Software Reliability Estimation Using Cubic Splines Network Model

A thesis submitted for the Degree of Master of Philosophy

K.W.K.B.P.L.M. Kelanibandara

University of Colombo School of Computing

2012 December

Declaration

The Thesis is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge it does not contain any material published or written by another person, except as acknowledge in the text.

Student Number: MPhil/PT/2006/012

Author's name: K.W.K.B.P.L.M. Kelanibandara Date ……

Signature …….

This is to certify that this thesis is based on the work of Ms K.W.K.B.P.L.M. Kelanibandara under our supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certify by

Supervisor 1 Name: Prof. G.N. Wikramanayake Date ……...

Signature ……..

Supervisor 2 Name: Dr. (Mrs) J.S. Goonathillake Date ……..

Signature ……

Acknowledgment

First of all I would like to express my sincere gratitude to Prof. G.N. Wikramanayake, Professor/Director University of Colombo School of Computing, who has been my principal supervisor since the beginning of my study. Especially he provided me with many helpful suggestions, important guidance and constant encouragement during the course of this work. I would like to take this opportunity in thanking him in getting this work successfully within his kind attention.

I also wish like to express my second thank and appreciation to Dr. J.S. Goonathillake, Senior Lecturer, Department of Information Engineering, University of Colombo School of Computing who also has been my supervisor since the beginning of my study, made many valuable suggestions and gave constructive advices and feedback from the beginning to the end.

Special thanks are due to Prof. N.D. Kodikara, the Head of the Department of Post Graduate Studies and University of Colombo School of Computing for providing me useful guidelines for the thesis.

I would like to draw my special thanks to Mr. M.T.W. Ponnamperuma, Former Chairman, Vocational Training Authority of Sri Lanka, for his valuable assistance and encouragements specially to attend for the conferences.

My sincere thanks go to Mr. Tissa Bandara and Mr. Gallaba, Directors of Vocational Training authority for the fullest support in succeeding my research work.

I would like to thank my colleague Ms. Probodini Jayasinghe who gave me support for my studies. Finally my thanks go to my parents, sisters and brother in giving me support and encouragement in succeeding this thesis.

Abstract

The term quality in general, is a feeling. Thus, it is hard to describe consistently as a feeling is not consistent. Software quality is essentially a kind of quality particularly associating with software. Thus, the term software quality is also hard to describe. Hence, researchers use software quality models. Each software quality model consists of several factors which affect the software quality and they are called software quality factors. Software reliability is one of such software quality factors in nearly all the software quality models. Hence, software in order to be a high quality one, all the quality factors including software reliability has to be guaranteed. However, it is evident that software reliability is not guaranteed in almost all the commercial software development. This has been due to the lack of accuracy of the reliability estimation and the time taken to estimate the reliability in existing software reliability estimation models or software reliability growth models.

Among the commonly used software reliability growth models, Non Homogeneous Poisson Model (NHPP model) shows more accuracy than the other models. However, in order to estimate the reliability, it requires more input data (i.e. a minimum of twenty five failure data). Thus, it takes considerable time. In this thesis, a novel software reliability growth model called Cubic Spline Network model (CSN model) has been introduced for improved accuracy with respect to the existing models.

The proposed model requires relatively smaller number of past failure data as input and thus, this research will prove that it is more practical to use in the commercial software developments. Cubic splines network model has sensitivity of tuning for smaller or higher reliability estimation which has also not been introduced in the literature.

Table of Contents

Content **Page Number**

List of Figures

List of Tables

Abbreviations

AIAA - American Institute of Aeronautics and Astronautics

ANN – Artificial Neural Network

ANSI – American National Standards Institute

CMM - Capability Maturity Model

CPU – Central Processing Unit

CS I – Case Study I

CS II – Case Study II

CSN - Cubic Splines Network

FURPS – Functionality Usability Reliability Performance Supportability

FURPS + - enhance FURPS by IBM

IBM – International Business Mechanic

IEEE – Institute of Electrical and Electronics Engineers

I / O – Input / Output

ISO – International Organization for Standardization

KLOC – thousand Lines Of Code

KSLOC – thousand Source Lines Of Code

LOC – Lines Of Code

MTBF - Mean Time Between Failures

NASA - National Aeronautics and Space Administration

NHPP - None Homogenous Poisson Process

SMERFS - Statistical Modeling and Estimation of Reliability Functions for Software

SLOC – Source Lines Of Code

SRE - Software Reliability Engineering

SRGM - Software Reliability Growth Model

URPS – Usability Reliability Performance Supportability

WASR - Workshop on Applied Software Reliability

1. Introduction

The increasing dependency on computer aided systems consequence more and more software systems in operation. However, some of them can be disruptive if the software fails to meet the required level of performance. People are with the attitude of striking off software due to lack of performance. This attitude is no longer practical if the software usage is mature or the software functions are safety critical. When the consequences of the software problems are significant enough, the software quality engineers has to come forward to give solutions.

1.1 Software Quality

Generally, the quality is neither visible, nor tangible and is immeasurable. The quality is rather a feeling such that, while using, if the product causes happiness or comfort-ability in any particular aspect, the quality of the product in that particular aspect is higher and vice versa. For example, if the users experience trouble-free manipulation of the software, then the quality aspect usability of that particular software is privileged.

Software quality has no constant definition (Hoyer, et al., 2001). It is situational and it depends on the application. The software quality has a broadened scope. In order to understand the term software quality, it is important to pay attention to the literature to find how the quality has been defined by the people who have studied deeply in this subject.

Crosby summarizes his perspective on quality (Crosby, 1979) as conformance to requirements. Walter Edwards Deming (Deming, et al., 1986) states that, quality must be defined in terms of customer satisfaction. Armand Villain Feigenbaum explains his perspective on quality through the following text (Feigenbaum, 1983) "*Quality is a customer determination, not an engineer's*

determination, not a marketing determination, nor a general management determination. Kaoru Ishikawa explains "What is quality control? in Japanese Way" (Ishika, 1985) "*We engage in quality control in order to manufacture products with the quality which can satisfy the requirements of consumers.* Juran Trilogy's Definition of the word qualit (Juran, 1988) as "*fitness for use*" for the task of managing quality. Shert Wart defines that there are two common aspects of quality: One of them has to do with the consideration of the quality of a thing as an objective reality independent of the existence of man. The other has to do with what we think, feel or sense as a result of the objective reality.

IEEE defines the software quality as the degree to which software possesses a desired combination of attributes (IEEE, 1990).

ISO9126 defines the software quality as: "*the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs*" (ISO, 2001).

All the above definitions confirm the fact that the quality has a varied meaning and generally it is engaged with the satisfaction of the requirements of the user. Hence, It is important to discuss how the term quality which is situational and has a broader scope with a varied meaning can be described.

Software quality is described in the means of models which are called software quality models and these have their own quality attributes (McCall et. al., 1977). Following presents some software quality models for a better understanding.

ISO 9126 defines software quality with six software quality attributes as functionality, reliability, usability, effectiveness, maintainability and portability (ISO, 2001).

Another famous and useful categorization of factors that affect the software quality was proposed by McCall, Richards, and Walters (MaCall, et al., 1998). According to this categorization, quality is described in three categories as product operation, product revision and product transition. There are eleven (11) quality factors which would fall into these three quality categories. Product operation quality attributes are correctness, reliability, efficiency, integrity and usability. Product revision quality attributes are maintainability, testability and flexibility. Product transition quality attributes are portability, re-usability and testability.

IEEE 1061 standard defines software quality in five main attributes and they are similar to the ISO main software quality attributes and software reliability is one of the five software quality attributes (IEEE, 1990).

Boahems describes (Boehm, et al., 1976) the quality model in three levels as high level, intermediate level and primitive level. High level characteristics represent the basic high level requirements and it includes as-is-utility (how well, i.e., easily, reliably, efficiently can I use it as-is), maintainability and portability. Intermediate level represents the software quality expected from the software, i.e., portability, reliability, efficiency, usability, testability, understandability and flexibility. Primitive level represents the foundation for defining the quality matrices.

FURPS originally proposed by Robert Graby as FURPS (Functionality, Usability, Reliability, Performance and Supportability) and later enhanced by IBM Rational Software as FURPS+. These models categorize the software quality attributes as Functional (F) (Functionality) and as None Functional (URPS) (Usability, Reliability, Performance, and Supportability). The quality attributes are Functionality, Usability, Reliability, Performance, and Supportability (Gray, 1992).

Dromey software quality model represented by R. Geoff Dromey considers the quality evaluation defers from the product (Dromey, 1995). A dynamic quality model which depends on the focused software is needed to be successfully applied for different systems. This model attempt to match product properties with the software quality attributes. There are three basic elements as such product properties, quality attributes and linking product properties with quality attributes in this model. Product properties are correctness, internal, contextual and descriptive. Functionality and reliability are the attributes which would contribute to the correctness product property and the attributes of the internal product property are maintainability, efficiency and reliability. Maintainability, re-usability, portability and reliability are the attributes of contextual product property and the attributes which would contribute to descriptive product property are maintainability, re-usability, portability and usability. The summary of these quality model attributes can be tabulated as shown in Table 1.1.

Quality Attribute	ISO 9126	McCall	IEEE 1061	Boahems	FURPS/FURPS +	Dromey
Functionality	X		X		X	
Reliability	X	X	X	X	X	X
Usability	X	X		X	X	X
Effectiveness	X		X			
Maintainability	X	X	X	X		X
Portability	X		X	X		
Correctness		X				
Efficiency		X		X		X
Integrity		X				
Flexibility		X		X		X
Testability		X		X		
Portability		X				X
Re Usability		X				X
Interoperability		X				
As-is Utility				X		
Understandability				X		
Performance					X	
Supportability					X	

Table 1.1: The quality attributes of famous software quality models

According to the Table 1.1, reliability and usability are the two quality attributes which are common to most of the listed software quality models. This depicts that the reliability and the usability essentially have a drastic impact on software quality. Hence if a company is to develop high quality software, it is important to employ some efforts on software reliability and usability. However, this thesis focuses only on software reliability.

1.2 Application of Software Reliability by Software Industry

It is evident that the degree of practicing software reliability in commercial software development is still a question. Following facts were taken from the keynote address of the chair of Workshop on Applied Software Reliability

(WASR) 2006 of International Conference on Dependable Systems and Networks (Agbari, et al., 2006):

"*Research on software reliability has been active for several decades now and has produced massive amount of literature to explore new ideas, and system prototypes to experiment with the proposed ideas. Notwithstanding this proficiency, only in a few instances has research work found its way into industrial applications. This apparent uncoordination is exacerbated by the increasing need for quality and dependability guarantees in the more and more computerized modern world.*"

This clearly elaborates that, there is an urgent need of more practical solutions for software quality guarantee for instance software reliability estimation.

"*Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability*" (Martin, et al., 2008)

"*Unfortunately the methods to handle software reliability by far did not get that development which hardware reliability has undergone in last 20 decades.* " (Hoppe, 1996)

The above texts elaborate the fact that a lack of usage of reliability in the industry. However, after interviewing some of engineers of several international and local software companies, it was found that there is very little usage of software reliability estimation in the industry.

This thesis discusses the reasons for lack of usage of software reliability estimate in the industry. Based on the reasons, the features of a useful model have been identified and new SRGM using Cubic Spline Network has been

designed. There are significant improvements in the new SRGM with respect to the existing models.

1.3 Objectives

The accuracy level of the existing software reliability estimation models is not up to the standard. New software reliability estimation model called Cubic Splines Network Software Reliability Growth Model is needed to be designed so as to improve the accuracy of the existing software reliability estimation models. Finally the model is needed to be tested for accuracy to prove that the model enhances the accuracy.

1.3.1 Sub Objectives

To mitigate input data size for software reliability estimation in order to improve the usage of reliability estimation by software industry.

To introduce an automated tool for estimating the reliability using Cubic Splines Network model (CSN).

1.4 Scope

In order to test the accuracy of CSN model, real failure data of a software project is required. However, it is a very time consuming process to carryout software testing for a real project and could even take years. This is not feasible within the available time and thus secondary data which have already been collected and published were used to test the accuracy (Lyu, 1996).

1.5 Methodology

The accuracies of the existing SRGMs are first analyzed. Then the facts which affect the accuracy are identified. The ways to avoid the effects of the facts

which impair the accuracy are studied. The most suitable way of avoiding the effects is to be understood. Then according to the findings, new SRGM is to be designed. The new model is to be tested using the existing datasets. Finally the new model is statistically proved for the accuracy.

1.6 Motivation

Today, people are highly dependent on computer aided systems. However, along with the computer aided systems, the major and important role is played by the software. Numerous types of software can be found ranging from free of charge to billions of dollars.

The size and complexity of computer-aided systems software have grown dramatically, and the trend will continue in the future too. Examples of highly complex computer aided systems software are in the aviation industry, in scientific researches, in telecommunication industry and for military activities. For example, the NASA Space Shuttle on board software is approximately 500,000 lines of code and the ground control and processing software is approximately 3.5 million lines of code. The software used in the telecommunication industry to control the phone carries hundreds of millions of lines of source code. Windows operating system used in many home and office computers is of 28 million lines of code.

In comparison with advancements of hardware components, software achieves less progress caring a larger burden of the total system. The potential of integrating the independently developed software into such hardware system has enabled software designers to develop high complex systems. However, in comparison to the hardware technology, the software technology has not succeeded in keeping measures such as quality. Software reports major source of outages in many systems.

The consequences of failures in software have been impacted in several major systems.

"*In the NASA Voyager project, the Uranus encounter was in jeopardy because of late software deliveries and reduced capability in the Deep Space Network. Several Space Shuttle missions have been delayed due to hardware/software interaction problems. Software glitches in an automated baggage- handling system forced Denver International Airport to sit empty more than a year after airplanes were to fill its gates and runways. The Hong Kong Airport experienced a similar problem"* (Martine, et al., 1976).

Unfortunately, software can also kill people. The following is an example for such situation.

"*The massive Therac-25 radiation therapy machine had enjoyed a perfect safety record until software errors in its sophisticated control systems malfunctioned and claimed several patients' lives in 1985 and 1986. On October 26, 1992, the Computer Aided Dispatch system of the London Ambulance Service broke down right after its installation, paralyzing the capability of the world's largest ambulance service to handle 5000 daily requests in carrying patients in emergency situations. In the recent aviation industry, although the real causes for several airliner crashes in the past few years remained mysteries, experts pointed out that software control could be the chief suspect in some of these incidences due to its inappropriate response to the pilots' desperate inquires during an abnormal flight conditions*" (Lee, 1992).

It is clear that the failures of the software systems carry disasters impacts even the human lives. Software reliability is the science of studying about the software failures. Software reliability estimation quantifies the software reliability using software failure data collected during the testing. This thesis

focuses on accurate software reliability estimation which can easily be used in software industry.

1.7 The Rest of the Thesis

The thesis contains six main chapters. The introduction chapter gives an approach to the work objectives, scope and the motivation. Chapter 2 presents the theories regarding the software reliability, its benefits, and key terminologies. A critical review about software reliability estimation, history, existing software reliability growth models and the problems with the existing software reliability estimation models has been done in chapter 3. In chapter 4, the research outcome, the cubic splines network model is described. The evaluation of research outcome, comparative findings of the cubic splines model, the special features of cubic splines model are described in the chapter 5. The conclusions and future work are described in the chapter 6.

2. Software Reliability

Software reliability is equated with the failures of the software system (Fries et. al., 1996; Lyu, 1996; AIAA/ANSI, 1993). Engineering discipline of satudying about the software reliability is Software Reliability Engineering (SRE). Important terminologies under the discipline are such as fault, error, failure, software reliability estimation, software reliability prediction, software reliability models, and software reliability activities. This chapter describes the term software reliability engineering, definitions of the software reliability, benefits of the software reliability application in each phase of the software development and three main activities in software reliability.

2.1 Software Reliability Engineering

Software Reliability Engineering (SRE) is established disciplines that can help organizations to improve the reliability of their products and processes. The American Institute of Aeronautics and Astronautics (AIAA) defines SRE as "the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems" (AIAA/ANSI, 1993).

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (Schneidewind, 1993) (AIAA/ANSI, 1993). SRE is therefore defined as the quantitative study of the operational behavior of software-based systems with respect to user requirements concerning reliability.

There are examples of applications of SRE in the large software companies. Ref. (Agbari, et al., 2006) states that, "*As a proven technique, SRE has been adopted either as standard or as best current practice by more than 50 organizations in their software projects and repor, including AT&T, Lucent,*

IBM, NASA, Microsoft, and many others in Europe, Asia, and North America". However, this is a very small number compared to the software development companies in the world.

Software reliability engineering is centered on a key attribute of software reliability. As discussed in the chapter 1, among the attributes of software quality such as functionality, usability, portability and maintainability etc., software reliability is generally accepted as the major factor in software quality since it quantifies software failures, which can make a powerful system inoperative. It is important to discuss the direct benefits that an organization or a customer can acquire by employing software reliability. The benefits are the reasons why an organization should promote the usage of software reliability.

2.2 Software Reliability Benefits

An organization attains several benefits through the usage of software reliability engineering. Using software reliability engineering disciplines, practitioners who develop the software system and the customer who acquire the product, can have a determination about the continuity of the software invocation and hence the software quality aspects. When the software system development is done through the agreement between vendor and customer, the reliability objective of the software should be either a pre agreed one of software quality metrics or it should be as a part of standard practice of the organization. From the customers view point, it is important to have a guarantee that the agreed reliability objective is achieved in the released software.

By employing the software reliability disciplines during the requirements formulation, the validity of the design can be improved. Some of reliability issues during the requirement formulation focus on reducing the erroneous

requirements in consideration, accounting of the risk of failure occurrences of each requirement, and the change management issues of future changes of the requirements.

Design phase of the software development process is the most important and high accuracy needed phase. Critical operations of the software must be considered and the reliability actions must be included in the design to improve the availability of such operations. Software safety criterion can be assisted and the release time of the software can be determined using the software reliability during the testing. The maintenance size and the effort could be determined through the software reliability engineering disciplines. According to the criticality of the operation, the maintenance efforts and resources can be allocated and this will improve the productivity.

The software reliability is comparable with the hardware reliability. The history of hardware reliability runs to the long past than software reliability. In fact, the software reliability has influenced by hardware reliability.

2.3 Software Reliability against Hardware Reliability

The total system consists of software and hardware components. However, software and hardware reliability is recognized differently. Changing of hardware takes time and need to undergo steps like component gathering, customization, assembly, inspection and testing etc. Software has no physical existence and has no life without hardware components. Any line of software code can be subjected to a failure.

More important factor with software reliability concerns is that software does not wear out or burn out. Furthermore any software fault can be subjected to a failure without any prior notice. However, hardware provides powerful notices of degradation.

Same software can be distributed with many copies without an additional cost where as manufacturing of the same hardware product takes the same time and requires the same cost. Similarly, repair of software requires changing software coding where as repair of hardware is generally done through the replacement of the component by a new one.

Software reliability is expressed in execution time but hardware reliability is expressed in clock time.

There are three important terminologies that we must concentrate on software reliability concerns namely fault, error and failure.

- **Fault**

The result that causes from the mistakes in the software is called as a fault (faulty instruction(s) or data).

- **Error**

When invocating a faulty instruction or a data by an appropriate input pattern, the fault produces an error.

- **Failure**

If the erroneous data affect the delivered service (in value and/or in the timing of their delivery), failure occurs.

2.4 Software Reliability Activities

Software reliability is comprised of three activities:

1. Error prevention
2. Fault detection and removal
3. Measurements to maximize reliability (Rosenberg, et al., 1998)

The error prevention techniques are more important and each of them will fit well in any development process. The error prevention techniques used in the industry are designed by contract, bug tracking systems, monitoring, coding standards, definitive programming, culture of development, and code reviews. When employing more techniques in a project, that project is closer to achieve a comprehensive and coherent error prevention program.

Software testing during the lifecycle is done for fault detection and for removal. During the coding, module testing is carried out. Making use of architectural design of the software, the integration testing is carried out. Integration of the modules is examined during the integration testing subsequent to the module testing. By encompassing the software requirements, the system testing is carried out. This will verify the faults in the complete system invocation. Finally the acceptance testing is done to ensure the requirements collected from the customer. Several strategies and techniques are used in test selection, test design and stop testing to high level fault detection and removal.

Software reliability estimation and prediction are used as measurements in maximizing the reliability. Usage of operational profile during the testing will enhance the efficiency in testing. Following section describes the term operational profile.

2.5 Operational Profile

Operational profile quantitatively characterizes how the software will be used during its operational phase. Operational profile includes developing customer type list, developing user type list, listing system models, developing functional profiles and converting functional profile to an operational profile. Customer type shows which type of customers use this software (e.g. large retail stores) and user type shows which kind of users use this software (e.g. computer program operator, clerk). System modes include which kinds of system modes are invoked by each user (e.g. retail sales, database cleanup) and each system mode includes several functions. A functional profile shows which kind of functions are used (number of functions, variable types, scope of the function etc.), converting functional profile to an operational profile includes calculating probabilities of operations. This can be used for testing procedure, i.e., allocation of test resources, input data for test.

Major changes can occur with respect to several of the factors when software becomes operational. In the operational environment, the failure rate is a function of the fault content of the program, of the variability of input and computer states, and of software maintenance policies. Software in the operational environment may not exhibit the reduction in failure rate with execution time that is an implicit assumption in most prediction models. Thus, the prediction of operational reliability from data obtained during test may not hold true during operation. To mitigate this problem, use an operational profile, for driving tests, that is representative of the operational environment.

The following section describes the important measures in software reliability concerns.

2.6 Software Reliability Metrics

Measurement is commonplace in other engineering fields, but not in software engineering. Quantifying the software reliability is still a problem. Measuring software reliability remains a difficult problem because of the lack of understanding of the nature of software. There is no clear definition to what aspects are related to software reliability and most of them are related. Hence a suitable way to measure software reliability cannot be found.

