed in section 1.5.1 that no other estimators are better than maximum likelihood estimate for large sample size so one can arise question why consider least square estimators? Our answer will be for small or medium size samples least square estimators may be better. For example, they may have smaller bias or they may approach normality faster. Least squares estimation, like maximum likelihood estimation, is a fairly general technique which can be applied in most practical situations.

The approach we take is to estimate the model parameters by fitting the functional relationship of the failure intensity with respect to the mean value function to the observed failure intensity. This method will be discussed in detail for two parameter model groups, especially the exponential class and geometric family.

We begin with a discussion on how to calculate the observed failure intensity (Section 1.5.2.1). Point and interval estimation of model parameters is discussed in section 1.5.2.2 and 1.5.2.3 respectively. Estimation of derived quantities is covered in section 1.5.2.4.

1.5.2.1 Estimation of failure Intensity

The determination of the observed failure intensity, using both failure time and grouped data, is a straightforward process. We begin by looking at failure time data. Let the observation interval \((0, t_e]\) be partitioned at
every \( k \) failure occurrence time so that there are \( p \) disjoint subintervals. (Note that \( p \) is the smallest integer greater than or equal to \( m_{ik}/k \)). The observed failure intensity \( r_{il} \) for the \( l \)th subinterval \((t_{i(l-1)}, t_{i,l}]\) is given by

\[
  r_{il} = \begin{cases} 
    \frac{k}{t_{i(l)} - t_{i(l-1)}}, & l = 1, \ldots, r - 1 \\
    \frac{m_{i} - k(p - 1)}{t_{i} - t_{i(l-1)}}, & l = p 
  \end{cases}
\]  

where the subscript notation denotes the product of the two indices. For example, for \( k = 5 \) we can obtain the observed failure intensity for the second \((l = 2)\) subinterval \((t_{5,2}, t_{5,0}]\) as \(5/(t_{5,0} - t_{5})\).

Note that the observed failure intensities as such as independent of each other since each subinterval is chosen to be disjoint with respect to the others. This independence of the data makes it possible to apply the least squares method, which will be discussed in Section 1.5.2.2. The ordinary least squares method can not be applied if the relationship of the cumulative number of failures with time is used, because cumulative failures are dependent on the previous data.

The estimate of the mean value function for the \( l \)th subinterval is given by

\[
m_{il} = k(l - 1), \quad l = 1, \ldots, p
\]

It should be pointed out that the use of the midpoint was also investigated. That is, the mean value function could be estimated by taking the number of failures at the midpoint of the interval rather than at the end. A simulation study indicated that estimates of model parameters obtained based on the midpoint of each interval were biased specifically; the initial failure intensity \( \lambda_{0} \) tends to be overestimated.

Grouping a small number of failures (a small value of \( k \)) will result in large variations in the observed failure intensity. On the other hand, grouping a large number of failures (a large value of \( k \)) will result in too
much smoothing. A group of five failures (that is, $k = 5$) has been selected as a reasonable compromise in the following analysis. Although some information may be lost because of grouping failures, an advantage of this approach is that no specific model or distribution is assumed.

1.5.2.2 Point Estimation of Model

The functional relationship of the failure intensity with respect to the expected number of failures is given in the form of model. The model parameters $\beta$ can be estimated by fitting the functional relationship $\lambda(m_i; \beta)$ to the observed failure intensity $[(m_i, r_i), i = 1, \ldots, p]$. It should be pointed out that there are some other approaches to the least squares method. For instance, the functional relationship $\lambda(t; \beta)$ may be used instead of $\lambda(m_i; \beta)$.

Let $\varepsilon_i$ represent the logarithm of the ratio of the $i^{th}$ data point to the model value. Then, we have

$$\ln r_i = \ln \lambda(m_i; \beta) + \varepsilon_i$$  \hspace{1cm} 1.138

It is assumed that all $\varepsilon_i$'s are independent, identically distributed normal random errors with zero mean and a common variance $\sigma^2$. The normality assumption is not needed for obtaining the least squares estimates, but it is necessary for constructing confidence intervals. The validity of this assumption for the geometric family has been investigated by a simulation study.

