2009

Development of a software tool for reliability estimation

Chihui Li
West Virginia University

Development of a Software Tool for Reliability Estimation

Chihui Li

Thesis submitted to the College of Engineering and Mineral Resources at

West Virginia University

in partial fulfillment of the requirements for the degree of

Master of Science
In
Industrial Engineering

Rashpal S. Ahluwalia, Ph.D., Chair
Majid Jaraiedi, Ph.D.
Alan R. McKendall, Ph.D.

Department of Industrial and Management Systems Engineering

Morgantown, West Virginia
2009

Keywords: Failure Distributions; Statistical Tests; Reliability-Software; Hardware and System; Connection Matrix; RBD; Fault Tree; Markov.

ABSTRACT

Development of a Software Tool for Reliability Estimation

Chihui Li

This thesis presents Version 2.0 of Software Tool for Reliability Estimation (STORE 2.0). It expands on the work done by Parekh [1] by revising the algorithm for tie-set and cut-set calculation, by including fault tree reliability analysis, by analyzing state dependent system, and by integrating component and system reliability analysis.

This thesis also presents an approach to the simplification of complex systems by collapsing series and parallel components into a sub-system. The approach was illustrated on an example described by Nelson et al. [2]. The example had 16 components resulting in ten cut-sets and fifty five tie-sets. Upon simplification, the problem was reduced to one tie-set only.

STORE 2.0 integrates parameter estimation, component reliability analysis, system reliability analysis, estimation of reliability of state dependent systems, and fault tree analysis. It was verified and validated on several examples taken from the open literature. The software was developed in Visual Basic 2008 with SQL as the database.

Acknowledgments

I would like to express my sincere thanks to my committee chair Dr. Rashpal Ahluwalia for his continuous support throughout this research. He was always available to answer my questions and explicate problems that I encountered. I also appreciate the time and effort of Dr. Majid Jaraiedi and Dr. Alan R. McKendall for their helpful comments and suggestions.

I am thankful to the Industrial and Management Systems Engineering Department for its continued financial support which enabled me to complete my graduate work. Finally, I would like to thank my parents and all my friends for their support all through my studies.

Table of Contents

List of Figures

List of Tables

List of Notations

Notation	Variable Name in Software	Meaning
PDF	-	Probability Density function
CDF	-	Cumulative Distribution Function
MTTF	-	Mean Time To Failure
MTBF	-	Mean Time Between Failures
R_s	Rel_sys	System reliability
R_i	Rel_comp(i)	Reliability of component *Xi*.
R(t)	Rel_comp_at_t	Reliability for mission time *t*
f(t)	-	Unreliability probability density function
F(t)	-	Unreliability cumulative distribution function
h(t)	-	Hazard rate probability density function
H(t)	-	Hazard rate cumulative distribution function
λ	Lumbda	Scale parameter of Exponential distribution
β	Beta	Shape parameter of Weibull distribution
θ	Theta	Scale parameter of Weibull distribution
μ	Mean	Mean of Normal distribution
σ	Sigma_normal	Standard deviation of Normal distribution
μ'	Median	Median of Lognormal distribution
σ'	Sigma_lognormal	Standard deviation of Lognormal distribution
a	a	Intercept of a straight line
b	b	Slope of a straight line
r^2	r_sqr	Coefficient of determination
f	Num_of_failure	Number of failures
MLE	-	Maximum likelihood estimation
B	Bartlett_test_statistic	Bartlett's test statistic
M	Mann_test_statistic	Mann's test statistic
D	KS_test_statistic	Kolmogorov Smirnov's test statistic
CM_{ij}	CM(i, j)	Connection matrix element in row *i* and column *j*
RA_{ij}	Rel_comp_array(i, j)	Reliability array element in row *i* and column *j*
C_i	Cut_Set_Result(i)	i_{th} Cut-set
T_i	Tie_Set_Result(i)	i_{th} Tie-set
R_L	Rel_LB	Lower bound of reliability
R_U	Rel_UB	Upper bound of reliability
$\lambda_{ij}(t)$	MarkovTransitionsMatrix.rate()	Failure rate (i, j)
$\mu_{ij}(t)$	MarkovTransitionsMatrix.rate()	Repair rate (i, j)

$\Pi(t)$	State_Prob_Vector ()	Markov state probability vector
$[A]$	A_Matrix (i, j)	Transition probability matrix
r_{ij} $(i \neq j)$	A_Matrix_ Element	The rate (failure rate λ or repair rate μ) from state i to state j
α	Alpha	Significance of a model or probability of rejecting a correct model.
RBD	-	Reliability Block Diagram

CHAPTER 1

INTRODUCTION

1.1 Background

Over the years, engineering of products have become more sophisticated and complex. For example, in 1935 a farm tractor had 1200 critical parts, and in 1990 the number increased to around 2900. Today, a Boeing 747 Jumbo Jet plane is made up of approximately 4.5 million parts including fasteners [3]. Needless to say, reliability and quality of systems such as these have become more important than ever before.

In addition, failures are much more significant in both their economic and safety impacts as illustrated by the following examples [4]. In 1979, the left engine of a DC-10 broke away from the aircraft during takeoff, killing 271 people. Poor maintenance procedures and a bad design led to this crash. The Three Mile Island disaster in 1979, which resulted in a partial meltdown of a nuclear reactor, was a result of both mechanical and human error. When a backup cooling system was down for routine maintenance, air cut off the flow of cooling water to the reactor. Warning lights were hidden by maintenance tags. An emergency relief valve failed to close, causing additional water to be lost from the cooling system. Operators were either reading gauges that were not working properly or taking the wrong actions on the basis of those that were operating. The 1986 explosion of the space shuttle Challenger was a result of the failure of the rubber O-rings that were used to seal the four sections of the booster rockets. The below freezing temperatures before the launch contributed to the failure by making the rubber brittle.

The demand for complex hardware systems has increased more rapidly than the ability to design, implement, test, and maintain them; the impact of some failures can lead to high

economic damage, even loss of life. There is a definite need for reliability engineers to improve system reliability.

1.2 Reliability Engineering – Present Status

There are a number of techniques that are available for system reliability evaluation. These techniques include conditional probability analysis, network reduction, cut-set and tie-set approaches, logic diagrams, tree diagrams, connection matrix techniques, and Markov analysis. These techniques can be applied to reliability analysis of components and system. For component reliability evaluation, probability analysis is widely used. By collecting failure data and fitting a suitable distribution to it, one can compute the reliability according to the fitted distribution and mission time.

When it comes to system reliability analysis, different techniques are suitable for different systems. For example, a series-parallel system is good for the application of network reduction because it does not require intensive calculation. For a complex system, which cannot be broken down to a series-parallel system, cut-set and tie-set approaches are appropriate. Markov analysis is a suitable technique to analyze state dependent system in which the failure of one component is dependent on the failure of another component.

The main advantage of using tie-set and cut-set approach to solve complex system is that it is easy to program and most of the commercial software for reliability prediction use this approach to evaluate the reliability of complex systems.

1.3 Problem Statement

The techniques mentioned above are powerful tools for reliability analysis; however, they require large amounts of computations which may take weeks and even months to evaluate a relatively simple system by hand. In such situations, a computer solution is necessary to handle tedious and time-consuming computations.

However, when the system becomes large and complex, there are still problems with computer solutions. For example, in the software tool developed by Parekh [1], the matrix size was limited to 100 by 100, thereby allowing software to estimate reliabilities of system having no more than 100 tie-sets or cut-sets. There are no restrictions on the size of the system in the Path Tracing Algorithm by Fotuhi-Firuzabad et al. [5]; nevertheless, the number of tie-sets can go out of control since every added parallel sub-system dramatically increases the number of tie-sets. For example, a system having ten sub-systems in series and each sub-system having ten different components in parallel will have 10 billion (10^{10}) minimal tie-sets. Improved techniques are required to enhance the modeling process and to reduce the time required for the analysis of the model.

This thesis proposes an efficient approach containing a revised connection matrix and a simplification method for large simple and complex network system. The revised connection matrix has only three column “begin node”, “end node”, and “component” which is much more concise than the traditional n x n matrix. The simplification method can simplify the system by identifying and combining the series and parallel sub-system until a pure complex system is attained. After simplification the number of minimal path and cut-set are reduced, so using the simplification method before applying minimal path or cut-set technique can improve the efficiency of the identification of paths and cuts, and save time from reliability calculation.

This research also makes an improvement on the element substitution algorithm [6], which is the latest and one of the most efficient method to determine minimal cut-sets. In the improved method, whenever a potential cut-set is deleted, all levels of its successors will also be
In this way, potential cut-set array will contain less non-minimal cut-sets and its size will be decreased leading to reduce computation time.

The computational techniques implemented in commercial software do not integrate component analysis, system analysis, state dependent system, and fault tree analysis. As a result, in this research the integration of all these functions are also considered.

1.4 Objectives

The objectives of this research are to:

1) Develop an integrated approach to parameter estimation, component reliability calculation, system reliability estimation, estimate reliability of state dependent system, and fault tree analysis.

2) Develop a common database for all analysis methods.

3) Develop an algorithm to simplify the Reliability Block Diagram (RBD).

4) Develop a better representation of RBD.

CHAPTER 2

LITERATURE REVIEW

2.1 Reliability Historical Review

The history of reliability engineering is not too long, but it grew fast and has made significant progress during recent decades. O'Connor et al. [7] described the development of reliability engineering as follows.

Reliability engineering, as a separate engineering discipline, originated in the United States during the 1950s. The increasing complexity of military electronic systems was generating failure rates, which resulted in generally reduced availability and increased costs. Solid state electronics technology offered long-term hope, but conversely miniaturization was to lead to proportionately greater complexity, which offset the reliability improvements expected. The gathering pace of electronic device technology meant that the developers of new military systems were making increasing use of large numbers of new components types, involving new manufacturing processes, with the inevitable consequences of low reliability. The users of such equipment were also finding that the problems of diagnosing and repairing the new complex equipment were seriously affecting its availability for use, and the costs of spares, training and other logistics support were becoming excessive. Against this background the US Department of Defense (DoD) and the electronics industry jointly set up the Advisory Group on Reliability of Electronic Equipment (AGREE) in 1952. The AGREE report concluded that, to break out of the spiral of increasing development and ownership costs due to low reliability, disciplines must be laid down as integral activities in the development cycle for electronic equipment. The report laid particular stress on the need for new equipment to be tested for several thousand hours in

high stress cyclical environments including high and low temperatures, vibration and switching, in order to discover the majority of weak areas in a design at an early enough stage to enable them to be corrected before production commenced. Until that time, environmental tests of tens of duration had been considered adequate to prove the suitability of a design. The report also recommended that formal demonstrations of reliability, in terms of statistical confidence that a specified Mean Time Between Failure (MTBF) had been exceeded, be instituted as a condition for acceptance of equipment by the procuring agency. A large part of the report was devoted to providing detailed test plans for various levels of statistical confidence and environmental conditions.

The AGREE report was accepted by the DoD, and AGREE testing quickly became a standard procedure. Companies that invested in the expensive environmental test equipment necessary soon found that they could attain levels of reliability far higher than by traditional methods. It was evident that designers, particularly those working at the fringes of advanced technology, could not be expected to produce highly reliable equipment without it being subjected to a test regime that would show up weaknesses. Complex systems and the components used in them included too many variables and interactions for the human designer to cope with infallibly, and even the most careful design reviews and disciplines could not provide sufficient protection. Consequently, it was necessary to make the product speak for itself, by causing it to fail, and then to eliminate the weaknesses that caused the failures. The DoD reissued the AGREE report on testing as US Military Standard (MIL-STD) 781, Reliability Qualification and Production Approval Tests.

Meanwhile the revolution in electronic device technology continued, led by integrated micro-circuitry. Increased emphasis was now placed on improving the quality of devices fitted

to production equipment. Screening techniques, in which devices are temperatures cycled, vibrated, centrifuged, operated at electrical overstress and otherwise abused, were introduced in place of the traditional sampling techniques. With component populations on even single printed circuit boards becoming so large, sampling no longer provided sufficient protection against the production of defective equipment. These techniques were formalized in military standards covering the full range of electronic components. Components produced to these standards were called 'Hi-rel' components.

Engineering reliability effort in the United States developed quickly, AGREE and reliability program concepts were adopted by NASA and many other major suppliers and purchasers of high technology equipment. In 1965, the DoD issued MIL-STD-785-Reliability Programs for Systems and Equipment. This document made mandatory the integration of a program of reliability engineering activities with the traditional engineering activities of design, development and production, as it was by then realized that such an integrated program was the only way to ensure that potential reliability problems would be detected and eliminated at the earliest, and therefore the cheapest, stage in the development cycle. Much written work appeared on the cost-benefit of higher reliability, to show that effort and resources expended during early development and during production testing, plus the imposition of demonstrations of specified levels of reliability to MIL-STD-781, led to reductions in service costs which more than paid the reliability program expenditure.

The concept of life cycle costs (LCC), or whole life costs, was introduced. In the United Kingdom, Defense Standard 00-40, The Management of Reliability and Maintainability was issued in 1981. The British Standards Institution has issued BS 5760- Guide on Reliability of Systems, Equipment's and Components.

Specifications and test systems for electronic components, based upon the US Military Standards, have been developed in the United Kingdom and in continental Europe. Electronic component standards including test and quality aspects are being harmonized internationally through the International Electro-technical Commission (IEC).

2.2 Concept of Reliability

In statistics, reliability is the consistency of a set of measurements or measuring often used to describe a test. This can either be whether the measurements of the same instrument give or are likely to give the same measurement (test-retest), or in the case of more subjective instruments, such as personality or trait inventories, whether two independent assessors give similar scores (inter-rater reliability). Reliability is inversely related to random error [8].

Reliability is usually contrasted with validity, but reliability does not imply validity. One may have a highly reliable measure which is not valid. The following example may illustrate the difference between reliability and validity. If a 5 feet long table is measured 4 times, and it reads 5 feet each time, then the measurement is valid and reliable. However, if it reads 8 feet each time instead of 5, then it is not valid, but it is still reliable because the readings are consistent.

In experimental sciences, reliability is the extent to which the measurements of a test remain consistent over repeated tests of the same subject under identical conditions. An experiment is reliable if it yields consistent results of the same measure. It is unreliable if repeated measurements give different results. It can also be interpreted as the lack of random error in measurement [8].

In engineering, The IEEE (Institute of Electrical and Electronics Engineers) defines reliability as ". . . the ability of a system or component to perform its required functions under stated conditions for a specified period of time." It is often reported in terms of a probability. Evaluations of reliability involve the use of many statistical tools.

2.3 Reliability Determination from Tie-set and Cut-set

There are a number of techniques that are available for system reliability evaluation, such as conditional probability analysis, cut-set and tie-set approaches, logic diagrams, tree diagrams, connection matrix techniques. Among these techniques, minimal tie-set and cut-set approaches are one of the most popular and widely used methods for complex system.

Bellmore and Jensen [9] first investigated the search for all minimal cut-sets. Plenty of improved approaches were developed after that, such as Rai and Aggarwal [10] used Boolean algebra to obtain minimal cut-sets by inverting from the minimal cut-sets; Yeh [11] applied a revised layered-network algorithm to search for all minimal paths. Recently some improved approaches are proposed to efficiently determine minimal tie-set and cut-set.

A new minimal cut-set enumeration approach was proposed by Lin et al. [12] in 2003. As we know, components connected to the source node consist of a minimal cut-set because the failure of all these components prevents the source node from arriving at the sink node and if one of these components works, the source node has a way to reach the sink node. The basic principle of this approach was to recursively combine adjacent nodes to the source node and to consider them as a new source node (called source set) so that a new cut-set can be generated. To guarantee every generated cut-set is a minimal cut-set, after a new source set is created all redundant nodes (a node adjacent to source set and has no way to reach the sink node without

going through any node in source set) need to be checked and absorbed to the new source set. To find redundant nodes is time consuming because it means we need to determine tie-sets between every adjacent node and the sink node.

In 2004, a novel approach to determine minimal tie-sets of complex network was developed by Fotuhi-Firuzabad et al. [5]. A technique designated as the "Path Tracing was presented, which can handle both directed and undirected network. There were two steps in the algorithm, tracing all minimal tie-sets and retrieving all of them. This algorithm was easy to program, did not require limits on the size of the network, and found to be computationally efficient.

In 2005, Yeh et al. [13] provided an improved algorithm to search for all minimal cut-sets based on the approach proposed by Jasmon and Foong [14]. One property of the network is that a connected network will be broken into two connected subgraphs by removing a cut. Moreover, these two subgraphs contain the source node and the sink node, respectively. So if MCVs (the set of nodes in the subgraph containing the sink node) are known, the cut-sets can then be determined. According to this property this paper developed an algorithm to search for all MCVs and then convert MCVs to minimal cut-sets.

In 2005, Younes and Girgis [15] proposed an algorithm to search all minimal tie-sets based on a different connection matrix called link matrix whose rows represent the link (component) and columns denote different nodes. For example, component X2 is connected between node 4 and node 6, then row 2 of the link matrix will be: 0 0 0 1 0 1. Minimal tie-set can then be developed by performing union of different rows of this matrix. This algorithm did not improve the efficiency of tie-set determination, but it is good for the reliability calculation because it is

easier to conduct the union of the minimal tie-set based on this matrix. The drawback of this algorithm is that it cannot handle directed network.

An element substitution approach was used to develop multistate minimal path vectors by Ramirez-Marquez et al. [6] in 2006. Based on this Gebre and Ramirez-Marquez [16] developed an improved algorithm for general two-terminal network reliability analyses in 2007 by using forward and backward element substitution approach simultaneously. The general rationale behind element substitution is that a new cut-set can be generated by replacing the element of the known cut-set with its preceding (backward) or succeeding (forward) elements. For example, components connected with the source node consist of a cut-set. Every time when a component is substituted by it succeeding components a new cut (not necessary a minimal cut) can be generated. From these generated cut-sets, minimal cut-sets can be determined. Both backward and forward recursion can generate minimal cut-sets independently for complex network. However, when they are integrated together the number of generated cut-sets can be significantly reduced which reduces time spent on deleting non-minimal cut-set.

CHAPTER 3

FAILURE DATA ANALYSIS

3.1 Types of Failure Data

There are two types of failure data, complete data and censored data. In reality most of data are censored data because testing components are removed from the testing prior to their failure, or because the test is finished prior to all components failing. For example, components may be removed if they fail because of other failure modes. Censoring may be further categorized as follows:

Single censored data. All units have the same test time, and the test is concluded before all units have failed.

Type I censoring: Testing is terminated after fixed length of time (f^*), has elapsed.

Type II censoring: Testing is terminated after a fixed number of failures (f) have occurred. The test time is then given by t_f, the failure time of the f^{th} failure.

Multiply censored data. Test times or operating times differ among the censored (removed but operating) units. Censored units are removed at various times from the sample, or units have gone into service at different times [17].

3.2 Reliability and Failure Functions

If T is the life of a system, sub-system, or a component, then reliability (R) is defined as the probability that it will not fail during time t, where $t \leq T$. Reliability is also defined as the *probability,* at a given *confidence level*, that the system/component will perform its *intended*

function, for a specified *mission time* (*t*), *without failure*, when used for the *intended purpose* under the *intended operational conditions*. The unreliability (*F*) is the probability that a system, sub-system, or a component will fail during time *t*. Failures can occur due to wear, corrosions, defects, etc. Reliability and unreliability can vary with time, *R(t)* typically decreases with time and *F(t)* typically increases with time. At any time *t*, the sum of *R(t)* and *F(t)* is 1. A system, sub-system, or a component may be repairable or non-repairable [18].

$$R(t) + F(t) = 1 \tag{3.1}$$

Let's say we subject a large number (*N*) of components to a life test. After an arbitrary time period *t*, $N_s(t)$ components will survive and $N_f(t)$ components will fail.

$$N_s(t) + N_f(t) = N \tag{3.2}$$

Component reliability can be expressed as:

$$R(t) = N_s(t) / [N_s(t) + N_f(t)] \tag{3.3}$$

$$R(t) = 1 - N_f(t) / N = 1 - F(t) \tag{3.4}$$

The above can be expressed in mathematical terms by defining a continuous random variable T ($T \geq 0$) as life of a system, sub-system or a component. The reliability can be expressed in terms of time to failure as the probability of component failure [$P(T \geq t)$], that is, failure occurs after time *t*.

or $$R(t) = \int_t^{\infty} f(t)dt \qquad \text{where } R(t) \geq 0 \tag{3.5}$$

where $0 < t < +\infty$, $0 \leq R(t) \leq 1$, $R(0) = 1$, and $R(t)_{t \to \infty} = 0$

For a given *t*, *R(t)* is the probability that the time to failure is greater than *t*. *F(t)* is defined as the probability that failure occurs during the period 0 - *t*.

$$R(t) = 1 - F(t) = \int_t^{\infty} f(t)dt \tag{3.6}$$

or $$\frac{dR(t)}{dt} = -f(t) \quad (3.7)$$

where $0 \leq F(t) \leq 1$, $F(0) = 0$, and $F(t)_{t\to\infty} = 1$

The hazard function is defined as the limit of the failure rate as Δt approaches zero. That is, hazard function $h(t)$ is the instantaneous failure rate, it is the conditional probability that the component will fail during the interval $[t, t+\Delta t]$, given that it did not fail until time t. It is given by:

$$h(t) = lim_{\Delta t\to 0} \frac{R(t)-R(t+\Delta t)}{\Delta t\, R(t)} = \frac{1}{R(t)}\left(\frac{R(t)-R(t+\Delta t)}{\Delta t}\right) = \frac{1}{R(t)}\frac{-dR(t)}{dt} = \frac{f(t)}{R(t)} \quad (3.8)$$

The cumulative hazard function $H(t)$ is the conditional probability of failure during the interval $[0, t]$.

$$H(t) = \int_0^t h(t)dt \quad (3.9)$$

3.3 Common Failure Distributions

In this section, four common failure distributions as shown in Table 3.1 are discussed.

Table 3.1: Common Failure Distributions

	f(t)	Parameters	*F(t)*	*h(t)*	*R(t)*
Exponential	$\lambda e^{-\lambda t}$	λ – scale parameter – Failure Rate – 1/MTTF or 1/MTBF	$1 - e^{-\lambda t}$	λ	$e^{-\lambda t}$
Weibull	$\frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1} e^{-(t/\theta)^{\beta}}$	β – shape parameter θ – scale parameter	$1 - e^{-(t/\theta)^{\beta}}$	$\frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1}$	$e^{-(t/\theta)^{\beta}}$
Normal	$\frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(t-\mu)^2}{2\sigma^2}\right]}$	μ – location parameter – mean σ – shape parameter – standard deviation	$\Phi\left(\frac{t-\mu}{\sigma}\right)$	$\frac{\phi(t)}{\Phi(-t)}$	$1 - \Phi\left(\frac{t-\mu}{\sigma}\right)$
Lognormal	$\frac{1}{\sqrt{2\pi}\sigma' t} e^{\left[-\frac{1}{2\sigma'^2}\left(ln\frac{t}{\mu'}\right)^2\right]}$	μ' – location parameter – mean σ' – shape parameter – standard deviation	$\Phi\left(\frac{1}{\sigma'} ln\frac{t}{\mu'}\right)$	$\left(\frac{1}{t\sigma'}\right)\phi\left(\frac{1}{\sigma'} ln\frac{t}{\mu'}\right) / \Phi\left(\frac{-1}{\sigma'} ln\frac{t}{\mu'}\right)$	$1 - \Phi\left(\frac{t-\mu'}{\sigma'}\right)$

3.3.1 The Exponential Model

Many electronic components (transistors, resistors, integrated circuits, etc) have a constant failure rate during their useful life. The exponential model is widely used to estimate reliability of hardware components with constant failure rate [17]. The probability density function (PDF) of exponential distribution is given by

$$f(t) = \lambda e^{-\lambda t} \tag{3.10}$$

where λ is the scale parameter. It is equal to the failure rate (1/ MTBF or 1/MTTF) of the exponential model. Mean time between failures (MTBF) is the arithmetic mean (average) time between failures of a system. The MTBF is typically part of a model that assumes the failed system is immediately repaired (zero elapsed time), as a part of a renewal process. This is in contrast to the mean time to failure (MTTF), which measures average time between failures with the modeling assumption that the failed system is not repaired.

The cumulative distribution function (CDF), hazard rate function *h(t)*, and reliability function *R(t)* are given by:

$$F(t) = 1 - e^{-\lambda t} \tag{3.11}$$

$$h(t) = \lambda \tag{3.12}$$

$$R(t) = e^{-\lambda t} \tag{3.13}$$

Common Statistics of exponential distribution are

Mean	$1/\lambda$
Median	$ln(2/\lambda)$
Standard Deviation	$1/\lambda$

3.3.2 The Weibull Distribution

The Weibull distribution is an approximate model for time to failure if the item is of a type in which a large number of flaws exist [17]. The PDF is given by

$$f(t) = \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1} e^{-(t/\theta)^{\beta}} \tag{3.14}$$

where β is the shape parameter, and θ is the scale parameter.

The CDF, *h(t)*, and *R(t)* are given by

$$F(t) = 1 - e^{-(t/\theta)^{\beta}} \tag{3.15}$$

$$h(t) = \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1} \tag{3.16}$$

$$R(t) = e^{-(t/\theta)^{\beta}} \tag{3.17}$$

Common Statistics of weibull distribution are

Mean	$\theta\,\Gamma((\beta+1)/\beta)$ where Γ is gamma function
Median	$\theta\, ln(2)^{1/\beta}$
Standard Deviation	$\theta\sqrt{\Gamma\left(\frac{\beta+2}{\beta}\right) - \left(\Gamma\left(\frac{\beta+1}{\beta}\right)\right)^2}$

3.3.3 The Normal Distribution

The normal probability distribution function can be used to model failures due to fatigue or wearout. The parameters of the normal PDF are its mean (μ) and variance (σ^2). The normal is not a true reliability distribution since the random variable ranges from minus infinity to plus infinity. The positive portion of the normal does provide a reasonable approximation to the failure process. The dispersion about the mean is dependent on the value of the variance (σ^2) or standard deviation (σ). The probability density function for the normal distribution provides the well-known bell shaped curve [17]. The PDF is given by

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(t-\mu)^2}{2\sigma^2}\right]} \tag{3.18}$$

where σ is the shape parameter and standard deviation and μ is the location parameter and mean. CDF, *h(t)*, and *R(t)* are given by

$$F(t) = \Phi\left(\frac{t-\mu}{\sigma}\right) \tag{3.19}$$

$$h(t) = \frac{\phi(t)}{\Phi(-t)} \tag{3.20}$$

where Φ is the CDF of the standard normal distribution and ϕ is the PDF of the standard normal distribution.

$$R(t) = 1 - \Phi\left(\frac{t-\mu}{\sigma}\right) \tag{3.21}$$

Common Statistics of normal distribution are

Mean	μ
Median	μ
Standard Deviation	σ

3.3.4 The Lognormal Distribution

The lognormal distribution is a good model for times to failure when failures are caused by fatigue cracks. The lognormal PDF is defined for only positive values of t and is more appropriate than the normal distribution as a failure distribution [17]. If t is a random variable with a lognormal distribution, its PDF is given by

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma' t} e^{\left[-\frac{1}{2\sigma'^2}\left(ln\frac{t}{\mu'}\right)^2\right]} \quad (3.22)$$

where σ' is the shape parameter, μ' is the scale parameter and median. CDF, *h(t)*, and *R(t)* are given by

$$F(t) = \Phi\left(\frac{1}{\sigma'} ln\frac{t}{\mu'}\right) \quad (3.23)$$

$$h(t') = \left(\frac{1}{t\sigma'}\right)\phi\left(\frac{1}{\sigma'} ln\frac{t}{\mu'}\right) / \Phi\left(\frac{-1}{\sigma'} ln\frac{t}{\mu'}\right) \quad (3.24)$$

where Φ is the CDF of the standard normal distribution and ϕ is the PDF of the standard normal distribution.

$$R(t) = 1 - \Phi\left(\frac{t-\mu'}{\sigma'}\right) \quad (3.25)$$

Common Statistics of lognormal distribution are

Mean	$exp\ (0.5\sigma'^2)$
Median	μ'
Standard Deviation	$\sqrt{e^{\sigma'^2}(e^{\sigma'^2} - 1)}$

3.4 Candidate Distribution Identification

After collection of failure or repair data, there are three steps for the fitting of a theoretical distribution which are 1) identifying candidate distributions, 2) estimating parameters, and 3) performing a goodness-of-fit test. In the first step, least square is used to identify candidate distribution.