It is important to have an understanding about the measurements that is related to reliability to reflect the characteristics, if reliability cannot be measured directly. The current practices of software reliability measurements can be divided into four categories (Rosenberg, et al., 1998):

- **Product metrics**

Software size is thought to be reflected the complexity, development effort and reliability. Lines Of Code (LOC), or LOC in thousands (KLOC), is an intuitive initial approach to measuring software size. But there is no standard way of counting. Typically, source code is used (SLOC, KSLOC) and comments and other non-executable statements are not counted. This method cannot faithfully compare software not written in the same language. The advent of new technologies of code reuses and code generation technique also cast doubt on this simple method.

Complexity is directly related to software reliability, therefore representing complexity is important. Complexity-oriented metric is a method of determining the complexity of a program's control structure, by simplifying the code into a graphical representation. Representative metric is McCabe's Complexity Metric.

Test coverage metrics are a way of estimating fault and reliability by performing tests on software products, based on the assumption that software reliability is a function of the portion of software that has been successfully verified or tested. Detailed discussion about various software testing methods can be found in topic the Software Testing (Glenford, 1979).

- **Project management metrics**

Researchers have realized that good management can result in better products. Research has demonstrated that a relationship exists between the development process and the ability to complete projects on time and within the desired quality objectives. Costs increase when developers use inadequate processes. Higher reliability can be achieved by using better development process, risk management process, configuration management process, etc.

- **Process metrics**

Based on the assumption that the quality of the product is a direct function of the process, process metrics can be used to estimate, monitor and improve the reliability and quality of software. ISO-9000 certification, or "quality management standards", is the generic reference for a family of standards developed by the International Standards Organization (ISO).

- **Fault and failure metrics**

The goal of collecting fault and failure metrics is to be able to determine when the software is approaching failure-free execution. Minimally, both the number of faults found during testing (i.e., before delivery) and the failures (or other problems) reported by users after delivery are collected, summarized and analyzed to achieve this goal. Test strategy is highly relative to the

effectiveness of fault metrics, because if the testing scenario does not cover the full functionality of the software, the software may pass all tests and yet be prone to failure once delivered. Usually, failure metrics are based upon customer information regarding failures found after release of the software. The failure data collected is therefore used to calculate failure density, Mean Time Between Failures (MTBF) or other parameters to measure or predict software reliability.

Some important facts regarding software reliability has been described in the following section.

2.7 Related Other Topics

Software Reliability relates to many areas where software quality is concerned.

2.7.1 Traditional/Hardware Reliability

The initial software reliability study is based on traditional and hardware reliability. Many of the concepts and analytical methods that are used in traditional reliability can be used to assess and improve software reliability as well.

2.7.2 Software Fault Tolerance

Software fault tolerance is a necessary part of a system with high reliability. It is a way of handling unknown and unpredictable software (and hardware) failures (faults), by providing a set of functionally equivalent software modules developed by diverse and independent production teams. The assumption is the design diversity of software, which itself is difficult to achieve.

2.7.3 Software Testing

Software testing serves as a way to measure and improve software reliability. It plays an important role in the design, implementation, validation and release phases. It is not a mature field. Advances in this field will have great impact on software industry.

2.7.4 Social & Legal Concerns

As software permeates to every corner of our daily life, software related problems and the quality of software products can cause serious problems, such as the Therac-25 accident (Lee, 1992). The defects in software are significantly different than those in hardware and other components of the system: they are usually design defects, and a lot of them are related to problems in specification. The unfeasibility of completely testing a software module complicates the problem because bug-free software cannot be guaranteed for a moderately complex piece of software. Bug-free software product cannot be achieved even with the hardest attempts during the development. Losses caused by software defects cause more and more social and legal concerns. Guaranteeing not known bugs is certainly not a good-enough approach to the problem.

Estimation of software reliability is one of most important topic in software reliability. It has taken many researchers attention and it is still a question. Following chapter describes the attempts taken by the researchers so far to estimate the software reliability.

3. Software Reliability Estimation

Software reliability models are used to estimate the software reliability. Further they are mathematical expressions those specify the general form of software failure process. This chapter critically reviews the types of software reliability models, existing SRGMs, models classifications, their assumptions and the issues of the existing models.

According to the phase in which software reliability models are used to get the reliability, there are two types of models as described below (Wood,1996 ; Lyu, 1996 ; ANSI/AIAAA, 1993).

1. Software reliability prediction models
2. Software Reliability Estimation Models or SRGM

3.1 Software Reliability Prediction Models

These models predict the software reliability based on the reliability metrics measured or calculated during early stages of software development life cycle (prior to the integrated testing) (AIAA/ANSI, 1993).

The reliability is forecasted by comparing the software with a similar project in which the failure probability is known. The software that used to compare the reliability of the developing software is known as the proof software (Lyu, 1996).

The reliability of the proof program is known at any time during the software development life cycle. As such there is an advantage for the developing software since its reliability can be calculated at any time comparing with the proof software (Lyu, 1996; AIAA/ANSI, 1993).

The similarities between the proof software and the developing software are compared in terms of architectural similarities, operational profile similarity, services delivery similarity, and the similarity of the reliability achievement compared to the proof program (Lyu, 1996).

However the validity of the prediction depends on the similarities between the proof software and the developing software. Generally the random behavior of the software failures, sleeping faults in the software, the change of the operational profile are affected to the software reliability behaviors. Changes in the operational profile take place due to the causes like hardware ware out or malicious program affection to the system software which affect the hardware resource usage and mismatch in software design especially in co-functional level. Due to the changes in the reliability behaviors there are no two software which show purely similarities in reliability concerns (Gray, 1992; Dolores, et al., 2001).

Since the difficulty of finding exactly similar software, there is an alternative method for software reliability prediction. That is, the software reliability is measured by calculating the software reliability matrices which are possible to compute or available to calculate. The reliability metrics are such as SLOC, fault density, initial number of faults, fault exposure ratio (the probability of executing single fault during single execution), the time for prediction is to be valid, failure probability per fault and unit and initial failure rate (Gray,1992; Lyu,1996; AIAA/ANSI,1993). References (Agbari,2006; Fries et. al.,1996; Wood,1996; William et. al.,1983; Lyu,1996; AIAA/ANSI,1993) state that none of these models (Martine,1976) show the accuracy or wide applicability to date. Hence inventing a new software reliability prediction model is not useful and this study focuses on software reliability estimation which is rather accurate and widely applicable.

3.2 Software Reliability Estimation Models or SRGMs

SRGM estimates the reliability based on observed failure data which are collected during the integrated testing and onwards. A numerous SRGMs can be found in literature and these models take observed failure time as input (Wood, 1996; Dolores, 2001; Lyu, 1996; AIAA/ANSI, 1993). The observed failure time data is twofold as interval data and failure time data. The interval data defines as the number of failures observed over a desired and constant period. The failure time data defines the time taken to occur a failure (next failure). According to the definition of the software reliability discussed in the Chapter 2, it is a time of failure free software operation. Hence the estimation of time to next failure is more useful than estimating number of failures for the period. As such this thesis focuses to utilize failure time data.

The reliability is quantified with respect to the time, which can be categorized into three as execution time, calendar time and clock time. It is possible to define reliability with respect to other basis such as Program Runs. The actual CPU usage time by the software, from the start of the program to the end is the execution time. The time people normally experience which includes the time during which a computer may not be running is considered as calendar time. However, the elapsed time from the start to end of the software running is called as clock time which includes the time which CPU doesn't use for program execution (Wood, 1996; AIAA/ANSI, 1993). However, finding the execution time is very difficult. The actual CPU usage time is not easy to measure as operating software and other auxiliary software are also executed in parallel to particular software execution. On the other hand, the calendar time does not give a reasonable sense since the all durations which the computer may not run during the testing period are not equal. Hence the measurements are not accurate. Most of the SRGMs consider the clock time and it is rather practical (Wood, 1996; Lyu, 1996; AIAA/ANSI, 1993).

3.3 History of SRGMs

Software reliability modeling was geared by Jalinski, Moranda, Shooman and Cousinhood doing pioneer work in early 70s. Their goals were to predict future failure behavior and approaches used time between failures or observed number of failures per given time period as data (Lyu, 1996; AIAA/ANSI, 1993).

3.4 The SRGM

The mathematical and statistical functions used in software reliability growth modeling, employ several computational steps. The equations for the models themselves have parameters that are estimated using techniques like least squares fit or maximum likelihood estimation. Then the models, usually equations in some exponential form, must be executed. Verifying that the selected model is valid for the particular data set may require iteration and study of the model functions. From these results predictions about the number of remaining faults or the time to next failure can be made (Wood,1996; AIAA/ANSI,1993). There are huge number of SRGMs can be found in the literature (Wood, 1996; Dolores, 2001; Hudepohl et. al., 1996; Lyu, 1996; Schneidewind, 1993; AIAA/ANSI, 1993). In order to understand the existing SRGMs, classification of them is needed. Following section describes the existing SRGMs based on the specific classification.

3.5 SRGM Classifications

SRGM have been classified in various ways in different literature (Wood, 1996; Lyu, 1996; AIAA/ANSI, 1993). Model classification has been done based on the model form. In all most all the literature discuses the model classification assuming the models are statistical models (Wood, 1996; Lyu,

1996; AIAA/ANSI, 1993). In this thesis, classification scheme of recommended practice of software reliability engineering are discussed since it is more popular and is recognized as the standard practice (AIAA/ANSI, 1993).

There are three classes of SRGMs according to the above classification. They are Exponential NHPP models, None Exponential NHPP models, and Bayesian models.

Generally, Poisson Process probability distribution function takes the form $f(t) = \lambda e^{-\lambda t}$. The mean time function is $\mu(t) = \lambda$. Homogeneous Poisson Process models assume a constant mean time function while Non-Homogeneous Poisson Process models assume a mean time function to be non-liner (i.e.,$\mu(t) = \lambda(t)$). Following sections describe the examples of software reliability estimation models, model forms and mean time functions in each class of models.

3.5.1 Exponential NHPP Models

Representative models in this class are Shooman's model, Musa's Basic Model, Jelinski and Moranda's model, Generalized Exponential model and Scheneidewind's models. Table 3.1 gives the reliability functions or mean time to failure functions for each of these Exponential NHPP models.

Following are the definitions of used parameters in the models

$E0$ - is the initial number of faults in the program that will lead to failures. It is also viewed as the number of failures that would be experienced if testing continued indefinitely

Ec - is the number of faults in the program, which have been found and corrected, once x units of time have been expended

K - is a constant of proportionality: failures per time unit per remaining fault

α Failure rate at the beginning of interval S

β Negative of derivative of failure rate divided by failure rate (i.e., relative failure rate)

N= $E0$

IT The number of instructions

ϕ *is a variable* $0 \leq \phi \leq 1$

Model	**f(t) or mvf(m(t))**
Shooman's model	$K'[E_O / I_T - \varepsilon_c(x)]$ $\varepsilon_c = E_c / I_T$ $K = K' / I_T$
Musa's Basic Model	$\mu(t) = \beta_0(1 - \exp(-\beta_1 t))$
Jelinski and Moranda's model	$\mu(t) = N(1 - \exp(-\phi t))$
Scheneidewind's models	$\mu(t_i) = \frac{\alpha}{\beta}[1 - \exp(-\beta i)]$

Table 3.1: Exponential NHPP Model examples

3.5.2 Non Exponential NHPP Models

Representative models in this class are Duane's model, Brook and MoHey's Binomial and Poison models, Yamada's S-Shaped model and Musa and Okumoto's Logarithmic Poisson models. Table 3.2 gives the reliability functions or mean time to failure functions for each of these Non Exponential NHPP models.

Model	**f(t) or m(t)**
Duane's model	$\mu(t) = \alpha t^{\beta}$
Brook and MoHey's Binomial and Poison models	$P(X = n_i) = \frac{(\overline{N}_i \phi_i)^{n_i} e^{-\overline{N}_i \phi_i}}{n_i!}$
Yamada's S-Shaped model	$\mu(t) = N \sum_{i=1}^{K} p_i[1 - \exp(-\beta_i t)]$
Musa and Okumoto-Logarithmic Poisson model	$\lambda_o \exp(-\phi\mu)$

Table 3.2: Non-Exponential NHPP Model examples

3.5.3 Bayesian Models

Representative models are those developed by Little Wood (AIAA/ANSI, 1993). Table 3.3 gives the reliability functions or mean time to failure functions for each of these Bayesian models.

Model	f(t) or m(t)
Little Wood	$\lambda_{linear}(t) = \frac{\alpha - 1}{\sqrt{\beta_0^2 + 2\beta_1 t(\alpha - 1)}}$ and $\lambda_{quadratic}(t) = \frac{\upsilon_1}{\sqrt{t^2 + \upsilon_2}}((t + (t^2 + \upsilon_2)^{1/2})^{1/3} - (t - (t^2 + \upsilon_2)^{1/2})^{1/3})$

Table 3.3: Bayesian Model examples

The models discussed so far are among the famous models. The real world applications of the recommended models can be tabulated as shown in Table 3.4 (AIAA/ANSI, 1993).

Model	Real world applications
Schneidewind	1. In Fortran by Naval Surface warfare center, Dahlgren Virgin as a part of SMERFS 2. IBM, Houston, Texas, Reliability prediction and assessment of On-board NASA space shuttle software 3. Naval surface warfare center, Dahlgren, Virginia, Research in reliability prediction and analysis of the TRIDENT I and II fire control software 4. NASA JPT, Pasadena, California, Experiments with multi model software reliability approach 5. Hughes Aircraft company, Fullerton, California, Integrated, multimode, approach to reliability prediction
Generalized exponential model	No implementation as a standalone application. But in tools such as SMERFS, SRMP(London) RELTOOLS at AT &T labs
Musa Okumoto Logarithmic Poisson Execution time model	1. In SMERFS 2. In AT&T labs set of programs
Littlewood/ Verrall model	1. In SMERFS 2. In SRMP

Table 3.4: Real world applications of the recommended SRGMs

However, the accuracy of the SRGMs is still a question (Wood, 1996; Dolores et. al., 2001; Lyu, 1996; AIAA/ANSI, 1993). The accuracy of the models in the same class is generally the same, as the general reliability function of them is same. Hence, it is enough to argue about the accuracy if at least one model in each class is considered. Following figure depicts the estimated reliability and the actual reliability. The calculations have been done using a tool which is discussed in Chapter 5.

According to the Figure 3.1, it is clear that the estimations of the SRGMs are average values except the None Homogeneous Poisson Process model. Hence

the accuracies of these models are not acceptable. Although NHPP model shows accuracy, there are limitations associating with the model.

FNO	Actual	Geometric	Jalinski Morenda	Little Liner	Musa Basic	Musa Log	NHPP	Littlewood
9	15830	2797	16373	19638	16682	11685	15626	2828
10	21932	5533	10503	14682	10210	7249	9770	5513
11	2485	33396	39643	46641	38598	36191	35525	33364
12	11000	21471	26421	34549	24273	24268	22461	21510
13	2880	37327	41576	51538	38213	40530	35053	37512
14	61182	5445	3241	24	6840	374185	2537	5135
15	4800	39038	41050	53580	36435	42688	32686	39677
16	38005	3324	3701	11637	1284	5237	2054	3755
17	16200	26524	26132	39907	19948	30392	18058	27690
18	6000	45922	44195	59810	35959	50390	31674	47538
19	1000	58724	55616	72149	45391	63553	39901	60726
20	10000	45880	41472	58265	29508	50932	26251	48197
21	220	67720	61742	79357	47249	73248	41399	70450
22	35580	18590	13147	26788	3077	23177	4590	20941
23	81000	262	2674	233	21350	41	4511	16
25	47857	16862	9212	20540	10302	22025	5386	19520
26	154800	44655	76698	39413	75482	31778	23237	37781
27	170460	56758	96444	56094	83614	41244	24991	49011
28	108540	391	5787	801	1878	57	1060	63
29	73800	10994	3334	8008	7646	16410	2210	12908
30	1860	111072	93218	101584	102676	121194	55820	114128

Table 3.5: The actual and the estimated

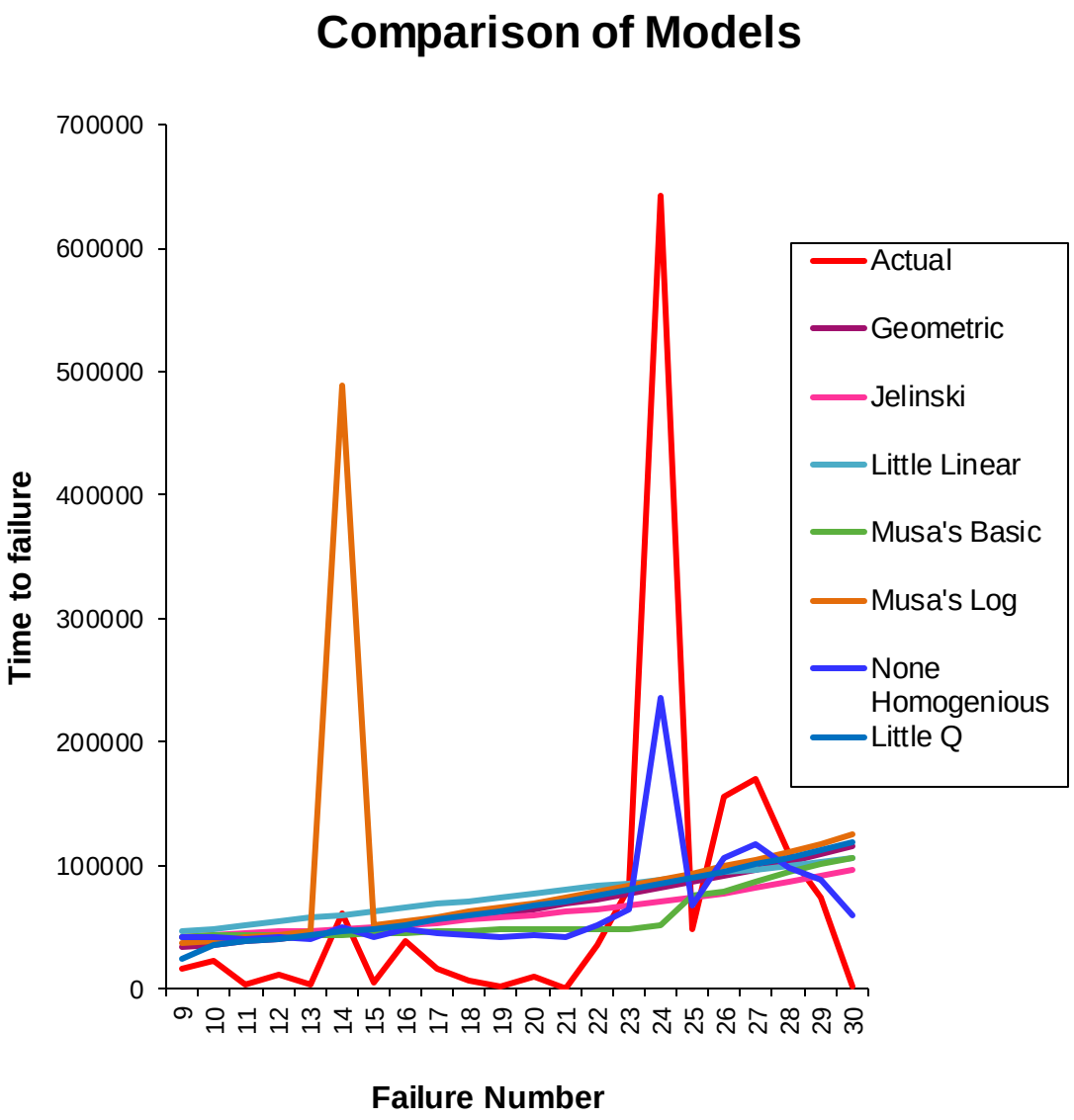


Figure 3.1: The actual and the estimated reliabilities of the famous models

Specially, NHPP model requires more data (25 failures) in order to make a valid estimation. The Figure 3.1 has been drawn based on the results at the 41st failure. The results at a different failures are different than these. Hence the accuracy varies with the size of the dataset. Further, the low the dataset size, the low the accuracy is.

The Figure 3.2 shows the software reliability estimation of NPHH model for CS I, at 25th, 30th, 35th, and 41st failures. According to the Figure 3.2, it is clear that the accuracy of the estimation varies with the size of the dataset. The accuracy of SRGM estimations is affected due to several reasons. First and foremost is the mismatching assumption. Following section describes the assumptions used in the models.

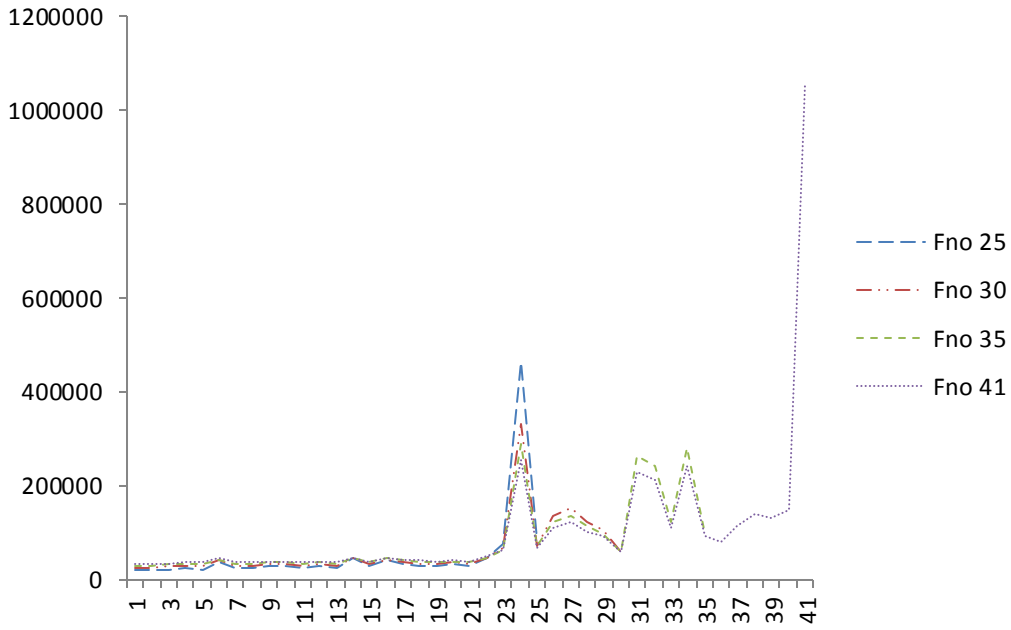


Figure 3.2: NHPP model reliability estimation at different failures

3.6 Assumptions Used in Software Reliability Growth Models

Numerous assumptions are associated with the SRGMs. The erroneous assumptions and the reality of them have been tabulated below (Wood, 1996).