The estimates of the model parameters can be found so that the sum of the squares of $\varepsilon_i$'s, that is,

$$S(\beta) = \sum_{i=1}^{p} \varepsilon_i^2 = \sum_{i=1}^{p} [\ln r_i - \ln \lambda(m_i; \beta)]^2$$  \hspace{1cm} 1.139

is minimized.
Note that this minimization is equivalent to the minimization of the sum of the squares of the relative errors (the absolute errors divided by the model values). The use of the relative errors yields the same weight for any level of failure intensity, that is, for all data points. On the other hand, if the absolute errors are used, the estimates will be significantly influenced by the data points with large failure intensity. In tracking and predicting reliability, managers usually need a constant relative (rather than absolute) accuracy for failure intensity. In general, the approach described yields a nonlinear regression problem. The one exception, resulting in an analytic solution for $\beta$, occurs for the geometric family of models.

1.5.2.3 Interval Estimation of Model Parameters

The next step after obtaining the point estimates of the parameters is the computation of confidence intervals for the parameters. These will be useful in evaluating the accuracy of the estimates. The method for generating these intervals will be described for two-parameter model groups such as the exponential class and the geometric family. Modification of the method for a model with more than two parameters is straightforward.

In addition to the assumption that the $\epsilon_i$s are independent random variables with zero mean, we also assume that they are normally distributed with a common variance $\sigma^2$. Then, we can estimate the common variance as

$$
\sigma^2 = \frac{\sum(\hat{\beta}_0, \hat{\beta}_1)}{p-2}
$$

1.140

This reflects the variation about the regression line. We estimate the variance of $\hat{\beta}_j (j = 0,1)$ as
\[ \text{var}[\hat{\beta}_j] = \frac{\hat{\sigma}^2}{SS} \sum_{i=1}^{n} \left[ \frac{\partial}{\partial \beta_i} \ln \lambda(m_i) \right]^2 \]  

1.141

Where

\[ SS = \sum_{i=1}^{n} \left[ \frac{\partial}{\partial \beta_0} \ln \lambda(m_i) \right]^2 \sum_{j} \left[ \frac{\partial}{\partial \beta_j} \ln \lambda(m_j) \right]^2 - \left[ \sum_{i} \left( \frac{\partial}{\partial \beta_0} \ln \lambda(m_i) \frac{\partial}{\partial \beta_j} \ln \lambda(m_j) \right) \right] \]  

1.142

The estimated covariance of \( \hat{\beta}_0 \) and \( \hat{\beta}_1 \) is

\[ \text{cov}[\hat{\beta}_0, \hat{\beta}_1] = \frac{\hat{\sigma}}{SS} \sum_{i=1}^{n} \left[ \frac{\partial}{\partial \beta_0} \ln \lambda(m_i) \frac{\partial}{\partial \beta_1} \ln \lambda(m_i) \right]^2 \]  

1.143

In Equations (1.142) and (1.143) \( \hat{\beta}_0 \) and \( \hat{\beta}_1 \) are used to evaluate the expression.

Assuming that the least squares estimates \( \hat{\beta}_0 \) and \( \hat{\beta}_1 \) are normally distributed, a 100(1 - \( \alpha \)) per cent confidence interval for the parameter \( \beta_j \) (\( j = 0,1 \)) is given by

\[ \hat{\beta}_j - t_{p-2,\alpha/2} \sqrt{\text{var}[\hat{\beta}_j]} \leq \beta_j \leq \hat{\beta}_j + t_{p-2,\alpha/2} \sqrt{\text{var}[\hat{\beta}_j]} \]  

1.144

where \( t_{p-2,\alpha/2} \) is the upper \( \alpha/2 \) percentage point of the t distribution with \( p-2 \) degrees of freedom. Note that the t distribution is used since the variance is estimated from the data.