3.4.1 Least Square Fitting

Least Square Fitting is a mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve (see Figure 3.1). The sum of the *squares* of the offsets is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity. Least squares problems fall into two categories, linear and non-linear. The linear one is discussed and used in this research.

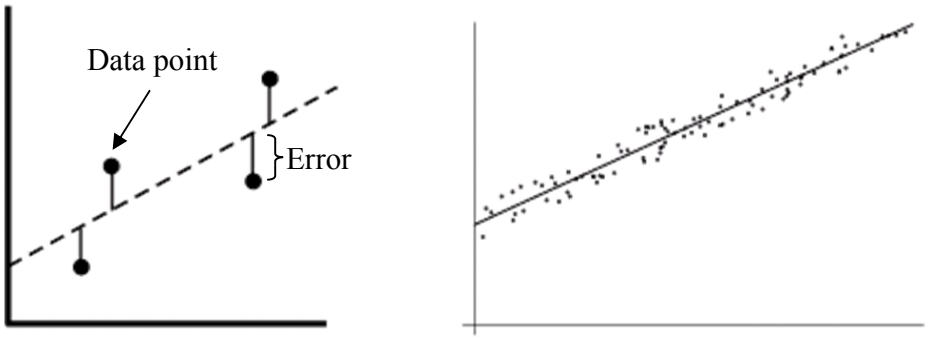


Figure 3.1: Least Square Fitting

A regression model is a linear one when the model comprises a linear combination of the parameters, i.e.

$$y_i = \sum_{j=1}^{n} x_{ij}\beta_j \quad i = 1, 2, \ldots, m \qquad (3.26)$$

has m linear equations in n unknown coefficients, $\beta_1, \beta_2, \ldots, \beta_n$, with $m > n$, written in matrix form as

$$\mathrm{Y} = \mathrm{X}\beta \tag{3.27}$$

where

$$X = \begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mm} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can then see that in that case the least square estimate β is given by

$$\hat{\beta} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{Y} \tag{3.28}$$

For a special case ($m = 2$), $y_i = a + bx_i$, using this method, a straight line with intercept a and slope b can be fixed by the following formulas.

$$b = \frac{\sum_{i=1}^{n} x_i y_i - \bar{x}\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \tag{3.29}$$

where

b is slope

x is independent variable

y is dependent variable

n is the number of data points

$$a = \bar{y} - b\bar{x} \tag{3.30}$$

where

a is intercept

$\bar{x}$ is average of x

$\bar{y}$ is average of y

The coefficient of determination, r^2, can be computed as

$$r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - a - bx_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (3.31)$$

The *coefficient of determination* measures the strength of the fit of the regression curve and can be interpreted as the proportion of the variation in the *y*'s explained by the *x* variables. The square root, *r*, here is called the *index of fit*. It will have a value between -1 and 1; a value $|r|$ of 1 is a perfect fit. However, for example, values of *r* equal to 0.3 and 0.6 only mean that we have two positive correlations, one somewhat stronger than the other. It is wrong to conclude that r = 0.6 indicates a linear relationship twice as good as that indicated by the value r = 0.3 [19].

3.4.2 Least Square Approach for Common Distributions

To make sure that the function are linear in parameters, transformations for *x* and *y* are necessary and listed in Table 3.2.

Table 3.2: Least Square Approach for Common Distributions

	x_i	y_i	**Parameters**
Exponential	t_i	$ln\left[\frac{1}{1-F(t_i)}\right]$	$\lambda = b$
Weibull	$ln\, t_i$	$ln\left[ln\left(\frac{1}{1-F(t_i)}\right)\right]$	$\beta = b$ $\theta = exp(-a/\beta)$
Normal	t_i	$F(t_i)$	$\sigma = 1/b$ $\mu = -a/b$
Lognormal	$ln\, t_i$	$F(t_i)$	$\sigma' = 1/b$ $\mu' = exp(-\sigma' a)$

where $F(t_i) = (i - 0.3) / (n + 0.4)$. This formula is often used as an approximation of the median positions.

The exponential CDF is $F(t) = 1 - e^{-\lambda t}$, or $1 - F(t) = e^{-\lambda t}$. Then taking the natural logarithm of both sides, $-ln\big(1 - F(t)\big) = ln\left(\frac{1}{1-F(t)}\right) = \lambda t$. So $x_i = t_i$ and

$y_i = ln\left[ln\left(\frac{1}{1-F(t_i)}\right)\right]$ are used for the transformation to keep dependent variable and independent variable linear.

From the Weibull cumulative distribution function, $F(t) = 1 - e^{-(t/\theta)^{\beta}}$, we get $ln\left[ln\left(\frac{1}{1-F(t)}\right)\right] = \beta\, ln\, t - \beta\, ln\, \theta$. Hence, the transformation will be $x_i = ln\, t_i$ and $y_i = ln\left[ln\left(\frac{1}{1-F(t_i)}\right)\right]$.

For normal distribution, $F(t) = \Phi\left(\frac{t-\mu}{\sigma}\right) = \Phi(z)$, the inverse function can be written as $z_i = \Phi^{-1}[F(t)] = \frac{t_i-\mu}{\sigma} = \frac{t_i}{\sigma} - \frac{\mu}{\sigma}$ which is linear in *t*. A least–squares fit is obtained by setting $x_i = t_i$ and $y_i = F(t_i)$.

Since lognormal distribution, $F(t) = \Phi\left(\frac{1}{\sigma'} ln \frac{t}{\mu'}\right) = \Phi(z)$, then $z_i = \Phi^{-1}[F(t)] = \frac{1}{\sigma} ln\, t - \frac{1}{\sigma} ln\, \mu'$. So $x_i = ln\, t_i$ and $y_i = F(t_i)$

3.5 Distribution Parameter Estimation

The previous discussion centered on the identification of candidate distributions for describing a failure or repair process. Once one or more distributions have been identified, the next step is to estimate the parameters of the distribution. Until the parameters are determined, the distribution is not completely specified. Although probability plots and least-squares fitting of the data provide a means of estimation of the parameters of the distributions, they are not necessarily the preferred, or "best" estimates of the distribution parameters. This is especially true in certain goodness-of-fit tests that are based on the maximum likelihood estimator (MLE) for the distribution parameters.

3.5.1 Maximum likelihood estimation (MLE)

This section presents the theory that underlies maximum likelihood estimation for complete data. If x is a continuous random variable with *PDF:*

$$f(x; \theta_1, \theta_2, \cdots, \theta_k) \tag{3.32}$$

where $\theta_1, \theta_2, \ldots, \theta_k$ are k unknown constant parameters which need to be estimated, conduct an experiment and obtain N independent observations, $x_1, x_2, \ldots, x_N$. Then the likelihood function is given by the following product:

$$L(x_1, x_2, \cdots, x_N | \theta_1, \theta_2, \cdots, \theta_k) = L = \prod_{i=1}^{N} f(x_i; \theta_1, \theta_2, \cdots, \theta_k) \tag{3.33}$$

The logarithmic likelihood function is given by:

$$\Lambda = ln\, L = \sum_{i=1}^{N} ln\, f(x_i; \theta_1, \theta_2, \cdots, \theta_k) \tag{3.34}$$

The maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are obtained by maximizing L or Λ. By maximizing Λ, the maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are the simultaneous solutions of k equations such that:

$$\frac{\partial(\Lambda)}{\partial \theta_j} = 0, \quad j = 1, 2, \cdots, k \tag{3.35}$$

Even though it is common practice to plot the MLE solutions using median ranks (points are plotted according to median ranks and the line according to the MLE solutions), this is not completely accurate. As it can be seen from the equations above, the MLE method is independent of any kind of ranks or plotting methods. For this reason, many times the MLE solution appears not to track the data on the probability plot. This is perfectly acceptable since the two methods are independent of each other and in no way suggests that the solution is wrong [17].

3.5.2 MLE Approach for Common Distributions

By using MLE, parameters can be obtained by the following formulas [17]:

Table 3.3: MLE Approach for Common Distributions

	Parameters
Exponential (λ)	$\lambda = f/T$
Weibull (θ, β)	$\frac{\sum_{i=1}^{f} t_i^{\beta} \ln t_i + (n-f) t_s^{\beta} \ln t_s}{\sum_{i=1}^{f} t_i^{\beta} + (n-f) t_s^{\beta}} - \frac{1}{\beta} - \frac{\sum_{i=1}^{f} \ln t_i}{f} = 0$ $Solve\ for\ \beta$ $\theta = \left\{ \frac{1}{f} \left[\sum_{i=1}^{f} t_i^{\beta} + (n-f) t_s^{\beta} \right] \right\}^{1/\beta}$
Normal (μ, σ)	$\mu = \bar{x}$ $\sigma^2 = \frac{(n-1)s^2}{n}$
Lognormal (μ', σ')	$\mu = \sum_{i=1}^{n} \frac{\ln t_i}{n}$ $\mu' = e^{\mu}$ $\sigma' = \sqrt{\frac{\sum_{i=1}^{n} (\ln t_i - \mu')^2}{n}}$

f = the number of failed items

n = the number of tested items

T = the sum of failed time t_i

For Weibull distribution MLE,

$$g(\beta) = \frac{\sum_{i=1}^{f} t_i^{\beta} \ln t_i + (n-f) t_S^{\beta} \ln t_S}{\sum_{i=1}^{f} t_i^{\beta} + (n-f) t_S^{\beta}} - \frac{1}{\beta} - \frac{\sum_{i=1}^{f} \ln t_i}{f} = 0 \quad (3.36)$$

The newton-Raphson method for solving a nonlinear equation may be used. This requires solving for β iteratively using.

$$\beta_{j+1} = \beta_j - \frac{g(\beta_j)}{g'(\beta_j)} \quad (3.37)$$

$$g'(x) = \frac{dg(x)}{dx} \quad (3.38)$$

3.6 Goodness-of-fit Tests

The final step in the selection of a theoretical distribution is to perform a statistical test for goodness of fit. Such a test compares a null hypothesis (H_0) with an alternative hypothesis (H_1) having the following form:

H_0: The failure times come from the specified distribution.

H_1: The failure times do not come from the specified distribution.

The test consists of computing a statistic based on the sample of failure times. This statistic is then compared with a critical value. The critical value depends on the level of significance of the test and the sample size [17].

There are two types of goodness-of-fit tests: general tests and specific tests. A general test is applicable to fitting more than one theoretical distribution, and a specific test is tailored to a single distribution. When available, specific tests will be more powerful (have a higher probability of correctly rejecting a distribution) than general tests.

Chi-square test is a general test which can test Exponential distribution, Weibull distribution, Normal distribution, and Lognormal distribution. However, the data for this test

must be grouped into classes. Another disadvantage is that it is valid for large sample size only; the sample size of each group should not be less than 5.

3.6.1 Goodness-of-fit Tests for Common Distributions

For this research, three goodness-of-fit tests are used. These three tests are designed for specific distributions. For instance, Kolmogorov-Smirnov test is designed for normal and lognormal distributions, Bartlett's test is designed for exponential distribution, and Mann's test is designed for the Weibull distribution. These specific tests are more powerful than the general test. For example, Monte Carlo power comparisons of the test based on Mann's test and analogs of the Kolmorov-Smirnov, the Kuiper, and the standard version, as well as a weighted version of the Cramer-von Mises tests, revealed that the Mann's test is most powerful against the alternatives studied [20].

Goodness-of-fit tests of exponential distribution, Weibull distribution, normal and lognormal distributions can be done using the criteria in Table 3.4.

Table 3.4: Goodness-of-fit Tests for Common Distributions

	Formulas	**Accept H_0 If**
Bartlett's Test (B) for Exponential distribution	$B = \frac{2f\left[ln\left((1/f)\sum_{i=1}^{f} t_i\right) - (1/f)\sum_{i=1}^{f} ln\, t_i\right]}{1+(f+1)/(6f)}$	$\chi^2_{1-\alpha/2,f-1} < B < \chi^2_{\alpha/2,f-1}$
Mann's Test (M) for Weibull distribution	$M = \frac{k_1 \sum_{i=k_1+1}^{f-1}[(ln\, t_{i+1} - ln\, t_i)/M_i]}{k_2 \sum_{i=1}^{k_1}[(ln\, t_{i+1} - ln\, t_i)/M_i]}$ $k_1 = \left\lfloor \frac{f}{2} \right\rfloor$ $k_2 = \left\lfloor \frac{f-1}{2} \right\rfloor$ $M_i = Z_{i+1} - Z_i$ $Z_i = ln\left[-ln\left(1 - \frac{i-0.5}{n+0.25}\right)\right]$	$M < F_{crit,\alpha,2k_2,2k_1}$
Kolmogorov-Smirnov Test (D) for Normal/ Lognormal distribution	$\bar{t} = \sum_{i=1}^{n} \frac{t_i}{n}$ $\sigma^2 = \frac{\sum_{i=1}^{n}(t_i - \bar{t})^2}{n-1}$ $D_1 = \max_{1 \le i \le n}\left\{\Phi\left(\frac{t_i - \bar{t}}{\sigma'}\right) - \frac{i-1}{n}\right\}$ $D_2 = \max_{1 \le i \le n}\left\{\frac{i}{n} - \Phi\left(\frac{t_i - \bar{t}}{\sigma'}\right)\right\}$	$D_n < D_{crit}$

CHAPTER 4

STATE INDEPENDENT SYSTEMS

4.1 State Independent Systems

To determine the reliability of a large system, it needs to be subdivided into smaller subsystems and components whose individual reliability factors are known or can be easily determined. Depending on the manner in which these subsystems and components are connected to constitute the given system, the combination rules of probability can be applied to obtain system reliability. From the point of view of interconnection of the subsystems, a system may be classified as series, parallel, series-parallel, or a complex system [21].

Finding the exact reliability for series and parallel networks is quite straightforward and is described briefly in next two sections. A series-parallel network consists of distinct series and parallel components within the given system. For such a system the reliability analysis is performed in steps as described in section 4.4 . K/N system is a special case of parallel system. It is discussed in section 4.5 . A complex system is one, which cannot be completely decomposed into independent sections of series and/or parallel sub-systems. Reliability analysis for such systems is significantly different from a series-parallel network. As a result, other approaches such as tie-set and cut-set are necessary to solve this kind of problem.

4.2 Series Systems

Consider a simple system consisting of *n* software or hardware components connected in series as shown in Figure 4.1.

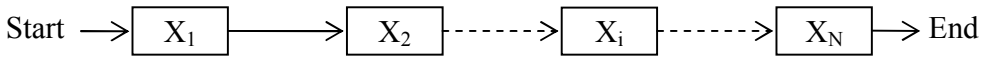


Figure 4.1: Series Systems

The reliability of the system, R_S, is given by:

$$R_s = R_1 * R_2 \ldots * R_i \ldots * R_N \qquad (4.1)$$

For example, if we have three components with known reliability values as shown in Figure 4.2, the system reliability will be:

$Rs = R_1 * R_2 * R_3 = (.99) * (.95) * (.98) = .92169$

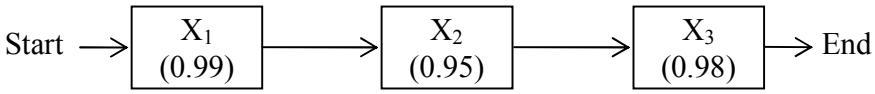


Figure 4.2: Series Systems Example

4.3 Parallel Systems

A parallel system is shown in Figure 4.3.

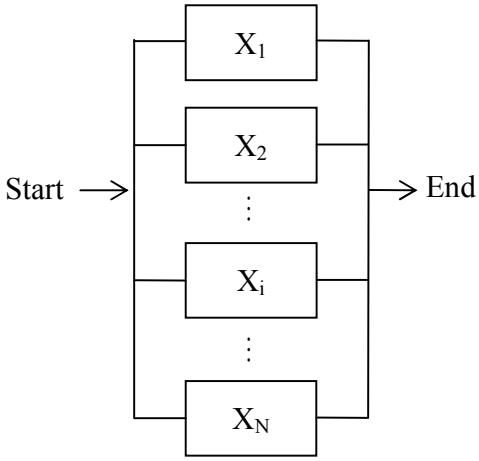


Figure 4.3: Parallel Systems

The reliability of the parallel system, R_S, is given by:

$$R_s = 1 - [(1 - R_1)*(1 - R_2)... *(1 - R_i) ...*(1 - R_N)] \quad (4.2)$$

For example, reliability of system described in Figure 4.4:

$Rs = 1 - [(1 - R_1)\ (1 - R_2)\ (1 - R_3)]$

$= 1 - [(0.05)\ (0.2)\ (0.3)]$

$= 0.997$

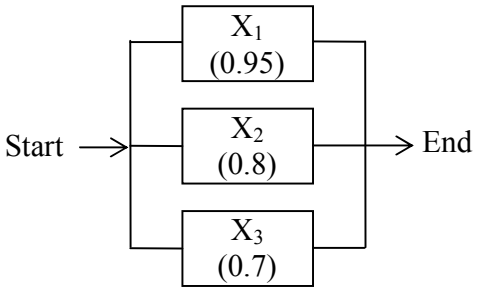


Figure 4.4: Parallel Systems Example

4.4 Series-Parallel Systems

An example of a series-parallel system is shown in Figure 4.5.

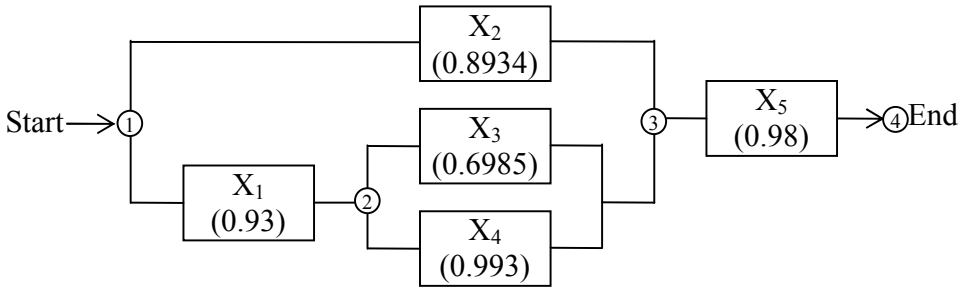


Figure 4.5: Series-Parallel Systems

Reliability analysis of such system is performed in steps. In each step the independent series and parallel structures are identified and solved separately. As a result of each step, the size of the system reduces until it becomes a simple series or parallel system. In the system shown in Figure 4.5, components X_3 and X_4 are in parallel and thus form a subsystem identified as subsystem X_{34} with reliability of R_{34}. The reliability of R_{34} is calculated as follow.

$$R_{34} = 1 - [(1 - R_3)(1 - R_4)] = .99789$$

See Figure 4.6 for the series-parallel system in Figure 4.5 with subsystem X_{34}.

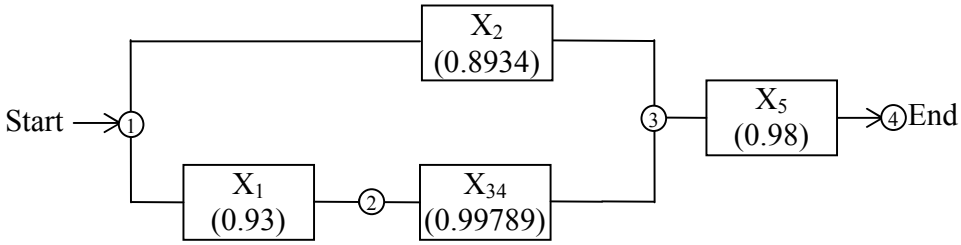


Figure 4.6: Series-Parallel System with Subsystem X_{34}

Component X_1 and subsystem X_{34} are in series and compose of another subsystem identified as $X_{1,34}$, with reliability of $R_{1,34}$ as shown in Figure 4.7, and $R_{1,34}$ is calculated as follow.

$$R_{1,34} = R_1 * R_{34} = .928037$$

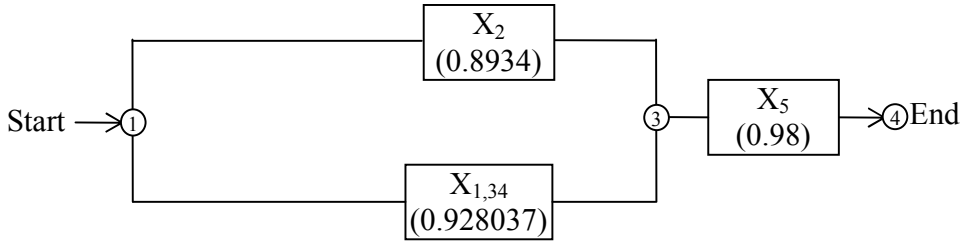


Figure 4.7: Series-Parallel System with Subsystem $X_{1,34}$

Then combine components X_2 and subsystem $X_{1,34}$ which are in parallel and thus form a subsystem identified as subsystem $X_{2,1,34}$ with reliability of $R_{2,1,34}$ as shown in Figure 4.8, and $R_{2,1,34}$ is calculated as follow.

$$R_{2,1,34} = 1 - [(1 - R_2)\,(1 - R_{1,34})] = .992329$$

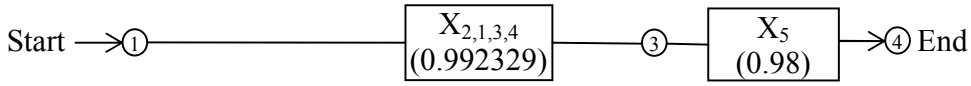


Figure 4.8: Series-Parallel System with Subsystem $X_{2,1,3,4}$

Continue to reduce the system until the whole systems' reliability is obtained:

$$R_s = R_5 * R_{2,1,34} = .972482$$

4.5 K/N Systems

The K-out-of-N system has a total of N components connected in parallel, and at least K components must operate for the system to function as shown in Figure 4.9.

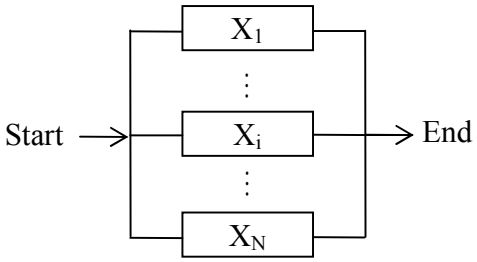


Figure 4.9: K/N Systems

The reliability of the system, R_S, is given by:

$$R_S = \sum_{j=k}^{N} \left[\left(\frac{N!}{j!(N-j)!} \right) * R^j * (1-R)^{N-j} \right] \quad (4.3)$$

where R is reliability of X_i (all components have the same reliability)

For example, an aircraft has four independent engines (Figure 4.10). Three out of the four engines must operate in order for the aircraft to fly. If each engine has reliability of .97, what is the aircraft reliability [18]?

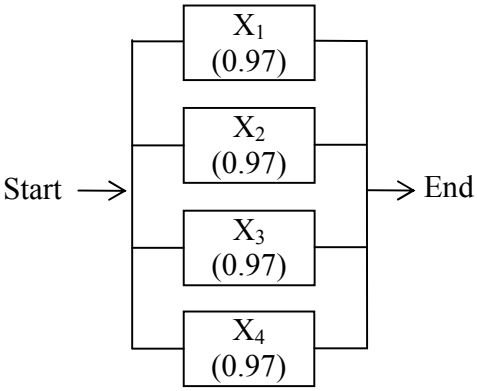


Figure 4.10: K/N Systems Example

In this problem, we know *k*=3, *N*=4, and *R*=0.97, so

$$R_s = \sum_{j=k}^{N} \left(\frac{N!}{j!(N-j)!} \right) * R^j * (1-R)^{N-j}$$

$$R_S = \sum_{j=3}^{4} \left(\frac{4!}{j!(4-3)!}\right) *. 97^j * (.03)^{4-j}$$

$$= 4 *. 97^3 * .03 +. 97^4$$

$$= .9948$$

4.6 Complex Systems

4.6.1 Existing Techniques for Complex System Reliability Evaluation

There are a number of techniques available to analyze a complex system. Some important approaches are: a) Conditional probability, b) Cut-set method, c) Tie-set method, d) Event Tree, and e) Fault trees.

The conditional approach splits the given system into subsystems, until they are simple series/parallel networks. The subsystems are then combined using the conditional probability method. This technique gives an accurate answer, but becomes tedious for very complex systems. It is also very difficult to implement in a computer program [22].

The cut-set method is a set of system components which, if the elements all fail, will result in system failure. A minimal cut-set is one in which all the components *must* fail in order for the system to fail and if any one element does not fail then the system does not fail [23].

The tie-set method is a set of system components whose functioning ensures that the system functions. A minimal tie-set (path) is one in which all the components within the set must function for the system to function, and if any one element does not function then the system is not guaranteed to function. A tie-set will fail if just one component of the tie-set fails and all the system tie-sets must fail for the system to fail [2].

An event tree is a pictorial representation of all the events which can occur in a system. The cut-sets and tie-sets can be developed from the event trees [23].

A useful tool in performing a system safety analysis is fault tree analysis. A fault tree analysis is a graphical design technique that provides an alternative to reliability block diagrams in several respects. It is a top-down, deductive analysis structured in terms of events rather than components. The perspective is on faults rather than reliability [17].

4.6.2 Revised Connection Matrix and System Simplification

4.6.2.1 Introduction

The first step of most techniques is to build a connection matrix which includes information on how the components are connected in the network. Most of the time nodes are added to make it easier to build the matrix and represent the direction of the branch. One of the good connection matrices for small size network is a n x n matrix which is applied in many
In such connection matrix, the row number of each component denotes the “begin” node, and the column number denotes the “end” node. Table 4.1 is an example of the connection matrix of bridge-type network shown in Figure 4.11 [5]. In this table component “1” means the begin node and the end node are the same node or the begin node and end node are connected without components between them.

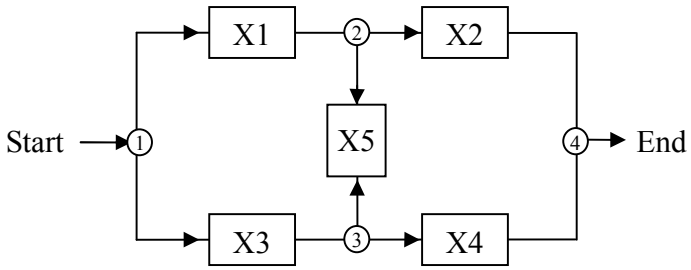


Figure 4.11: Bridge-type Network

Table 4.1: Connection Matrix for Example 1

		End Nodes			
		1	**2**	**3**	**4**
Begin Nodes	**1**	1	X1	X3	0
	2	0	1	X5	X2
	3	0	X5	1	X4
	4	0	0	0	1

Based on the connection matrix, a number of techniques are available for system reliability analysis. Minimal path (minimal tie-set) is one of the most important and widely used approaches. The system shown in Figure 4.11 has four minimal tie-set (Ti) as shown below:

T1: X1, X2

T2: X3, X4

T3: X1, X5, X4

T4: X3, X5, X2

System reliability can then be determinate from the minimal paths or the sets obtained. If each component has a reliability of 0.9, the system reliability of the above bridge-type network is 0.97848 obtained by the following calculation.