No	Assumption	Reality
01	Defects are repaired immediately when they are discovered	Defects are not repaired immediately, but this can be partially accommodated by not counting duplicates. Test time may be artificially accumulated if a non-repaired defect prevents other defects from being found.
02	Defect repair is perfect	Defect repair introduces new defects. The new defects are less likely to be discovered by test since the retest for the repaired code is not usually as comprehensive as the original testing.
03	No new code is introduced during QA test	New code is frequently introduced throughout the entire test period, both defect repair and new features. This is accounted in parameter estimation since actual defect discoveries are used, but may change the shape of the curve, i.e., make it less concave. There are techniques to account for new code introduction.
04	Defects are only reported by the product testing group	Defects are reported by lots of groups because of parallel testing activity. If we add the test time for those groups, we have the problem of equivalency between an hour of QA test time and an hour of test time from a group that is testing a different product. This can be accommodated by restricting defects to those discovered by QA, but that eliminates important data. This problem means that defects do not correlate perfectly with test time.
05	Each unit of time (calendar, execution, number of test cases) is equivalent	This is certainly not true for calendar time or cases as discussed earlier. For execution time, "corner" tests sometimes are more likely to find defects, so those tests create more stress on a per hour basis. When there is a section of code that has not been as thoroughly tested as other code, e.g., a product that is under schedule pressure, tests of that code will usually find more defects. Many tests are return to ensure defect repair has been done properly, and these returns should be less likely to find new defects. However, as long as test sequences are reasonably consistent from release to release, this can be accounted for if necessary from lessons learned on previous releases.
06	Test represent operational profile	Customers run so many different configurations and applications that it is difficult to define an appropriate operational profile. In some cases, the sheer size and transaction volume of the production system makes the operational environment impractical to replicate. The tests contained in operational environment impractical to replicate. The tests contained in the QA test library test basic functionality and operation, error recovery, and specific areas with which we have had problems in the past. Additional tests are continually being added, but the code also learns the old tests, i.e., the defects that the old tests would have uncovered have been repaired.

07	Failures are independent	Our experience is that this is reasonable except when there is a section of code that has not been as thoroughly tested as other code, e.g., a product behind schedule that was not thoroughly unit tested. Tests run against this section of code may find a disproportionate share of defects.

Table 3.6: The erroneous assumptions appeared in the SRGMs

Table 3.6 summarizes how these assumptions are used in SRGMs. According to the table, all the models use the 1-6 assumptions where as Jelinski-Moranda de- eutrophication model uses the assumptions 1-7. The accuracy of those models is not doubt to be low. However, the assumptions like '6' are unavoidable because in-house software testing is essential.

Model	**Assumptions**
Jelinski-Moranda model	1,2,3,4,5,6,7
Nonhomogeneous Poisson process model	1,2,3,4,5,6
Scheidewind's model	1,2,3,4,5,6
Musa's basic execution time model	1,2,3,4,5,6
Hyper-exponential model	1,2,3,4,5,6
Weibull model	1,2,3,4,5,6
S-shaped reliability growth model	1,2,3,4,5,6
Duane's model	1,2,3,4,5,6
Geometric model	1,2,3,4,5,6
Musa-Okumoto logarithmic poisson model	1,2,3,4,5,6
Littlewood-Verrall reliability growth model	1,2,3,4,5,6

Table 3.7: Assumptions of SRGMs

3.7 Existing Neural Network Based SRGMs

Usage of artificial neural networks for software reliability estimation is an emerging technique and it is not yet widely used in software reliability estimating. However, neural network based SRGMs show more accuracy in predicting. Some of the attempts of applying ANN in software reliability growth modeling are described below.

The first attempt of applying some kind of ANN architecture to estimate the software reliability was done by (Karunaniti et. al., 1991 ; Karunanithi et.al.,

1992a ; Karunanithi et. al., 1992b ; Karunanithi et. al., 1992 ; Karunanithi et. al., 1993 ; Karunanithi et. al., 1996). Subsequently, many research papers in the literature, discuss the usage of ANN for various aspects in software reliability modeling such as software reliability prediction, assessment, predicting fault prone modules, combinational software reliability growth modeling etc. There are attempts of usage of ANN for reliability growth modeling.

There are two classes of ANNs used for software reliability modeling (Yu-Shen Su, 2007). First class uses the cumulative execution time as input and produces accumulated number of failures as output while the second class uses the multiple failure time as input and produces the time to next failure as output. The ANN discusses in this thesis comes under the second class. The ANN models comes under the first class are comparable with software estimation models those use interval data will be discussed in Chapter 5. It is evident to say that there exist issues with the architecture of existing ANN models which comes under the second class (Yu-Shen Su, 2007). Those issues affect to the accuracy of the estimating software reliability and hence a more accurate ANN models is needed.

3.8 Issues Associated with the Current SRGMs

As already discussed in the previous section, the accuracy and time taken to give a valid estimation are the main issues. This section describes the factors which affect to the accuracy.

-Uncertainty of the software behavior;

- Lack of flexibility of the SRGM to adapt for changes occurs in the software;

-Complexity of the parameter estimation of the SRGM;

-The premise of most prediction models is that the failure rate is a direct function of the number of faults in the program and the failure rate will be reduced (reliability will be increased) as faults are detected and eliminated during test or operations;

-Change in failure criteria;

-Significant changes in the code under test;

-Significant changes in the computing environment;

-More failure data are needed to SRGM to estimate the reliability;

These factors are discussed in the following sections.

3.8.1 Uncertainty in the Software Behavior

Software behavior is uncertain, (Lyu,1996; AIAA/ANSI,1993). Consequently, it is not accurate to assume that the failure data would not follow a particular distribution. Assuming such a distribution, the SRGM implicitly expects a pattern for failure data and with respect to such distribution, parameters like mean value of time to failure, total number of remaining failures are also calculated. A valid mean time to represent the failure dataset can be considered if the values of the dataset are most likely distributed around the mean time. Uncertainty of dataset visualizes the fact that there is no value in which the dataset is most likely distributed around. To achieve the uncertainty of the failure occurrence and the uncertain software behavior, *the assumptions like dataset to follow a particular distribution*, calculation of *mean* has to be eliminated in the SRGM. Similarly, the parameters associated with the SRGM have to be re-estimated for each new estimation. That is *when a new estimation has to be made, the parameters of the SRGM equation should be estimated again using the current failure data*.

3.8.2 Lack of Flexibility of the SRGM

Changes in the software are twofold as code changes and operational profile changes (Schneidewind, 1993). During the software testing phase and during maintenance phase bugs are encountered and fixed. When fixing bugs, the software codes are normally changed. As a result of large scale code changes, the behavior of the software may also change. Hence, the previously calculated reliability is no longer relevant to the current software. Therefore, SRGM must be capable of accommodating the software code changes. This can be done considering only recent failure data (i.e., without considering all past data as all the past data do not represent the current software behavior) when calculating the reliability (Schneidewind, 1993).

When changing the software operational environment, such as software is installed in the real hardware (i.e., real operational environment) or existing hardware changes, generally the software behavior is changed. To estimate the changed reliability accurately by SRGM, it should be capable of estimating reliability with *a minimum number of failure data*. This feature is used only in Scheniedewinds models (Schneidewind, 1993).

3.8.3 Complexity of Parameter Estimation of the SRGM

In most of SRGM's, parameter estimation has complex calculations. Most literatures address numerical methods such as least square method, and maximum likelihood method for parameter estimation. When the calculations are complex, those models are difficult to use and hence they are most likely to be rejected in the commercial environments. This is one of the reasons why the software reliability is not practiced in the commercial environments. Simple calculations for parameter estimation are required when designing an SRGM. Similarly when the complex mathematics is used, it is difficult to understand for

people who have low mathematical background. The reality is, it is difficult to assume highly mathematical people in the commercial software development.

3.9 Features of a Useful Software Reliability Growth Model

- According to the listed features, it is clear that any useful model should not follow any statistical distribution. Hence the usage of parametric statistical methods will not contribute to enhance the accuracy of reliability estimation.
- Similarly, the reliability is a random process. It is important to give a considerable contribution to achieve the randomness in the estimation process.
- Recent past failure data are only implied the exact figure of the software reliability.

Basically, the research focuses on employing the randomness, when estimating the reliability. As such the aim was to design an SRGM without employing the statistical distributions. Considering these features, the cubic spline network model was designed.

The following chapter describes new Cubic Spline Network SRGM.

4. Cubic Spline Network Software Reliability Growth Model

New model discusses in this thesis is Cubic Spline Network Software Reliability Growth model (CSN Model). CSN model is based on artificial neural network. This chapter describes the CSN model architecture, calculations associated with the model, numerical example to explain the calculations, the role of the boundary condition which is a feature of CSN model and the tool developed to automate the calculations associate with the model. Theory of Artificial neural network is described in the following section.

4.1 Artificial Neural Networks

Neural network models in artificial intelligence are usually referred to as Artificial Neural Networks (ANNs); these are essentially simple mathematical models defining a function $f: X \rightarrow Y$.

Neural networks are made up of many artificial neurons. An artificial neuron is simply an electronically modeled biological neuron. How many neurons are used depends on the task at hand. It could be as few as three or as many as several thousand. There are many different ways of connecting artificial neurons together to create a neural network and most common is called a feed forward network.

ANNs are used especially for computational pattern recognition since it has self learning ability. This ability has been used to overcome the random behavior of the failures. Furthermore, during the training of ANN, the situational feature of the failure behaviors can be employed. Number of inputs can be limited without affecting the accuracy of the estimation. Especially the tunable feature which

will be discussed in this chapter is due to the ANN. In this thesis, cubic splines are used to generate the activation functions.

4.2 Spline Interpolation

In the mathematical field of numerical analysis, spline interpolation is a form of interpolation where an interpolate is a special type of piecewise polynomial called a spline. Spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small. In cubic spline interpolation, the polynomial type concern is cubic polynomial. It takes the form of a third order polynomial. In mathematics, any natural phenomena can be describes in third order polynomials. As such the cubic spline interpolation is used as the activation function. Why the spline interpolation is suitable to use in activation function is described below.

By using the cubic spline interpolation as the activation function of this ANN, arbitrary pattern of the input dataset can be employed as cubic spline interpolation does not assume any pre defined distribution for the dataset. Similarly, this is not a statistical distribution and as a result, the issues with the statistical SRGMs are not affected to this. The accuracy does not vary with the dataset size in spline interpolation and hence the input dataset size can be reduced.

4.3 Software Reliability Growth Model with Cubic Splines

This is an ANN based model to capture the input–output (I/O) relationships of software systems to corresponding failures and to improve the accuracy of reliability estimation.

With the input vector of $[x_{n-5}, x_{n-4}, ..., x_{n-1}]$ (where $n \geq 6$ in considering the boundary condition), the corresponding mapping of cubic spline network can be written as $\hat{x}_n = g(x_{n-5}, x_{n-4}, ...x_{n-1})$. This model for software reliability estimation is designed as a three-layer structure with an input layer, cubic spline layer, and output layer. It is the minimum number of layers according to the design. Each layer has fixed number of nodes. Input layer has five nodes corresponding to each input (five is the minimum expected inputs to make estimation). Cubic spline layer has three nodes each corresponding to the boundary conditions (only three boundary conditions are considered as per the derivates considered. It is described in section 4.5.1). The input data vector is connected to the input nodes of the networks:

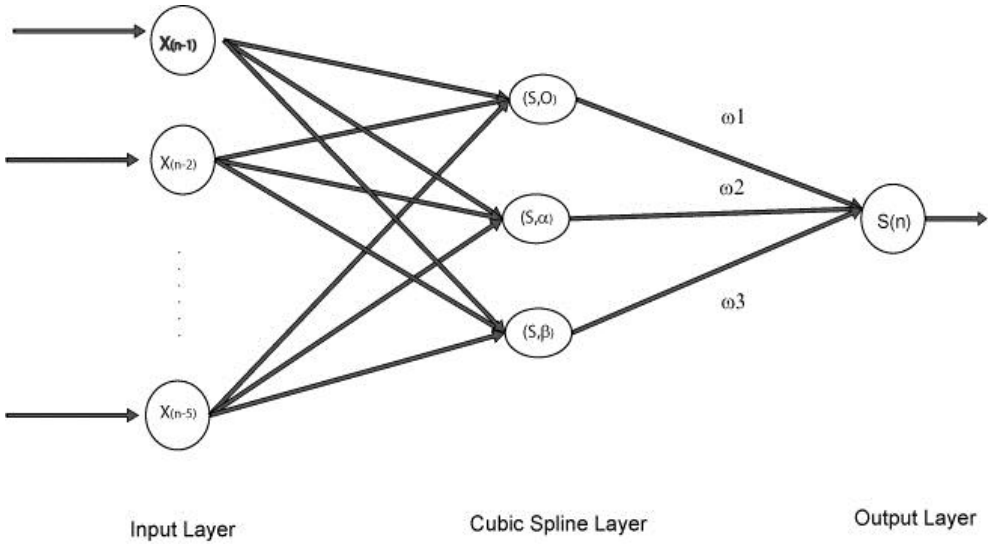


Figure 4.1: CSN for software reliability estimation

$X = [x_{n-5}, x_{n-4}, \ldots, x_{n-1}]$, for predicting the time to n^{th} failure.

We can derive the activation functions using cubic splines.

Given a function f which passing through the $x_{n-5}, x_{n-4}, \ldots, x_{n-1}$ nodes, can be represented in splines S_i defined on $[x_{i-1}, x_i]$, where n-5≤i≤n-1.

$$f(x) = S = \sum_{i=n-5}^{n-1} S_i(x)$$

A cubic spline interpolate S, for f is a function that satisfies the following six conditions,

i. The spline forms a continuous function
   i.e. $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ for each $i = n-5, n-4, \ldots, n-2$

ii. The spline forms a smooth function
   i.e. $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ for each i.

iii. The second derivative is continuous
   i.e. $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ for each i.

iv. S is a cubic polynomial, denoted S_i on subinterval X_i, X_{i+1} for each i.

v. The spline passes through each node $S(x_i) = f(x_i)$ for each i.

vi. One of following the boundary conditions is satisfied
   $S''(x_{n-5}) = S''(x_{n-1}) = 0$

a. Here $O = (x_{(n-1)-}x_{(n-6)})/6$ Set boundary derivatives for specified values.

b. Here $\alpha = (x_{(n-1)-}x_{(n-4)})/3$ Set boundary derivatives for specified values

c. Here $\beta = (x_{(n-1)-}x_{(n-2)})$

The existing dataset is considered as intervals (i.e., i^{th} interval is $x_{i-1} \leq X \leq x_i$). However in this reliability estimation model, the cubic splines are used for future estimation and hence the boundary conditions are important. This network captures three network nodes in the cubic spline layer; each is corresponding to the each boundary condition above. At the output layer, the resultant value from the cubic spline layer is multiplied by respective omega values and then they are summed in order to get the final estimation value. By varying the omega values the results can be enhanced.

The formulas for calculations can be derived as follows.

$$S_i(X) = a_i + b_i(X - x_i) + c_i(X - x_i)^2 + d_i(X - x_i)^3$$

For each i and a_i, b_i, c_i and d_i are real constants.

Let $h_i = x_{i+1} - x_i$ for all $i \geq 1$

The equations are simplified in finding the coefficients as follows.

$$S''(x_i) = 2 * c_i \qquad \rightarrow (1)$$

$$f(x_i) = a_i \qquad \rightarrow (2)$$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} =$$
$$(3/h_i)(a_{i+1} - a_i) - (3/h_{i-1})(a_i - a_{i-1}) \qquad \rightarrow (3)$$

$$c_{i+1} = c_i + 3 * d_i * h_i \qquad \rightarrow (4)$$

$$b_i =$$
$$(1/h_i)(a_{i+1} - a_i) - (h_i/3)(2c_i + c_{i+1}) \rightarrow (5)$$

The activation function of the k[th] cubic spline node for estimating the n[th] failure is

$$S_{n-2,k}(x_n) = a_{n-2,k} + b_{n-2,k}(x_n - x_{n-2}) + c_{n-2,k}(x_n - x_{n-2})^2 + d_{n-2,k}(x_n - x_{n-2})^3$$

for $k = 1,2,3$.

When $k = 1$, the boundary condition $S''(x_{n-5}) = S''(x_{n-1}) = 0$ is applied.

When $k = 2$, the boundary condition $S''(x_{n-5}) = S''(x_{n-1}) = \alpha$ is applied.

When $k = 3$, the boundary condition $S''(x_{n-5}) = S''(x_{n-1}) = \beta$ is applied.

The weight ω_k that connects the k[th] weighted node and the output node are indicated by the weighting vector $\omega = [\omega_1, \omega_2, \omega_3]$.

The final output of the cubic spline networks summing layer is:

$$y = \sum_{k=1}^{3} S_{n-2,k}(X_{n+1}) * \omega_k$$

The reliability estimation of the n[th] failure is y.

4.3.1 The Role of Boundary Conditions

In this SRGM, the boundary conditions are used as the c coefficients. They are the coefficients of second order polynomial segment of S. The most important knowledge we find about the boundary conditions is that they are equivalent to the second derivatives of the curve (S).

In general, the first derivative of the curve is its tangent. The second derivative is the rate of change of the tangent (i.e., tangent of the tangent). Within given two consecutive points, there can only be one exact line and there can be infinite number of other type curves.

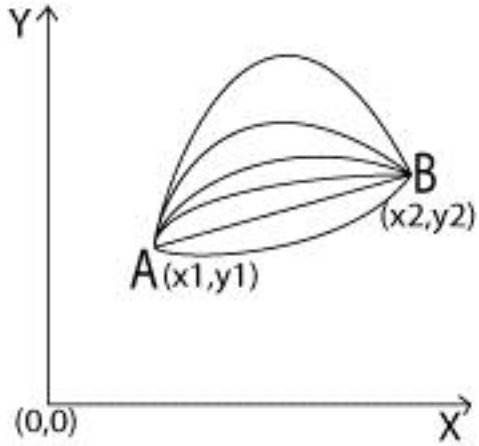


Figure 4.2: The number of graphs which can be drawn between 2 points

A and B are any given points on xy-plane. The curves are differed according to the second derivatives at A and B.

The tangent of the line is proportional to the deference of y coordinates. In this SRGM the tangent of the curve is assumed to be proportional to the relative difference of the consecutive time to failure values. Hence the second derivative is assumed to be proportional to the change of relative difference of the consecutive tangent values.

Following boundary conditions are used in this SRGM.

a. $O = (x_{(n-1)} - x_{(n-6)})/6$

b. $\alpha = (x_{(n-1)} - x_{(n-4)})/3$

c. $\beta = (x_{(n-1)} - x_{(n-2)})$

The boundary conditions are derived from the assumptions of the tangent proportional to the relative difference of the consecutive time to failure values and the second derivative is proportional to the change of relative difference of the consecutive tangent values.

The derivation of boundary condition O:

$$O = ((x_{(n-1)} - x_{(n-2)}) + (x_{(n-2)} - x_{(n-3)}) + (x_{(n-3)} - x_{(n-4)}) + (x_{(n-4)} - x_{(n-5)}) + (x_{(n-5)} - x_{(n-6)}))/5$$

$$= (x_{(n-1)} - x_{(n-6)})/5$$

Similarly α is calculated considering three recent derivatives and β is calculated considering one recent derivative.

4.3.2 Calculations of the SRGM

A numerical example is considered here to understand the calculations used in this model. Five recent failure data are taken as input and for calculating the boundary conditions six recent data are taken. Following dataset is used to show the calculations.

FNO(ti ,i=1.2. ..,6)	Time To Failure (f(ti), i=1,2,..,6)
1	20336
2	11776
3	40933
4	34794
5	17136
6	148446

Table 4.1 CS I – Dataset (1 to 6 of 41data)

Using the equations (1) to (5) and the three boundary conditions following parameter values can be calculated.

$h_i = 1$For all $i = 1,2,...,5$

Suppose 3 metrics A, B and C as follows, $A = \begin{bmatrix} 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \end{bmatrix}$ $B = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$

$$C = \begin{bmatrix} a_2 - 2a_1 + a_0 \\ a_3 - 2a_2 + a_1 \\ a_4 - 2a_3 - a_2 \end{bmatrix}$$

A*B=C

When c_1, c_5 are defined form the boundary conditions, we can solve the above for c_2, c_3, c_4

Let's take k_1, k_2, k_3 as constants:

$$k_1 = 2 * \left(f(t_4) - 2 * f(t_3) + f(t_2)\right)$$

=-70592

$$k_2 = 2 * \left(f(t_5) - 2 * f(t_4) + f(t_3)\right)$$

=-23038

$$k_3 = 2 * \left(f(t_6) - 2 * f(t_5) + f(t_4)\right)$$

= 297936

Now we can calculate C values as follows :

$$c_1 = (f(t_6) - f(t_1))/5$$

$$c_5 = c_1$$

$$c_3 = -1 * (k_1 - 4 * k_2 + k_3 - c_1 - c_5)/14$$

$$c_2 = (k_1 - c_3 - c_1)/4$$

$$c_4 = (k_3 - c_3 - c_5)/4$$

The Table 4.2 shows the calculated values for C.

c1	25622
c2	-19263.29
c3	-19160.86
c4	72868.71
c5	25622

From the equation (5) we get the following relationship:

$$b_i = (1/h_i)(a_{i+1} - a_i) - (h_i/3)(2c_i + c_{i+1}) \rightarrow (5)$$

The following are the calculated vales for bi

b1	18496.76
b2	13090.14
b3	-29173.67
b4	74190.19
b5	-165527.33

Table 4.3: The calculated b coefficient values for the data in Table 4.1

According to the equation (4)

$$c_{i+1} = c_i + 3 * d_i * h_i .$$

The d_i values are as follows

d1	-14961.76
d2	34.14
d3	30676.52
d4	-15748.9
d5	-8540.67

Table 4.4: The calculated d coefficient values for the data in Table 4.1

According to the equation

$$S_i(X) = a_i + b_i(X - x_i) + c_i(X - x_i)^2 + d_i(X - x_i)^3$$

The following are the calculated values for the boundary condition. By varying the boundary conditions the c,b,d metrics can be calculated. The estimated reliability for each boundary condition is as follows.