1.5.3 Bayesian Inference

The third and final method of statistical inference to be considered is the so-called Bayesian approach. Several authors including Langberg and Singpurwalla (1985), Meinhold and Singpurwalla (1983), Littlewood (1981), Okumoto and Goel (1978b), and Littlewood and Verrall (1973 and 1974) have developed or extended software reliability models based on the Bayesian viewpoint over the past 15 years or so. Here we will present a basic introduction to Bayesian inference. The Bayesian approach provides an efficient method for incorporating various subjective and objective data sources into the analysis. However, it is
markedly inferior to maximum likelihood from a practical viewpoint. The analysis is more complex. Hence it is more difficult to convey the concepts. Also, computation is more complicated. Thus, no attempt will be made to be as comprehensive as we were with maximum likelihood estimation.

Like the two previous approaches the aim is to estimate the unknown parameters $\beta$ based on the observed data $Y_0$. In maximum likelihood, least squares, and other classical methods of estimation $\beta$ is treated as a vector of constants having unknown values. A major part of classical inference centers around determining estimators for $\beta$ and their statistical properties (distribution, bias, efficiency, etc.). Confidence statements must be interpreted properly. What the interval really means is that in repeated sampling the computed confidence intervals include the true value of $\beta$ a proportion $(1-\alpha)$ of the time. We cannot properly say that $\beta$ lies within the interval with probability $1-\alpha$. However, the Bayesian interpretation, while still considering $\beta$ as unknown, treats the parameters as random variables with known distributions. All inference is based directly on these distributions. Questions on bias and efficiency no longer exist. Confidence statements directly represent valid probability statements on the model parameters. Thus, a statement such as

$$P|\beta_{low} \leq \beta \leq \beta_{high}| = 1-\alpha$$

1.145

gives the probability that $\beta$ lies between $\beta_{low}$ and $\beta_{high}$.

In the following development we will look at the case of one unknown parameter, say $\beta_1$. The generalization to two or more unknown parameters is straightforward and will be indicated later.
Baye’s theorem is the foundation of Bayesian inference. Suppose $\beta_k$ has a density function given by $f(\beta_k)$. Then applying Baye’s theorem, given the observed data $Y_D$, the conditional density function of $\beta_k$ is

$$f(\beta_k/Y_D) = \frac{f(Y_D/\beta_k)f(\beta_k)}{f(Y_D)}$$

where the marginal distribution $f(Y_D)$ may be obtained according to

$$F(Y_D) = \int f(Y_D/\beta_k)f(\beta_k)d\beta_k$$

The density function $f(\beta_k)$ is called the prior distribution of $\beta_k$. This function is supposed to represent what is known about $\beta_k$ before the data are available. It plays an important role in Bayesian analysis, for it is used to represent prior knowledge or relative ignorance. Correspondingly, $f(\beta_k/Y_D)$ is called the posterior distribution of $\beta_k$ given knowledge of the data, $Y_D$. This function represents what is known about $\beta_k$ after the data have been made available. The denominator of Equation (1.146) is a normalizing constant which insures that $f(\beta_k/Y_D)$ integrates to 1.

Given the data, $f(Y_D/\beta_k)$ may be regarded as a function of $\beta_k$ and not of $Y_D$. In so doing we can call it the likelihood function of $\beta_k$.

$$f(Y_D/\beta_k) = L(\beta_k; Y_D)$$

Baye’s theorem can now be written as

$$f(\beta_k/Y_D) \propto L(\beta_k; Y_D)f(\beta_k)$$

This says that the posterior distribution of $\beta_k$ is proportional to the product of the prior distribution of $\beta_k$ and the likelihood function. The likelihood function plays an important role here as it does with maximum likelihood estimation. All the information about $\beta_k$ coming from the data is contained in it. As the sample size increases the terms contributed by the likelihood function will tend to overwhelm the single term contributed by the prior. A typical relationship between the three
functions of Equation (1.149). Note how the occurrence of the data $Y_D$ increases the relative probability of certain values of $\beta$.