First calculate the reliability of all tie-set combinations. The result is shown in Table 4.2. Base on Table 4.2, we attain the values for the following terms:

$$\sum_{i=1}^{4} P(T_i) = P(T_1) + P(T_2) + P(T_3) + P(T_4)$$

$$= 0.81 + 0.81 + 0.729 + 0.729 = 3.078$$

$$\sum_{i=1}^{3}\sum_{j=i+1}^{4} P(T_i * T_j) = P(T_1 * T_2) + P(T_1 * T_3) + P(T_1 * T_4) + P(T_2 * T_3) + P(T_2 * T_4)$$

$$+P(T_3 * T_4)$$

$$= 0.6561 * 5 + 0.59049$$

$$= 3.87099$$

$$\sum_{i=1}^{2}\sum_{j=i+1}^{3}\sum_{k=j+1}^{4} P(T_i * T_j * T_k) = P(T_1 * T_2 * T_3) + P(T_1 * T_2 * T_4) + P(T_1 * T_3 * T_4)$$

$$+P(T_2 * T_3 * T_4)$$

$$= 4 * (0.59049)$$

$$\sum_{i=1}^{1}\sum_{j=i+1}^{2}\sum_{k=j+1}^{3}\sum_{l=k+1}^{4} P(T_i * T_j * T_k * T_l) = P(T_1 * T_2 * T_3 * T_4)$$

$$= 0.59049$$

The system reliability is obtained by the following equation:

$$R_s = P(T_1 + T_2 + T_3 + T_4)$$

$$= \sum_{i=1}^{4} P(T_i) - \sum_{i=1}^{3}\sum_{j=i+1}^{4} P(T_i * T_j)$$

$$+ \sum_{i=1}^{2}\sum_{j=i+1}^{3}\sum_{k=j+1}^{4} P(T_i * T_j * T_k)$$

$$- \sum_{i=1}^{1}\sum_{j=i+1}^{2}\sum_{k=j+1}^{3}\sum_{l=k+1}^{4} P(T_i * T_j * T_k * T_l)$$

$$= 3.078 - 3.87099 + 4(0.59049) - 0.59049$$

$$= 0.97848$$

Table 4.2: Tie-Set Reliability

Probability of Tie-set Combination	Break down Tie-set to components	Apply Boolean Algebra	Reliability	Value
$P(T_1)$	$P\{(X_1*X_2)\}$	$P\{X_1*X_2\}$	R_1*R_2	0.81
$P(T_2)$	$P\{(X_3*X_4)\}$	$P\{X_3*X_4\}$	R_3*R_4	0.81
$P(T_3)$	$P\{(X_1*X_5*X_4)\}$	$P\{X_1*X_5*X_4\}$	$R_1*R_5*R_4$	0.729
$P(T_4)$	$P\{(X_3*X_5*X_2)\}$	$P\{X_3*X_5*X_2\}$	$R_3*R_5*R_2$	0.729
$P(T_1*T_2)$	$P\{(X_1*X_2)*(X_3*X_4)\}$	$P\{X_1*X_2*X_3*X_4\}$	$R_1*R_2*R_3*R_4$	0.6561
$P(T_1*T_3)$	$P\{(X_1*X_2)*(X_1*X_5*X_4)\}$	$P\{X_1*X_2*X_4*X_5\}$	$R_1*R_2*R_4*R_5$	0.6561
$P(T_1*T_4)$	$P\{(X_1*X_2)*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_5\}$	$R_1*R_2*R_3*R_5$	0.6561
$P(T_2*T_3)$	$P\{ (X_3*X_4)*(X_1*X_5*X_4)\}$	$P\{X_1*X_3*X_4*X_5\}$	$R_1*R_3*R_4*R_5$	0.6561
$P(T_2*T_4)$	$P\{ (X_3*X_4)*(X_3*X_5*X_2)\}$	$P\{X_2*X_3*X_4*X_5\}$	$R_2*R_3*R_4*R_5$	0.6561
$P(T_3*T_4)$	$P\{ (X_1*X_5*X_4)*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049
$P(T_1*T_2*T_3)$	$P\{(X_1*X_2)*(X_3*X_4)$ $*(X_1*X_5*X_4)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049
$P(T_1*T_2*T_4)$	$P\{(X_1*X_2)*(X_3*X_4)$ $*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049
$P(T_1*T_3*T_4)$	$P\{ (X_1*X_2)* (X_1*X_5*X_4)*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049
$P(T_2*T_3*T_4)$	$P\{ (X_3*X_4)* (X_1*X_5*X_4)*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049
$P(T_1*T_2*T_3*T_4)$	$P\{ (X_1*X_2)* (X_3*X_4)*$ $(X_1*X_5*X_4)*(X_3*X_5*X_2)\}$	$P\{X_1*X_2*X_3*X_4*X_5\}$	$R_1*R_2*R_3*R_4*R_5$	0.59049

A slightly more complex system with eleven components and nine nodes (example 2) is shown in Figure 4.12. Table 4.3 shows the corresponding connection matrix.

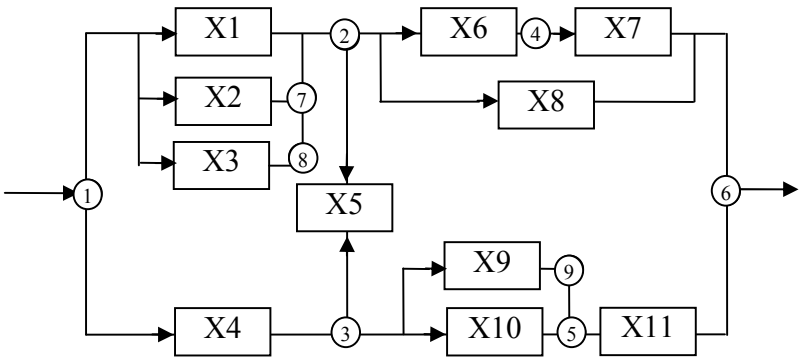


Figure 4.12: Complex System for Example 2

Table 4.3: Connection Matrix for Example 2

Begin Nodes \ End Nodes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	1	X1	X4	0	0	0	X2	X3	0
2	0	1	X5	X6	0	X8	1	1	0
3	0	X5	1	0	X10	0	0	0	X9
4	0	0	0	1	0	X7	0	0	0
5	0	0	0	0	1	X11	0	0	1
6	0	0	0	0	0	1	0	0	0
7	0	1	0	0	0	0	1	1	0
8	0	1	0	0	0	0	1	1	0
9	0	0	0	0	1	0	0	0	1

Minimal paths for the example shown in Figure 4.12 can be determined visually and the result is listed below:

Table 4.4: Tie-set Result for Example 2

NO.	Tie-set	NO.	Tie-set
T1	X1, X6, X7	T9	X3, X5, X9, X11
T2	X2, X6, X7	T10	X1, X5, X10, X11
T3	X3, X6, X7	T11	X2, X5, X10, X11
T4	X1, X8	T12	X3, X5, X10, X11
T5	X2, X8	T13	X4, X9, X11
T6	X3, X8	T14	X4, X10, X11
T7	X1, X5, X9, X11	T15	X4, X5, X6, X7
T8	X2, X5, X9, X11	T16	X4, X5, X8

To determine the system reliability, the method described for bridge-type network can be applied to this case. Assuming reliability of each component is 0.9, the whole system reliability is 0.99765, computed according to the minimal paths.

4.6.2.2 Revised Connection Matrix

The connection matrix shown in Table 4.1 is not very efficient for systems containing parallel sub-systems. If there are parallel sub-systems in the network, additional nodes have to be added to ensure that connection matrix works, because only one or zero component can exist between two nodes in the matrix. For example, in Figure 4.11, if component X11 is added between node 1 and node 2 (in parallel with Component X1), there will be no place for X11 in the connection matrix (Table 4.1) because cell (1,2) is already occupied by X1. This is the reason why additional nodes 7, 8, and 9 are added in Figure 4.12. While additional nodes are added in the system to overcome the parallel problem, the size of connection matrix increases accordingly.

Moreover, when the system becomes large, the inefficiency of this connection matrix is obvious since only the cells having component names are valuable. For example, the connection matrix (Table 4.3) in example 2 is sparsely populated with 69 out of 81 cells containing "1" or "0" which provide no useful information.

A revised connection matrix in the form of

Table 4.5 is proposed to overcome the problems highlighted above. The revised representation is better suited for large systems.

Table 4.5: Revised Connection Matrix for Example 1

Begin Node	**End Node**	**Component**
1	2	X1
1	3	X3
2	3	X5
3	2	X5
2	4	X2
3	4	X4

Applying this revised connection matrix to example 2 (Figure 4.12), nodes 7, 8, and 9 can be removed (Figure 4.13) and a concise matrix Table 4.6 can be attained having only 36 cells instead of 81 cells.

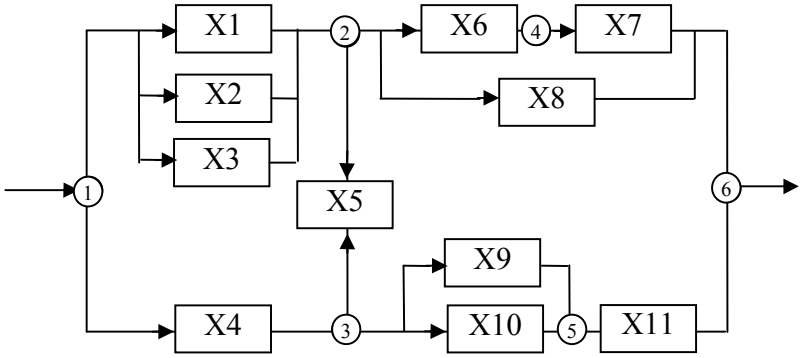


Figure 4.13: Revised Complex System for Example 2

Table 4.6: Revised Connection Matrix (Connection Array) for Example 2

Begin Node	End Node	Component
1	2	X1
1	2	X2
1	2	X3
1	3	X4
2	3	X5
3	2	X5
2	4	X6
2	6	X8
4	6	X7
3	5	X9
3	5	X10
5	6	X11

The same minimal paths can be obtained from the revised connection matrix.

4.6.2.3 System Simplification

The identification of minimal path is difficult for larger and more complex system, especially for complex systems containing a large number of parallel sub-systems. For example, a system having ten sub-systems in series and each sub-system having ten different components

in parallel will have 10 billion (10^{10}) minimal paths. However, this is not an exceptional case. In real world, most large systems have thousands of components, and most of the sub-systems in that system are in series and parallel. Simply using minimal path and minimal cut-set techniques to solve those large size network problems will lead to waste of time and resources.

An efficient way to handle this problem is to simplify the system before applying minimal path approach. As we know, computation of reliability with series/parallel laws is much more efficient than that with minimal path and minimal cut-set approach. In this section, based on the revised connection matrix introduced above, a method to simplify all series or parallel systems in the system until a pure complex system is obtained is described below.

Series Components Identification

Two components (sub-systems) are in series when (1) a node is only connected to these two components (sub-systems) and (2) the node has input from one of those components (sub-systems) and output to another component (sub-systems). So if a node satisfies the above criteria and then we know that components between them are in series.

For example, in Figure 4.12, node 2 has six components connected, so it is not a series node. Node 1 has output to four components but has no input from other components, so node 1 is not satisfied. Node 4 is a series node because it has X6 as input and X7 as output and no more components connect to it. In Figure 4.12 we have only component X6 and X7 in series (sub-system will be discussed later).

Components (sub-systems) are in parallel when they have the same input nodes and the same output nodes. For example, in Figure 4.12, component X1, X2, and X3 are in parallel

because they share the same input node 1 and the same output node 2. So identify all parallel components by checking their connected nodes.

Simplification Algorithm

Simplification algorithm pseudo-code

Initialize:

CM_Rows =the number of rows of connection matrix

Pre_CM_Rows = CM_Rows+1

CM_{ij} = Connection matrix element in row i and column j

n_k = the k^{th} node

nn1 = the number of n_k in CM_{i1} (first column of connection matrix)

nn2 = the number of n_k in CM_{i2} (second column of connection matrix)

RA_{ij} = Reliability array element in row i and column j

R(x) = Reliability of component x

Simplification:

While Pre_CM_Rows > CM_Rows

Pre_CM_Rows = CM_Rows

(Series simplification)

For each n_k

If nn1 = 1 and nn2 = 1 and $CM_{a1} = CM_{b2} = n_k$

$CM_{b2} = CM_{a2}$

$X = CM_{b3}$

$CM_{b3} = CM_{b3}$ + "*" + CM_{a3} (create a new component name)

$R(CM_{b3}) = R(X) * R(CM_{a3})$

Add component CM_{b3} and its reliability $R(CM_{b3})$ to Reliability array

Delete row a of CM

(Parallel simplification)

For any row of CM

If $CM_{c1} = CM_{d1}$ and $CM_{c2} = CM_{d2}$

$Y = CM_{c3}$

$CM_{c3} = CM_{c3}$ + "+" + CM_{d3} (create a new component name)

$R(CM_{c3}) = 1- (1- R(Y)) * (1- R(CM_{d3}))$

Add component CM_{c3} and its reliability $R(CM_{c3})$ to Reliability array

Delete row d of CM

CM_Rows =the number of rows of connection matrix

The algorithm is illustrated using the network shown in Figure 4.12.

1. Build an array (denoted as **connection matrix**) for connection matrix. The result of this step is shown in Table 4.6.

2. Build an array (denoted as reliability array) with two columns. The first column is used to store component name and the second one for reliability value as shown in Table 4.7.

Table 4.7: Reliability Array

Component	Reliability
X1	0.9
X2	0.9
X3	0.9
X4	0.9
X5	0.9
X6	0.9
X7	0.9
X8	0.9
X9	0.9
X10	0.9
X11	0.9

3. Set up an array (denoted as **nodes array**) for storing all nodes and put them into it as shown in Table 4.8.

Table 4.8: Nodes Array

Nodes
1
2
3
4
5
6
7
8
9

4. Get a node (denoted as **current node**) from nodes array at a time. If the first column (begin node) of the connection matrix has only one current node and second column (end node) has exact one too (it means two components are in series), go to step 5 to combine these two

components. Get the next node in the nodes array and do it again until all nodes in the nodes array are checked.

5. Combine the two rows containing the current node in the connection matrix by doing the following.
   a. Go to the row having the current node as end node (denoted as **row 1**).
   b. In the second column of row 1 (end node), cover that end node with the end node of another row (the row having current node as begin node, denoted as **row 2**).
   c. In the third column of row 1 (Component), combine the component name (in row 1) by adding "*" following by another component name from row 2.
   d. Multiple these two components' reliability and add the result into the reliability array.
   e. Delete row 2.

   After this step, all series sub-systems are simplified to be one component. In this example, component X6 and X7 which were in series are combined to form a new component X6*X7. The new connection matrix is shown in Table 4.9. The reliability array is also updated as shown in Table 4.10.

Table 4.9: Connection Matrix

Begin Node	End Node	Component
1	2	X1
1	2	X2
1	2	X3
1	3	X4
2	3	X5
3	2	X5
2	6	X6*X7
2	6	X8
3	5	X9
3	5	X10
5	6	X11

Table 4.10: Reliability Array

Component	Reliability
X1	0.9
X2	0.9
X3	0.9
X4	0.9
X5	0.9
X6	0.9
X7	0.9
X8	0.9
X9	0.9
X10	0.9
X11	0.9
X6*X7	0.81

6. Go to the first row of the connection matrix (initiate **checked row**).
7. Compare the checked row with all following rows (**current row**). If both the begin node and the end node are the same with those of the checked row (it means two components in parallel are found), call step 9 to combine these two components.
8. If checked row is the second row from the bottom of the connection matrix, go to step 10, else set next row to be the checked row and go to step 7.
9. Combine those two rows in the connection matrix by doing the following.
   a. In the third column of the checked row (Component), combine the component name (in checked row) by adding "+" following by another component name from current row.
   b. Multiple these two components' reliability and add the result into the reliability array.
   c. Delete current row.

   After this step, parallel sub-systems are eliminated. In the example, components X1, X2, X3 are combined to be X1+X2+X3; X6*X7 and X8 to be X6*X7+X8; X9, X10 to be X9+X10.

The connection matrix is simplified to be Table 4.11 and the reliability array is changed to be Table 4.12.

Table 4.11: Connection Matrix

Begin Node	End Node	Component
1	2	X1+X2+X3
1	3	X4
2	3	X5
3	2	X5
2	6	X6*X7+X8
3	5	X9+X10
5	6	X11

Table 4.12: Reliability Array

Component	**Reliability**
X1	0.9
X2	0.9
X3	0.9
X4	0.9
X5	0.9
X6	0.9
X7	0.9
X8	0.9
X9	0.9
X10	0.9
X11	0.9
X6*X7	0.81
X1+X2+X3	0.999
X6*X7+X8	0.981
X9+X10	0.99

10. Go to step 4 and do it again until the number of rows of the connection matrix is unchanged. After this step all series and parallel sub-system are simplified to be components, and the

reliability for these special components are computed and stored in the reliability array. The connection matrix final turn out to be a bridge-type bridge (Table 4.13).

Table 4.13: Connection Matrix

Begin Node	End Node	Component
1	2	X1+X2+X3
1	3	X4
2	3	X5
3	2	X5
2	6	X6*X7+X8
3	6	(X9+X10)* X11

Table 4.14: Reliability Array

Component	**Reliability**
X1	0.9
X2	0.9
X3	0.9
X4	0.9
X5	0.9
X6	0.9
X7	0.9
X8	0.9
X9	0.9
X10	0.9
X11	0.9
X6*X7	0.81
X1+X2+X3	0.999
X6*X7+X8	0.981
X9+X10	0.99
(X9+X10)* X11	0.891

If the network is a simple or series-parallel system, after this simplification, the system reliability and all reliability of series or parallel sub-system can be obtained. If the network is

complex, like the example above, a minimal path or cut-set approaches [24] - [25] can then be applied and the whole system reliability can be computed.

After simplification, the number of minimal paths reduces from 16 to be 4. The new minimal paths are as follows:

T1: X1+X2+X3, X6*X7+X8

T2: X4, (X9+X10)* X11

T3: X1+X2+X3, X5

T4: X4, X5, X6*X7+X8

Considered X1+X2+X3, X6*X7+X8 and (X9+X10)* X11 as sub-system and assuming component reliability to be 0.9, they have reliability as follow which were already exist in the reliability array:

X1+X2+X3 : 0.999 =1-(1-0.9)*(1-0.9) *(1-0.9)

X6*X7+X8 : 0.981=1-(1-0.9*0.9)*(1-0.9)

(X9+X10)* X11 : 0.891=[1-(1-0.9)*(1-0.9)]*0.9

From these sub-system reliability and the above four minimal paths, the system reliability is computed to be 0.99765.

This simplification method can not only reduce the number of minimal path but also the number of minimal cut-set. Every time the parallel components are combined to a sub-system, the number of minimal path of the system is reduced; the serried components are combined, the number of minimal cut-set is reduced. So this simplification approach is also suitable to cut-set method.

4.6.2.4 Application to Complex System

The revised connection matrix and the simplification approach are applied to a system in this section. The system in Figure 4.14 is from Nelson et al. [2]. The system has 55 minimal paths with system reliability of 0.972302.

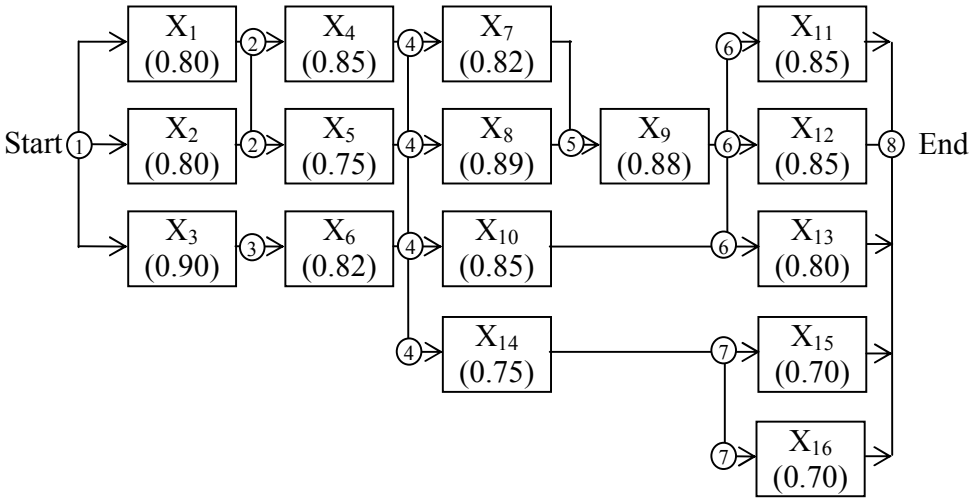


Figure 4.14: Nelson's Example

The revised connection matrix for Nelson's example is shown in Table 4.15. After simplification, the system has only one minimal path with system reliability of 0.972302, which means this system is a series-parallel system instead of complex system. The sub-system reliability is shown in Table 4.16.

Table 4.15: Revised Connection Matrix—Nelson's Example

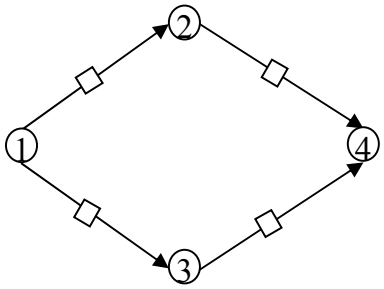
Begin Node	End Node	Component
1	2	X1
1	2	X2
1	3	X3
2	4	X4
3	4	X6
2	4	X5
4	5	X7
4	5	X8
5	6	X9
4	6	X10
6	8	X11
6	8	X12
6	8	X13
4	7	X14
7	8	X15
7	8	X16

Table 4.16: Sub-System Reliability—Nelson's Example

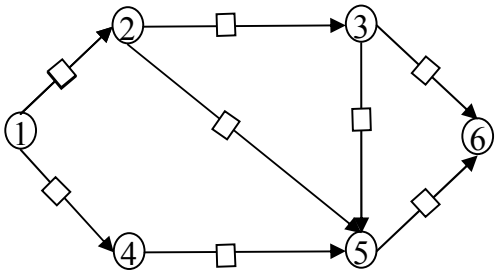
Sub-system	Reliability
X1+X2	0.96
X3*X6	0.738
X4+X5	0.9625
X7+X8	0.9802
X15+X16	0.91
(X7+X8)*X9	0.862576
X11+X12+X13	0.9955
X14*(X15+X16)	0.6825
(X1+X2)*(X4+X5)	0.924
((X7+X8)*X9)+X10	0.979386
((X1+X2)*(X4+X5))+(X3*X6)	0.980088
(((X7+X8)*X9)+X10)*(X11+X12+X13)	0.974979
((((X7+X8)*X9)+X10)*(X11+X12+X13))+(X14*(X15+X16))	0.992056
(((X1+X2)*(X4+X5))+(X3*X6))*(((((X7+X8)*X9)+X10)*(X11+X12+X13))+(X14*(X15+X16)))	0.972302

The following networks in Figure 4.15 are from Gebre and Ramirez-Marquez [16], Fotuhi-Firuzabad et al. [5], Ramirez-Marquez et al. [26], and Lin et al. [12]. Comparison of the number of tie-set and cut-set is shown in Table 4.17.