Estimate1	331000
Estimate2	351431.33
Estimate3	542376

Table 4.5: The estimated output from the cubic spline layer

Final estimation can be done using the following equation.

$$y = \sum_{k=1}^{3} S_{n-2,k}(X_{n+1}) * \omega_k$$

In fact, the omega ω_k values play a significant role in this SRGM. Omega values can be used to tune the estimation. If the omega value is high then this SRGM can be used to estimate the higher reliability more accurately. Similarly when the omega value is smaller, smaller reliability will be estimated more accurately. This flexibility is gained by using Artificial Neural Networks. NHPP model doesn't have this flexibility.

Here the omega values are considered as all equal values and it is equals to 0.4/6 .

The final output is

Final Estimation	81653.82

Table 4.6: The final estimated output from the network

4.4 Tool to Estimate Software Reliability

A software tool has been developed to estimate the reliability using the cubic spline network model. It is a PHP application which uses MySql as the data

storage. The reason to choose PHP is it runs in Linux platform which is less vulnerable to the malicious attacks. This tool was developed as a web based tool since it is easy to hire the tool without needing to install in a local computer. Mysql is mostly compatible to the PHP and again it is open source. There is an ANN simulation software. However, none of such tools was used for this development since this tool is developed to have more flexibility in commercializing.

The tool has the capability of import failure data from a text file saved in the home folder or one at a time. The time measurement unit has to be the same measurement units and the practicality of each failure is also in the same level. The calculations are automated. It is flexible with the ability of changing the logic without needing many changes to the database layer or to the front layer. The calculations can be done for simultaneous failure values in the same dataset or at a given failure value of the dataset. The simultaneous calculations can be exported Excel file format and the new Excel file is created in the home folder.

Accuracy and Flexibility are the important features of the tool.

4.4.1 Accuracy

The foremost important feature identified when developing this tool was the accuracy. First, manual calculations was done in an excel file for the dataset. While coding the activation function in PHP, manual output of the Excel file was referred in each step. The white box testing was carried out by generating the additional outputs to assure the accuracy in calculations.

4.4.2 Flexibility

The tool was designed in 3-Tire architecture. The business logic was coded in an independent file whereas the output is displayed in a different file. In this way, the ease of change is achieved. Furthermore, there is an input flexibility. If a user has dataset in a text file, the tool provides a way to upload the data directly from the text file. However, some users will need to enter the data one by one through the interface. The tool has the capability to handle that too. The output can be taken into excel format so that users has the flexibility to use them for analytical purposes as required.

5. Evaluation

This chapter aims to discuss the evaluation of the CSN model estimation. In order to perform this task, datasets are required. First sections of this chapter describe the selection of datasets. Then the CSN model estimations have been done for the selected datasets. In order to compare the accuracy of CSN model, it is required to compare this model with the most accurate famous model. As such, the famous SRGMs have been evaluated to find the most accurate model and then it has been compared with CSN model for the accuracy.

5.1 The Selection of the Data

To validate the models, a quality datasets are needed. (The quality of the dataset will be discussed in section 5.2). According to the time frame and the availability of resources, it was unable to develop a software system and test for long time to collect the data. Similarly, it was unable to collect failure data from the organizations or from individuals who are engaging in software testing as they are reluctant to share them.

There are several reasons why individuals as well as organizations hesitate to share data. Individuals worry about revealing illegal (perhaps not illegal) activities that may be reverse-engineered from the collected data. As failure data can potentially reveal product dependability statistics or loopholes that can be misused. Organizations fear that competitors may assess and/or misuse data and analyse results.

Often, industrial researchers are willing to share data with academic researchers and there are long delays exists due to the legal concerns. As well as there are often so limitations that be impossible to publish any meaningful research results based on the shared data (Lyu, 1996). There are publicly available

failure reports. However, the quality (i.e., the accuracy) cannot be assured as many failure data collection exercises have been based on manual (i.e., not automated) collections using pen and paper.

5.2 Quality of the Failure Data

A high quality dataset represents all failure data (i.e., all the occurred failures have been recorded in the dataset) during the testing period as well as the dataset represents failure data only (i.e., the recorded problems are all failures). The failure data collection process has been found to have a number of problems. The importance of these problems is not to be underestimated as there are often critical negative impacts on the quality of the collected data set. The problems associated with the data collection process are as follows.
During the data collection, people have to make the decisions like how to classify a particular failure or how to deal with unusual circumstances. In order to do so, they must have to have the understanding about the requirements. However, usually people engaging with the data collections are testers who do not have full familiarity with the requirements. Furthermore, it is not practical to give full familiarity about the requirements to them.

All failure detection efforts are not the same. Some failures are easily detectable while others are difficulty. (i.e., application crashes are easy to detect whereas the un-expected output time is not easy to detect until some of the following events occurs). This is also affected to the quality of the collected data set.

5.3 Data Sets Used

The datasets considered are taken from (Lyu, 1996). The Software Reliability Datasets used in this thesis were compiled by Prof. John Musa of Bell Telephone Laboratories. The dataset consists of software failure data of 16

projects. Careful controls had been employed during data collection to ensure that the data would be of high quality. It represents projects from a variety of applications including real time command and control, word processing, commercial, and military applications. Further, these datasets have been used by many researchers in many literatures to validate the software SRGMs (Okamura, et al., 2004; Cai, et al., 2001).

There are two case studies used to test the accuracy of the models. The reasons to select the particular data sets are there. They show a high variations which drastically affects the accuracy and there are considerable low number of data (i.e., 41 and 54 data) contain in the dataset. Hence it is easy to present them in the graphs and in calculations.

5.3.1 Case Study I - Time between failure data

Failure Number	Time to Failure	Failure Number	Time to Failure
1	20336	22	35580
2	11776	23	81000
3	40933	24	643095
4	34794	25	47857
5	17136	26	154800
6	148446	27	170460
7	7995	28	108540
8	1636	29	73800
9	15830	30	1860
10	21932	31	336600
11	2485	32	268140
12	11000	33	74880
13	2880	34	286200
14	61182	35	25320
15	4800	36	7080
16	38005	37	59820
17	16200	38	87900
18	6000	39	76200
19	1000	40	89280
20	10000	41	1209600
21	220		

Table 5.1: The failure data taken from (Lyu, 1996) of CS I

5.3.2 Case Study II – Time between failure data

FailureNumber	Time to Failure	Failure Number	Time to Failure	Failure Number	Time to Failure
1	191	19	625	37	661
2	222	20	912	38	50
3	280	21	638	39	729
4	290	22	293	40	900
5	290	23	1212	41	180
6	385	24	612	42	4225
7	570	25	675	43	15600
8	610	26	1215	44	0
9	365	27	2715	45	0
10	390	28	3551	46	300
11	275	29	800	47	9021
12	360	30	3910	48	2519
13	800	31	6900	49	6890
14	1210	32	3300	50	3348
15	407	33	1510	51	2750
16	50	34	195	52	6675
17	660	35	1956	53	6945
18	1507	36	135	54	7899

Table 5.2: The failure data taken from (Lyu, 1996) of CS II

5.4 Cubic Splines Network (CSN) Software Reliability Estimation Model

For the above datasets, the reliability estimations using CSN model are discussed in this section.

5.4.1 CSN Model Estimation for Case Study I

Following Table 5.3 shows the actually and estimated reliability values for each Failure Number.

FNo	Actual Time	Estimated Time	FNo	Actual Time	Estimated Time
9	15830	24696	25	47857	394748
10	21932	28946	26	154800	79836
11	2485	2696	27	170460	67951
12	11000	9042	28	108540	7202
13	2880	4972	29	73800	10374
14	61182	569	30	1860	11319
15	4800	41287	31	336600	1027239
16	38005	8765	32	268140	20089620
17	16200	23063	33	74880	21488
18	6000	317	34	286200	15821
19	1000	1506	35	25320	122321
20	10000	3369	36	7080	33926
21	220	5692	37	59820	111515
22	35580	3701	38	87900	105684
23	81000	23608	39	76200	13531
24	643095	28156	40	89280	1744

Table 5.3: CSN model Estimated Reliability for CS I

5.4.2 Comparison of the Results with the Actual Data

Validity of the estimation was checked using estimated and actual time to failure values for the dataset. The dataset was considered from the 9^{th} failure onwards (for x =9, 10…39).

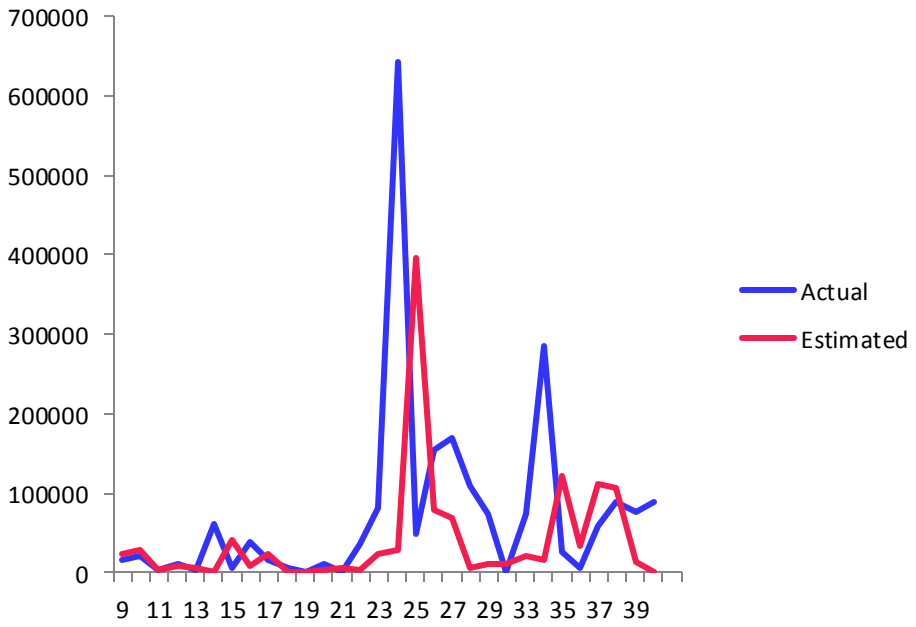


Figure 5.1: Estimated and actual reliability for the CS II

5.4.3 Comparison of the Results with the Other Models

In order to find the accuracy of CS model, it is essential to compare the output with the other models. Since the calculations from scratch of the other models are not available, it was difficult to find the output values of them. There are software reliability modeling tools available on the Internet. One of these is SMERFS; a public domain tool developed by Dr. William Farr of the Naval Surface Warfare Laboratory and employs several others of the models. This tool was selected since it is utilized for the real world applications especially like NASA's Software Assurance Technology Center at the Goddard Space

Flight Center. Following Table 5.4 shows the software reliability models used in SMERFS^3(Version 3).

Time Between Failure Models
Geometric Model
Jelinski / Moranda Model
Littlewood and Verrall Linear Model
Littlewood and Verrall Quadratic Model
Musa's Basic Model
Musa's Logarithmic Model
Non-homogeneous Poisson Model

Table 5.4: Software Reliability Models facilitated in SMERFS^3

5.4.3.1 Finding the Most Accurate Model in SMERFS^3

Figure 5.3 and 5.4 are taken through SMERFS by entering the datasets of Table 5.1 and 5.2 respectively. The doted points in each graph show the actual values whereas the other curves show the estimations of each model.

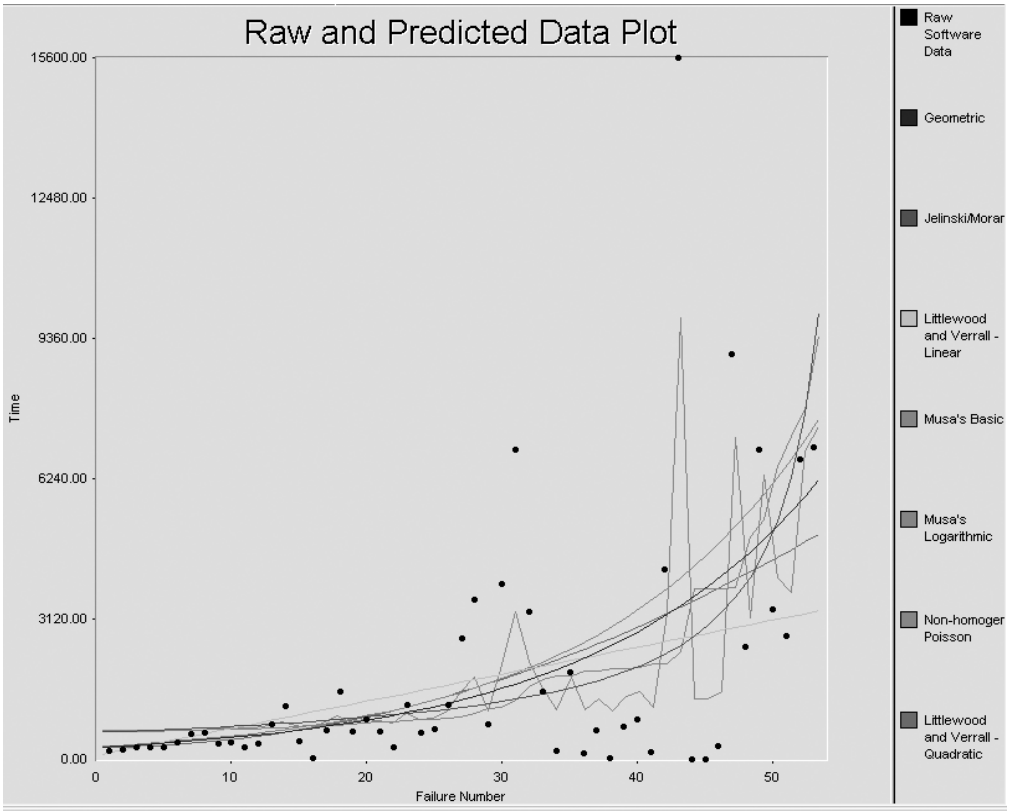


Figure 5.2: The output from SMERFS by each model for CS I

These curves show that, NHPP model curve is more sensitive towards the actual values and the curves of the other models show an average value. This can be further clarified by the following Figure 5.3.

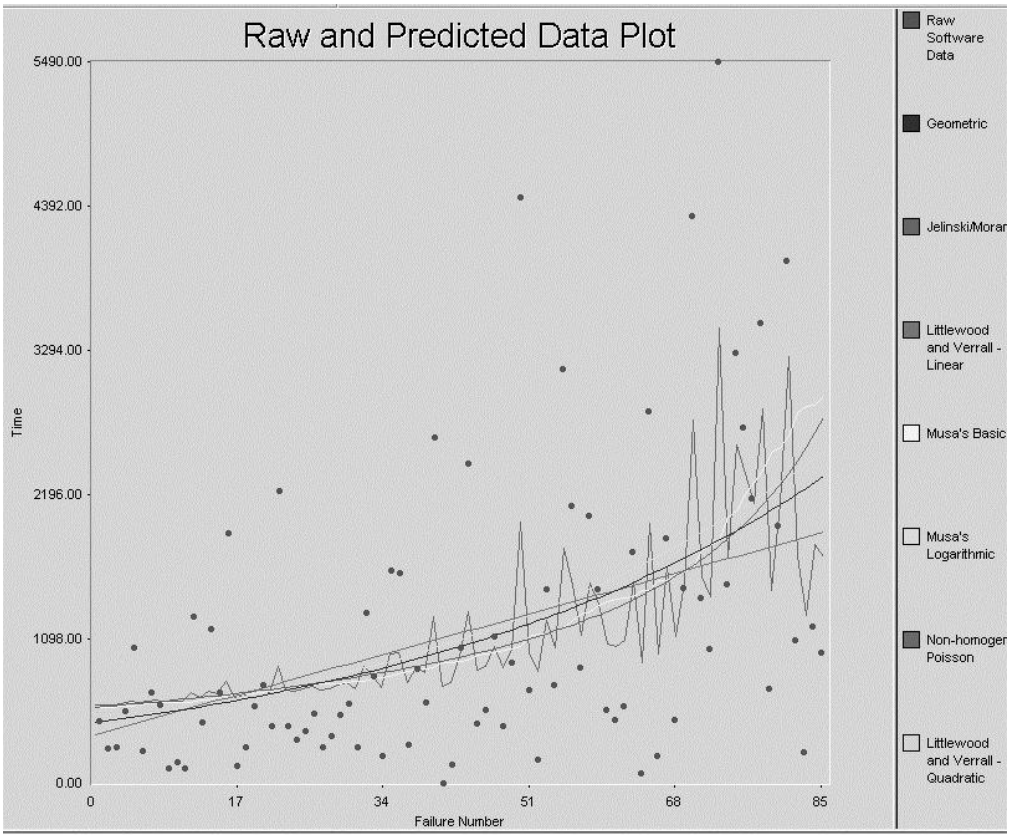


Figure 5.3: The output from SMERFS by each model for CS II

According to the interpretation which can be made by analyzing the two figures is that NHPP model is more accurate than the other models. However this can be tested further statistically to prove that NHPP model is more accurate.

5.4.3.2 Comparison of the CSN Results with NHPP Model

The statistical test used is goodness of fit test (Kleinbaum, et al., 1997). The error which can be calculated through "Actual Time – Calculated Time" is standardized and then are plotted against the failure numbers. If the curve lies between -3 to +3 and the curve doesn't show any pattern, then the dataset is statistically said to fit with the Calculated Time values. Following figures show the graphs for the standardized errors.

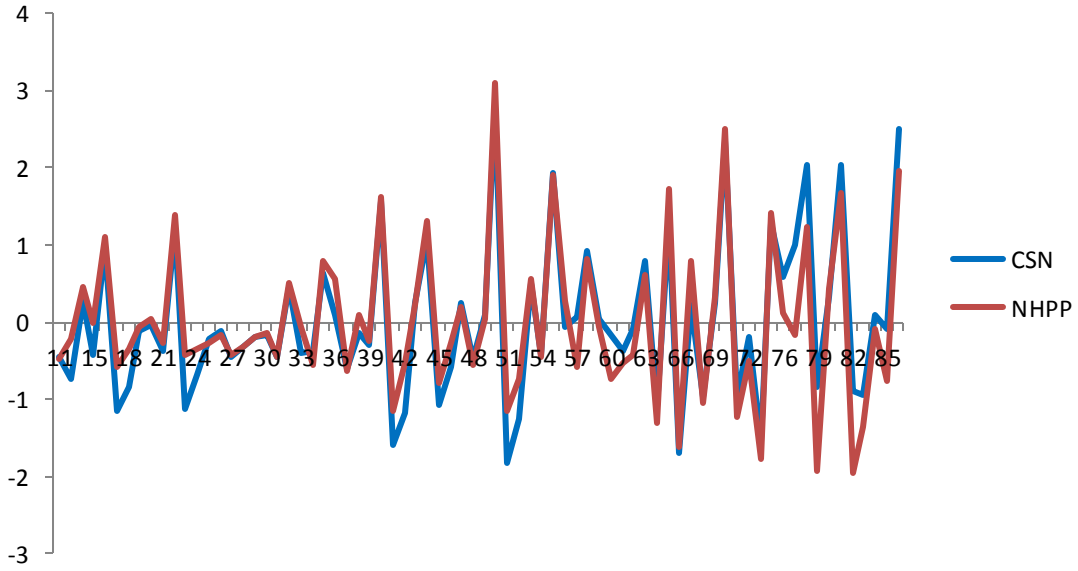


Figure 5.4: The standardized errors for dataset with 86 records in (Lyu, 1996)

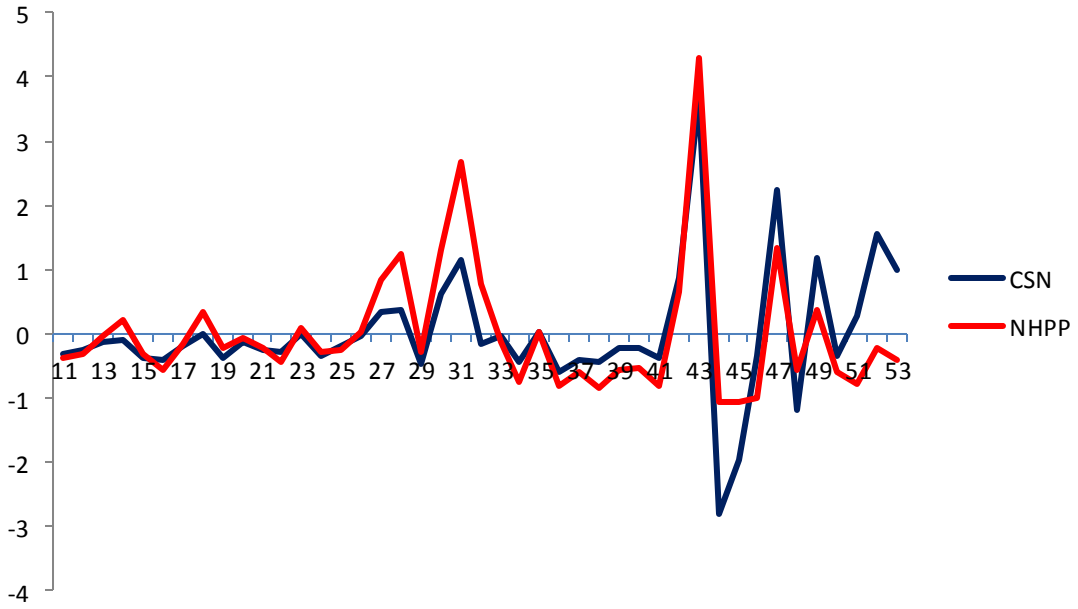


Figure 5.5: The standardized errors for CS II

According to the graphs it can be seen that CSN curve lies between -3 and +3 while NHPP curve is distributed beyond 3. Hence it can be statistically concluded that CSN model is more suitable for those datasets than NHPP

model. Further this has been tested mathematically and the calculations are available in Appendix B. The conclusions drawn through the said calculation is also CSN model is more suitable.