Note that we have the posterior distribution for $\beta$, what about point and interval estimation for $\beta$? For point estimation any measure of the central tendency of the posterior distribution will be adequate. Denote the Bayesian point estimate of $\beta$ as $\beta^*$. Then three such estimates are

1. the posterior mean, that is,

$$\beta^*_k = \int \beta_k f(\beta_k / Y_D) d\beta_k$$

2. the posterior median, that is, the value of the value of $\beta^*_k$ such that

$$\int_{-\infty}^{\beta^*_k} f(\beta_k / Y_D) d\beta_k \geq 0.5$$

and

$$\int_{\beta^*_k}^{\infty} f(\beta_k / Y_D) d\beta_k \leq 0.5$$

3. and the posterior mode or generalized maximum likelihood estimator Martz and Waller (1982), that is, the value of $\beta^*_k$ that maximizes

$$f_x(\beta_k / Y_D)$$

The Bayesian approach to interval estimation is direct. A $100(1-\alpha)$ percent probability interval for $\beta$ is easily obtained once we know the posterior distribution of $\beta$. We need only solve the two equations

$$\int_{-\infty}^{\beta_{up}} f(\beta_k / Y_D) d\beta_k = \frac{\alpha}{2}$$

and

$$\int_{\beta_{low}}^{\infty} f(\beta_k / Y_D) d\beta_k = \frac{\alpha}{2}$$
for the interval, lower and upper limits, \( \beta_{\text{low}} \) and \( \beta_{\text{high}} \), respectively. A 95 per cent probability interval for \( \beta \) is also obtained.

These integrals often cannot be expressed in closed form. This is also true (more so) when there is more than one unknown parameter. This does not present a problem since the integrals can be computed numerically with the use of computers.

Inference about some function of \( \beta \), say \( Q(\beta) \), can be made in a direct way. One good approach is to start with the posterior distribution of \( \beta \) and apply a change of variables to obtain the posterior distribution of \( Q \). A point estimate of \( Q \) can now be obtained as before by using the posterior mean, median, or mode and an interval estimate by using Equation (1.152) with the appropriate notational changes.

A criticism and key difficulty in carrying out a Bayesian analysis is the identification of the prior distribution. Often mathematical simplicity and convenience and the ability to describe various distributional shapes are the key considerations. See Box and Tiao (1973) for a complete discussion on the identification of an appropriate prior. In this chapter we will assume that little or nothing is known a priori about \( \beta \). Such a case can be handled using a locally uniform prior. A locally uniform prior is one that does not vary appreciably over the region in which the likelihood function is large. Outside this region a locally uniform prior also does not assume large values. Doing Equation (1.149) becomes

\[
f(\beta / Y) \propto L(\beta / Y) \tag{1.154}
\]

Thus the posterior distribution of \( \beta \) is directly proportional to the likelihood function.
The generalization to more than one unknown parameter is straightforward. If we let \( f(\beta) \) denote the joint prior distribution of \( \beta \), then the joint posterior distribution \( \beta \) is

\[
f(\beta/Y_d) \propto L(\beta; Y_d) f(\beta)
\]

1.155

Point and interval estimates for each \( \beta_k (k = 0,\ldots,w) \) can be obtained using the marginal posterior distribution of \( \beta_k \). For example, the distribution for \( \beta_k \) can be obtained from

\[
f(\beta_k/Y_d) \propto \int \cdots \int f(\beta_k/Y_d) \, d\beta_0 \cdots d\beta_{k-1} d\beta_{k+1} \cdots d\beta_w
\]

1.156

All the previous discussion regarding point and interval estimation of \( \beta_k \) can now be applied. Inferences about a function of \( \beta \) say \( Q(\beta) \), can be carried out by determining the posterior distribution of \( Q \).

Finally, a locally uniform prior for multiple parameters is one that can be approximated by a constant for all practical values of \( \beta \). The posterior distribution of \( \beta \) when a locally uniform prior is used is directly proportional to the likelihood function.