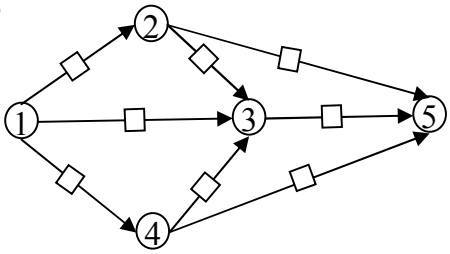
1



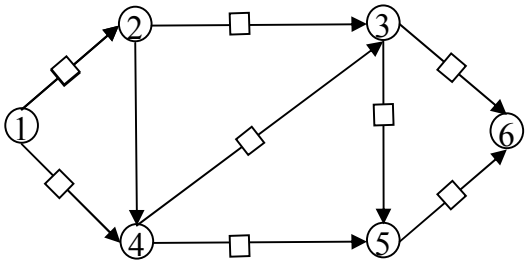
2



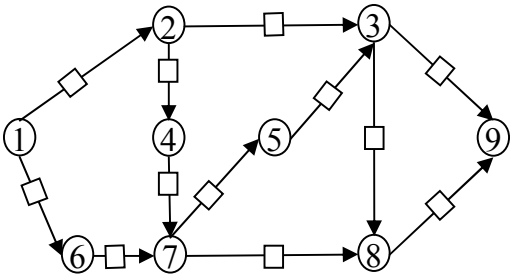
3



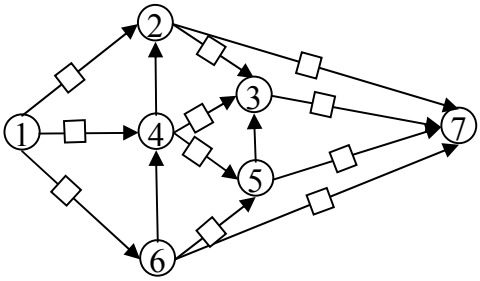
4



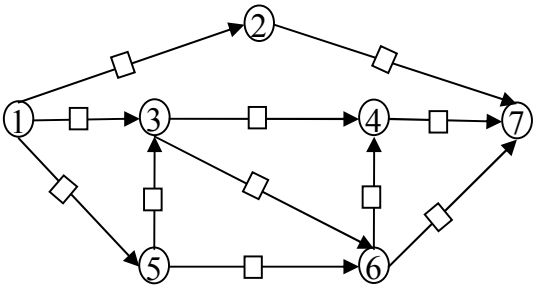
5



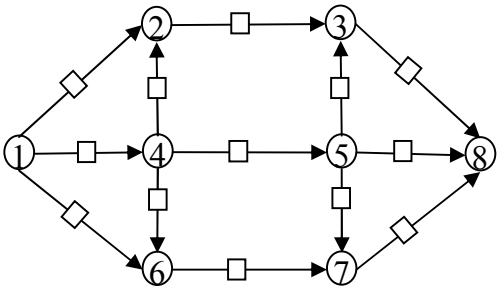
6



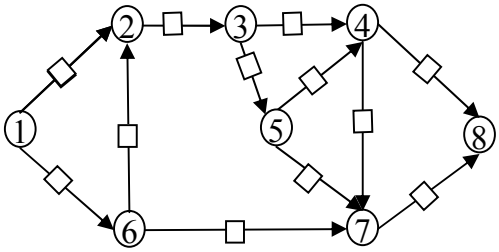
7



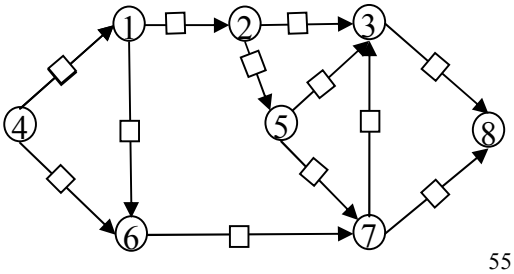
8



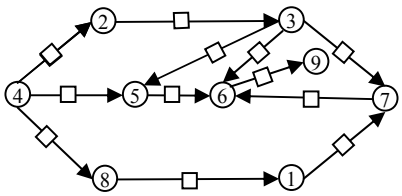
9



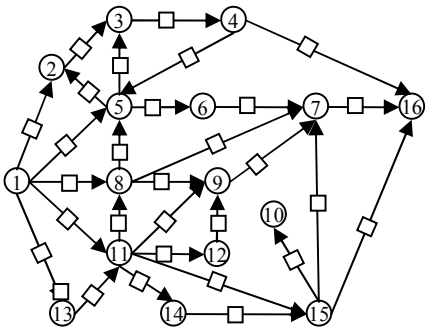
10



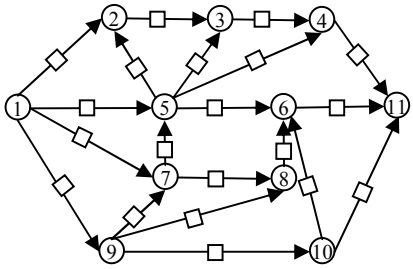
11



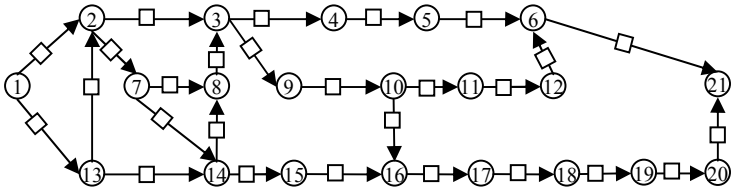
12



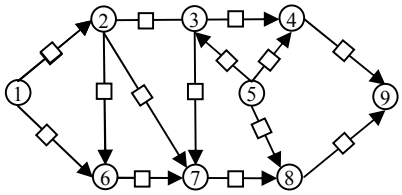
13



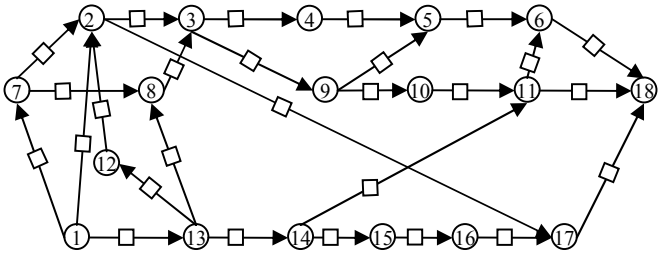
14



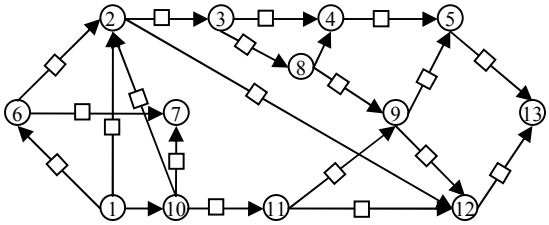
15



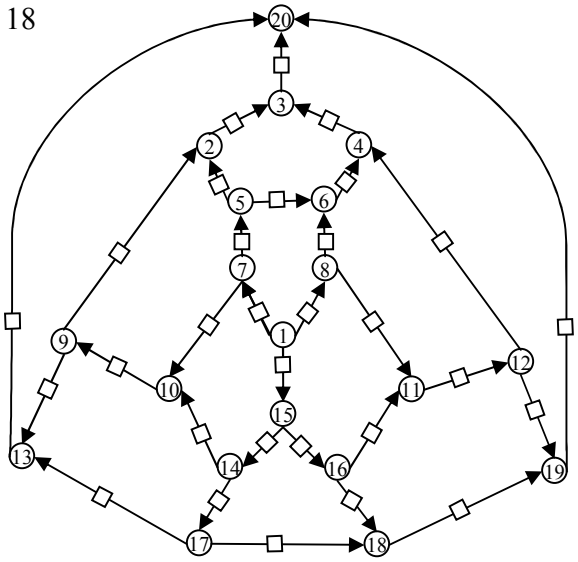
16



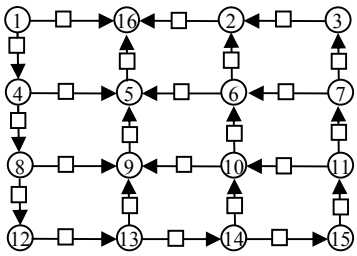
17



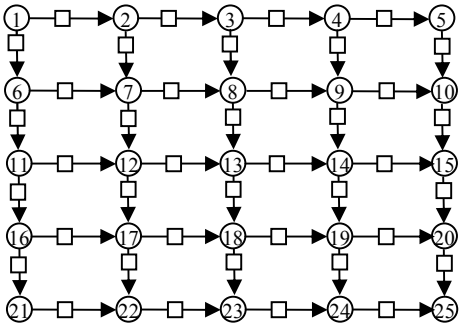
18



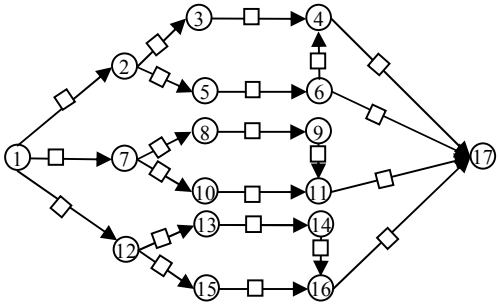
19



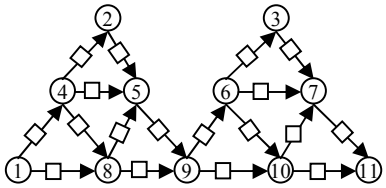
20



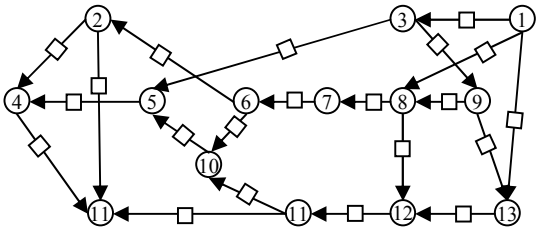
21



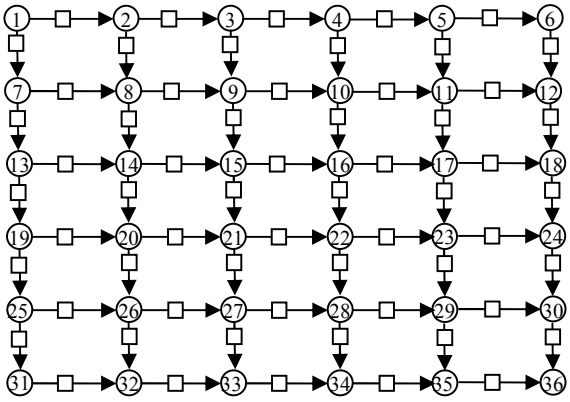
22



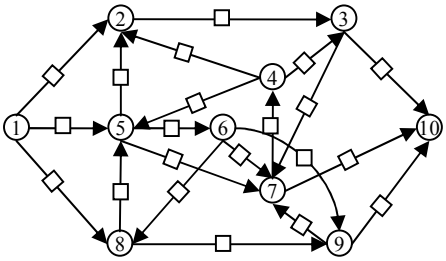
23



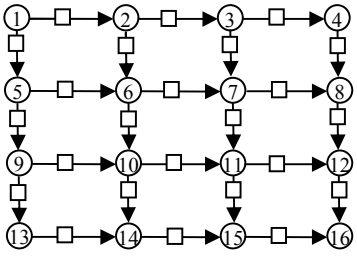
24



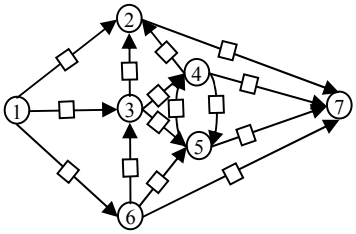
25



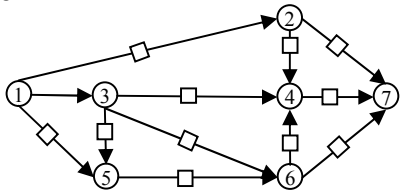
26



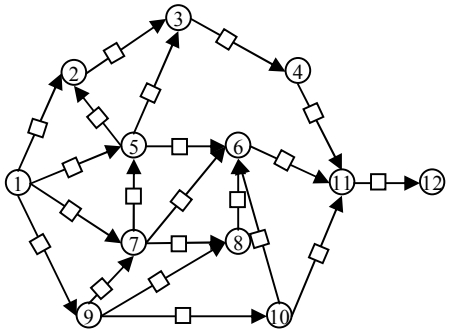
27



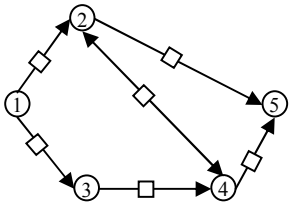
28



29



30



31

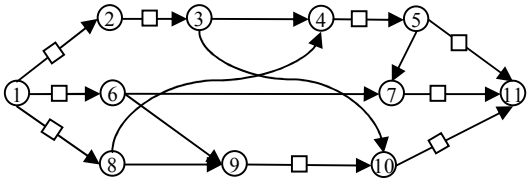


Figure 4.15: Complex Networks

Table 4.17: Simplification Comparison

Network	NO. of Minimal Cut-set		CPU Time (second)	
	Before Simplification	After Simplification	Before Simplification	After Simplification
Network 1 [16]	4	4	0.03	0.03
Network 2 [16]	9	6	0.05	0.03
Network 3 [16]	8	8	0.03	0.03
Network 4 [16]	9	9	0.05	0.05
Network 5 [16]	20	9	0.06	0.02
Network 6 [16]	25	25	0.08	0.08
Network 7 [16]	18	9	0.05	0.04
Network 8 [16]	27	27	0.08	0.08
Network 9 [16]	17	17	0.05	0.06
Network 10 [16]	19	9	0.05	0.04
Network 11 [16]	20	13	0.06	0.05
Network 12 [16]	396	110	2.00	0.39
Network 13 [16]	107	107	0.31	0.32
Network 14 [16]	222	23	0.55	0.11
Network 15 [16]	19	19	0.06	0.06
Network 16 [16]	615	191	3.45	0.72
Network 17 [16]	140	140	0.48	0.48
Network 18 [16]	3037	3037	30.09	30.10
Network 19 [16]	67	45	0.22	0.14
Network 20 [16]	6441	4530	185.00	91.89
Network 21 [16]	888	250	4.06	0.84
Network 22 [16]	16	12	0.06	0.06
Network 23 [16]	166	91	0.44	0.27
Network 24 [16]	—	—	—	—
Network 25 [16]	58	58	0.19	0.19
Network 26 [16]	330	214	1.08	0.58
Network 27 [16]	23	23	0.08	0.08
Network 28 [16]	20	20	0.06	0.06
Network 29 [26]	111	86	0.42	0.30
Network 30 [5]	6	4	0.03	0.02
Network 31 [12]	15	9	0.19	0.16

4.6.3 Determination of Minimal Tie-Set from Block Diagram

This technique has been designated as the Path Tracing Algorithm [5]. It is "the most-efficient reliability analysis methods currently available based on minimal path set enumeration" as mentioned by Gebre and Ramirez-Marquez [16]. It handles complex systems, and considers both unidirectional and bi-directional branches, but does not require any Boolean algebra for programming. The first step in this algorithm is to determine the connection matrix for the network. The connection matrix is a representation of the connections between the components of a network. It is constructed from the system network or reliability diagram that defines which components are connected between the nodes of the network. The row number of each component of the connection matrix denotes the "from" node, and the column number denotes the "to" node. Once the connection matrix is deduced and the input and output nodes determined, the algorithm moves through the rows and columns of the connection matrix, and traces all the minimal paths which exist between the two nodes of interest (input and output nodes). The process of the path tracing algorithm consists of two steps:

i) tracing all minimal paths and storing them in a specific format (Figure 4.16), and

ii) retrieving all minimal paths from the stored format(Figure 4.17).

By doing this all minimal tie-sets can be determined. Notations are defined as follow to illustrate these two steps.

- *Input node (source node), output node (sink node)*: The nodes of interest.
- *Branch number*: The number assigned to the components as the process moves through the connection matrix.

- *Parent branch*: The "branch number" of the element from which the current component has been branched.
- *Status*: A flag that indicates if a component has been branched and is represented by:
    - True – indicates that the element can be branched further, and
    - False – indicates that the element cannot be branched further.
- *Component Name*: The name of the element connected between two nodes (node number and the previous node).
- *Counter*: The number of times the process is required to be continued until all minimal paths are traced.
- *Path tracing array (PTA)*: Array used to develop paths. The structure of PTA is shown in Table 4.18. *PTA(Node_Number, j)* denotes the element of PTA in row Node_Number column j.

Table 4.18: Path Tracing Array

Node_Number	…	…	…
Parent_Branch	…	…	…
Branch_Number	…	…	…
Component	…	…	…
Status	…	…	…

Pseudo-code for step 1:

Initialize:

Counter =1

PTA(Node_Number, 0) = Input_Node

PTA(Parent_Branch, 0) = 0

PTA(Branch_Number, 0) = 0

PTA(Component, 0) = "" (empty)

PTA(Status, 0) = True

```
j=0

Develop paths:
    While PTA(Status, j) = True or j < counter
        If PTA(Status, j) = False
            j=j+1
            Continue while
        Else
            Find out all successor nodes of node PTA(Node_Number, j)
            For each successor node
                PTA(Node_Number, Counter) = Name of successor node
                PTA(Parent_Branch, Counter) = PTA(Branch_Number, j)
                PTA(Branch_Number, Counter) = Counter
                PTA(Component, Counter) = Component name between node
                                    PTA(Node_Number, j) and node PTA(Node_Number, Counter)
                If PTA(Node_Number, Counter) = Output Node
                    PTA(Status, Counter) = False
                Else
                    PTA(Status, Counter) = True
                Counter= Counter+1
            PTA(Status, j) = False
            j=j +1
```

Pseudo-code for step 2:

Retrieve all minimal paths:

```
For each PTA(Node_Number, j) = Output_Node
    k=j
    While Parent_Branch ≠ 0
        If PTA(Component, k) ≠ 1
            Store PTA(Component, k)
        Find PTA(Node_Number, i)= PTA(Parent_Branch, k)
        k=i
```

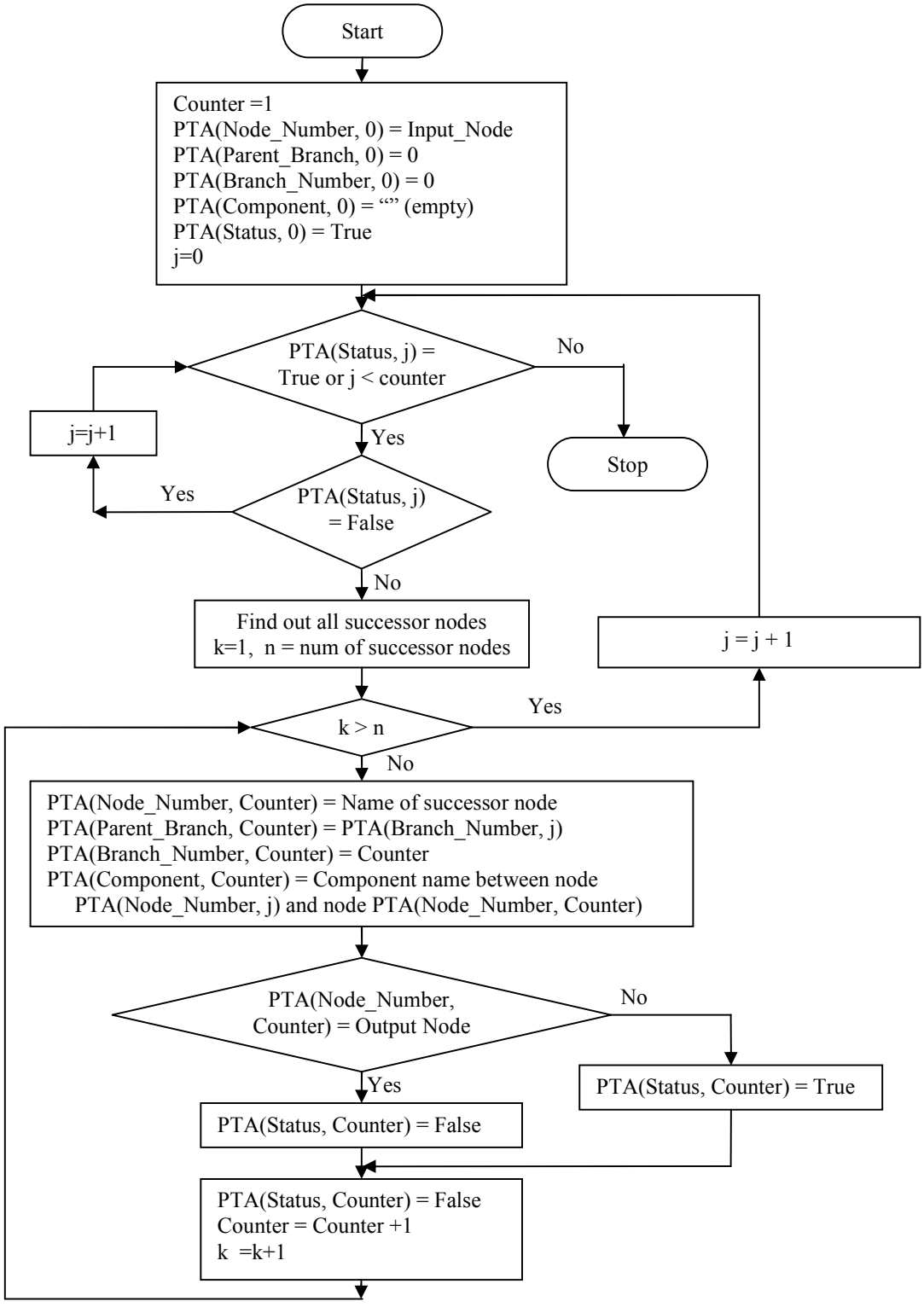


Figure 4.16: Flowchart to store paths

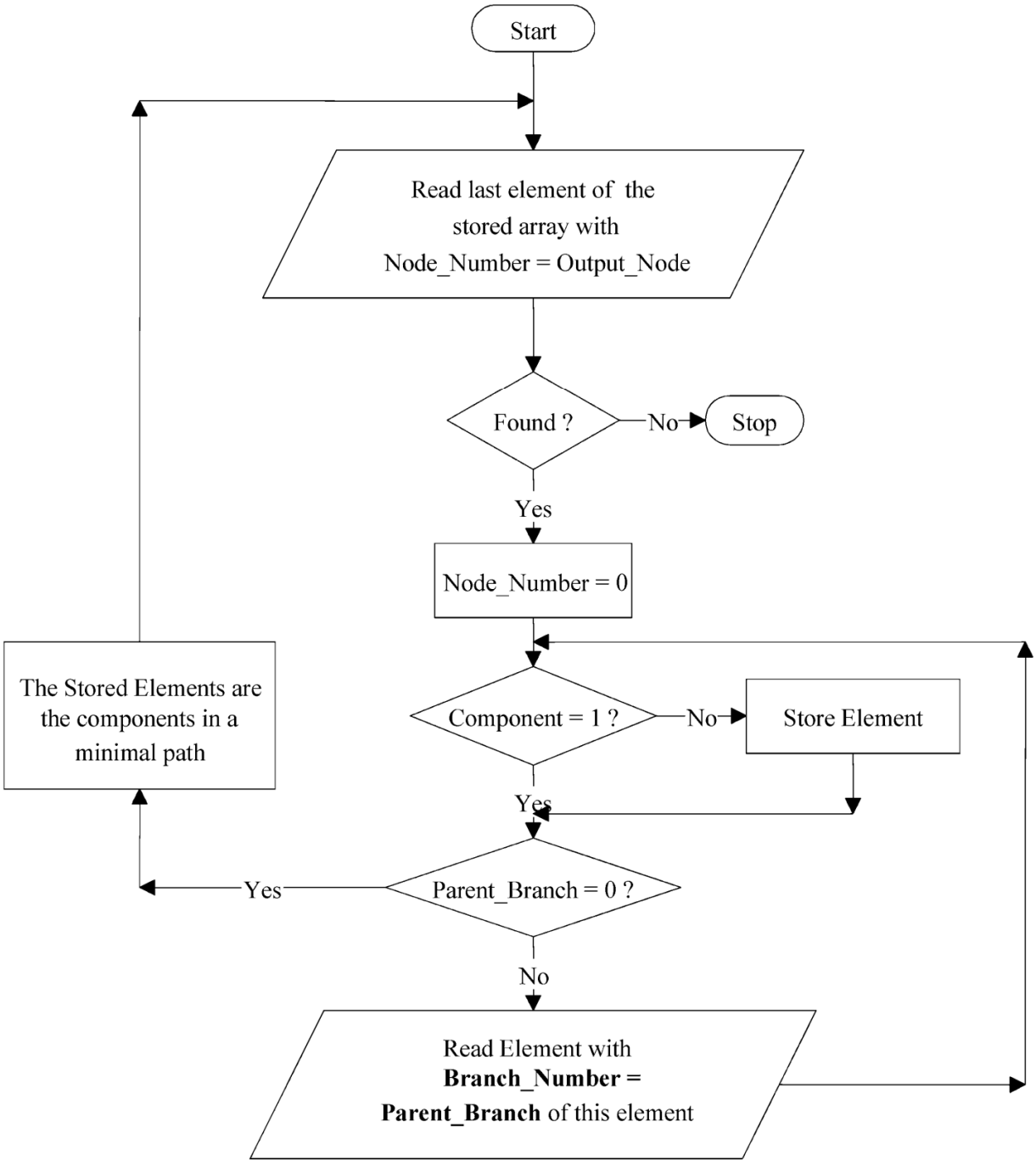


Figure 4.17: Flowchart to retrieve paths

4.6.4 Determination of Cut-Set from Given Minimal Tie-Set

Boolean algebra technique is used to find the cut-sets of a network from given minimal tie-set.

Boolean Algebra Technique

The Boolean Algebra technique, which is used to find the cut-sets of a network, also assumes that all the minimal paths of the network are known. This technique is simple and can be easily illustrated through an example. The technique involves some manipulation of Boolean expressions and therefore is computation-intensive. For the same system discussed in Figure 4.11, the minimal paths are

Tl = X1, X2

T2= X3, X4

T3 = X1, X5, X4

T4= X3, X5, X2

The system will fail if one or more components in each path fail. Hence for T1, either X1 or X2 or both must fail to cause the path to fail. This is expressed as (X1 + X2). Since each path must fail to cause the system failure, it imposes a Boolean AND condition to find the cut-sets. Hence the expression (X1+X2) * (X3+X4) * (X1+X4+X5)* (X3+X2+X5) will give all possible cut-sets. To get the minimal cut-sets, Boolean simplification has to be performed on this expression. The product of the first three terms is

(X1+X2) * (X3+X4)*(X1+X4+X5)

= X1*X3+X1*X3*X5+X1*X3*X4+X1*X3*X2+X1*X3*X4+X1*X4+X1*X4+X1*X4*X5 +
X1*X2*X4 + X2*X4 + X2*X4*X5 + X3*X2*X5

=X1*X3+X1*X4+X2*X4+X3*X2*X5

Multiplying with the fourth term

(X1+X2)*(X3+X4)*(X1+X4+X5)*(X3+X2+X5)

=(X1*X3+X1*X4+X2*X4+X3*X2*X5) * (X3+X2+X5)

= X1*X3 + X1*X3*X5 + X1*X3*X2 + X1*X3*X4 + X1*X4*X5 + X1*X2*X4 + X3*X2*X4 + X2*X4*X5 + X2*X4 + X3*X2*X5 + X3*X2*X5 + X3*X2*X5

=X1*X3 + X2*X4 + X1*X4*X5 + X3*X2*X5

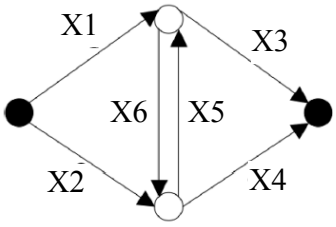
Therefore {X1, X3}, {X2, X4}, {X1, X4, X5}, and {X3, X2, X5} are the minimal cut-sets for the given network.

4.6.5 Determination of Minimal Cut-Set from Block Diagram

There are different algorithms currently available for minimal cut-sets enumeration, such as nodes merging approach [12], Yeh's MCV approach [13], and element substitution approach [6]. Element substitution algorithm is the latest and the most efficient method to determine minimal cut-sets, so this algorithm is selected and make some improvement to implement to the software.

4.6.5.1 Element Substitution Approach

An example network shown in Figure 4.18 is used to illustrate this approach.



Component	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	0	0	0	1	0
4	0	1	0	0	0	1
5	0	1	0	0	0	1
6	1	0	0	0	1	0

Figure 4.18: Predecessor matrix construction

Step 1: Build the connection matrix of the network which indicates the connection relationship of components. The result is shown in Figure 4.18.

Step 2: Initiate primary set (a cut-set used to generate other cut-sets using forward method) by putting all components connected to the source node; Initiate parent set (a cut-set used to generate other cut-sets using backward method) by putting all components connected to the sink node. In this example, the primary set is {X1, X2} and the parent is {X3, X4}.

Step3: Develop cut-sets from primary set by substituting elements of primary set with their successors and put the result in the primary potential cut-set array. The first element X1 of primary set {X1, X2} is replaced with its successors X3 and X6 forming a new cut-set {X3, X6, X2}. Before this new cut-set can be put to the potential cut-set array, we need to 1) check elements within the new cut-set for predecessor relationship and eliminate the predecessor (eliminate successor if it is developed from parent set) from the cut-set; 2) check the new cut-set with the primary potential cut-sets and parent potential cut-set. If the new cut-set is dominated or duplicated, eliminate the new cut-sets and go to next step, else if the new cut-set dominates the cuts in primary potential cut-set or parent potential cut-set, eliminate the dominated cut-sets. In this example, the new cut-set {X3, X6, X2} can be added to the primary potential cut-set because there is no predecessor relationship between the elements X3, X6 and X2 and no domination relationship between new cut-set {X3, X6, X2} and the primary potential cut-set

{X1, X2} and parent potential cut-set{X3, X4} as shown in Figure 4.19 ("→" means it is a primary set or parent set).

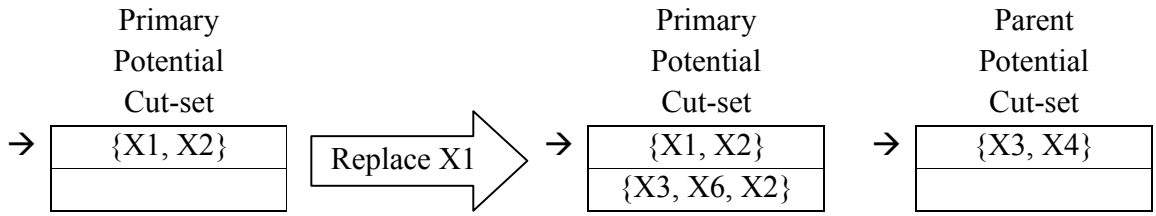


Figure 4.19: Element Substitution 1

Step4: Develop cut-sets from parent set by substituting elements of parent set with their predecessors and put the result in the parent potential cut-set array. The first element X3 of primary set {X3, X4} is replaced with its predecessors X1 and X5 forming a new cut-set {X1, X5, X4}. This new cut-set can be added to the parent potential cut-set because there is no predecessor relationship between the elements X1, X5 and X4 and no domination relationship between new cut-set {X1, X5, X4} and the primary potential cut-set {X1, X2}, {X3, X6, X2} and parent potential cut-set{X3, X4} as shown in Figure 4.20.

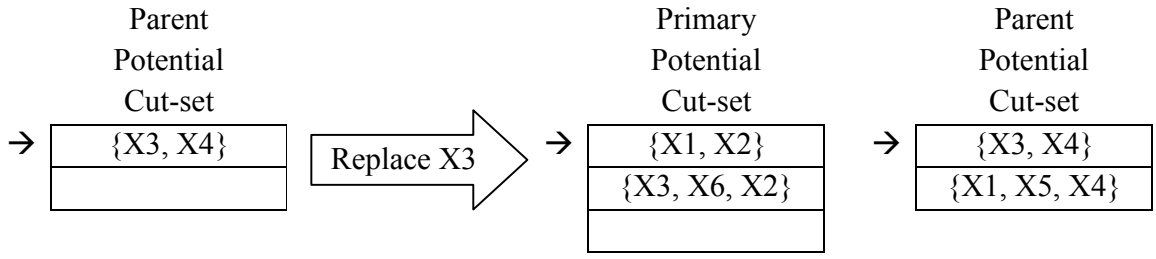


Figure 4.20: Element Substitution 2

Step5: Repeat step 3 and step 4 until all potential cut-sets have been developed. In this example, we will continue to

1) Go to step 3 and replace X2 of primary set {X1, X2} with it successors X4 and X5 forming new cut-set {X1, X4, X5}. This new cut-set cannot be put into the potential

cut-set array because there is already a same cut-set in parent potential cut-set array as shown in Figure 4.21.