5.4.3.3 Comparison of the Results with Other SRGMs

The (Table 5.1) dataset with 41 failures are entered to this model and output of each model are collected and tabulated as follows.

FNO	Actual	Geometric	Jelinski/Mo	Little Line	Musa's Ba	Musa's Log	None-homog	Littlewood Q
6	148446	28602.92	38931.44	37338.67	38536.71	30594.34	47662.55	29481.49
7	7995	30307.59	39926.68	40169.2	42188.26	32440.77	40128.51	30810.64
8	1636	32113.86	40974.13	42999.73	42394.47	34398.65	39739.67	32344.26
9	15830	34027.77	42078.03	45830.26	42436.79	36474.69	41204.47	24082.37
10	21932	36055.76	43243.05	48660.79	42848.46	38676.02	42248.94	36024.96
11	2485	38204.6	44474.43	51491.32	43425.42	41010.2	40342.13	38172.04
12	11000	40481.51	45777.99	54321.86	43491.28	43485.26	41548.51	40523.6
13	2880	42894.13	47160.27	57152.39	43784.02	46109.69	40608.52	43079.64
14	61182	45450.52	48628.62	59982.92	43860.99	488892.52	49928.01	45840.17
15	4800	48159.28	50191.35	62813.45	45528.46	51843.29	41733.79	48805.18
16	38005	51029.47	51857.86	65643.98	45661.93	54972.16	47927.12	51974.67
17	16200	54070.71	53638.83	68474.51	46732.62	58289.85	44569.54	55348.65
18	6000	57293.21	55546.49	71305.04	47196.61	61807.78	42833.07	58927.11
19	1000	60707.77	57598.83	74135.57	47369.63	65538.02	41876.78	62710.05
20	10000	64325.82	59800.03	76966.1	47398.53	69493.38	43977.53	66697.48
21	220	68159.5	62180.82	79796.63	47688.47	73687.47	41838.02	70889.39
22	35580	72221.66	64759.04	82627.16	47694.87	78134.67	50857.93	75285.78
23	81000	76525.92	67560.32	85457.69	48741.1	82850.28	64008.16	79886.66
24	643095	81086.7	70614.9	88288.22	51209.29	87850.48	235008.8	84692.02
25	47857	85919.29	73958.77	91118.75	75801.4	93152.46	66829.03	89701.86
26	154800	91039.9	77635.06	93949.28	78046.34	98774.42	105327.9	94916.19
27	170460	96465.68	81695.96	96779.81	85773.37	104735.68	116501.69	100335
28	108540	102214.83	86205.13	99610.34	95170	111056.72	98331.92	105958.29
29	73800	108306.61	91241.15	102440.87	101682.84	117759.25	87723.99	111786.07
30	1860	114761.46	96902.07	105271.4	106363.88	124866.28	59481.61	117818.33
31	336600	121600.99	103311.91	108101.93	106484.6	132402.25	209240.29	124055.08
32	268140	128848.15	110629.79	110932.46	130748.83	140393.03	191928.22	130496.3
33	74880	136527.23	119063.41	113762.99	153977.29	148866.06	105274.03	137142.02
34	286200	144663.96	128888.97	116593.52	161171.86	157850.47	220042.78	143992.21
35	25320	153285.62	140482.08	119424.06	191907.26	167377.1	87182.72	151046.89
36	7080	162421.11	154366.83	122254.59	194893.61	177478.69	77613.3	158306.05
37	59820	172101.07	171297.25	125085.12	195736.94	188189.94	108840.11	165769.7
38	87900	182357.92	192398.9	127915.65	203009.59	199547.63	128071.09	173437.82
39	76200	193226.06	219429.87	130746.18	214189.18	211590.78	123960.42	181310.44

Table 5.5: Output values for CSN model and other models in SMERFS for CS I

The following Figure 5.6 shows output values of other SRGMs and CSN model outputs.

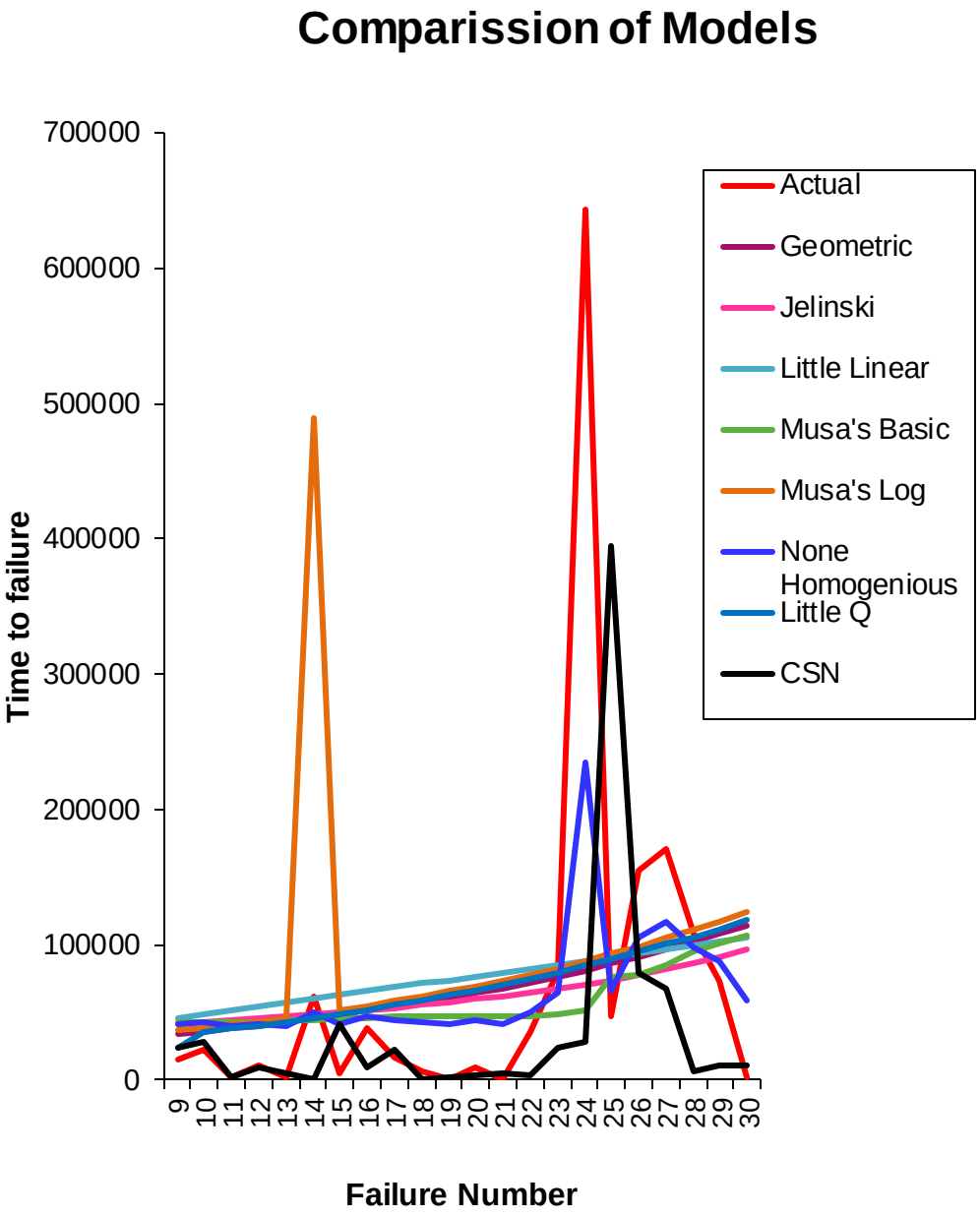


Figure 5.6: Comparison of estimation ability

5.5 Tunable Feature of CSN Model

Following Table 5.6 shows the CSN model estimation for the same dataset with varied omega value:

FNO	TTF	Omega=0.4/6	Omega=0.5/6	Omega=0.7/6
11	2485	7167.72	8959.64	12543.5
12	11000	9839.89	12299.87	17219.81
13	2880	4903.8	6129.75	8581.65
14	61182	2944.25	3680.31	5152.44
15	4800	35488.54	44360.68	62104.95
16	38005	18566.41	23208.01	32491.22
17	16200	21177.64	26472.06	37060.88
18	6000	5888.87	7361.08	10305.52
19	1000	2063.47	2579.33	3611.07
20	10000	4716.19	5895.23	8253.33
21	220	4863.11	6078.89	8510.44
22	35580	4480.49	5600.61	7840.86
23	81000	20956.36	26195.44	36673.62
24	643095	36495.56	45619.44	63867.22
25	47857	361678.76	452098.44	632937.82
26	154800	187286.1	234107.63	327750.68
27	170460	74009.8	92512.25	129517.15
28	108540	21902.8	27378.5	38329.9
29	73800	4499.42	5624.28	7873.99
30	1860	15601.2	19501.5	27302.1
31	336600	32327.92	40409.9	56573.86
32	268140	193884	242355	339297
33	74880	42050.13	52562.67	73587.73
34	286200	47096.27	58870.33	82418.47
35	25320	131104	163880	229432
36	7080	92062.4	115078	161109.2
37	59820	16464.53	20580.67	28812.93
38	87900	13927.47	17409.33	24373.07
39	76200	30068.53	37585.67	52619.93
40	89280	8812	11015	15421
41	1209600	25230.93	31538.67	44154.13

Table 5.6: The calculations of CSN model at different omega values for CS I

These data can be shown in the following Figure 5.7.

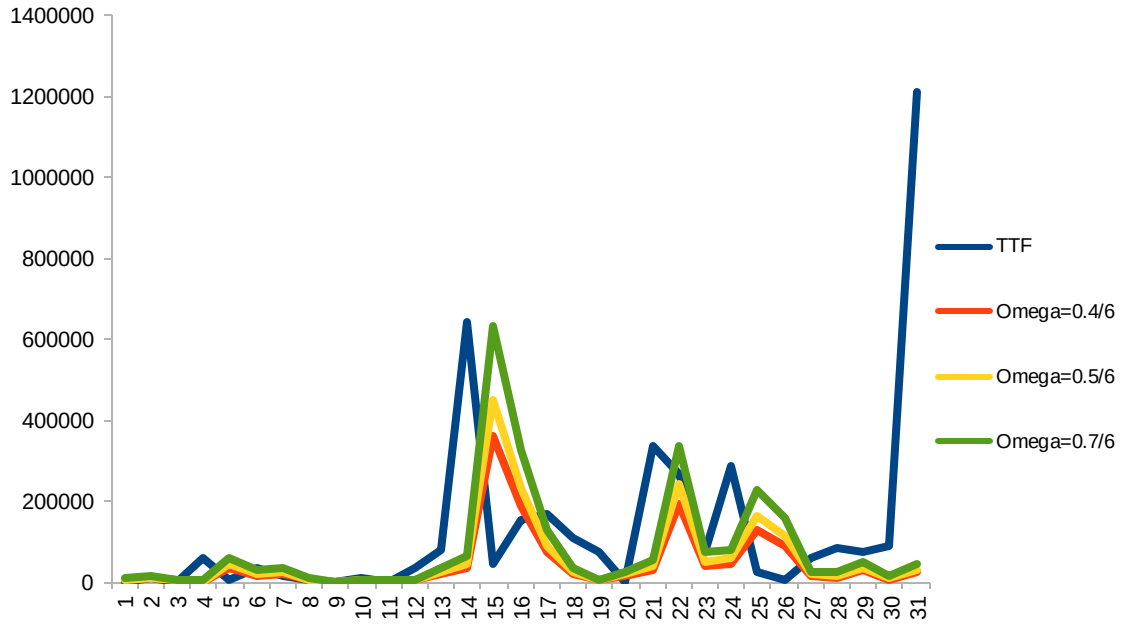


Figure 5.7: The output from CSN model at different omega values for CS I

The red colored curve shows the sensitivity towards the small values while the green colored graph shows the sensitivity towards the big values.

6. Conclusions and Future Work

In this thesis, accurate software reliability estimation has been addressed. Though software reliability is an important software quality factor, it is not widely used in the software industry. This is due to the lack of accuracy in software reliability estimation models and time taken to give a valid estimation for existing software reliability estimation models. Usage of statistical distribution for software reliability models, usage of invalid assumptions and usage of failure data which had collected early in the testing are mainly affected the accuracy of existing reliability estimation models.

CSN model does not employ the statistical distributions. Similarly, the model does not assume invalid assumptions like no new code is introduced during the testing and failures are independent. The model takes only recent failure data and thus the accuracy improvement initiatives have been employed in designing the model. Further, as CSN model takes only six recent failure data, the time taken to estimate the reliability has been reduced. Hence this model increases the opportunity to apply it in software industry.

According to the results in the thesis, NHPP model is more accurate than the other famous models. CSN model has been statistically compared with the NHPP model for accuracy and it proves that CSN model is more accurate than NHPP model.

CSN model requires only six recent consecutive failure data to make estimation whereas NHPP model requires at least twenty five recent failure data to give a single estimation. Hence CSN model is more desirable towards the achievement of business objectives than NHPP model. Further, CSN model is tunable.

However estimation accuracy of CSN model is higher when there are less sudden variations. CSN model shows the less accuracy with the presence of

sudden variations in the failure data. This can be considered as a limitation of the CSN model.

CSN model can be used to assure reliability at the release time of the software because there are no sudden changes expected in the reliability when releasing the software.

Since software tool to automate calculations of CSN model has been developed, this model can be used in the industry without any overhead of the interior functions of the model.

Future work includes the improvement of omega calculation for a given dataset.

References

Agbari Adnan Basile Claudio & Zbigniew Kalbarczyk (2006). Proceedings of the International Conference on Dependable Systems & Networks, Washington DC, IEEE Computer Societry.

AIAA/ANSI (1993). Recommended Practice for Software Reliability, The American Institute of Aeronautics and Astronautics, Washington DC, Aeruspace Center, R-013, 1992, - ISBN 1-56347-024-1.

Boehm B.M., Brown J. R. & Lipow M. **(**1976). Quantitative evaluation of software quality, In Proceedings of the International Conference on Software Engineering, San Fransisco, Vol 13-15 pp 592-605, IEEE Computer Society Press.

Cai K.Y, Cai L, Wang W.D & Yu Z.Y (2001). On the Neural Network Approach in Software Reliability Modeling, The Journal of Systems & Software - Vol 47.

Crosby P. B. (1979). Quality is Free The Art of Making Quality Certain NY, McGraw -Hill.

Deming & Edwards W (1986). Out of Crisis, MIT Press.

Dromey R. G. (1995). A Model for Software Product Quality, IEEE Transactions on Software Engineering - Vol 2.

Feigenbaum A. V. (1983). Total Quality Control, McGraw - Hill.

Fries A. & Sen A. (1996). A Survey of Descrete Reliability Growth Models, IEEE Transactions Reliability - Vol 45.

Glenford J. (1979). The Art of Software Testing, NY, John Wiley & Sons.

Gray R.B (1992). Practical Software Matrics for Project Management and Process Improvement, Prentice Hall.

Hoppe Wolfgang (1996). An Indusry Applicable Approach to Predict Software Reliability, Bremen, STN ATLAS Elektronik Gmbh.

Hudepohl J. P, Aud S. J, Khoshgoftaar T. M. & Allen E. B. (1996). EMERALD: Software Matrics and Models on the Desktop, IEEE Software - Vol 13(5) pp 56-60.

IEEE Std 610.12 (1990). IEEE Standard Glossary of Software Engineering Terminology, NY.

Ishika K (1985). What is Quality Control? The Japanese Way, Prentice Hall, NY.

ISO/IEC Std 9126-1 (2001). Software Engineering – Product Quality, Part 1: Quality Models, International Organisation for Standards.

Juran J.M (1988). Juran's Quality Control Handbook, McGraw - Hill.

Karunaniti N, Malaiya Y. K. & Whitly D. (1991). Prediction of software reliability using neural networks, Proceedings of the 2nd IEEE International Symposium on Software Reliability Engineering, Los Alamitos CA - pp 124–130.

Karunanithi N, Malaiya Y. K. (1992). The scaling problem in neural networks for software reliability prediction, Proceedings of the Third International IEEE Symposium of Software Reliability Engineering, Los Alamitos CA, pp 76–82.

Karunanithi N, Whitley D, Malaiya Y. K. (1992a). Using neural networks in reliability prediction, IEEE Software - Vol 9 pp 53–59.

Karunanithi N, Whitley D, Malaiya Y. K. (1992b). Prediction of software reliability using connectionist models, IEEE Transactions on Software Engineering - Vol 18 pp 563–574.

Karunanithi N. (1993). A Neural Network Approach for Software Reliability Growth Modeling In the Presence of Code Churn, ISSRE.

Karunanithi N, Malaiya Y. K. (1996). Neural networks for software reliability engineering - NY.

Kleinbaum David G. (1997). Applied Regression Analysis and Multivariable Methods, Thompson Higher Education, USA, ISBN- 0-495-38496-8.

Lee L (1992). The Day the Phones Stopped How People Get Hurt When Computers Go Wrong, Donald I Fine Inc.

Lyu M.R (1996). Handbook of Software Reliability Engineering, IEEE Computer Society Press.

Martin Jedlicka, Moravcik Oliver & Schreiber Peter (2008). Survey to Software Reliability, 19th Central European Conference on Information and Intelligent Systems, Varazdin.

Martine L. & Shooman (1976). Structural Models for Software Reliability Prediction, 2nd International Conference on Software Engineering.

McCall J.A, Richards P.K & G.F Walters (1998). Software Quality Assurance.

McCall J.A, Richards P.K & Walters G.F (1977). Factors in Software Quality, Nat'1 Tech Information Services, - Vol 1-2.

Okamura H, Murayama A. & Dohi T. (2004). EM Algorithm for Descrete Software Reliability Models: A Unified Parameter Estimation Mothod, IEEE Int.Symp.

Rosenberg Linda, Hammer Ted & Shaw Jack (1998). Software Matrics and Reliability, ISSRE.

Schneidewind N. F. (1993). Software Reliability Model with Optimal Selection of Failure Data, IEEE Transactions on Software Engineering - Vol 19(11) pp 1095-1104.

Sua Yu-Shen & Chin-Yu Huang (2006). Neural-network-based Approaches for Software Reliability Estimation using Dynamic Weighted Combinational Models, Journal of Systems and Software.

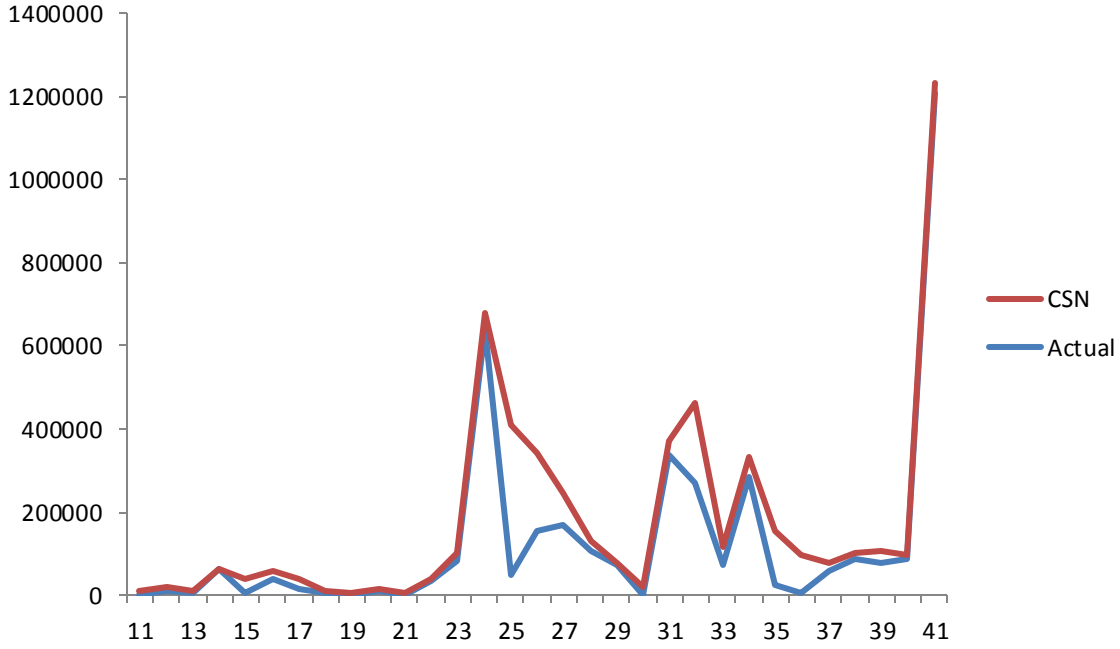
William H. & Far (1983). A Survey of Software Reliability Modeling and Estimating, Naval Surface Weapons Center, NSWC, TR 82-171 pp 4-88.

Wood Alan (1996). Software Reliability Growth Models , Tandem Technical Report- Vol 19.1-part number 130056.

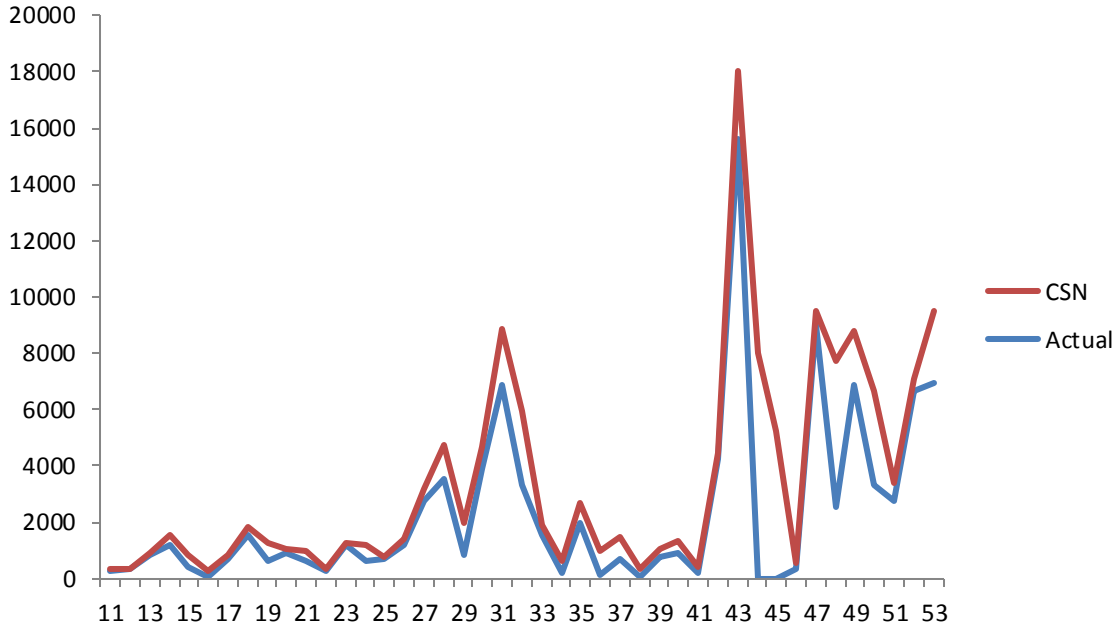
Appendix A – Graphical Representations of CSN Model Estimations

Appendix A presents the output of CSN model. Notice that CSN model estimations are possible from 7th failure onwards. But in these data, the range has been considered from 11th failure. The graphs are of datasets from I to X. The omega value used for these calculations is 0.7.