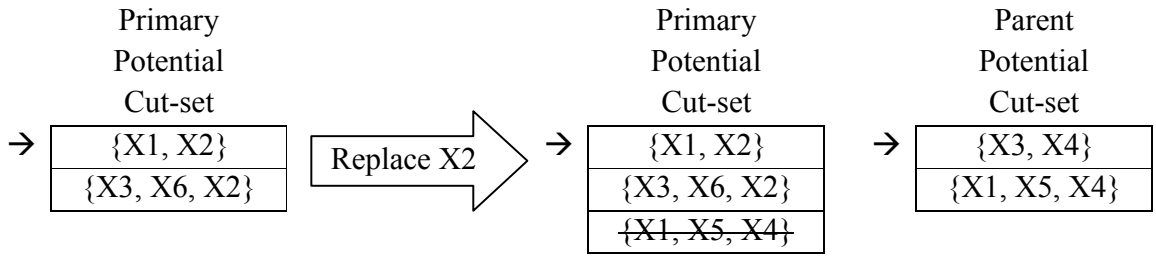


Figure 4.21: Element Substitution 3

2) Go to step 4 replace X4 of parent set {X3, X4} with its predecessors X2 and X6 forming new cut-set {X3, X2, X6} which is already exist in the primary potential cut-set array, so delete the new cut-set and go to next step.

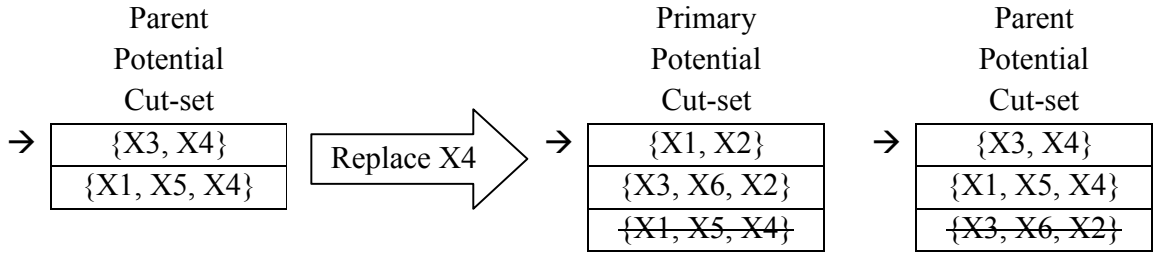


Figure 4.22: Element Substitution 4

3) Go to step 3. Because each element of primary set {X1, X2} has already been replace {X1, X2} with its following cut-set {X3, X6, X2} in primary potential cut-set array. So move "→" to {X3, X6, X2} and continue step 3. Since there is no successor for the first element X3 of the primary set, go to the second element X6 and replace X6 with its successor X4 and X5 forming a new cut-set {X3, X4, X5, X2}. Within this new cut-set, X5 is a predecessor of X3, so delete element X5; X2 is a predecessors of X4, so delete X2. So the new cut-set is updated to be {X3, X4}. This new cut-set

cannot enter the primary potential cut-set because the same cut-set already exists in the parent potential cut-set array. The result is shown in Figure 4.23.

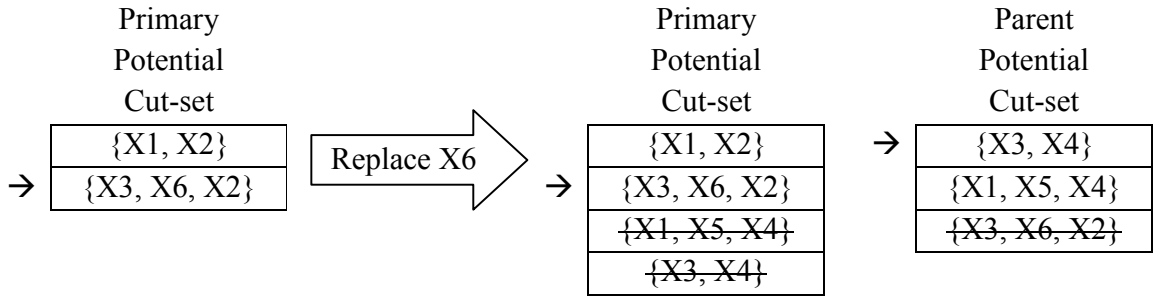


Figure 4.23: Element Substitution 5

4) Go to step 4. Because each element of parent set {X3, X4} has already been replaced, replace {X3, X4} with its following cut-set {X1, X5, X4} in parent potential cut-set array. So move "→" to {X1, X5, X4} and continue step 4. Since there is no predecessor for the first element X1 of the parent set, go to the second element X5 and replace X5 with its predecessor X2 and X6 forming a new cut-set {X1, X2, X6, X4}. Within this new cut-set, X4 is a successor of X2, so delete element X4; X6 is a successor of X1, so delete X6. So the new cut-set is updated to be {X1, X2}. This new cut-set cannot enter the primary potential cut-set because the same cut-set already exists in the primary potential cut-set array. The result is shown in Figure 4.24.

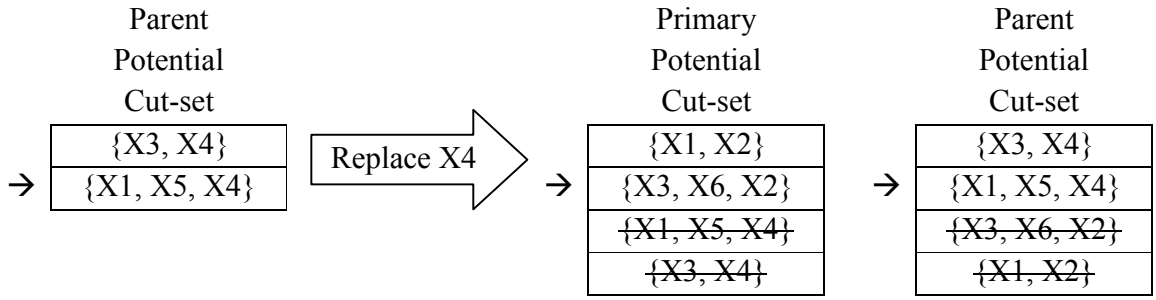


Figure 4.24: Element Substitution 6

5) Go to step 3 and replace element X2 with its successor X4 and X5 form a new cut-set {X3, X6, X4, X5} which becomes {X3, X4} after eliminating predecessors. Because {X3, X4} already exist in parent potential cut-set, the new cut-set is deleted. Since the last element of the last primary potential cut-set has already substituted and there is no more primary potential cut-sets generated, stop the loop of step 3.

6) Go to step 4 and replace element X4 with its predecessor X2 and X6 form a new cut-set {X1, X5, X2, X6} which becomes {X1, X2} after eliminating successors. Because {X1, X2} already exist in parent primary cut-set, the new cut-set is deleted. Since the last element of the last parent potential cut-set has already substituted and there is no more parent potential cut-sets generated, stop the loop of step 4.

Step6: Retrieve minimal cut-sets from the primary potential cut-set array and parent potential cut-set array. In this example, the minimal cut-sets are {X1, X2}, {X3, X6, X2}, {X3, X4}, {X1, X5, X4}.

4.6.5.2 Improvement to Element Substitution Approach

This algorithm continuously uses the generated potential cut-set as a primary set or parent set to generate more potential cut-sets, so the primary or parent set is wanted to be a minimal set. If the primary or parent cut-set is not a minimal cut-set, all its generated cut-sets will not be

minimal and these non-minimal cut-sets will continue to generate more useless cut-sets. The size of potential cut-set array will increase significantly resulting in inefficiency of the cut-set determination. This is the reason why every time when a new cut-set is generated we need to compare the new cut-set with each potential cut-set to eliminate those potential cut-sets which are dominated by the new cut-set. However, this still cannot guarantee every potential cut-set in the potential cut-set array is a minimal cut-set until the final cut-set is generated and the last domination elimination is done.

The efficiency can be improved if the potential cut-set array is kept leaner (less non-minimal cut-set). In current algorithm when a potential cut-set is found to be dominated by the new cut-set, this potential cut-set will be eliminated. Now we can do it further. All its successors (the potential cut-sets generated by this eliminated cut-set) will also be deleted, and continue to delete successors in the next level until the end of the potential cut-set array is reached. That is, whenever a potential cut-set is deleted, all levels of its successors will also be eliminated. In this way, potential cut-set array will contain less non-minimal cut-sets and its size will be decreased. Because when a new cut-set is generated, it is required to compare with every cut-set in the potential array to check the domination relationship, the decrease of the size of potential cut-set array will increase the efficiency.

This improved method is applied to Gebre and Ramirez-Marquez's 28 networks [16] in Figure 4.15 and obtain exactly the same minimal cut-set results with a shorter computation time as shown in Table 4.19.

Table 4.19: Element Substitution Result Comparison

Net-work	NO. of Comp onent	NO. of minimal Cuts	Gebre's CPU time (sec.) [16]	New CPU Time (sec.)	Improved Approach CPU time (sec.)	Non-minimal Cut-set Generated	Non-minimal Cut-set Generated (Improved)	Simplified CPU time (sec.)
1	5	4	0	0.05	0.03	0	0	0.03
2	8	9	0.01	0.05	0.05	0	0	0.03
3	8	8	0.01	0.03	0.03	0	0	0.03
4	9	9	0.01	0.06	0.05	0	0	0.05
5	12	20	0.01	0.06	0.06	0	0	0.02
6	14	25	0.02	0.09	0.08	0	0	0.08
7	11	18	0.01	0.08	0.05	1	1	0.04
8	13	27	0.02	0.09	0.08	0	0	0.08
9	12	17	0.01	0.06	0.05	1	1	0.06
10	12	19	0.01	0.08	0.05	3	3	0.04
11	13	20	0.01	0.08	0.06	0	0	0.05
12	30	396	10.11	2.55	2.00	461	350	0.39
13	20	107	0.51	0.40	0.31	22	22	0.32
14	26	222	2.07	0.64	0.55	178	134	0.11
15	14	19	0.03	0.06	0.06	3	3	0.06
16	29	615	35.06	3.97	3.45	472	379	0.72
17	23	140	1.071	0.56	0.48	48	33	0.48
18	30	3037	355.16	31.69	30.09	1026	997	30.10
19	24	67	0.3	0.25	0.22	17	17	0.14
20	40	6441	3329.7	216.0	185.00	7433	6760	91.89
21	25	888	26.638	4.73	4.06	268	267	0.84
22	18	16	0.1	0.08	0.06	14	14	0.06
23	21	166	0.911	0.52	0.44	52	52	0.27
24	60	—	—	—	—	—	—	—
25	21	58	0.21	0.20	0.19	9	9	0.19
26	24	330	5.137	1.15	1.08	129	122	0.58
27	15	23	0.08	0.11	0.08	0	0	0.08
28	12	20	0.09	0.08	0.06	0	0	0.06

4.6.6 Calculation of System Reliability from Tie-Sets and Cut-Sets

Let T_i correspond to the i^{th} tie-set, and C_i correspond to the i^{th} cut-set. The system reliability is computed as follows:

$R_s = P(T_1+T_2\ldots\ldots+T_m) = P$(at least one tie-set is good)

$R_s = P(C_1 *C2 \ldots\ldots *C_m) = P$(at least one element of the cut-set is operative)

Expressed in terms of unreliability, we have,

$1 - R_s = P(T'_1*T'_2\ldots\ldots*T'_m) = P$(all tie-sets failed)

$1 - R_s = P(C'_1+C'_2\ldots\ldots+C'_m) = P$(at least one cut-set fails)

where T'_i and C'_i are the compliments of T_i and C_i respectively. From the above reliability expression bounds can be obtained by using the basic probabilistic inequalities as follows:

$$R_s = P(T_1 + T_2 \cdots + T_m) \leq \sum_{i=1}^{m} P(T_i) \tag{4.4}$$

$$R_s = P(T_1 + T_2 \cdots + T_m) \geq \sum_{i=1}^{m} P(T_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P(T_i * T_j) \quad 1 \leq i,j \leq m \tag{4.5}$$

The upper (R_{U1}) and lower bounds (R_{L1}) to the reliability are:

$$R_{U1} = \sum_{i=1}^{m} P(T_i) \tag{4.6}$$

$$R_{L1} = \sum_{i=1}^{m} P(T_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P(T_i * T_j), \quad 1 \leq i,j \leq m \tag{4.7}$$

In the same manner, another upper bound is obtained

$$R_{U2} = \sum_{i=1}^{m} P(T_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P(T_i * T_j) + \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} P(T_i * T_j * T_k), \quad 1 \leq i,j,k \leq m \tag{4.8}$$

By using the above equation, upper and lower bounds on reliability are computed till the desired precision on system reliability is obtained. The application of this method is illustrated through an example as shown in Figure 4.11 in section 4.6.2.1

Similarly inequalities can be applied to cut-sets to obtain another upper and lower bounds.

$$R_{L1} = 1 - \sum_{i=1}^{m} P(C'_i) \tag{4.9}$$

$$R_{U1} = 1 - \left(\sum_{i=1}^{m} P(C'_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P\left(C'_i * C'_j\right)\right)$$

$$= R_{L1} + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P\left(C'_i * C'_j\right) \quad 1 \le i,j \le m \tag{4.10}$$

$$R_{L2} = 1 - \left(\sum_{i=1}^{m} P(C'_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} P\left(C'_i * C'_j\right)\right.$$

$$\left. + \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} P\left(C'_i * C'_j * C'_k\right)\right)$$

$$= R_{U1} - \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} P\left(C'_i * C'_j * C'_k\right)$$

$$1 \le i,j,k \le m \tag{4.11}$$

Example: A system has four minimal cut-sets (developed from a fault tree in section 6.2): {X3, X4, X5}, {X2, X3}, {X1, X3}, and {X1, X2}. Assuming the unreliability of components X1, X2, X3, X4, X5 are R_1'=0.1, R_2'=0.2, R_3'=0.3, R_4'=0.4, R_5'=0.5 respectively, what is the system's reliability?

To solve this problem, we first calculate the reliability of all combinations of cut-sets. The result is shown as Table 4.20.

Table 4.20: Cut-set Reliability

Probability of Cut-set Combination	Break down Cut-set to components	Apply Boolean Algebra	Reliability	Value
$P(C_1')$	$P\{(X_1'*X_2')\}$	$P\{X_1'*X_2'\}$	$R_1'*R_2'$	0.02
$P(C_2')$	$P\{(X_1'*X_3')\}$	$P\{X_1'*X_3'\}$	$R_1'*R_3'$	0.03
$P(C_3')$	$P\{(X_2'*X_3')\}$	$P\{X_2'*X_3'\}$	$R_2'*R_3'$	0.06
$P(C_4')$	$P\{(X_3'*X_4'*X_5')\}$	$P\{X_3'*X_4'*X_5'\}$	$R_3'*R_4'*R_5'$	0.06
$P(C_1'*C_2')$	$P\{(X_1'*X_2')*(X_1'*X_3')\}$	$P\{X_1'*X_2'*X_3'\}$	$R_1'*R_2'*R_3'$	0.006
$P(C1'*C3')$	$P\{(X1'*X2')*(X2'*X3')\}$	$P\{X1'*X2'*X3'\}$	$R1'*R2'*R3'$	0.006
$P(C1'*C4')$	$P\{(X1'*X2')*(X3'*X4'*X5')\}$	$P\{X1'*X2'*X3'*X4'*X5'\}$	$R1'*R2'*R3'*R4'*R5'$	0.0012
$P(C2'*C3')$	$P\{(X1'*X3')*(X2'*X3')\}$	$P\{X1'*X2'*X3'\}$	$R1'*R2'*R3'$	0.006
$P(C2'*C4')$	$P\{(X1'*X3')*(X3'*X4'*X5')\}$	$P\{X1'*X3'*X4'*X5'\}$	$R1'*R3'*R4'*R5'$	0.006
$P(C3'*C4')$	$P\{(X2'*X3')*(X3'*X4'*X5')\}$	$P\{X2'*X3'*X4'*X5'\}$	$R2'*R3'*R4'*R5'$	0.012
$P(C1'*C2'*C3')$	$P\{(X1'*X2')*(X1'*X3')*(X2'*X3')\}$	$P\{X1'*X2'*X3'\}$	$R1'*R2'*R3'$	0.006
$P(C1'*C2'*C4')$	$P\{(X1'*X2')*(X1'*X3')*(X3'*X4'*X5')\}$	$P\{X1'*X2'*X3'*X4'*X5'\}$	$R1'*R2'*R3'*R4'*R5'$	0.0012
$P(C1'*C3'*C4')$	$P\{(X1'*X2')*(X2'*X3')*(X3'*X4'*X5')\}$	$P\{X1'*X2'*X3'*X4'*X5'\}$	$R1'*R2'*R3'*R4'*R5'$	0.0012
$P(C2'*C3'*C4')$	$P\{(X1'*X3')*(X2'*X3')*(X3'*X4'*X5')\}$	$P\{X1'*X2'*X3'*X4'*X5'\}$	$R1'*R2'*R3'*R4'*R5'$	0.0012
$P(C1'*C2'*C3'*C4')$	$P\{(X1'*X2')*(X1'*X3')*(X2'*X3')*(X3'*X4'*X5')\}$	$P\{X1'*X2'*X3'*X4'*X5'\}$	$R1'*R2'*R3'*R4'*R5'$	0.0012

According to the formulas above,

$$\sum_{i=1}^{4} P(C_i') = P(C_1') + P(C_2') + P(C_3') + P(C_4')$$

$$= 0.02 + 0.03 + 0.06 + 0.06 = 0.17$$

$$\sum_{I=1}^{3}\sum_{j=i+1}^{4} P(C_i' * C_j') = P(C_1' * C_2') + P(C_1' * C_3') + P(C_1' * C_4') + P(C_2' * C_3') + P(C_2' * C_4') + P(C_3' * C_4')$$

$$= 0.006 + 0.006 + 0.0012 + 0.006 + 0.006 + 0.012$$

$$= 0.0372$$

$$\sum_{i=1}^{2}\sum_{j=i+1}^{3}\sum_{k=j+1}^{4} P(C_i' * C_j' * C_k') = P(C_1' * C_2' * C_3') + P(C_1' * C_2' * C_4')$$

$$+P(C_1' * C_3' * C_4') + P(C_2' * C_3' * C_4')$$

$$= 0.006 + 0.0012 + 0.0012 + 0.0012$$

$$= 0.0096$$

$$\sum_{I=1}^{1}\sum_{j=i+1}^{2}\sum_{k=j+1}^{3}\sum_{l=k+1}^{4} P(C_i' * C_j' * C_k' * C_l') = P(C_1' * C_2' * C_3' * C_4')$$

$$= 0.0012$$

And then the system reliability 0.8588 is obtained by the following steps shown in Table 4.21.

Table 4.21: Determination of System Reliability from Cut-set

Step	Lower Bound	Upper Bound	Reliability	Tolerance
Step 1	R_{L1} =1 - 0.17 =0.83	—	—	—
Step 2	R_{L1} =0.83	$R_{U1} = R_{L1} + 0.0372$ =0.8672	$R_{sys}= (R_{L1}+ R_{U1})/2$ = 0.84866	±0.0186 (=0.0372/2)
Step 3	R_{L2} = R_{U1} - 0.0096 =0.8576	R_{U1} = 0.8672	$R_{sys}= (R_{L2}+ R_{U1})/2$ = 0.8624	±0.0048 (=0.0096/2)
Step 4	R_{L2} =0.8576	$R_{U2} = R_{L2} + 0.0012$ =0.8588	$R_{sys}= (R_{L2}+ R_{U2})/2$ = 0.8582	±0.0006 (=0.0012/2)
Step 5	R_{L3} = R_{U2} - 0 =0.8588	R_{U2} = 0.8588	$R_{sys}= (R_{L3}+ R_{U2})/2$ = 0.8588	0

CHAPTER 5

STATE DEPENDENT SYSTEM

In the previous section we made an important assumption of independent failures among the individual components. However, if the failure of one component is dependent on the failure of another component then we need a different analysis method ---Markov analysis.

5.1 State Diagram

Figure 5.2 is state diagram of a general standby system shown in Figure 5.1 [18].

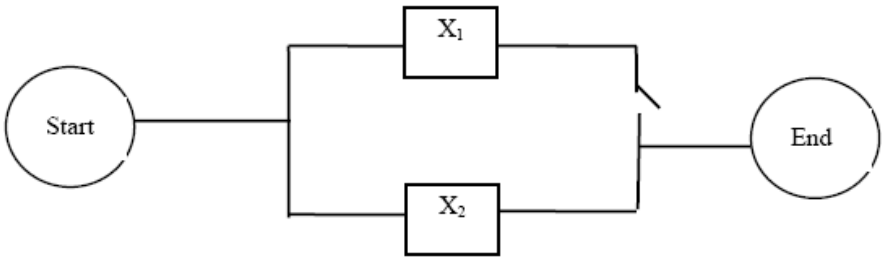


Figure 5.1: Standby System

States: 1. Both components operating

2. Component X1 fails

3. Component X2 fails

4. Both components fail

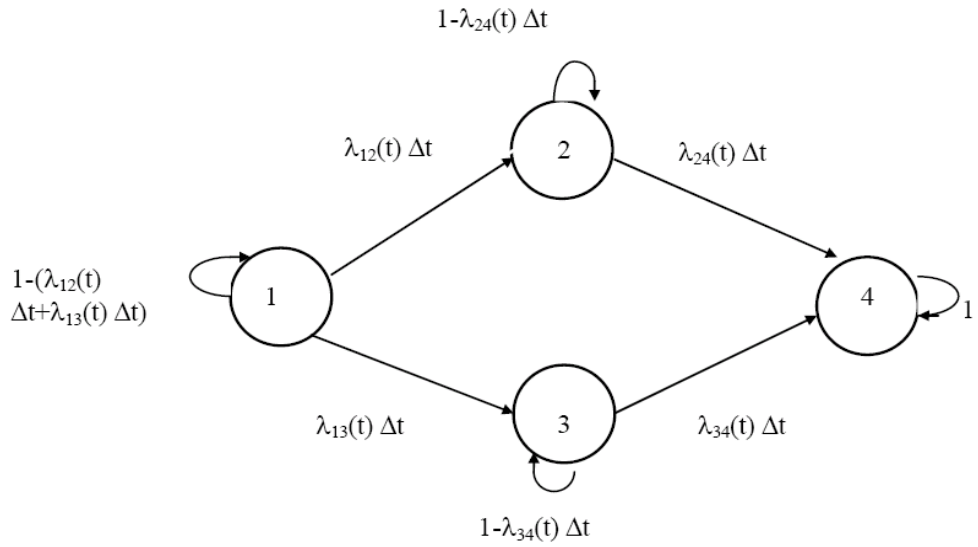


Figure 5.2: State Diagram

Probability

$\lambda_{12}(t)\,\Delta t$ Probability of going from state 1 to state 2 during Δt

$\lambda_{13}(t)\,\Delta t$ Probability of going from state 1 to state 3 during Δt

$\lambda_{24}(t)\,\Delta t$ Probability of going from state 2 to state 4 during Δt

$\lambda_{34}(t)\,\Delta t$ Probability of going from state 3 to state 4 during Δt

where, $\lambda_{ij}(t)$ are failure rates for each ij transition.

State Equations

1. $P_1(t+\Delta t) = P_1(t)\,[\,1 - (\lambda_{12}(t)\,\Delta t + \lambda_{13}(t)\,\Delta t)\,]$
2. $P_2(t+\Delta t) = P_1(t)\,\lambda_{12}(t)\,\Delta t + P_2(t)\,[1 - \lambda_{24}(t)\,\Delta t)]$
3. $P_3(t+\Delta t) = P_1(t)\,\lambda_{13}(t)\,\Delta t + P_3(t)\,[1 - \lambda_{34}(t)\,\Delta t)]$
4. $P_4(t+\Delta t) = P_2(t)\,\lambda_{24}(t)\,\Delta t + P_3(t)\,\lambda_{34}(t)\,\Delta t + P_4(t)$

5.2 Markov Analysis

5.2.1 Calculation of System Reliability from Markov Model

From Markov Model the exact solution for the reliability of a complex system with N success states over the time t is

$$R(t)=\sum_{i=1}^{N} P_i(t) \tag{5. 1}$$

where the $P_i(t)$ values are the solutions to the set of N differential equations

$$\begin{aligned}
\frac{d}{dt}P_1(t) &= -r_{11}P_1(t) + r_{21}P_2(t) + r_{31}P_3(t) + \cdots + r_{N1}P_N(t) \\
\frac{d}{dt}P_2(t) &= -r_{12}P_1(t) + r_{22}P_2(t) + r_{32}P_3(t) + \cdots + r_{N2}P_N(t) \\
&\vdots \\
\frac{d}{dt}P_N(t) &= -r_{1N}P_1(t) + r_{2N}P_2(t) + r_{3N}P_3(t) + \cdots + r_{NN}P_N(t) \\
P_1(0) &= 1.0 \quad and \quad P_i(0) = 0 \quad for\ all\ i \neq 1
\end{aligned}$$

where r_{ij} ($i \neq j$) represent the rate (failure rate λ or repair rate μ) from state i to state j. r_{ii} represent the sum of all transition rates out of state i:

$$r_{ii} = \sum_{all\ k \neq i} r_{ik}$$

The above equations can be solved on the computer by approximating them by difference equations with sufficiently small Δt.

$$\begin{aligned}
P_1(t+\Delta t) &= P_1(t)(1 - r_{11}\Delta t) + P_2(t)\, r_{21}\Delta t + \ldots + P_N(t)\, r_{N1}\Delta t \\
P_2(t+\Delta t) &= P_1(t)\, r_{12}\Delta t + P_2(t)\,(1 - r_{22}\Delta t) + \ldots + P_N(t)\, r_{N2}\Delta t \\
&\vdots
\end{aligned}$$

$$P_N(t+\Delta t) = P_1(t)\, r_{1N}\Delta t \quad + P_2(t)\, r_{2N}\Delta t + \ldots \quad + P_N(t)\,(1 - r_{NN}\Delta t)$$

That is,

$$P_i(t+\Delta t) = \sum_{all\ j \neq i} P_j(t)\left[r_{ji}\Delta t\right] + P_i(t)\left[1 - r_{ii}\Delta t\right] \tag{5.2}$$

Let

$$n = t/\Delta t \tag{5.3}$$

Then,

$$P_i(t) = P_i(n\ \Delta t) \tag{5.4}$$

And

$$P_i(t+\Delta t) = P_i(\left[n+1\right]\Delta t) \tag{5.5}$$

The set of N independent difference equations can be written as

$$\begin{aligned} &P_i([n+1]\Delta t) = \sum_{all\ j \neq i} P_j(n\Delta t)[r_{ji}\Delta t] + P_i(n\Delta t)[1 - r_{ii}\Delta t] \\ &where\ i = 1,2,3\cdots, N \end{aligned} \tag{5.6}$$

The probability vector $\Pi(t)$ and the matrix $[A]$ are defined as follows:

$$\Pi = \begin{bmatrix} P_1(t) \\ P_2(t) \\ \vdots \\ P_N(t) \end{bmatrix} \tag{5.7}$$

$$[A] = \begin{bmatrix} (1 - r_{11}\Delta t) & r_{21}\Delta t & r_{31}\Delta t & \cdots & r_{N1}\Delta t \\ r_{12}\Delta t & (1 - r_{22}\Delta t) & r_{32}\Delta t & \cdots & r_{N2}\Delta t \\ r_{13}\Delta t & r_{23}\Delta t & (1 - r_{33}\Delta t) & \cdots & r_{N3}\Delta t \\ \vdots & \vdots & \vdots & & \vdots \\ r_{1N}\Delta t & r_{2N}\Delta t & r_{3N}\Delta t & \cdots & (1 - r_{NN}\Delta t) \end{bmatrix} \tag{5.8}$$

In matrix form, the set of equations is written

$$\Pi([n+1]\Delta t) = [A]\Pi(n\Delta t) \tag{5.9}$$

From which follows

$$\Pi(n\Delta t) = [A]([n-1]\Delta t) = [A]^2\,\Pi([n-2]\Delta t) = \cdots = [A]^n\,\Pi(0) \tag{5.10}$$

Hence, the solution to $\Pi(t)$ is

$$\Pi(t) = [A]^n\,\Pi(0) \tag{5.11}$$

where [A] is the coefficient matrix of the set of difference equations and Π(0) is known to be

$$\Pi(0) = \begin{bmatrix} P_1(t) \\ P_2(t) \\ \vdots \\ P_N(t) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.12}$$

So we can solve $R(t) = \sum_{i=1}^{N} P_i(t)$ by doing the following:

1. Select a sufficiently small Δt.
2. Determine $n = t/\Delta t$.
3. Determine the coefficient matrix [A] from the state-to-state transition rates and raising [A] to the n^{th} power—that is, performing n successive matrix multiplications of [A].
4. Finally, determine the $P_i(t)$ values.