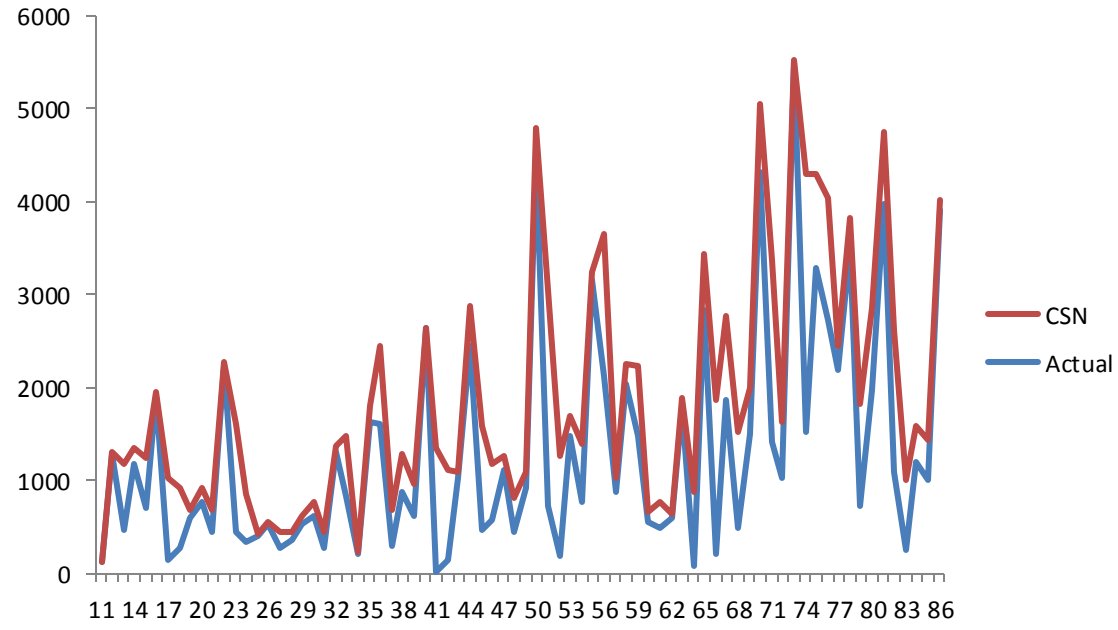
Calculations for dataset I



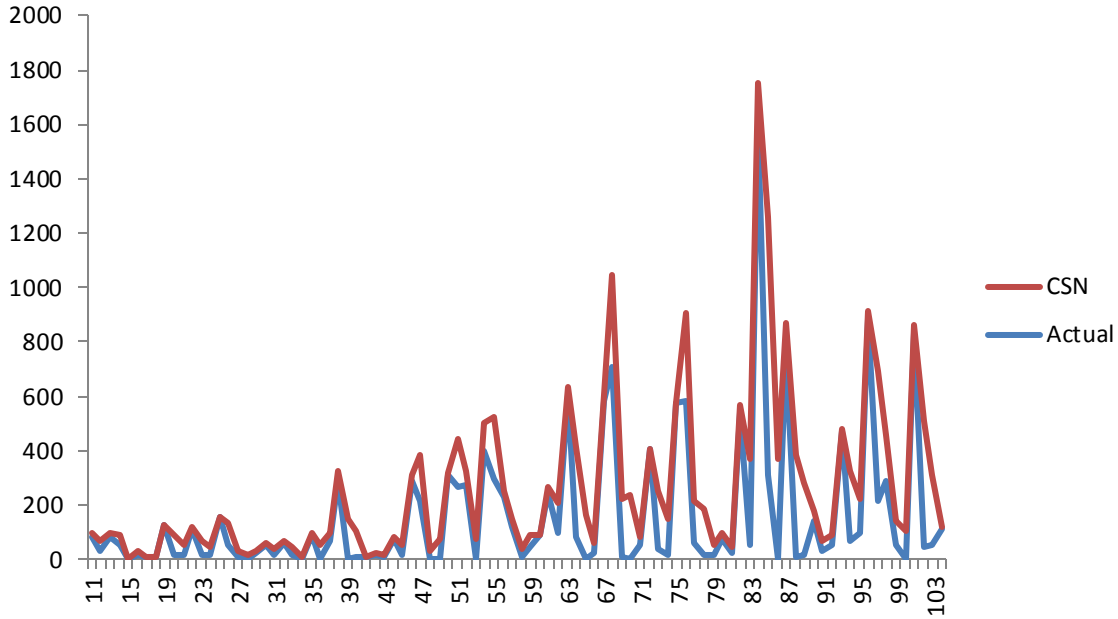
Calculations for dataset II



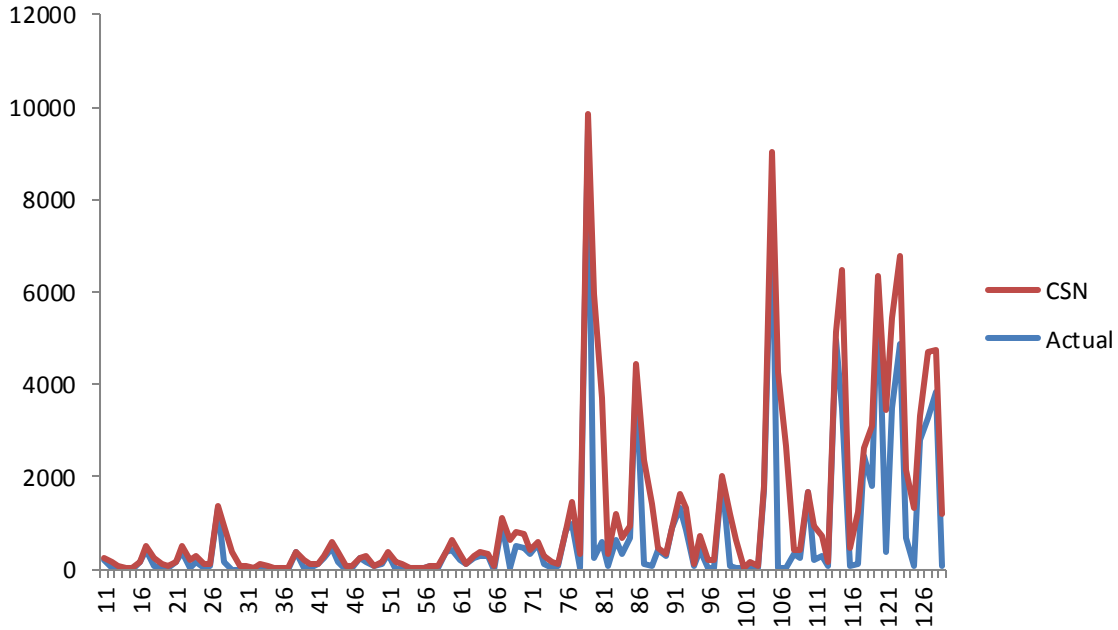
Calculations for dataset III



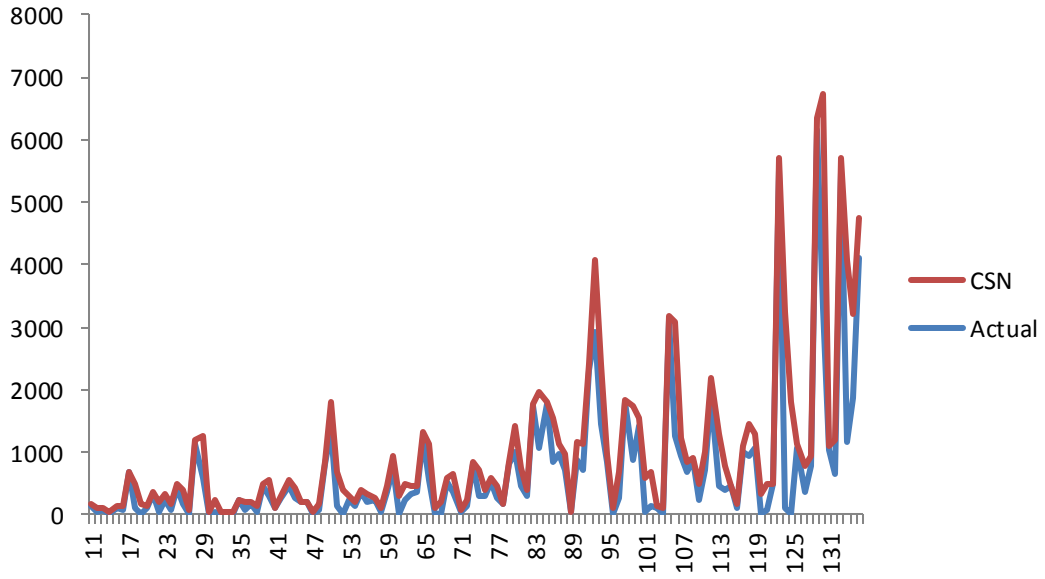
Calculations for dataset IV



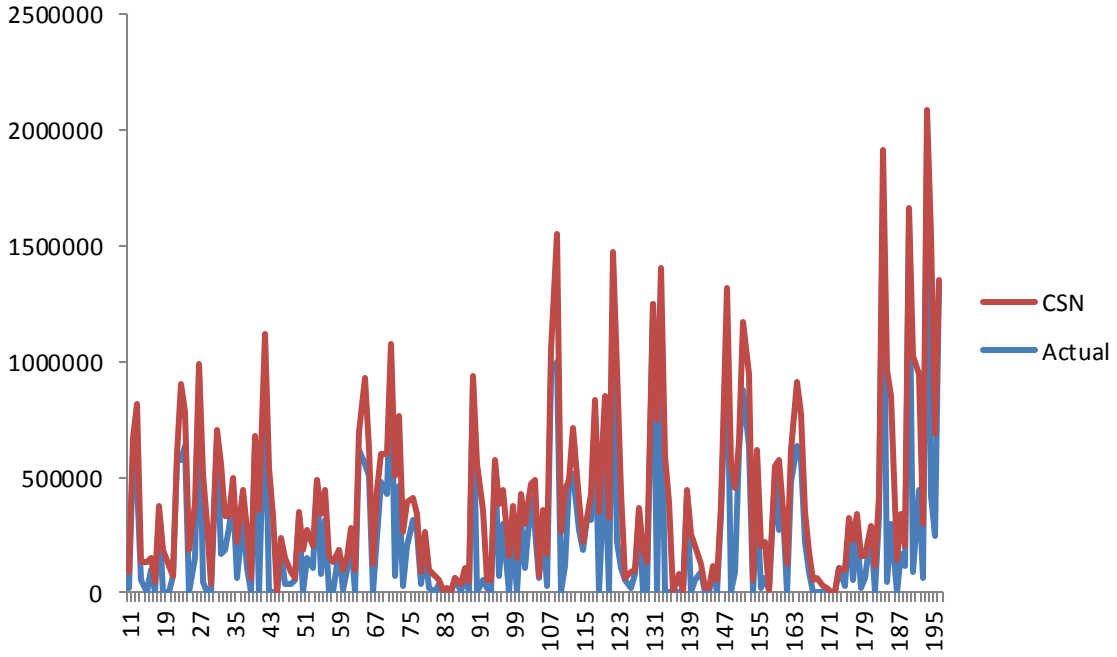
Calculations for dataset V



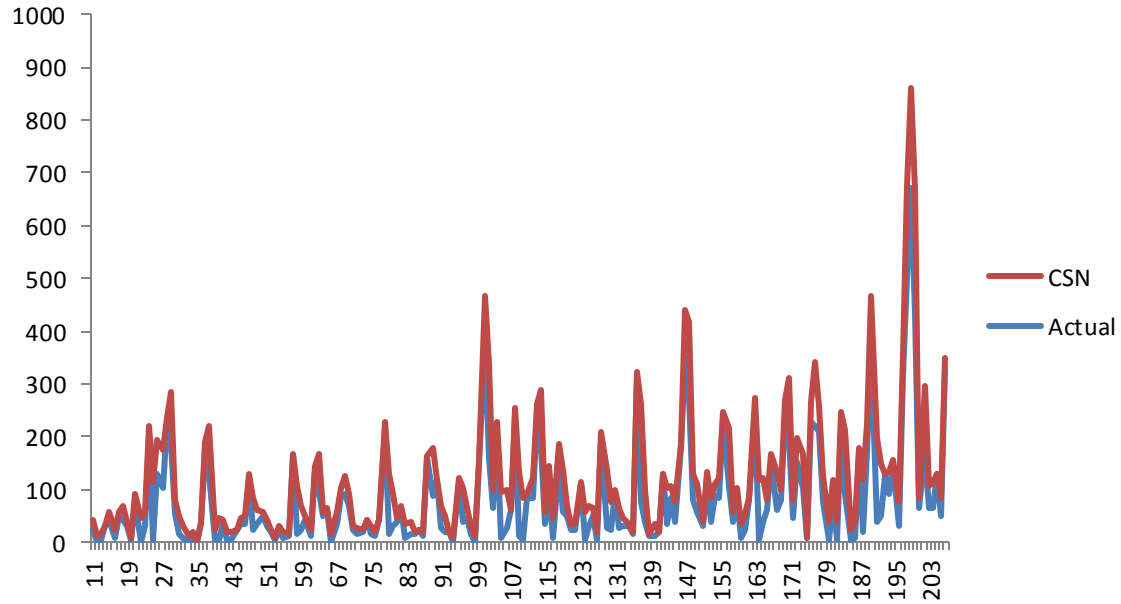
Calculations for dataset VI



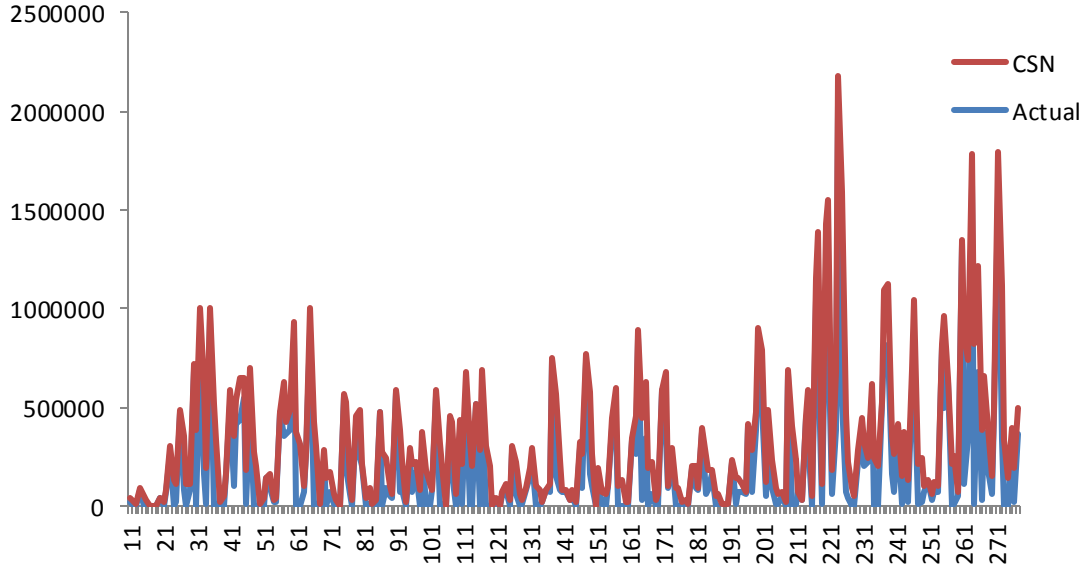
Calculations for dataset VII



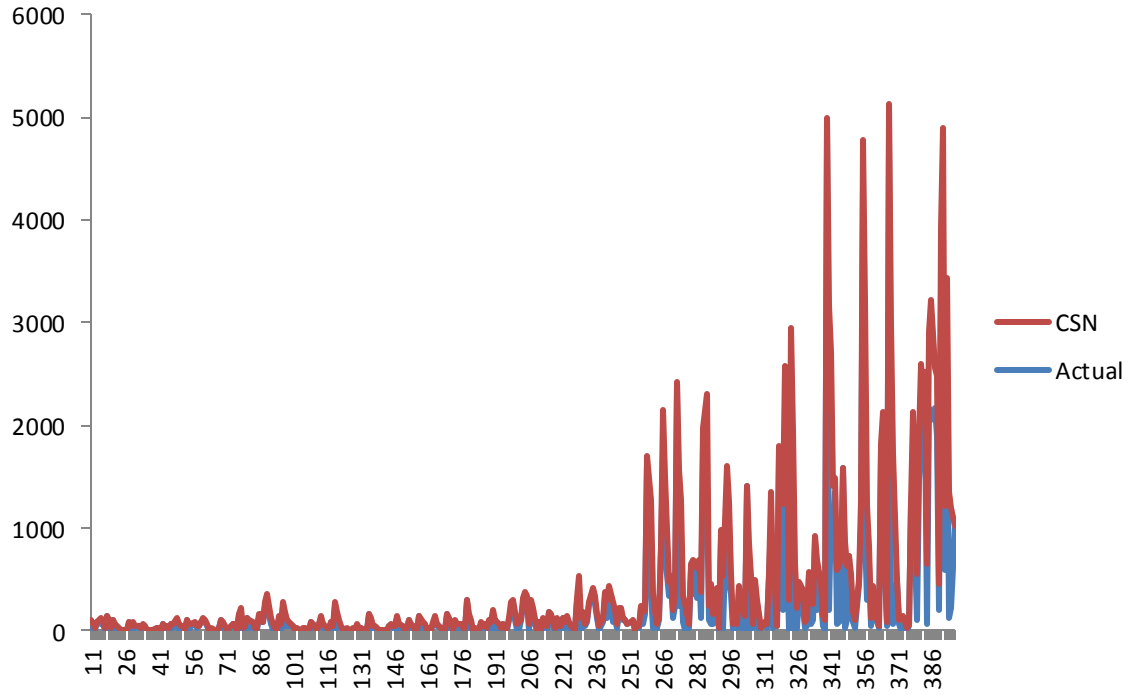
Calculations for dataset VIII



Calculations for dataset IX



Calculations for dataset X



Appendix B – Comparative study of CSN model and NHPP model

Results of the comparative study of CSN model and NHPP model have been listed here for the datasets. (The comparative studies on dataset I and II are in the thesis)

Comparative study for dataset III

FNO	Error CS	Erro NHPP	Difference	FNO	Error CS	Erro NHPP	Difference
11	97.58	434.13	-336.54	49	91.12	9.54	81.57
12	99.21	45.73	53.48	50	96.18	55.33	40.86
13	20.82	39.5	-18.69	51	71.03	40.14	30.89
14	92.64	39.89	52.75	52	216.23	372.71	-156.48
15	58.71	1.47	57.25	53	92.94	16.42	76.52
16	99.08	58.96	40.12	54	56.49	37.46	19.03
17	249.34	386.5	-137.15	55	98.44	43.21	55.23
18	21.4	142.03	-120.63	56	62.09	28.75	33.34
19	93.03	17.87	75.16	57	91.57	28.32	63.24
20	89.47	4.35	85.12	58	94.61	24.89	69.72
21	71.84	59.94	11.9	59	73.17	7.6	65.57
22	98.79	59.78	39.01	60	90.42	90.81	-0.39
23	40.82	64.04	-23.22	61	70.09	114.09	-44
24	19.67	108.76	-89.08	62	96.72	84.2	12.52
25	96.82	78.09	18.73	63	96.86	13.3	83.56
26	98.81	38.72	60.09	64	389.44	986.93	-597.49
27	70.19	157.39	-87.2	65	88.9	30.16	58.74
28	87.64	100.48	-12.85	66	305.36	366.02	-60.66
29	90.2	44.54	45.66	67	74.45	11.29	63.16
30	86.15	26.19	59.96	68	11.14	128.77	-117.63
31	67.13	162.7	-95.57	69	82.38	3.61	78.77
32	97.28	31.13	66.15	70	91.21	35.93	55.27
33	57.61	1.04	56.58	71	27.5	11.18	16.32
34	93.59	244.32	-150.73	72	68.64	38.96	29.68
35	94.45	39.12	55.33	73	99.71	36.83	62.88
36	72.65	37.45	35.2	74	4.25	12.94	-8.69
37	32.38	158.72	-126.35	75	83.78	21.39	62.38
38	76.11	1.51	74.61	76	74.57	13.08	61.49
39	71.54	36.74	34.79	77	93.57	1.74	91.83
40	99.89	51.73	48.16	78	95.29	18.69	76.6
41	13899.53	14666.8	-767.27	79	20.25	102.56	-82.31
42	237.73	416.72	-178.99	80	76.16	8.55	67.61
43	97.31	6	91.31	81	89.96	18.35	71.62
44	90.65	46.22	44.43	82	26.67	59.14	-32.48
45	27.15	88.45	-61.3	83	62.37	422.8	-360.43
46	43.07	58.91	-15.84	84	82.59	52.36	30.23
47	93.33	6.7	86.64	85	76.56	73.33	3.23
48	54.88	101.56	-46.68				