5.2.2 Calculation of System Reliability from Markov Model --Example

Example [17]: An active generator has a failure rate (failures per day) of 0.01. An older standby generator has a failure rate of 0.001 while in standby and a failure rate of 0.10 when online. Determine the system reliability for a planned 3 day use.

To solve this problem, we first define the states as follow:

States: 1. Both generators operating

2. New generator X1 fails

3. Old generator X2 fails

4. Both generators fail

The failure rates are defined as follow:

λ_1 : The failure rate for new generator X1 which is equal to 0.01

λ_{2I} : The failrue rate for old generator X2 in standby mode (idle) which is equal to 0.001

λ_{2F} : The failure rate for old generator X2 when online (functioning) which is equal to 0.1

And then obtain the state diagram as shown in Figure 5.3.

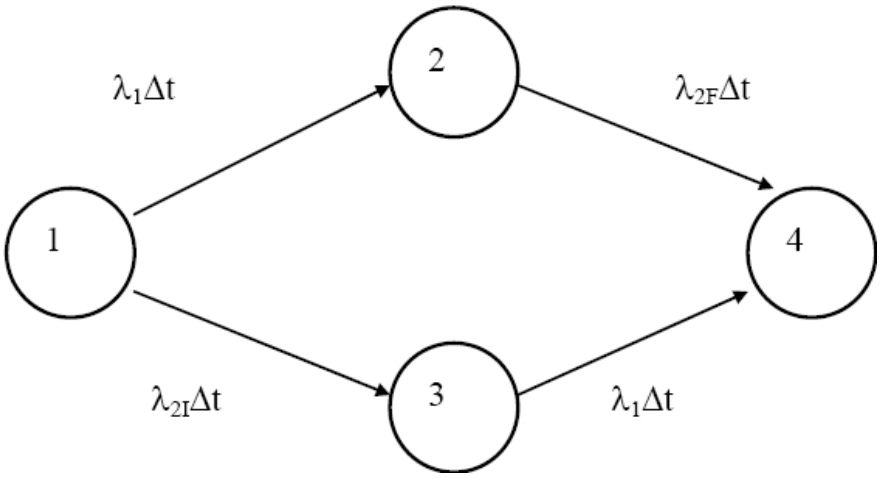


Figure 5.3: State Diagram Example

From the given data we know $\lambda_1 = .01$, $\lambda_{2I} = .001$, $\lambda_{2F} = .1$, mission time is 3 days. If we partition mission time to 1000 units, that is n=1000, Δt will be equal to 3/1000=0.003 days. So

$$[A] = \begin{bmatrix} (1 - r_{11}\Delta t) & r_{21}\Delta t & r_{31}\Delta t & r_{41}\Delta t \\ r_{12}\Delta t & (1 - r_{22}\Delta t) & r_{32}\Delta t & r_{42}\Delta t \\ r_{13}\Delta t & r_{23}\Delta t & (1 - r_{33}\Delta t) & r_{43}\Delta t \\ r_{14}\Delta t & r_{24}\Delta t & r_{34}\Delta t & (1 - r_{44}\Delta t) \end{bmatrix}$$

since $r_{12}\,\Delta t = \lambda_1\,\Delta t = 0.01(0.003) = 0.00003$

$r_{13}\,\Delta t = \lambda_{2I}\,\Delta t = 0.001(0.003) = 0.000003$

$r_{24}\,\Delta t = \lambda_{2F}\,\Delta t = 0.1(0.003) = 0.0003$

$r_{34}\,\Delta t = \lambda_{1}\,\Delta t = 0.01(0.003) = 0.00003$

$r_{11}\,\Delta t = (\lambda_{1} + \lambda_{2I})\Delta t = 0.011(0.003) = 0.000033$

$r_{22}\,\Delta t = \lambda_{2F}\,\Delta t = 0.1(0.003) = 0.0003$

$r_{33}\,\Delta t = \lambda_{1}\,\Delta t = 0.01(0.003) = 0.00003$

$r_{44}\,\Delta t = 0\,\Delta t = 0$

$$[A] = \begin{bmatrix} 0.999967 & 0 & 0 & 0 \\ 0.00003 & 0.9997 & 0 & 0 \\ 0.000003 & 0 & 0.99997 & 0 \\ 0 & 0.0003 & 0.00003 & 1 \end{bmatrix}$$

$$\Pi(t) = \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} = [A]^{1000}\ \Pi(0) = \begin{bmatrix} 0.967538 & 0 & 0 & 0 \\ 0.025478 & 0.740785 & 0 & 0 \\ 0.002907 & 0 & 0.970445 & 0 \\ 0.004077 & 0.002996 & 0.029555 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.967538 \\ 0.025478 \\ 0.002907 \\ 0.004077 \end{bmatrix}$$

That means during the mission the probability of state 1 (both generators operating) is $P_1(t)$ = 0.967538, the probability of state 2 (generator X1 fails, but X2 runs) is $P_2(t)$ = 0.025478, the probability of state 3 (generator X2 fails, but X1 runs) is $P_3(t)$ = 0.002907, and the probability of state 4 (both generators fail) is $P_4(t)$ = 0.004077. So the reliability of the system is R_s= $P_1(t)+P_2(t)+P_3(t)$ = 0.99592. This is an approximate result. An exact reliability value 0.99596 for this example can be obtained by the formula below which is derived by Laplace method.

$$R(t) = e^{-\lambda_1 t} + \frac{\lambda_1}{\lambda_1+\lambda_{2F}-\lambda_{2I}}\left[e^{-\lambda_{2I} t} - e^{-(\lambda_1+\lambda_{2F})t}\right] \qquad (5.13)$$

CHAPTER 6

FAULT TREE ANALYSIS

6.1 Fault Tree Configuration

The fault tree technique was introduced in 1962 at Bell Telephone Laboratories, in connection with a safety evaluation of the launching system for the intercontinental Minuteman missile. Today fault tree analysis is one of the most commonly used techniques for reliability and safety analysis.

A fault tree is a logic diagram that displays the interrelationships between a potential critical event (accident) in a system and the causes for this event. The cause may be environmental conditions, human errors, normal events, and specific component failures [27].

The basic symbols used in the construction of a fault tree include logic gates and events. Logic gate symbols are used to represent when a particular event can occur. The AND gate (*) describes the logical operation that requires the coexistence of all input events to produce the output event. The OR gate (+) describes that an output event occurs if any of the input events occur. There are three basic types of events that can occur in a fault tree. These are top event, intermediate events and terminal events. The top event (undesired event) appears at the top of the fault tree and is placed within a rectangle. An intermediate event is any event within the fault tree that is further resolved into events that could cause it. These are represented by rectangles. A terminal event (Sink event) is an event that is not resolved into further causes and is represented by either circles or diamonds [18].

6.2 Determination of Minimal Cut-Set and Tie-Set from Fault Tree

The major Fault Tree Algorithms are MOCUS [28] which is based on PREP – KITT (Downward) [29], TREEL based on MICSUP (upward) [30], ELRAFT [31] based on unique factorization property of natural numbers 1 11…19, 111…119, …

MOCUS is probably the most famous algorithm to compute minimal cut-sets of fault trees, event trees, block diagrams, etc. It represents the class of top-down algorithms. The following example is from McCalley [32]. Consider the power system illustrated in Figure 6.1.

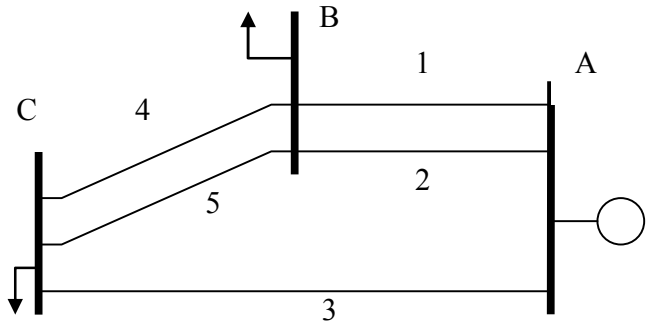


Figure 6.1: System for Fault Tree Example

The generator at station A represents power inflow that can be perfectly reliable for purposes of this example. Define "system failure" to be

1. Station B is isolated or
2. Station C is isolated or
3. The combined load of stations B and C are carried by a single circuit.

A fault tree is drawn for this system (note the answer is not unique) as shown in Figure 6.2.

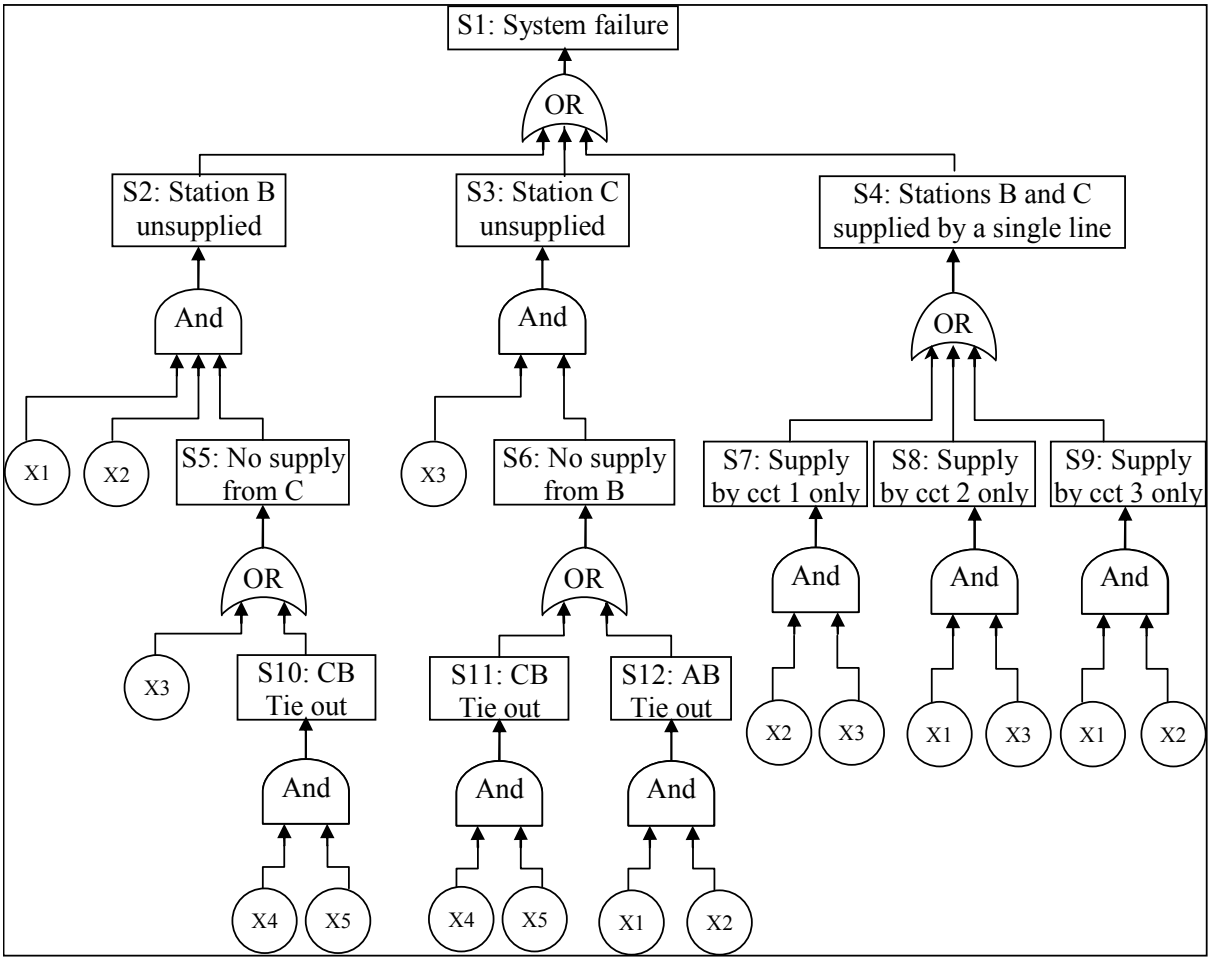


Figure 6.2: Fault Tree

Our approach to identifying the minimal cut-sets from a fault tree assumes that we will make the identification by analyzing the fault tree from top to bottom. In doing so, we make two important observations:

1. As we proceed down the tree from the top event, whenever we pass through an AND-gate, it means that all of the inputs to the gate must occur in order for the output to occur; as a result, the cut-set of interest increases in cardinality by adding the events corresponding to the AND-gate inputs.

2. As we proceed down the tree from the top event, whenever we pass through an OR-gate, it means that any of the inputs to the gate can occur in order for the output to occur; as a result, the number of cut-sets increases by adding additional cut-sets corresponding to the original cut-set plus one of the OR-gate input events.

The following cut-set identification algorithm follows from these two observations.

Step1: Alphabetize the gates.

Step2: Label each basic failure event.

Step3: Locate the uppermost gate in the first row and first column of a matrix.

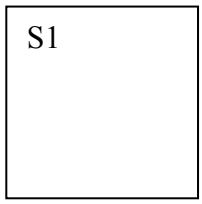
Step4: Iterate either of the fundamental permutations (a) or (b) in a top-down fashion.

a. Replace AND gates by a horizontal arrangement of the input to the gates, and enlarge the size of the cut-sets.
b. Replace OR gates by a vertical arrangement of the input to the gates, and increase the cut-sets

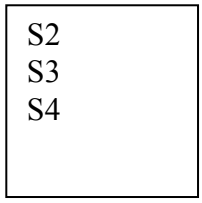
Step5: When all gates are replaced by basic events, obtain the minimal cut-sets by removing supersets. A superset is a cut-set that includes some other cut-sets.

This algorithm is illustrated on the example described Figure 6.2, as follows:

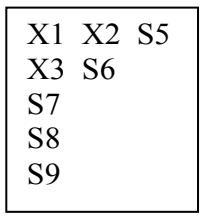
We begin with the top event, label it as system S1.



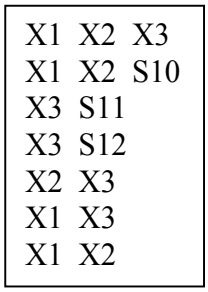
The gate below S1 is an OR gate, so we replace S1 with a vertical arrangement of the inputs to S1, resulting in:



Sub-system S2 and S3 are both AND gates, with inputs {X1, X2, S5}, and {X3, S6}, respectively, so replace S2 and S3 with these horizontal expansions. Sub-system S4 is an OR gate, with inputs {S7, S8, S9}, so we replace S4 with this vertical expansion. These changes result in:



Sub-systems S5 and S6 are OR gates, with inputs {X3, S10} and {S11, S12}, respectively, so replace S5 and S6 with these vertical expansions. Sub-systems S7, S8, and S9 are AND gates with inputs {X2, X3}, {X1, X3}, and {X1, X2}, respectively, so replace S7, S8, and S9 with these horizontal expansions.



Sub-systems S10, S11, and S12, are all AND gates with inputs {X4, X5}, {X4, X5}, and {X1, X2}, respectively, so replace S10, S11, and S12 with these horizontal expansions.

X1 X2 X3
X1 X2 X4 X5
X3 X4 X5
X3 X1 X2
X2 X3
X1 X3
X1 X2

There are 3 sets which are not minimal cut-sets: {X1, X2, X3}, {X1, X2, X4, X5}, {X3, X1, X2}, and these can be eliminated. We have remaining the minimal cut-sets of {X3, X4, X5}, {X2, X3}, {X1, X3}, and {X1, X2}. There are no other minimal cut-sets because a properly constructed fault tree must produce all cut-sets. In this example, we may verify the list by observing its significance with respect to Figure 6.2.

Once the minimal cut-sets are obtained, system reliability can be calculated according to the method described in section 4.6.6 . Assuming the unreliability of components X1, X2, X3, X4, X5 are R_1'=0.1, R_2'=0.2, R_3'=0.3, R_4'=0.4, R_5'=0.5 respectively, the system's reliability will be 0.85880 which was obtained in section 4.6.6 .

CHAPTER 7

SOFTWARE DESIGN AND VALIDATION

7.1 Software Development

A Software Tool for Reliability Estimation 2.0 (STORE2.0) was developed under Microsoft Windows XP operating system using Microsoft Visual Basic 2008 development tool, and Microsoft SQL server 2005 as Database Engine. It is significantly different from the earlier version developed by Parekh [1]. Some of major enhancements are listed below:

1. Fault tree reliability analysis was added.
2. A more efficient algorithm for tie-set and cut-set calculation was developed.
3. State Dependent Systems can be used to analyze any multiple states Markov model.
4. The software is developed under the latest development tool Visual Basic 2008 and database is implemented for user data storage.

This software tool also differs significantly from commercial software. The comparison is shown in Table 7.1.

Table 7.1: Comparison with commercial software

	STORE2.0	Isograph [33]	Relex [34]	Item [35]	ReliaSoft [36]
Failure Data Analysis	√	×	×	×	×
Parameter Estimation	√	×	×	×	×
Cut-set Identification	√	√	√	√	×
Tie-set Identification	√	×	√	×	√
RBD	√	√	√	√	√
Fault Tree Analysis	√	√	√	√	√
State Dependent Systems	√	√	√	√	√

7.2 Database Structure

STORE 2.0 uses the SQL data base to store data and settings for users. When a user creates a new project, a SQL database file (with extension of mdf) is created automatically. When a project is saved, all information including component, system, fault tree, block diagram, and Markov model are saved to this file, so that the user can simplify open only one file and easily get all needed information to continue their work later. A database file is designed to have different tables for different functions and each table has its own data structure. The database has seven tables as shown in Table 7.2.

Table 7.2: Database Tables

Table Name	**Description**
Components	Store information for components
Systems	Store information for systems
FaultTreeConnection	Store the fault tree configuration
RBDcnMatrix	Store connection information for Reliability Block Diagram
MarkovState	Store State information for Markov Model
MarkovTransitions	Store connection matrix for Markov Model
OtherInfo	Other information needed to be stored, such as setup information

Detail structure information for each table is as follow:

Table 7.3: Component Database Table

Field Name	Data Type	Description
Comp_ID	text	Key field. Component identification
Comp_Name	text	Component name
Reliability	float	Reliability of the component
Failure_Data	text	Failure data of the component
Description	text	Description of the component
Mission_Time	float	Mission time for the reliability calculation
Mission_Time_Unit	text	Mission time unit for the reliability calculation
Distribution	text	Distribution of the failure data
Dist_Parameter1	float	Distribution parameter one
Dist_Parameter2	float	Distribution parameter two
Dist_Parameter3	float	Distribution parameter three
Data_Type	text	Data type of the failure data
Test_Time	float	Test time of the failure data
Total_NO_of_Data	int	Total number of the tested items when collecting failure data
Level_of_Significance	float	Level of significant required for reliability computation

Table 7.4: System Database Table

Field Name	Data Type	Description
System_ID	text	Key field. System identification
Description	text	Description of the system
Structure	text	Structure of the system
Reliability	float	Reliability of the system
System_Name	text	System name
Gate	text	Gate of the fault tree

Table 7.5: FaultTreeConnection Database Table

Field Name	Data Type	Description
Output_Event_Name	text	Output event name of the fault tree
Gate	text	Gate of the fault tree
Input_Event_Name	text	Input event name of the fault tree
Event_Description	text	Description of the event

Table 7.6: RBDcnMatrix Database Table

Field Name	Data Type	Description
Begin_Node	int	Begin node
End_Node	int	End node
Component	text	Component between begin node and end node
Reliability	float	Reliability of the component

Table 7.7: MarkovState Database Table

Field Name	Data Type	Description
State	text	The name of the state
Description	text	Description of the state

Table 7.8: MarkovTransitions Database Table

Field Name	**Data Type**	**Description**
State_From	text	Begin state
State_To	text	End state
Description	text	Description of the transitions between begin state to end state
Rate	float	Rate of the transitions between begin state to end state

Table 7.9: OtherInfo Database Table

Field Name	Data Type	Description
RBD_Description	Text	Description of the Block Diagram project
FLT_Description	Text	Description of the Fault Tree project
Markov_Description	Text	Description of the Markov Model project
Markov_Initial_State	Text	Initial state of in the markov project
Markov_Unavailability_State	Text	Unavailability state of in the markov project
Markov_Mission_Time	Float	Mission of in the markov project
Markov_NO_of_Partitions	Int	Number of partitions of mission time
RBD_Source_Node	Text	Source node in the Block Diagram project
RBD_Sink_Node	Text	Sink node in the Block Diagram project
RBD_Accuracy	Float	Accuracy required in the Block Diagram project
FLT_Accuracy	Float	Accuracy required in the Fault Tree project
Active_Tab	Int	Which tab should be shown when the software is opened
Markov_Time_Unit	Text	Time unit of the mission time in Markov project
RBD_Mission_Time	Float	Mission time in Block Diagram project
RBD_Time_Unit	Text	Time unit of the mission time in Block Diagram
RBD_K	Int	Value of K in the K/N system
RBD_N	Int	Value of N in the K/N system
RBD_KN_Reliability	float	Value of reliability in the K/N system
FLT_Mission_Time	Float	Mission time in Fault Tree project
FLT_Time_Unit	text	Time unit of the mission time in Fault Tree project

7.3 Software Functions

The opening screen of STORE 2.0 is shown in Figure 7.1. There are five tabs (Component, Reliability Block Diagram, Fault Tree Analysis, Markov Analysis, and Tools) available for users to choose from. The following sections describe each of the five tabs.

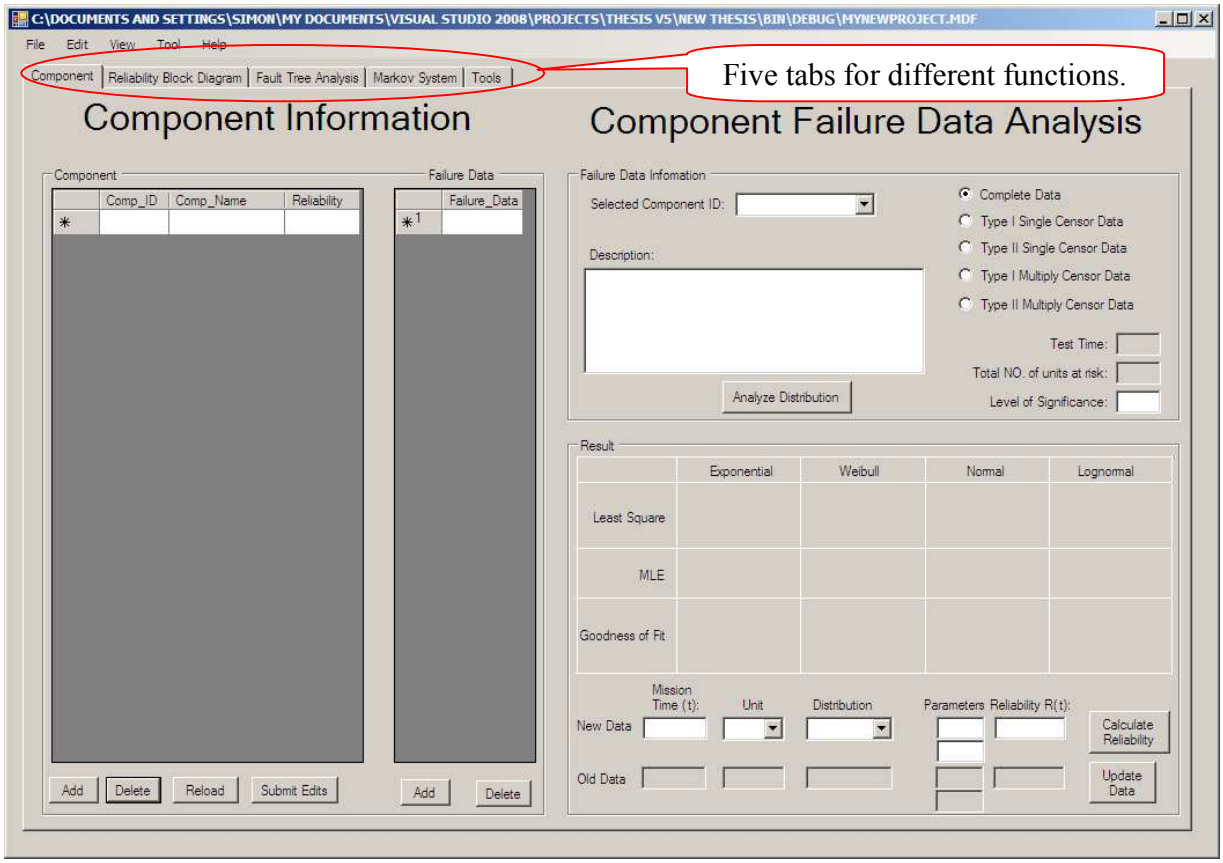


Figure 7.1: Software Functions Screen

7.3.1 Component Reliability Analysis

The following example will be used to illustrate how to compute component reliability and how to analyze the RBD.

Example: Let us say we have a six components system connected as shown in Figure 7.2.

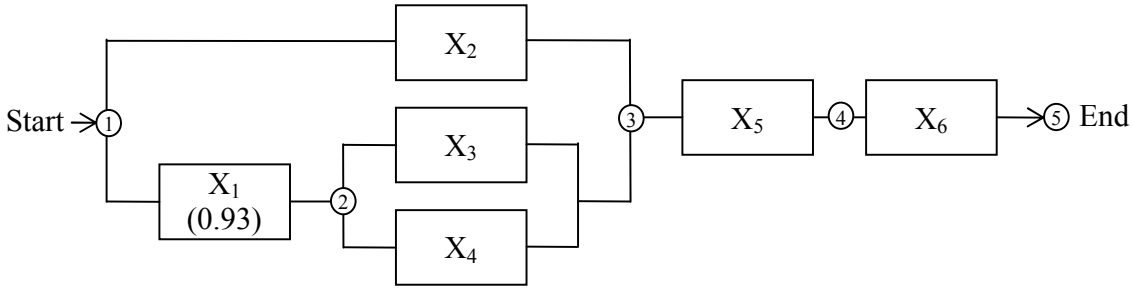


Figure 7.2: Series-Parallel Block Diagram System

The reliability of component X1 is known to be 0.93. For other components we don't know the reliability but we have the failure data as follows. We want to know all components' reliability and the system reliability for a mission time of 50 days with α=0.05.

Failure data of X2 (Complete Data) [17]

The following failure times (days) was obtained by testing 15 units until each unit failed:

Failure Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time (days)	25.1	73.9	75.5	88.5	95.5	112.2	113.6	138.5	139.8	150.3	151.9	156.8	164.5	218	403.1

Failure data of X3 (Type I Single Censor Data)

Twenty units were placed on a test for 90 days. The following 15 failure were observed prior to concluding the test.

Item Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time (days)	61.6	70	78.4	75.3	83.5	72.3	65.1	77.1	83.2	63.4	72.7	72.5	84.3	73	65.5

Failure data of X4 (Type II Single Censor Data) [17]

The following 35 failure times (days) were observed from among 50 units placed on test. The test was terminated at the 35^{th} failure (type II censoring).

Failure Number	1	2	3	4	5	6	7	8	9	10	11	12
Time (days)	1.3	7.3	7.8	13.3	13.9	19.4	19.7	22.3	22.8	26.7	29.7	30.2
Failure Number	13	14	15	16	17	18	19	20	21	22	23	24
Time (days)	31.9	32.2	33	36.8	37	41.7	46.7	50.4	51.4	60	61.3	61.4
Failure Number	25	26	27	28	29	30	31	32	33	34	35	
Time (days)	65.6	65.8	72.6	78.4	100.4	110.6	111.4	118.2	119.4	132.1	139.7	

Failure data of X5 (Type I Multiple Censor Data) [17]

Fifteen units were placed on test for 500 days. The following failure times and censored times were observed prior to concluding the test ("+" represent it is a censored (removal) time).

Failure Number	1	2	3	4	5	6	7	8	9	10	11
Time (days)	34	136	145+	154	189	200+	286	287	334	353	380+

Failure data of X6 (Type II Multiple Censor Data) [17]

Thirty motors are placed on test with failures occurring at the following cycle times. A cycle consists of a motor starting up to its maximum number of revolutions per minutes and then shutting down until it has come to a complete stop. Censored units resulted from motors being removed from test to satisfy other demands.

Failure Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Time	141	391	399	410+	463	465	497	501+	559	563	579	580+	586	616
Failure Number	15	16	17	18	19	20								
Time	683	707	713	742+	755+	764								

When the tab “Component” is clicked, the window of component reliability analysis will be shown like Figure 7.1. It contains two parts, Component Information (left hand side) and Component Failure Data Analysis (right hand side). Through the above example, we will illustrate how to use part one to enter data and analyze the result from part two.

In the first part Component Information, a list of component information can be created in the component table. For this example, we put X1, X2, X3, X4, X5 and X6 in the first column (Comp_ID) of the component table and put their corresponding names in the column of Comp_Name (suppose their names are A, B, C, D, E, F respectively). Since the reliability of component X1 is known, enter its value to the third column (Reliability) of the component table. The result is shown in Figure 7.3.

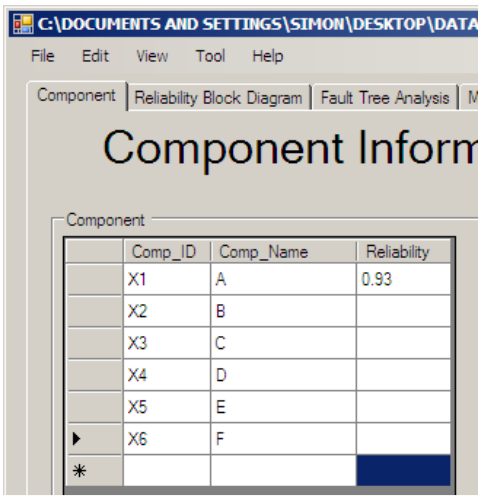


Figure 7.3: Entering Component Name

For each created component, failure data can be enter/display in the failure table. In this case, we first select component X2 in the drawing list “Selected Component ID” and then put all those failure data into the failure table. After that select day as the failure data units in the drawing list “Unit” and select “Complete data” as the data type (see section 3.1 for more information about data type). The result is shown in Figure 7.4.

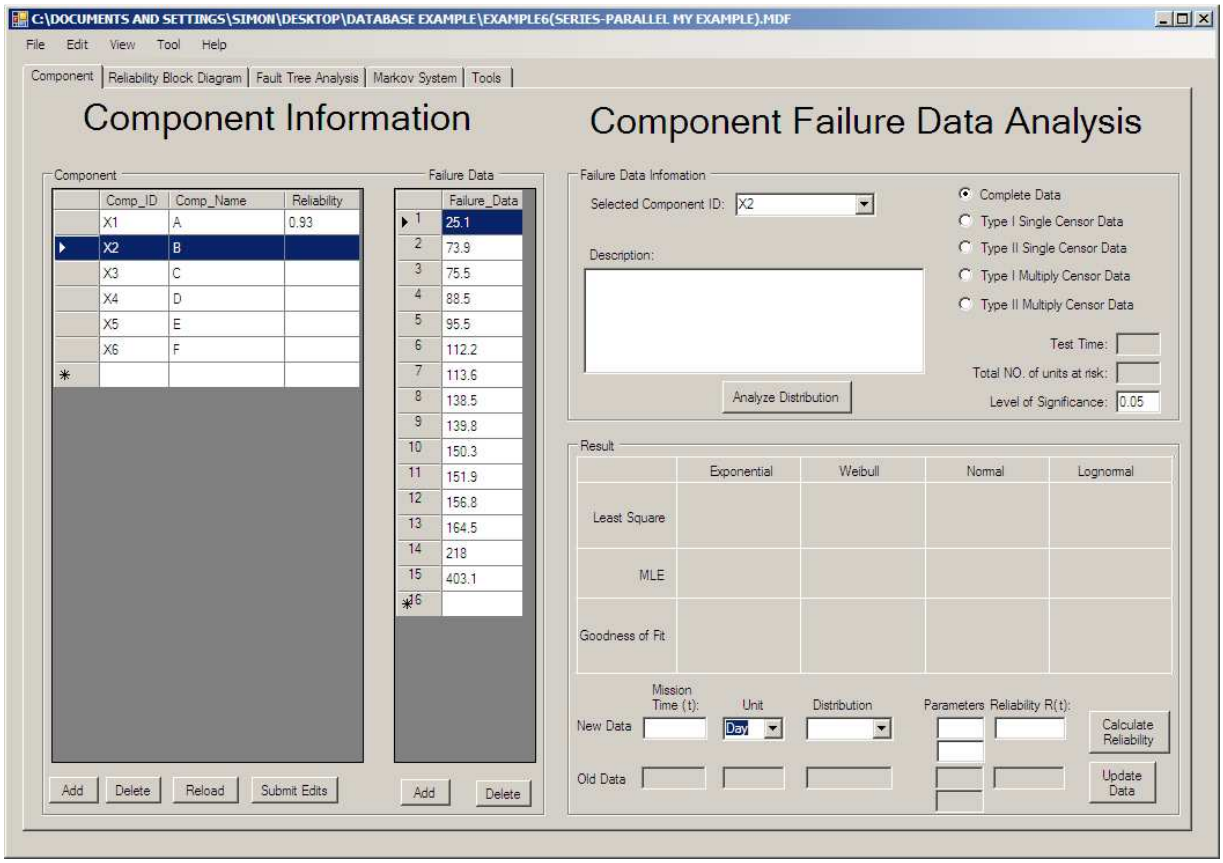


Figure 7.4: Input Component Failure Data

Failure data analysis can be done in the second part. After finishing the failure data information, and entering the required significant of level in the "Significant of Level" text box, user is ready to analyze the data. When the button "Analyze Distribution" is clicked, failure data are fitted to four common distributions (Exponential, Weibull, Normal, and Lognormal) and results are shown in the result panel. By choosing the best distribution and enter the mission time, reliability can be computed. In this example, the index of fit r of Weibull is higher than those of other distributions and the goodness-of-fit of Weibull is accepted, so Weibull distribution is the best choice. After selecting Weibull in the "distribution" drawing list, the fitted parameters will go to the "parameters" text box automatically. And then enter 50 in the mission time text box and then click the button "Calculate Reliability". The reliability comes out

to be 0.882271 which is shown in text box "Reliability *R(t)*". Click "Update Data", the reliability will be update to the component table. The result is shown in Figure 7.5.

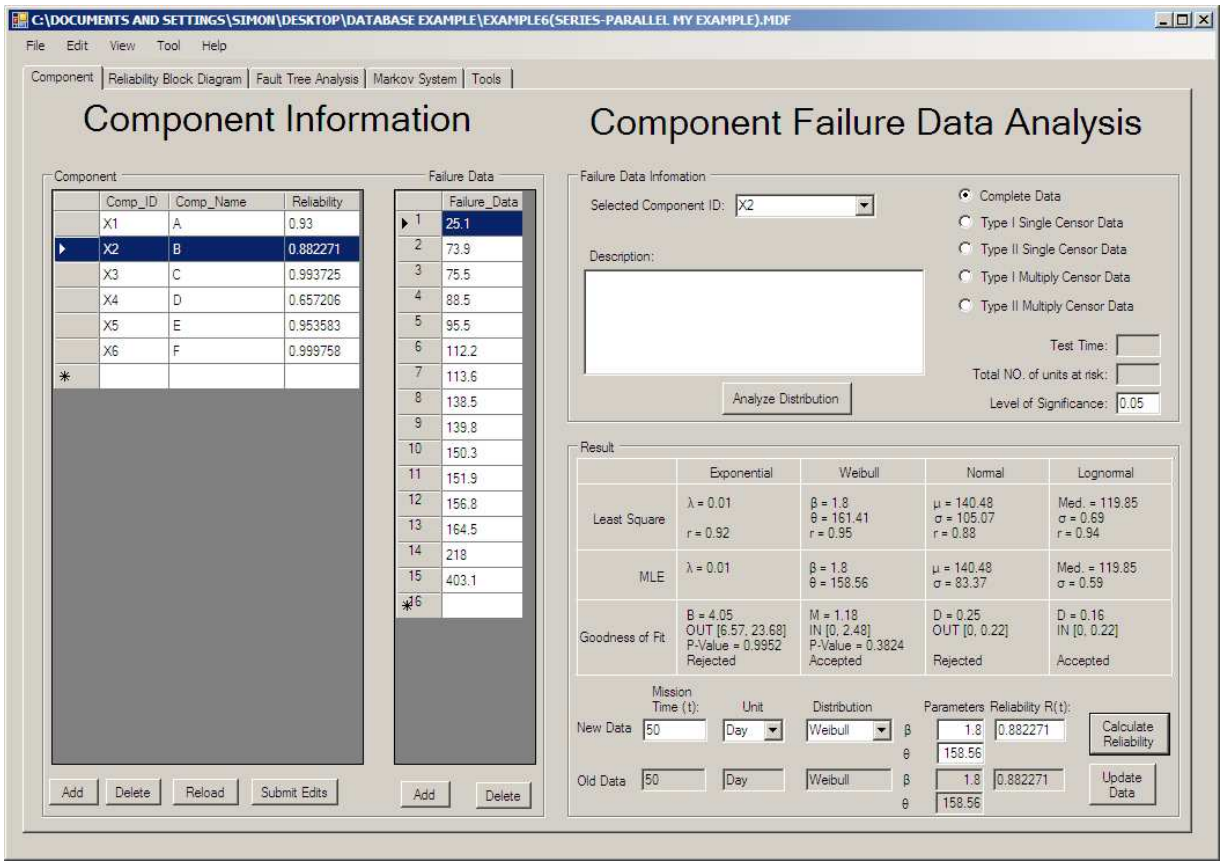


Figure 7.5: Component X2 Reliability Analysis

To analyze the reliability of component X3, we first put the failure data in the table. Since these are type I single censor data, we select "Type I single Censor Data". For this type of data, we need to enter test time (90) and total number of units (20) in the test. Select the best distribution (weibull) according to the analysis result and enter mission time (50) and time units (day). By clicking the button of "Calculate Reliability", the component reliability (0.993725) is attained as shown in Figure 7.6.

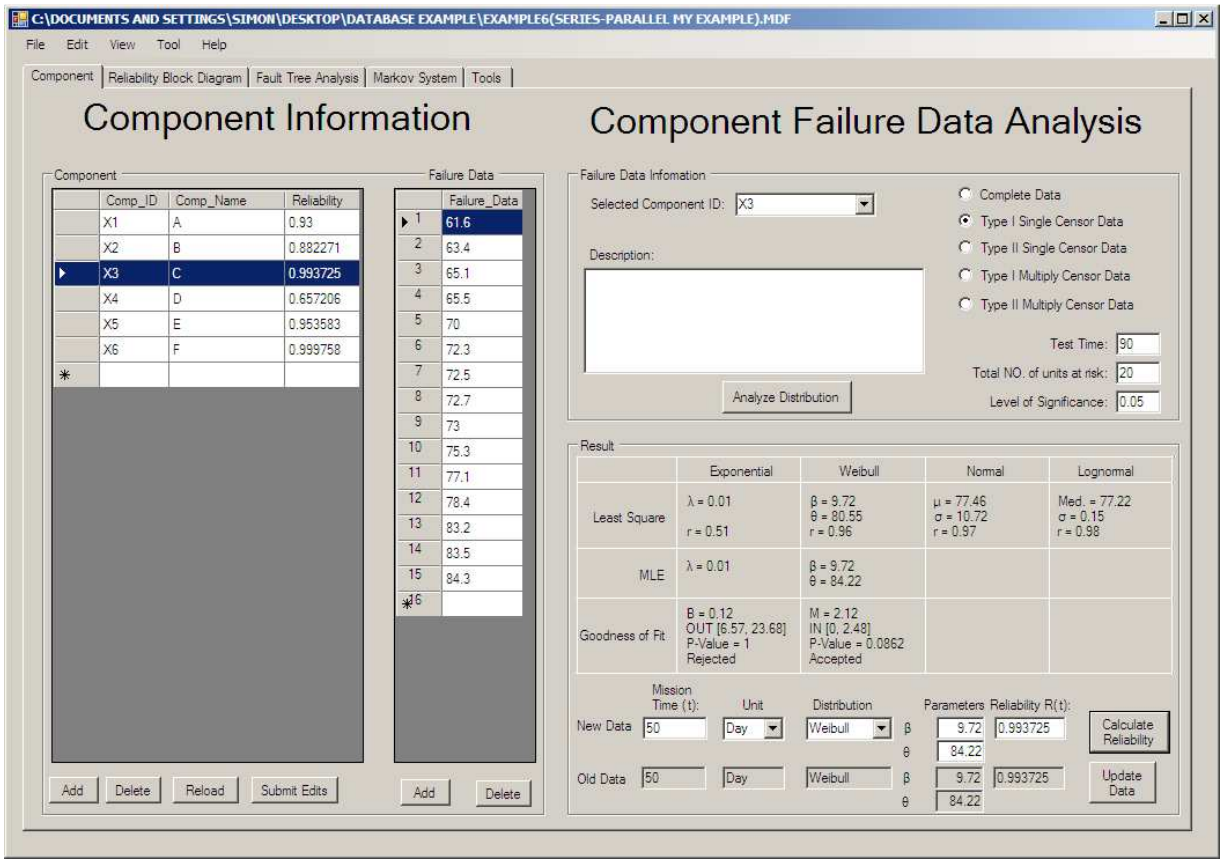


Figure 7.6: Component X3 Reliability Analysis

Reliability of component X4 can be computed in the same manner. The only difference is we need to choose “Type II single Censor Data” instead of “Type I single Censor Data”. The result is shown in Figure 7.7.

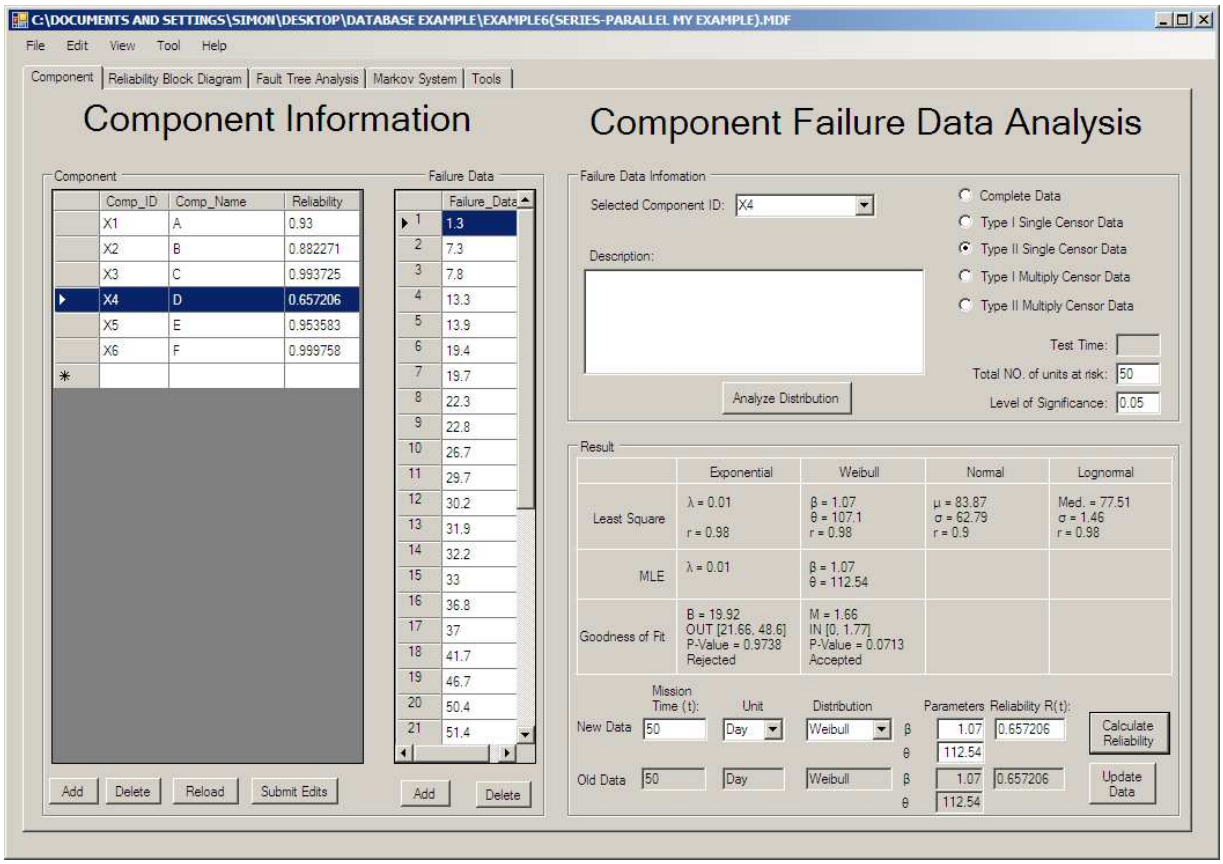


Figure 7.7: Component X4 Reliability Analysis

For component X5 and X6 with multiply censor failure data, "+" should be added following the time if it is a censored (removal) time. The analysis result of component X5 and X6 is shown in Figure 7.8 and Figure 7.9 respectively.

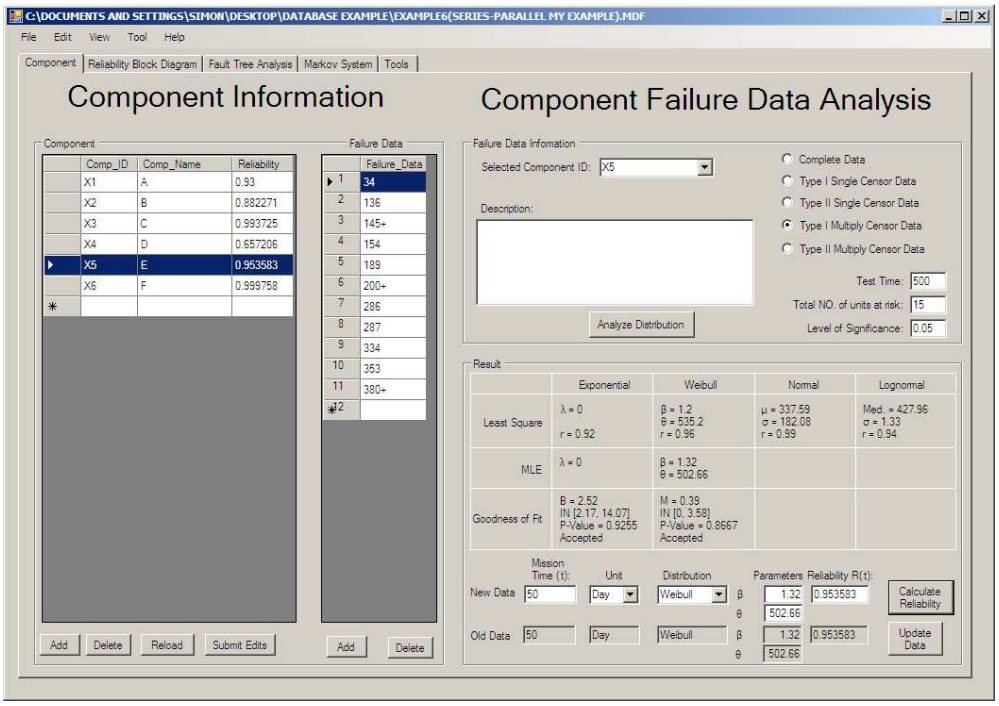


Figure 7.8: Component X5 Reliability Analysis

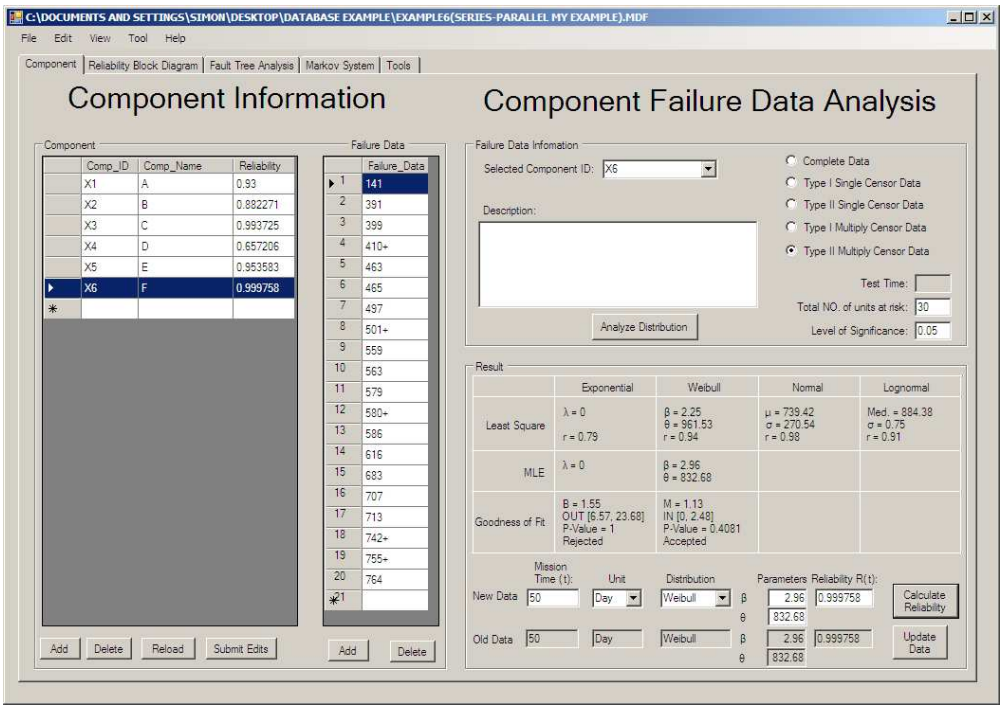


Figure 7.9: Component X6 Reliability Analysis

7.3.2 Reliability Block Diagram

To analyze the reliability of system in Figure 7.2, we first need to enter the structure of the system as shown in Figure 7.10.

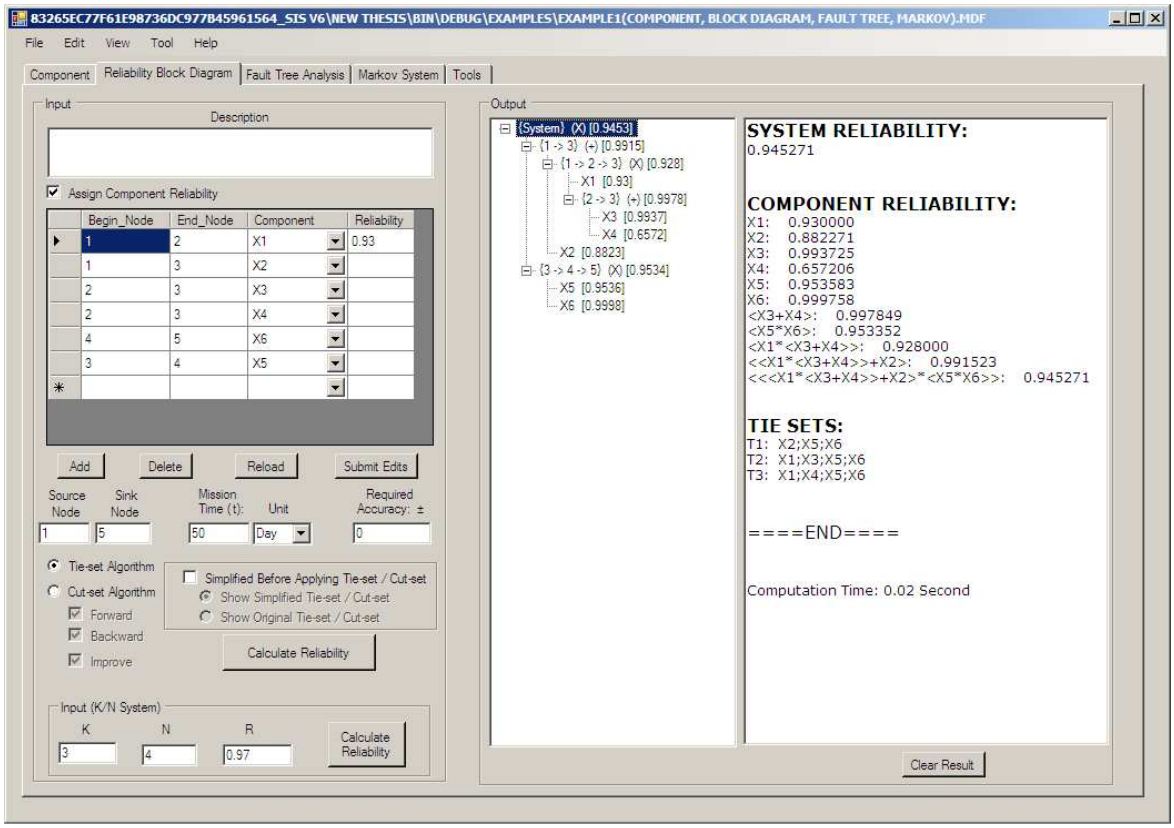


Figure 7.10: Reliability Block Diagram

Since the example has assigned reliability, the check box “Assign Component Reliability” should be checked. And then enter the assigned reliability 0.93 to the table and leave the other reliability to be blank. After entering the required accuracy (0, means we want an exact reliability value), source node (1) and sink node (5), mission time (50) and time unit (day), reliability result (0.945271) can be achieved by click the button “Calculate Reliably”. The analysis result is shown in Figure 7.10.

The Reliability Block Diagram not only can analyze series-parallel system such as example in Figure 7.2 but also can solve complex system. Figure 7.11 shows the result of the application

of block diagram analysis for complex example described in Figure 4.12, assuming each component has reliability of 0.9. The output shows the system reliability to be 0.997650 and reliability of all components and sub-system. Tie-sets and cut-sets are also included in the

The tree view shows the structure of the system. In this example, the system is complex as it is shown "System(complex)". This system has 3 sub-systems (1→2, 2→6, 3→5→6) and two components (X4, X5). 1→2 means the begin node and end node of this sub-system are 1 and 2 respectively. The reliability of sub-system 1→2 is 0.999 (shown as [0.999]) and it is composed of 3 parallel (shown as +) components X1, X2, and X3. In the same way, sub-system 3→5→6 means it's begin node is 3, end node is 6, and this sub-system is composed of two sub-systems (components) in series, 3→5 and 5→6.

On the lower left corner, a K/N system is available to calculate the system reliability.