Total	-1074.34
Count of -ve values	26
Count of +ve values	49

Comparative study for dataset IV

FNO	Error% CS	Erro% NHPP	Difference	FNO	Error% CS	Erro% NHPP	Difference	FNO	Error% CS	Erro% NHPP	Difference
11	19.98	17.57	2.41	43	5.36	692.11	-686.75	75	5.95	40.47	-34.52
12	33.53	91.29	-57.77	44	9.32	12.96	-3.64	76	347.29	38.88	308.41
13	11.92	19.75	-7.83	45	44.85	386.73	-341.88	77	170.2	118.73	51.47
14	40.26	24.22	16.04	46	18.24	54.65	-36.4	78	182.36	496.58	-314.22
15	0.33	2041	-2040.67	47	182.93	44.36	138.58	79	37.56	470.15	-432.59
16	20.46	333.93	-313.48	48	29.38	1730.25	-1700.87	80	22.1	79.52	-57.42
17	2.29	975.83	-973.54	49	74.34	1369.6	-1295.26	81	24.87	386.54	-361.67
18	5.03	709.5	-704.47	50	14.24	52.84	-38.61	82	30.33	32.15	-1.81
19	0.47	42.35	-41.88	51	187.78	48.14	139.64	83	339.6	158.4	181.2
20	79.34	249.47	-170.13	52	55.89	47.62	8.26	84	164.03	42.15	121.89
21	34.61	250.37	-215.76	53	76.63	7569	-7492.37	85	1016.69	9.27	1007.42
22	4.72	32.38	-27.65	54	113.08	54.41	58.67	86	395.93	12022	-11626.07
23	59.59	346.13	-286.55	55	246.56	46	200.56	87	115.04	29.31	85.73
24	31.53	319.75	-288.22	56	25.68	36.71	-11.03	88	403.69	1211.1	-807.41
25	0.39	46.37	-45.98	57	34.39	2.56	31.83	89	283.4	584.5	-301.1
26	88.44	44	44.44	58	31.69	557.08	-525.39	90	36.39	44.71	-8.32
27	23.96	576.4	-552.44	59	47.91	103.66	-55.75	91	40.95	410.36	-369.41
28	15.39	3236.5	-3221.11	60	3.57	22.52	-18.95	92	33.91	185.29	-151.37
29	10.07	215.18	-205.11	61	26.92	34.78	-7.86	93	3.53	13.42	-9.89
30	10.73	39.04	-28.31	62	118.23	16.33	101.89	94	282.06	161.22	120.85
31	21.66	265.58	-243.92	63	28.02	53.32	-25.3	95	130.24	94.95	35.29
32	8.18	29.71	-21.53	64	354.31	38.89	315.42	96	33.58	22.19	11.39
33	29.54	250.65	-221.11	65	176.48	4319.5	-4143.02	97	515.74	29	486.74
34	10.38	2157.33	-2146.96	66	35.72	272.23	-236.51	98	183.73	13.59	170.13
35	6.76	11.13	-4.37	67	19.76	49.23	-29.47	99	92.19	231.13	-138.94
36	54.41	1270	-1215.59	68	364.65	50.04	314.61	100	112.5	4658.33	-4545.83
37	31.22	19.3	11.91	69	232.56	1518.83	-1286.27	101	29.98	14.89	15.09
38	39.29	59.18	-19.89	70	253.42	2310.5	-2057.08	102	499.84	309.81	190.02
39	159.17	2228.67	-2069.5	71	27.61	111.78	-84.18	103	278.44	236.84	41.61
40	101.93	688.78	-586.84	72	0.27	36.64	-36.37	104	11.96	105.85	-93.89
41	1.02	496.92	-495.9	73	228.96	211.69	17.27				
42	9.38	305	-295.62	74	143.65	592.73	-449.08				

Total	-43000.13
Count of -ve values	50
Count of +ve values	25

The omega value for this calculation is 0.43

Comparative study for dataset V

FNO	Error% CS	Error% NHPP	Difference	FNO	Error% CS	Error% NHPP	Difference
11	99.76	7.55	92.21	41	99.94	127.11	-27.17
12	77.14	1302.36	-1225.22	42	99.6	0.88	98.71
13	88.32	1209.93	-1121.61	43	99.36	36.21	63.15
14	99.78	383	-283.22	44	97.43	37.59	59.84
15	80.7	19500	-19419.3	45	58.46	7064.67	-7006.21
16	99.77	34.35	65.42	46	92.78	812.42	-719.64
17	99.48	45.25	54.22	47	99.91	4.32	95.58
18	91.02	274.06	-183.04	48	98	52.25	45.76
19	89.15	734.04	-644.89	49	99.92	320.87	-220.95
20	99.35	358.95	-259.6	50	99.5	78.89	20.61
21	99.99	18.66	81.33	51	99.59	12.46	87.14
22	99.45	41.05	58.4	52	20.59	7201.67	-7181.08
23	77.9	966.68	-888.79	53	67.59	2349	-2281.41
24	97.86	48.19	49.66	54	98.75	1743.08	-1644.33
25	94.95	469.92	-374.96	55	98.37	1137	-1038.63
26	97.9	283.94	-186.05	56	99.14	2352.11	-2252.97
27	99.97	74.02	25.95	57	99.68	215.44	-115.76
28	88.14	38.58	49.56	58	93.05	1383.93	-1290.89
29	9.14	2511.63	-2502.48	59	99.88	15.29	84.59
30	39.41	20715	-20675.59	60	98.74	23.05	75.69
31	90.82	1136.29	-1045.48	61	98.43	30.82	67.61
32	99.8	1213.63	-1113.82	62	99.66	120.55	-20.88
33	99.08	152.71	-53.63	63	99.82	11.66	88.16
34	94.36	1011.32	-916.96	64	99.05	11.01	88.04
35	98.41	633.55	-535.15	65	99.5	9.83	89.67
36	0	0	0	66	51.59	22649	-22597.41
37	93.01	4089.4	-3996.39	67	99.73	48.24	51.48
38	99.99	26.48	73.51	68	50.96	701.4	-650.44
39	46.92	2018	-1971.08	69	98.31	23.05	75.26
40	73.99	1827.73	-1753.74	70	98.51	19.7	78.8

FNO	Error% CS	Error% NHPP	Difference	FNO	Error% CS	Error% NHPP	Difference
71	99.5	0.08	99.42	100	18.4	1864.84	-1846.44
72	99.78	24.89	74.89	101	97.49	1770.5	-1673.01
73	96.77	115.61	-18.84	102	97.4	416.34	-318.95
74	93.3	444.57	-351.27	103	96.04	1471.58	-1375.54
75	98.81	195.04	-96.23	104	99.91	20.64	79.27
76	99.99	37.98	62.02	105	99.67	34.33	65.34
77	98.8	40.39	58.41	106	960.72	4159.7	-3198.98
78	78.96	648.2	-569.24	107	232.11	2062.35	-1830.24
79	99.92	59.54	40.38	108	99.48	88.64	10.84
80	42.56	58.8	-16.23	109	98.42	133.88	-35.46
81	87.32	10.28	77.04	110	99.99	8.71	91.27
82	92.19	302.51	-210.32	111	91.51	171.55	-80.03
83	97.54	9	88.55	112	96.1	119.26	-23.17
84	97.65	28.96	68.69	113	95.93	747.88	-651.94
85	98.98	13.48	85.5	114	99.93	21.18	78.75
86	99.83	47.65	52.19	115	97.88	14.43	83.45
87	49.14	243.71	-194.57	116	83.04	802.88	-719.84
88	55.21	387.81	-332.61	117	70.94	473.86	-402.92
89	99.77	29.11	70.66	118	99.81	4.68	95.13
90	99.78	63.91	35.86	119	98.23	4.51	93.72
91	99.96	40.19	59.77	120	99.94	13.12	86.82
92	99.37	24.6	74.77	121	80.21	125.18	-44.97
93	98.59	8.56	90.04	122	98.61	2.54	96.07
94	97.37	585.47	-488.11	123	99.03	4.78	94.24
95	98.17	37.42	60.75	124	94.73	74.52	20.2
96	88.69	928.44	-839.75	125	48.35	988.48	-940.13
97	70.67	2296.33	-2225.66	126	99.54	8.76	90.78
98	99.95	27.83	72.12	127	98.9	7.55	91.34
99	52.54	559.18	-506.64	128	99.4	6.69	92.71

Total	-121128.53
Count of -ve values	60
Count of +ve values	57

The omega value for this calculation is 0.01

Comparative study for dataset VI

FNO	Error CS	Erro NHPP	Difference	FNO	Error CS	Erro NHPP	Difference
11	77.8	52.19	25.61	43	78.08	33.36	44.72
12	44.56	314.5	-269.94	44	34.75	6.6	28.15
13	73.65	171.25	-97.6	45	93.55	33.96	59.59
14	8.81	761.54	-752.74	46	92.73	37.19	55.54
15	81.69	95.76	-14.07	47	360.11	3775.67	-3415.56
16	39.44	139.84	-100.4	48	2.44	338.04	-335.6
17	98.53	63.12	35.41	49	97.02	52.68	44.34
18	203.26	80.43	122.83	50	67.41	63.09	4.32
19	491.19	711.38	-220.19	51	255.14	82.36	172.78
20	75.34	90.77	-15.43	52	1615.68	1067.1	548.59
21	91.73	27.9	63.82	53	69.42	24.27	45.15
22	167.27	291.35	-124.08	54	47.84	101.77	-53.93
23	62.08	4.45	57.63	55	93.17	10.35	82.82
24	71.59	220.38	-148.79	56	32.22	48.52	-16.29
25	87.36	40.62	46.74	57	93.12	26.28	66.84
26	22.92	28.22	-5.3	58	125.82	714.84	-589.02
27	400.09	2053.6	-1653.51	59	78.33	9.9	68.43
28	95.03	70.99	24.04	60	73.86	42.89	30.97
29	7.92	52.79	-44.87	61	0	0	0
30	89.47	1383.53	-1294.07	62	7.53	32.79	-25.26
31	448.79	525.19	-76.41	63	65.27	1.54	63.73
32	849.93	5438	-4588.07	64	76.48	4.89	71.58
33	0	0	0	65	91.96	52.64	39.32
34	296.83	2675.88	-2379.05	66	5.35	25.11	-19.76
35	94.52	10.46	84.05	67	814.44	2538	-1723.56
36	100.95	254.91	-153.96	68	1102.7	1559.81	-457.12
37	79.32	40.09	39.23	69	89.54	22.01	67.53
38	41.63	298.45	-256.82	70	27.83	1.36	26.47
39	92.81	36.53	56.28	71	56.3	531.7	-475.4
40	17.52	9.98	7.54	72	13.13	135.58	-122.45
41	84.8	148.4	-63.61	73	95.24	36.57	58.67
42	83.07	1.95	81.11	74	44.34	23.99	20.36

FNO	Error CS	Erro NHPP	Difference	FNO	Error CS	Erro NHPP	Difference
75	64	21.81	42.19	105	98.11	33.16	64.94
76	87.31	16.4	70.91	106	43.49	12.03	31.46
77	36.42	30.59	5.83	107	70.17	0.03	70.14
78	85.73	105.16	-19.43	108	79.04	16.56	62.47
79	96.28	32.82	63.47	109	96.4	5.94	90.45
80	59.89	37.78	22.11	110	4.44	131.45	-127.01
81	31.06	1.01	30.05	111	64.1	17.76	46.34
82	70.61	32.28	38.33	112	84.86	16.87	67.98
83	99.24	47.07	52.17	113	83.13	59.55	23.59
84	16.34	34.84	-18.5	114	1.06	76.35	-75.3
85	99.11	44.83	54.28	115	93.75	62.15	31.6
86	22.57	25	-2.42	116	59.99	328.47	-268.48
87	83.7	28.31	55.38	117	88.75	9.29	79.46
88	65.2	14.58	50.63	118	49.61	12.74	36.86
89	22.08	908.21	-886.13	119	80.96	8.22	72.74
90	68.29	21.27	47.02	120	1328.34	2069.41	-741.07
91	43.11	12.66	30.45	121	424.24	584.28	-160.04
92	96.06	42.27	53.79	122	98.51	64.08	34.43
93	62.23	43.45	18.78	123	96.6	21.99	74.62
94	37.32	29.68	7.64	124	2964.39	468.01	2496.38
95	85.35	10.3	75.05	125	17528.72	4947	12581.72
96	743.75	2974.75	-2231	126	95.51	18.85	76.67
97	58.12	87.48	-29.36	127	4.38	108.98	-104.59
98	98.54	30.61	67.93	128	81.48	37.99	43.49
99	0.2	6.73	-6.53	129	96.73	16.07	80.66
100	92.07	22.69	69.39	130	0.07	5.9	-5.82
101	1671.14	1236.1	435.04	131	94.46	33.07	61.39
102	268.54	221.83	46.71	132	18.9	68.08	-49.19
103	71.99	310.41	-238.42	133	95.74	8.33	87.41
104	0	0	0	134	146.33	35.24	111.09

Total	-4548.93
Count of -ve values	46
Count of +ve values	75

The omega value for this calculation is 0.39

Comparative study for dataset VII

Fno	Error CSN	Error NHPP	Difference	Fno	Error CSN	Error NHPP	Difference
11	77.68	538.54	-460.86	60	97.50	76.61	20.89
12	363.45	12368.26	-12004.81	61	76948.16	109507.31	-32559.15
13	99.20	12368.51	-12269.31	62	2182.28	3758.41	-1576.13
14	96.73	51.63	45.09	63	91.80	85.81	5.99
15	4758.44	49811.87	-45053.42	64	99.63	41.26	58.37
16	5651.50	99722.22	-94070.72	65	96.48	76.23	20.25
17	92.35	6556.83	-6464.49	66	29150.40	36898.23	-7747.83
18	31.61	24857.83	-24826.22	67	807.42	1540.95	-733.53
19	88.91	15259.74	-15170.83	68	864.65	73894.21	-73029.56
20	98.78	224.02	-125.24	69	95.41	34.69	60.72
21	163.99	5609.09	-5445.10	70	28386.13	111155.54	-82769.41
22	96.12	69.08	27.04	71	63.61	98.17	-34.56
23	95.87	50.73	45.13	72	17.56	940.55	-922.99
24	6706.43	24956.36	-18249.93	73	2113.19	44446.20	-42333.01
25	1.76	452.28	-450.52	74	764.33	111251.13	-110486.79
26	99.54	71.57	27.96	75	99.57	65.90	33.68
27	16.28	53.12	-36.84	76	55.20	18.20	37.00
28	2247.25	9986.35	-7739.10	77	66.94	149.69	-82.75
29	89.45	53.62	35.83	78	1626.95	31982.14	-30355.18
30	99.02	78.60	20.42	79	97.39	56.72	40.66
31	4239.01	5398.34	-1159.33	80	74.87	32.89	41.99
32	92.06	79.83	12.23	81	98.79	23.29	75.50
33	32.15	21.97	10.18	82	108.91	2777.86	-2668.95
34	452.22	1391.86	-939.64	83	111.87	1251.46	-1139.58
35	98.61	82.54	16.07	84	413.62	45211.06	-44797.44
36	9751.95	9329.03	422.92	85	41.52	3194.23	-3152.71
37	450.65	751.07	-300.41	86	99.78	59.41	40.36
38	11.11	3543.00	-3531.88	87	23067.09	45429.44	-22362.35
39	391.65	4842.62	-4450.97	88	55.10	67.98	-12.87
40	99.98	18.24	81.75	89	72.85	195.56	-122.71
41	96.57	70.72	25.85	90	87.46	238.42	-150.95
42	32.76	35.03	-2.27	91	99.60	63.73	35.87
43	93.10	64.40	28.69	92	10.72	104.28	-93.57
44	87.74	65.27	22.47	93	54.59	78.54	-23.95
45	95.39	70.83	24.57	94	513.59	15224.77	-14711.18
46	396.41	1348.98	-952.57	95	94.37	29.51	64.86
47	89.33	67.75	21.58	96	52.32	100.62	-48.31
48	13977.93	26556.58	-12578.65	97	91.96	5.15	86.82
49	9056.94	23595.69	-14538.75	98	1902386.71	13847612.00	-11945225.29
50	33.10	5668.87	-5635.77	99	94.25	40.45	53.80
51	711.12	11751.67	-11040.56	100	1234.89	4047.71	-2812.82
52	99.94	0.00	99.94	101	1103.89	5849.97	-4746.09
53	81.06	46.45	34.60	102	99.49	83.03	16.46
54	89.44	469.54	-380.10	103	98.62	60.83	37.78
55	81.57	466.86	-385.29	104	9863.82	17837.35	-7973.53
56	100.00	65.49	34.50	105	352.36	1216.31	-863.95
57	80.67	56.12	24.56	106	245.30	77589.37	-77344.07
58	96.48	57.27	39.20	107	98.35	52.61	45.75
59	93.32	57.98	35.34	108	61.30	0.68	60.62

Fno	Error CSN	Error NHPP	Difference	Fno	Error CSN	Error NHPP	Difference
109	2326.81	23356.86	-21030.04	154	24585.33	248080.38	-223495.05
110	96.33	47.19	49.14	155	98.31	74.64	23.68
111	918.24	2683.77	-1765.53	156	96.80	39.94	56.85
112	94.10	58.56	35.53	157	86.71	38.91	47.80
113	14666.21	26204.29	-11538.08	158	13868.78	83359.66	-69490.88
114	5567.15	15689.17	-10122.02	159	18632.41	83359.94	-64727.54
115	99.99	56.55	43.45	160	1740.36	62501.01	-60760.65
116	16585.33	33891.26	-17305.93	161	822.19	27735.84	-26913.66
117	91.63	55.93	35.70	162	99.23	31.78	67.45
118	7219.97	13978.38	-6758.41	163	81.76	19.20	62.56
119	1080.22	3381.37	-2301.14	164	98.60	58.11	40.49
120	10.40	29799.55	-29789.15	165	300.44	499.88	-199.44
121	5183.42	79604.31	-74420.90	166	79.86	31.95	47.92
122	95.51	23823.48	-23727.97	167	1067.62	3671.22	-2603.60
123	93.62	184.25	-90.62	168	37.92	161.11	-123.18
124	90.22	91.67	-1.45	169	778.17	18134.79	-17356.62
125	4315.56	79712.79	-75397.23	170	2639.19	28251.76	-25612.58
126	97.49	29.50	67.99	171	99.96	48.82	51.15
127	32.13	158.56	-126.43	172	74.10	28.23	45.87
128	1640.65	17073.43	-15432.78	173	98.93	89.58	9.34
129	26.84	1936.62	-1909.77	174	94.21	9.83	84.38
130	96.63	107.31	-10.69	175	709.99	5660.22	-4950.22
131	93.09	15.64	77.45	176	2488.07	17129.27	-14641.20
132	94.00	20.47	73.53	177	369.28	14262.35	-13893.07
133	405.96	3093.83	-2687.87	178	58.60	1263.49	-1204.89
134	940.86	7459.76	-6518.90	179	53.60	7095.49	-7041.89
135	89.39	777.98	-688.60	180	98.83	249.18	-150.35
136	99.01	107.36	-8.35	181	67.09	106.75	-39.67
137	87.70	113.39	-25.69	182	96.54	114.48	-17.94
138	99.56	60.62	38.94	183	98.57	30.68	67.89
139	28.50	19.31	9.20	184	18.14	198.76	-180.62
140	47.61	110.87	-63.26	185	71.79	117.02	-45.23
141	97.73	127.02	-29.29	186	96.48	28.14	68.35
142	93.34	141.76	-48.42	187	1747.78	24020.84	-22273.06
143	92.85	452.85	-360.00	188	350.71	4928.76	-4578.04
144	87.05	187.21	-100.16	189	94.06	10121.93	-10027.87
145	3443.72	122184.02	-118740.29	190	31.39	4929.28	-4897.90
146	98.04	29.56	68.48	191	96.21	2718.72	-2622.51
147	51.14	84.34	-33.20	192	99.78	0.81	98.97
148	98.32	58.31	40.01	193	242.88	1548.83	-1305.95
149	39.08	10.33	28.76	194	74.73	160.12	-85.39
150	63.47	61.62	1.85	195	86.15	161.04	-74.89
151	133.38	30904.09	-30770.71	196	96.77	173.46	-76.69
152	83.98	48.75	35.22	197	99.01	30.53	68.48
153	135.13	2219.80	-2084.67				

Total	**-13839003.26**
Count of +ve values	**70.00**
Count of -ve values	**117.00**

FNO	Error CS	Error NHPP	Difference
171	4.34	22.63	-18.28
172	15.95	86.46	-70.51
173	51	11.93	39.07
174	3.48	9.9	-6.42
175	97.38	678.78	-581.4
176	73.12	24.88	48.25
177	7.88	23.07	-15.19
178	59.37	20.57	38.8
179	1.41	34.36	-32.95
180	1827.2	2212	-384.8
181	32	30	2
182	830.76	1084.33	-253.58
183	74.03	19.01	55.02
184	118.42	24.78	93.64
185	1078.57	2248.67	-1170.1
186	376.33	639.7	-263.37
187	93.68	9.67	84.01
188	657.65	281.38	376.27
189	51.9	8.87	43.03
190	57.88	29.65	28.23
191	555.29	129.48	425.81
192	246.13	100.42	145.71
193	95.54	9.17	86.37
194	20.97	32.56	-11.59
195	93.34	1.62	91.72
196	164.19	194.57	-30.38
197	80.12	22.95	57.16
198	41.81	30.44	11.37
199	54.26	33.12	21.14
200	9.71	25.66	-15.95
201	53.19	74.88	-21.69
202	24.86	3.92	20.94
203	2.96	76.7	-73.74
204	21.6	77.33	-55.73
205	96.15	20.22	75.93
206	22.51	120.51	-98

Total	-3042.94
Count of -ve values	84
Count of +ve values	112

The omega value for this calculation is 0.66

Comparative study for dataset VIII

FNO	Error CS	Error NHPP	Difference	FNO	Error CS	Error NHPP	Difference
11	5.68	63.12	-57.44	51	39.8	48.81	-9.01
12	1649	3931	-2282	52	89.09	121.76	-32.67
13	312.97	909.75	-596.78	53	10.08	1019.75	-1009.67
14	97.86	37.07	60.8	54	47.4	104.04	-56.64
15	43.5	0.98	42.52	55	88.26	404.78	-316.52
16	180.76	352.44	-171.68	56	43.65	253.31	-209.66
17	68.33	14.12	54.21	57	96.17	61.21	34.97
18	5.29	4.32	0.97	58	1005.7	232.07	773.63
19	57.25	30.72	26.53	59	252.84	116.05	136.8
20	91.09	1260.67	-1169.58	60	70.06	21.98	48.08
21	74.3	43.69	30.61	61	35.77	287.33	-251.56
22	7151.51	3990	3161.51	62	91.17	54.42	36.75
23	54.69	40.73	13.96	63	28.59	39.11	-10.52
24	86.73	75.31	11.42	64	68.92	6.27	62.66
25	3516.42	731.4	2785.02	65	91.56	12.79	78.77
26	16.3	62.8	-46.49	66	1047.93	2199	-1151.07
27	16.77	54.24	-37.47	67	13.14	44.74	-31.6
28	92.55	75.74	16.8	68	72.88	34.17	38.71
29	11.25	71.45	-60.2	69	32.46	34.28	-1.82
30	17.31	13.53	3.78	70	44.19	18.22	25.98
31	308.66	211.43	97.23	71	36.27	125.59	-89.32
32	314.33	381.56	-67.22	72	50.79	224.33	-173.54
33	744.8	2045.5	-1300.7	73	72.21	159.68	-87.47
34	66.98	334.6	-267.62	74	95.89	26.33	69.56
35	66.06	4187	-4120.94	75	84.73	248.29	-163.55
36	77.95	33.18	44.78	76	20.69	339.36	-318.68
37	80.83	67.26	13.56	77	97.79	30.05	67.74
38	16.06	58.74	-42.69	78	85.19	61.04	24.15
39	774.97	995	-220.03	79	1056.41	211.44	844.98
40	1647.35	995.25	652.1	80	240.65	74.37	166.29
41	69.37	32.66	36.71	81	52.7	45.05	7.65
42	335.03	779.8	-444.77	82	93.16	10.65	82.51
43	340.69	828.6	-487.91	83	372.16	444.89	-72.73
44	83.03	99.95	-16.93	84	142.26	214.69	-72.43
45	54.3	30.42	23.88	85	45.79	257.57	-211.79
46	33.54	34.31	-0.77	86	95.84	116.58	-20.75
47	89.09	54.15	34.94	87	16.21	315.25	-299.04
48	333.02	101.3	231.71	88	96.66	50.96	45.7
49	40.83	43.67	-2.84	89	69.16	26.75	42.41
50	65.15	2.29	62.86	90	99.03	39.55	59.47

FNO	Error CS	Error NHPP	Difference	FNO	Error CS	Error NHPP	Difference
91	131.85	86.79	45.05	131	91.38	11.1	80.27
92	131.85	150.86	-19	132	152.29	145.54	6.75
93	88.53	189.83	-101.3	133	2.95	117.43	-114.48
94	362.47	2347.5	-1985.03	134	82.34	117.93	-35.6
95	87.77	36.55	51.22	135	69.44	260.82	-191.38
96	186.12	53	133.12	136	99.2	49.94	49.26
97	38.26	27.63	10.63	137	288.94	5.65	283.29
98	43.87	209.71	-165.84	138	160.4	79.08	81.32
99	1649.94	4823	-3173.06	139	49.45	371.69	-322.24
100	92.56	45.22	47.34	140	155.95	372.15	-216.2
101	63.18	64.59	-1.41	141	96.58	234.16	-137.58
102	84.75	45.74	39.01	142	95.99	21.06	74.92
103	20.27	0.23	20.04	143	231.83	103.41	128.42
104	83.53	51.9	31.63	144	53.38	3.36	50.03
105	1467.66	492.33	975.32	145	57.79	79.4	-21.61
106	371.75	120.92	250.83	146	90.81	31.78	59.03
107	98.92	6.94	91.98	147	56.68	47.27	9.41
108	89.38	53.9	35.48	148	12.8	41.88	-29.08
109	1443.82	324.38	1119.44	149	6.23	9.8	-3.57
110	3136.36	1224.25	1912.11	150	47.57	40.6	6.97
111	90.47	13.45	77.02	151	52.89	131.35	-78.47
112	24.82	12.93	11.89	152	74.68	9.36	65.32
113	85.69	52.15	33.54	153	93.02	93.15	-0.13
114	2.14	43.87	-41.73	154	54.42	7.28	47.14
115	13.89	83.56	-69.67	155	28.05	11.14	16.91
116	30.98	20.65	10.33	156	89.45	34.5	54.95
117	585.46	523.33	62.12	157	67.55	3.4	64.15
118	56.22	35.69	20.53	158	17.48	102.61	-85.13
119	116.28	19.39	96.89	159	65.84	12.24	53.59
120	23.39	40.94	-17.54	160	528.35	828.14	-299.8
121	46.94	145.4	-98.46	161	136.06	223.27	-87.22
122	33.36	145.92	-112.56	162	93.62	18.86	74.76
123	94.18	22.09	72.09	163	76.31	32.38	43.93
124	1640.14	1021.8	618.34	164	6305.77	2042.33	4263.44
125	95.14	105.74	-10.6	165	243.22	103.74	139.48
126	52.51	37.02	15.49	166	50.69	42.65	8.04
127	309.51	844	-534.49	167	84.95	15.55	69.39
128	87.81	41.29	46.52	168	79.7	44.24	35.46
129	574.18	135.48	438.7	169	57.32	23.29	34.04
130	232.72	152.36	80.36	170	83.52	29.95	53.57

FNO	Error CS	Error NHPP	Difference
171	4.34	22.63	-18.28
172	15.95	86.46	-70.51
173	51	11.93	39.07
174	3.48	9.9	-6.42
175	97.38	678.78	-581.4
176	73.12	24.88	48.25
177	7.88	23.07	-15.19
178	59.37	20.57	38.8
179	1.41	34.36	-32.95
180	1827.2	2212	-384.8
181	32	30	2
182	830.76	1084.33	-253.58
183	74.03	19.01	55.02
184	118.42	24.78	93.64
185	1078.57	2248.67	-1170.1
186	376.33	639.7	-263.37
187	93.68	9.67	84.01
188	657.65	281.38	376.27
189	51.9	8.87	43.03
190	57.88	29.65	28.23
191	555.29	129.48	425.81
192	246.13	100.42	145.71
193	95.54	9.17	86.37
194	20.97	32.56	-11.59
195	93.34	1.62	91.72
196	164.19	194.57	-30.38
197	80.12	22.95	57.16
198	41.81	30.44	11.37
199	54.26	33.12	21.14
200	9.71	25.66	-15.95
201	53.19	74.88	-21.69
202	24.86	3.92	20.94
203	2.96	76.7	-73.74
204	21.6	77.33	-55.73
205	96.15	20.22	75.93
206	22.51	120.51	-98

Total	-3042.94
Count of -ve values	84
Count of +ve values	112

The omega value for this calculation is 0.66

Comparative study for dataset IX

FNO	Error CS	Error NHPP	Difference	FNO	Error CS	Error NHPP	Difference
11	77.68	538.54	-460.86	56	100	65.49	34.5
12	363.45	12368.26	-12004.81	57	80.67	56.12	24.56
13	99.2	12368.51	-12269.31	58	96.48	57.27	39.2
14	96.73	51.63	45.09	59	93.32	57.98	35.34
15	4758.44	49811.87	-45053.42	60	97.5	76.61	20.89
16	5651.5	99722.22	-94070.72	61	76948.16	109507.31	-32559.15
17	92.35	6556.83	-6464.49	62	2182.28	3758.41	-1576.13
18	31.61	24857.83	-24826.22	63	91.8	85.81	5.99
19	88.91	15259.74	-15170.83	64	99.63	41.26	58.37
20	98.78	224.02	-125.24	65	96.48	76.23	20.25
21	163.99	5609.09	-5445.1	66	29150.4	36898.23	-7747.83
22	96.12	69.08	27.04	67	807.42	1540.95	-733.53
23	95.87	50.73	45.13	68	864.65	73894.21	-73029.56
24	6706.43	24956.36	-18249.93	69	95.41	34.69	60.72
25	1.76	452.28	-450.52	70	28386.13	111155.54	-82769.41
26	99.54	71.57	27.96	71	63.61	98.17	-34.56
27	16.28	53.12	-36.84	72	17.56	940.55	-922.99
28	2247.25	9986.35	-7739.1	73	2113.19	44446.2	-42333.01
29	89.45	53.62	35.83	74	764.33	111251.13	-110486.79
30	99.02	78.6	20.42	75	99.57	65.9	33.68
31	4239.01	5398.34	-1159.33	76	55.2	18.2	37
32	92.06	79.83	12.23	77	66.94	149.69	-82.75
33	32.15	21.97	10.18	78	1626.95	31982.14	-30355.18
34	452.22	1391.86	-939.64	79	97.39	56.72	40.66
35	98.61	82.54	16.07	80	74.87	32.89	41.99
36	9751.95	9329.03	422.92	81	98.79	23.29	75.5
37	450.65	751.07	-300.41	82	108.91	2777.86	-2668.95
38	11.11	3543	-3531.88	83	111.87	1251.46	-1139.58
39	391.65	4842.62	-4450.97	84	413.62	45211.06	-44797.44
40	99.98	18.24	81.75	85	41.52	3194.23	-3152.71
41	96.57	70.72	25.85	86	99.78	59.41	40.36
42	32.76	35.03	-2.27	87	23067.09	45429.44	-22362.35
43	93.1	64.4	28.69	88	55.1	67.98	-12.87
44	87.74	65.27	22.47	89	72.85	195.56	-122.71
45	95.39	70.83	24.57	90	87.46	238.42	-150.95
46	396.41	1348.98	-952.57	91	99.6	63.73	35.87
47	89.33	67.75	21.58	92	10.72	104.28	-93.57
48	13977.93	26556.58	-12578.65	93	54.59	78.54	-23.95
49	9056.94	23595.69	-14538.75	94	513.59	15224.77	-14711.18
50	33.1	5668.87	-5635.77	95	94.37	29.51	64.86
51	711.12	11751.67	-11040.56	96	52.32	100.62	-48.31
52	99.94	0	99.94	97	91.96	5.15	86.82
53	81.06	46.45	34.6	98	1902386.71	13847612	-11945225.29
54	89.44	469.54	-380.1	99	94.25	40.45	53.8
55	81.57	466.86	-385.29	100	1234.89	4047.71	-2812.82

FNO	Error CS	Error NHPP	Difference	FNO	Error CS	Error NHPP	Difference
101	1103.89	5849.97	-4746.09	146	98.04	29.56	68.48
102	99.49	83.03	16.46	147	51.14	84.34	-33.2
103	98.62	60.83	37.78	148	98.32	58.31	40.01
104	9863.82	17837.35	-7973.53	149	39.08	10.33	28.76
105	352.36	1216.31	-863.95	150	63.47	61.62	1.85
106	245.3	77589.37	-77344.07	151	133.38	30904.09	-30770.71
107	98.35	52.61	45.75	152	83.98	48.75	35.22
108	61.3	0.68	60.62	153	135.13	2219.8	-2084.67
109	2326.81	23356.86	-21030.04	154	24585.33	248080.38	-223495.05
110	96.33	47.19	49.14	155	98.31	74.64	23.68
111	918.24	2683.77	-1765.53	156	96.8	39.94	56.85
112	94.1	58.56	35.53	157	86.71	38.91	47.8
113	14666.21	26204.29	-11538.08	158	13868.78	83359.66	-69490.88
114	5567.15	15689.17	-10122.02	159	18632.41	83359.94	-64727.54
115	99.99	56.55	43.45	160	1740.36	62501.01	-60760.65
116	16585.33	33891.26	-17305.93	161	822.19	27735.84	-26913.66
117	91.63	55.93	35.7	162	99.23	31.78	67.45
118	7219.97	13978.38	-6758.41	163	81.76	19.2	62.56
119	1080.22	3381.37	-2301.14	164	98.6	58.11	40.49
120	10.4	29799.55	-29789.15	165	300.44	499.88	-199.44
121	5183.42	79604.31	-74420.9	166	79.86	31.95	47.92
122	95.51	23823.48	-23727.97	167	1067.62	3671.22	-2603.6
123	93.62	184.25	-90.62	168	37.92	161.11	-123.18
124	90.22	91.67	-1.45	169	778.17	18134.79	-17356.62
125	4315.56	79712.79	-75397.23	170	2639.19	28251.76	-25612.58
126	97.49	29.5	67.99	171	99.96	48.82	51.15
127	32.13	158.56	-126.43	172	74.1	28.23	45.87
128	1640.65	17073.43	-15432.78	173	98.93	89.58	9.34
129	26.84	1936.62	-1909.77	174	94.21	9.83	84.38
130	96.63	107.31	-10.69	175	709.99	5660.22	-4950.22
131	93.09	15.64	77.45	176	2488.07	17129.27	-14641.2
132	94	20.47	73.53	177	369.28	14262.35	-13893.07
133	405.96	3093.83	-2687.87	178	58.6	1263.49	-1204.89
134	940.86	7459.76	-6518.9	179	53.6	7095.49	-7041.89
135	89.39	777.98	-688.6	180	98.83	249.18	-150.35
136	99.01	107.36	-8.35	181	67.09	106.75	-39.67
137	87.7	113.39	-25.69	182	96.54	114.48	-17.94
138	99.56	60.62	38.94	183	98.57	30.68	67.89
139	28.5	19.31	9.2	184	18.14	198.76	-180.62
140	47.61	110.87	-63.26	185	71.79	117.02	-45.23
141	97.73	127.02	-29.29	186	96.48	28.14	68.35
142	93.34	141.76	-48.42	187	1747.78	24020.84	-22273.06
143	92.85	452.85	-360	188	350.71	4928.76	-4578.04
144	87.05	187.21	-100.16	189	94.06	10121.93	-10027.87
145	3443.72	122184.02	-118740.29	190	31.39	4929.28	-4897.9

FNO	Error CS	Error NHPP	Difference	FNO	Error CS	Error NHPP	Difference
191	96.21	2718.72	-2622.51	235	5887.44	17545.62	-11658.18
192	99.78	0.81	98.97	236	5078.26	18646.43	-13568.17
193	242.88	1548.83	-1305.95	237	99.09	26.56	72.54
194	74.73	160.12	-85.39	238	91.47	39.27	52.2
195	86.15	161.04	-74.89	239	90.08	38.3	51.78
196	96.77	173.46	-76.69	240	69.34	53.44	15.91
197	99.01	30.53	68.48	241	32.62	195.39	-162.77
198	29.58	142.41	-112.83	242	96.8	9.92	86.88
199	93.95	27.59	66.36	243	303.82	1964.63	-1660.81
200	92.91	46.7	46.21	244	84.12	20.81	63.31
201	86.18	37.23	48.94	245	60.27	1013.2	-952.93
202	69.88	230.16	-160.28	246	96.4	28.67	67.74
203	87.2	17.82	69.38	247	87.19	32.33	54.86
204	62.89	102.91	-40.02	248	328.74	1560.71	-1231.96
205	299.01	3138.6	-2839.58	249	134.64	723.6	-588.97
206	99.82	2552.96	-2453.14	250	92.73	187.36	-94.64
207	87.06	262.64	-175.58	251	98.62	93.35	5.27
208	1216.7	38897.99	-37681.29	252	68.65	660.59	-591.94
209	99.06	43.34	55.73	253	85.14	190.46	-105.32
210	83316.94	137169.22	-53852.27	254	87.84	216.76	-128.92
211	680.21	2219.79	-1539.58	255	99.69	34.58	65.11
212	98.5	171.56	-73.06	256	76.71	19.52	57.18
213	92.27	547.3	-455.03	257	99.03	26.69	72.33
214	99.62	30.23	69.39	258	1453.28	5663.1	-4209.82
215	82.02	17.98	64.04	259	736.08	2556.21	-1820.13
216	2637.15	34529.35	-31892.19	260	89.04	347.41	-258.37
217	97.26	50.39	46.87	261	99.56	41.86	57.7
218	80.49	44.48	36.01	262	76.18	119.7	-43.52
219	235.36	2034.93	-1799.57	263	71.4	0.45	70.96
220	94.24	50.92	43.32	264	97.38	43.32	54.06
221	82.31	46.33	35.97	265	1243.43	1258.96	-15.54
222	45.48	229.38	-183.9	266	80.33	25.88	54.45
223	80.78	19.84	60.94	267	239.24	698.71	-459.47
224	98.28	55.85	42.43	268	81	1.74	79.26
225	49.05	30.21	18.85	269	71.31	54.92	16.39
226	50.62	204.38	-153.76	270	65.5	254.75	-189.25
227	0.83	343.46	-342.63	271	98.75	11.19	87.56
228	977.32	8618.81	-7641.49	272	96.39	40.51	55.88
229	1846.5	29456.78	-27610.28	273	41.32	7.71	33.62
230	93.63	15.23	78.41	274	46293.48	112540.44	-66246.96
231	90.79	8.64	82.14	275	4121.9	24968.25	-20846.35
232	85.01	25.82	59.18	276	94.49	10.66	83.83
233	98.76	16.16	82.59	277	90.24	873.39	-783.14
234	97.29	29.88	67.41				

Total	-14131293.91
Count of -ve values	156
Count of +ve values	111

The omega value used for the calculations is 0.1

Appendix C - Other Research Findings

The research focused on designing an accurate software reliability estimation model. While achieving the objective, first outcome was a model which uses a curve for the dataset. This was named as curve fitting model and then was enhanced to use intelligence in selecting the optimal dataset for estimation. However, this is not directly related to the finding of the research which have been described in the chapters of this thesis. Hence, it has been added to the appendix.

Two research papers were sent to the conferences from curve fitting model and optimal selection of data. The research paper which describes optimal selection of data was published. The curve fitting model procedure is described in the following sections.

Curve fitting method for reliability estimation

Objectives of the model:

The objectives of our curve fitting model for software reliability estimation are to estimate following attributes of the software.

(1) To estimate the time to next failure at a preferred confidence level without assuming a distribution.

(2) To estimate the probability of observed failure at a given time period

(3) To estimate the optimal number of failures to be considered at each failure point.

Assumptions of the model:

(i) "new" failure data (which have not been counted previously) are only collected

(ii) Time is measured by the calendar time

In practice, correction of encountered failure takes time. Due to none attention for the correction, the same failure can be occurred repeatedly. This impacts the accuracy since the same failure is being countered twice. CPU time provides the most resolution, but in many applications only calendar time is available. Hence we have considered taking the calendar time.

Philosophy of this model is failure detection has a random fashion whereas all parametric statistical models assume that failure detection follows a distribution. Further the recent failure data are more useful than previously observed data. This is because of the software code that is changed during the testing.

Curve fitting procedure:

(1) Use of operational profile: The testing is to be done using operational profile.

(2) Collect the data: The data (time to next failure) are to be collected.

(3) Software reliability estimation based on the curve: The time to next failure data are plotted against the failure number (Figure 1). The curve is to be drawn as the best curve, i.e., smooth curve. The accuracy of the curve influence to the estimation. Hence the maximum possible scale is to be used.

The associated end point error of each measurement (which is half of the minimum error) is to be drawn as shown in the Figure 1. The sketching is to be done in such a way that maximum possible failure data points are to be crossed.

Optimal selection of failure data:

It is clear that most recent data are only useful when estimating the reliability. The method used select the optimal dataset is "optimal selection of failure data based on the standard deviation" (the research publication regarding the optimal selection of failure data can be found in the number one research paper in Appendix D) . Suppose that the optimal dataset is at "r" th failure data onwards. The selected range data curve is drawn to a maximum possible scale to enhance the accuracy of the estimation.

Suppose the desired confidence level is α.

Our hypothesis is

H0: Software will not fail at least for "ti" hours

H1: Software will fail before "ti" hours

Calculate the area of the curve in the range of fi and fi-r. Let it to be S.

Calculate the area below ti, as marked by the blue color area in the Figure 1. Suppose that area is S1.

The probability of reliability objective is not achieved is S1/S.

Let C=(S1/S)*100%

If C <=(100- α) we can conclude that our null hypothesis is true at α confidence level. Hence we need to find ti in such a way that C =S1/S = (100 – α) to be true at α confidence level. The respective ti time is the estimation of the reliability.

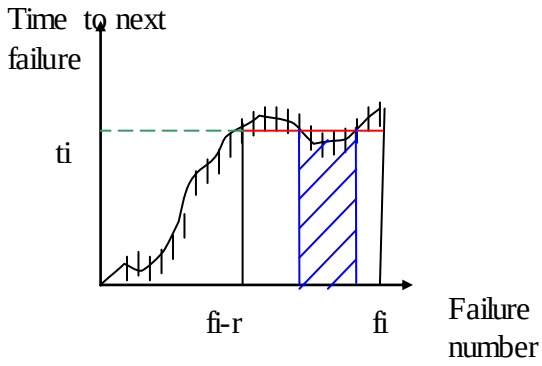


Figure C1: Time to taken to occur failures

Conclusions of the Curve Fitting Model

This model doesn't assume any statistical distribution in estimating software reliability. The parameter estimation here is also not difficult. Further this model doesn't require a through statistical knowledge. Hence this curve fitting model is easy to use by any practitioner. However, the accuracy of the model is not acceptable and need to enhance further.

Appendix D - List of Publications

List of Publications through the research is as follows. First and second papers were also presented in the particular conferences.

Research Papers

1. P.L.M Kelanibandara, G.N.Wikramanayake, and J.S. Goonethillake, "*Optimal Selection of Failure Data for Reliability Estimation Based on Standard Deviation Method*" , International Conference of Industrial and Information Systems, August 2007, Peradeniya, Sri Lanka.

2. P.L.M Kelanibandara, G.N.Wikramanayake, and J.S. Goonethillake, *"Software Reliability Estimation Based on Cubic Splines Network"* , International Conference of Computer Science and Engineering,WCE 2009 , July 2009, London, U.K.

3. P.L.M Kelanibandara, G.N.Wikramanayake, and J.S. Goonethillake, "*Software Reliability Estimation Model Based on Curve Fitting Model*" , International Journal of Information and Communication Technology emerging Regions, To be submitted, UCSC, Sri Lanka.