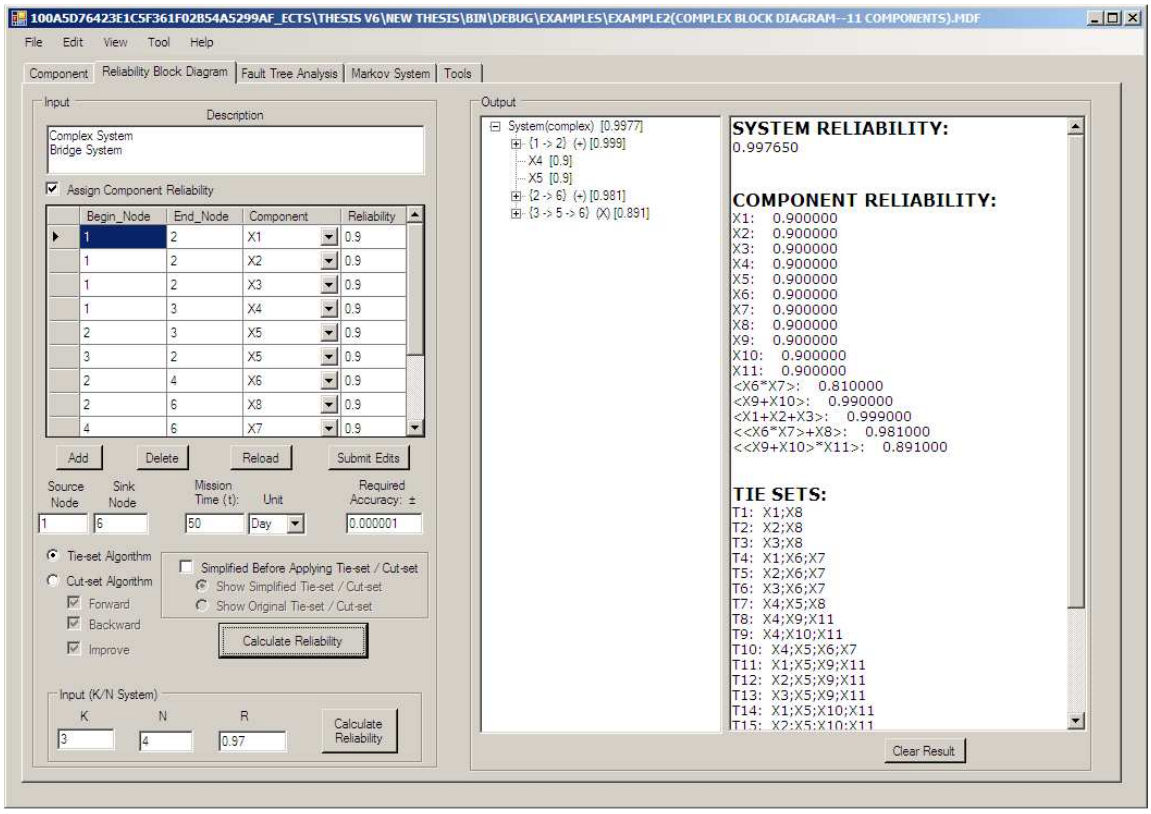


Figure 7.11: Software Block Diagram Screen

7.3.3 Fault Tree Analysis

Figure 7.12 shows the screen of Fault Tree Analysis when the tab "Fault Tree Analysis" is clicked. Those two tables in the left hand side are used to build the structure of the system. We will show how to use this function by building an example system shown in section 6.2 .

First, we build the top level system. In Figure 6.2 we see the top level system S1 has an "or" gate with sub-system S2, S3, and S4, so in the first row we put "S1" in the column of "System_ID", "System Failure" in the column of "System_Name", and "OR" in the column of "Gate". In the input event table, we put "S2", "S3", and "S4" as shown in Figure 7.12. And then break down sub-system S2, S3 and S4. For sub-system S2 which has an "and" gate with component X1, component X2 and system S5, we will enter "S2" in the column of "System_ID",

"Station B unsupplied" in the column of "System_Name", and "AND" in the column of "Gate"; In the input event table, we put "X1", "X2", and "S5". After breaking down all sub-systems and we get the same table shown in Figure 7.12.

A tree view of the system is available on the middle of the window showing the structure of selected sub-system. We can change the selected sub-system by selecting the sub-system in the drawing list "Selected System ID", or click the sub-system in the table. When the selected system is changed the tree view will be updated automatically.

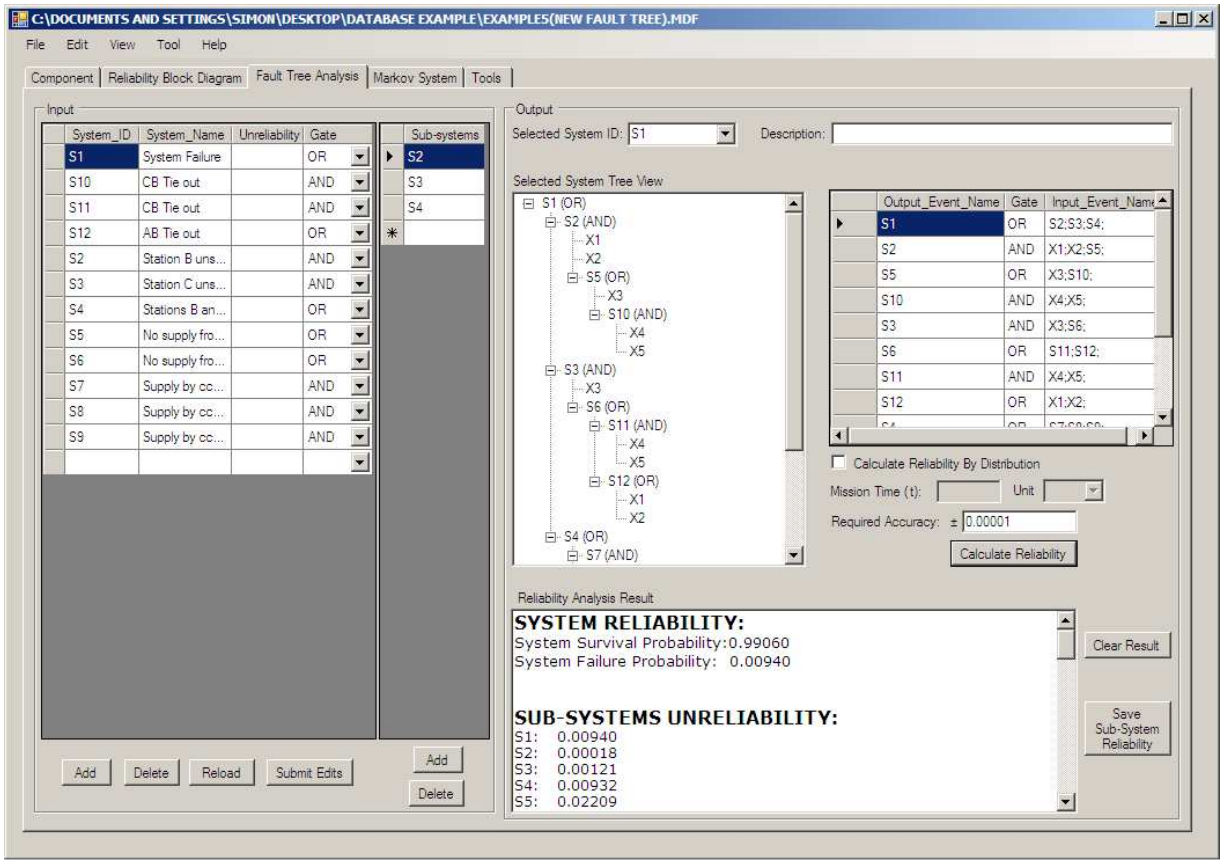


Figure 7.12: Fault Tree Analysis

To calculate the reliability, select the system wanted to be analyzed, enter the require accuracy, and then click the button "Calculate Reliability", the result will be shown on the bottom. But before clicking the "Calculate Reliability" button, make sure the component

database has all the components and their reliability. In this example, click tab "Component", we will see we already have those components (X1, X2, X3, X4, and X5) which are created in section 7.3.1 Component Reliability Analysis. Now we go back to the Fault Tree analysis and click the button "Calculate Reliability". The result in Figure 7.12 shows the reliability (0.99060) of the selected sub-system (sub-system S1). In the result we can also find the unreliability of all components and sub-systems under S1. The result also shows the tie-sets and cut-sets of sub-system S1.

7.3.4 Markov Reliability Analysis

Two tables are used for the Markov reliability analysis (Figure 7.13). The first one is for the state information and the second one for the state transition matrix information. To calculate the reliability, we need to partition the mission time to many parts. In theory, the more parts we partition to the more accuracy we get, however, when the number of parts increases, computer will lose some of the accuracy.

Figure 7.13 shows the result of the application of Markov Model for example in Figure 5.2, assuming $\lambda_{12} = 0.01$, $\lambda_{13} = 0.1$, $\lambda_{24} = 0.001$, $\lambda_{34} = 0.01$, mission time =3 days. The output shows the system reliability and the each state's reliability.

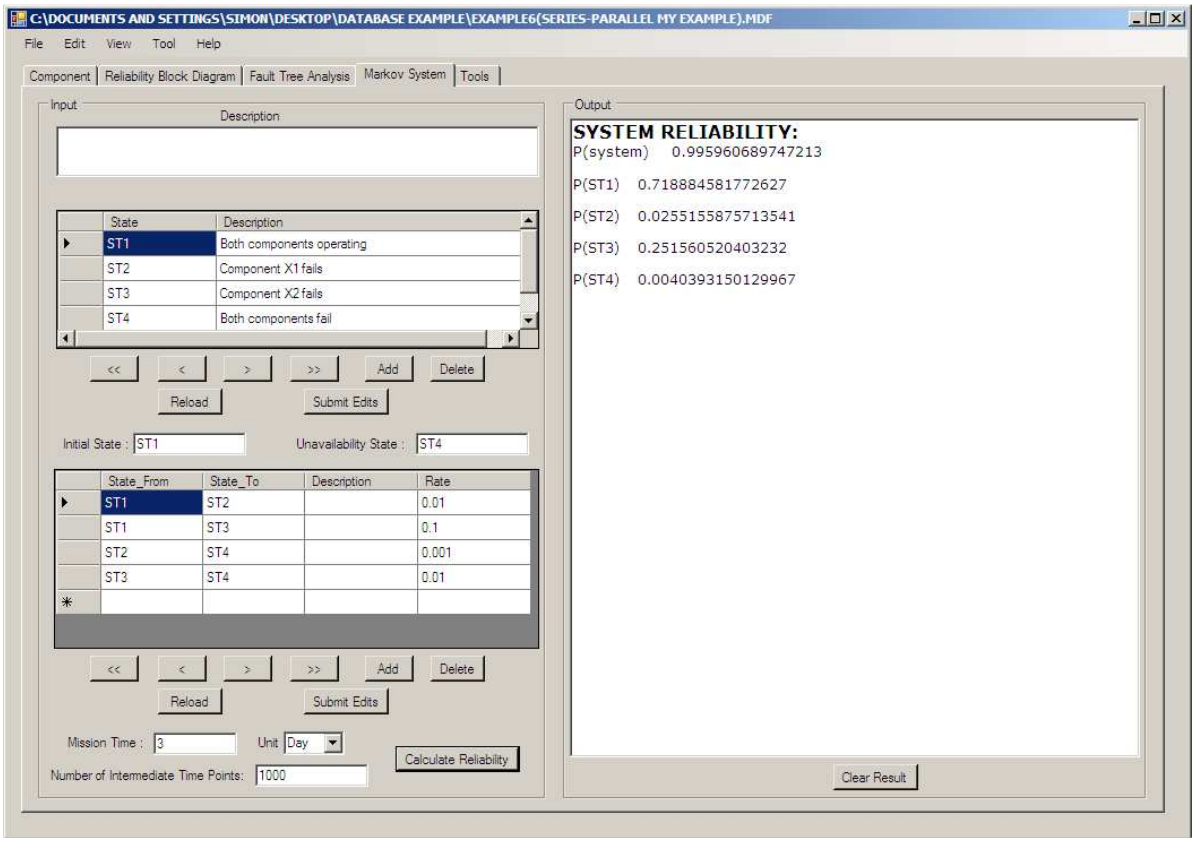


Figure 7.13: Software Markov Model Screen

7.4 Software Validation

7.4.1 Failure Data Analysis

In this section examples for different distribution will be described and the results obtained will be compared with the ones existing in the literature.

Exponential Distribution

The following are complete data representing the failure time (hours) of the tested items [17] (assuming α =0.05):

Failure Number	1	2	3	4	5	6	7	8	9	10
Time (Hours)	3.3	4.2	12.9	13.8	14.3	14.8	18.5	22.8	27.1	29.7
Failure Number	11	12	13	14	15	16	17	18	19	20
Time (Hours)	32	39.5	41.3	41.6	51.1	61.7	92.2	106.6	148.8	198.1

a. Least Square

According to Table 3.2 and formulas in section 3.4.1 , the calculation results for this example are: $\lambda = 0.02$ $r = 0.98$

b. MLE

As discussed in section 3.5.2 , the formula for exponential MLE analysis is as below:

$$\lambda = f/T$$

since $T = 3.3+4.2+12.9+\ldots+198.1=974.3$

$$f=20$$

$$\lambda=0.02$$

c. Goodness-of-fit

As discussed in section 3.6.1 , Bartlett's Test should be used for exponential. The calculation results for this example are:

$$B= 16.49$$

$$10.12 < B< 30.14$$

As the value of B falls within the range, the distribution is proved to be exponential.

d. Reliability

Assume mission time to be 10 hours, the formula for the reliability calculation, as shown in section 3.3.2 , is:

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

$$R(10) = e^{-0.02(10)} = 0.818731$$

Doing this example by hand involves lot of tedious calculation, but when the same set of failure data is run in the software, it gives the accurate results as shown in Figure 7.14.

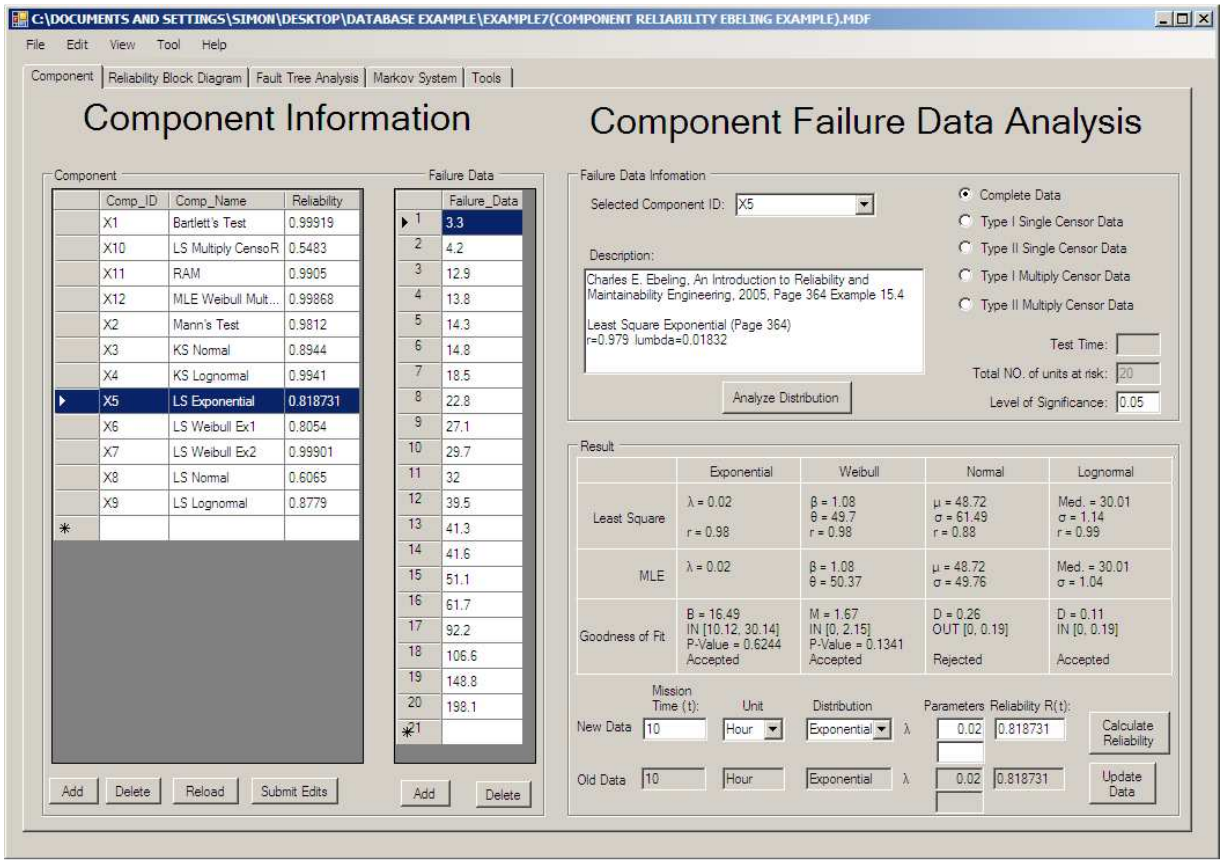


Figure 7.14: Software Exponential Result

Weibull Distribution

The following failure times were obtained from testing 15 units until each had failed [17] (assuming α =0.05):

Failure Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time (days)	25.1	73.9	75.5	88.5	95.5	112.2	113.6	138.5	139.8	150.3	151.9	156.8	164.5	218	403.1

a. Least Square

According to Table 3.2 and formulas in section 3.4.1 , the calculation results for this example are:

$$\beta = 1.8$$

$$\theta = 161.41$$

$$r = 0.95$$

b. MLE

As discussed in section 3.5.2 , newton-Raphson method for solving a nonlinear equation is used. By initiating $\beta = 1.8$ which is obtain in MLE, the answer for β and θ are calculated as follow

$$\beta = 1.8$$

$$\theta = 158.56$$

c. Goodness-of-fit

As discussed in section 3.6.1 , Mann's Test should be used for Weibull. The calculation results for this example are:

$$M = 1.18$$

$$M < 2.48$$

As the value of M is smaller than the critical value, the distribution is proved to be Weibull.

d. Reliability

Assume mission time to be 10 hours, the formula for the reliability calculation, as shown in section 3.3.2 , is:

$$R(t) = e^{-(t/\theta)^{\beta}}$$

$$R(10) = e^{-(10/158.56)^{1.8}} = 0.993111$$

When the same set of failure data is run in the software, it gives the accurate results as shown in Figure 7.15.

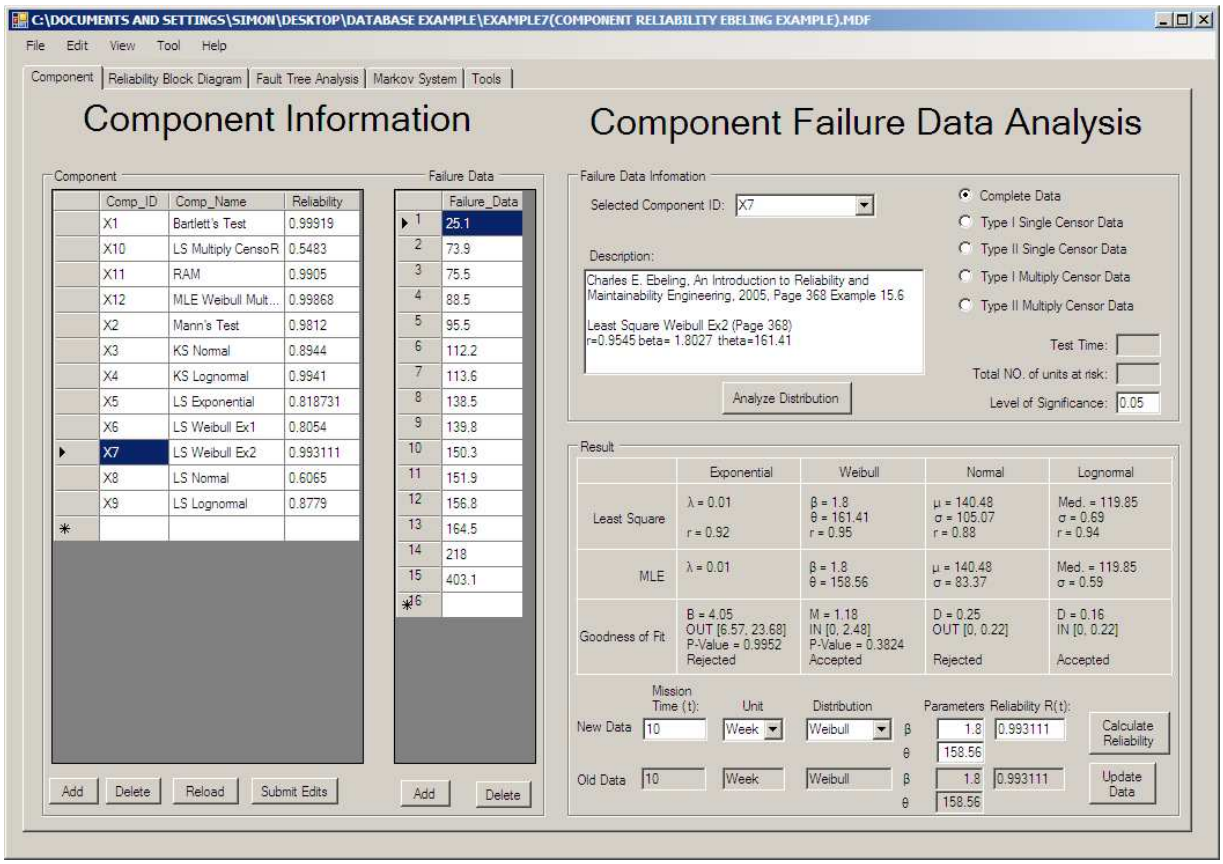


Figure 7.15: Software Weibull Result

Normal Distribution

The following 15 observations represent a sample of the repair times, in hours, of a complex piece of machinery [17] (assuming α =0.10):

Item Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time (Hours)	61.6	70	78.4	75.3	83.5	72.3	65.1	77.1	83.2	63.4	72.7	72.5	84.3	73	65.5

a. Least Square

According to Table 3.2 and formulas in section 3.4.1 , the calculation results for this example are:

σ =7.93

$\mu = 73.19$

r=0.98

b. MLE

As discussed in section 3.5.2 , the formula for Normal MLE analysis is as below:

$$\mu = \bar{x}$$

$$\sigma^2 = \frac{(n-1)s^2}{n}$$

The calculation results for this example are: σ =7.04

μ = 73.19

c. Goodness-of-fit

As discussed in section 3.6.1 , Kolmogorov-Smirnov Test should be used for Normal. The calculation results for this example are:

$D = 0.13$

$D < 0.2$

As the value of *D* is smaller than the critical value, the null hypothesis regarding the normality of data is accepted.

d. Reliability

Assuming mission time to be 65 hours, the formula for the reliability calculation, as shown in section 3.3.3 , is:

$$R(t) = 1 - \Phi\left(\frac{t-\mu}{\sigma}\right)$$

$$R(65) = 0.88$$

When the same set of failure data is run in the software, it gives the accurate results as shown in Figure 7.16.

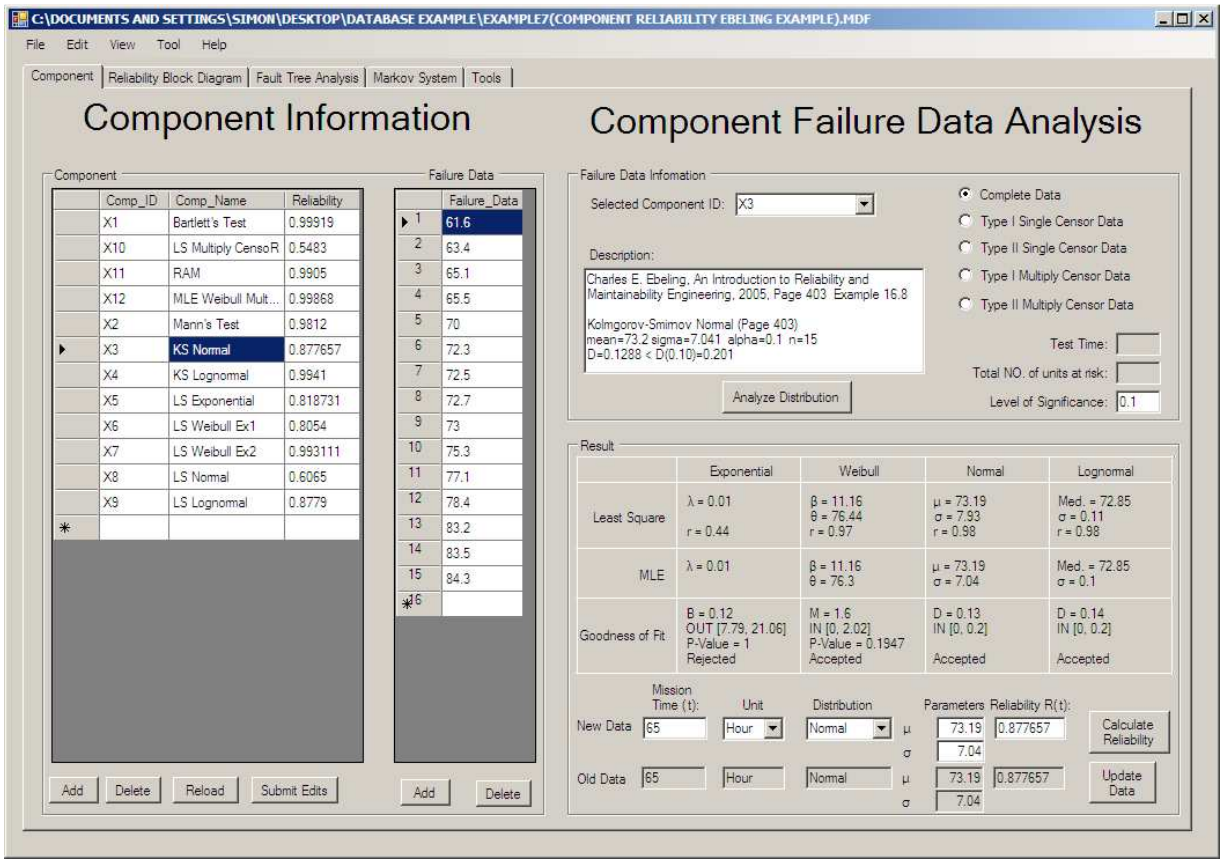


Figure 7.16: Software Normal Result

Lognormal Distribution

The time to failure of hose assemblies, due to structural fatigue and chemical breakdown, is believed to have a lognormal distribution. The following 25 failure times were obtained from environmental stress testing (complete data) [17] (assuming α =0.10):

Item Number	1	2	3	4	5	6	7	8	9	10	11	12	13
Time (Hours)	240.5	511.8	1083.4	821.3	1725.4	629.4	326.9	964.8	1677.8	282.3	652.3	639.2	1847.8
Item Number	14	15	16	17	18	19	20	21	22	23	24	25	
Time (Hours)	670.8	338.8	818.1	1407.5	4991	452	464.9	734.9	220.2	1078.1	1077.3	1773	

a. Least Square

According to Table 3.2 and formulas in section 3.4.1 , the calculation results for this example are:

$$s = 0.79$$

$$\mu' = 765.43$$

$$r = 0.99$$

b. MLE

As discussed in section 3.5.2 , the calculation results for this example are:

$$\sigma = 0.73$$

$$\mu' = 765.43$$

c. Goodness-of-fit

As discussed in section 3.6.1 , Kolmogorov-Smirnov Test should be used for Lognormal. The calculation results for this example are:

$$D = 0.08$$

$$D < 0.16$$

As the value of D is smaller than the critical value, the null hypothesis regarding the normality of data is accepted.

d. Reliability

Assuming mission time to be 200 hours, the formula for the reliability calculation, as shown in 3.3.4 , is:

$$R(t) = 1 - \Phi\left(\frac{1}{\sigma'} ln \frac{t}{\mu'}\right)$$

$$R(200) = 0.97$$

When the same set of failure data is run in the software, it gives the accurate results as shown in Figure 7.17.

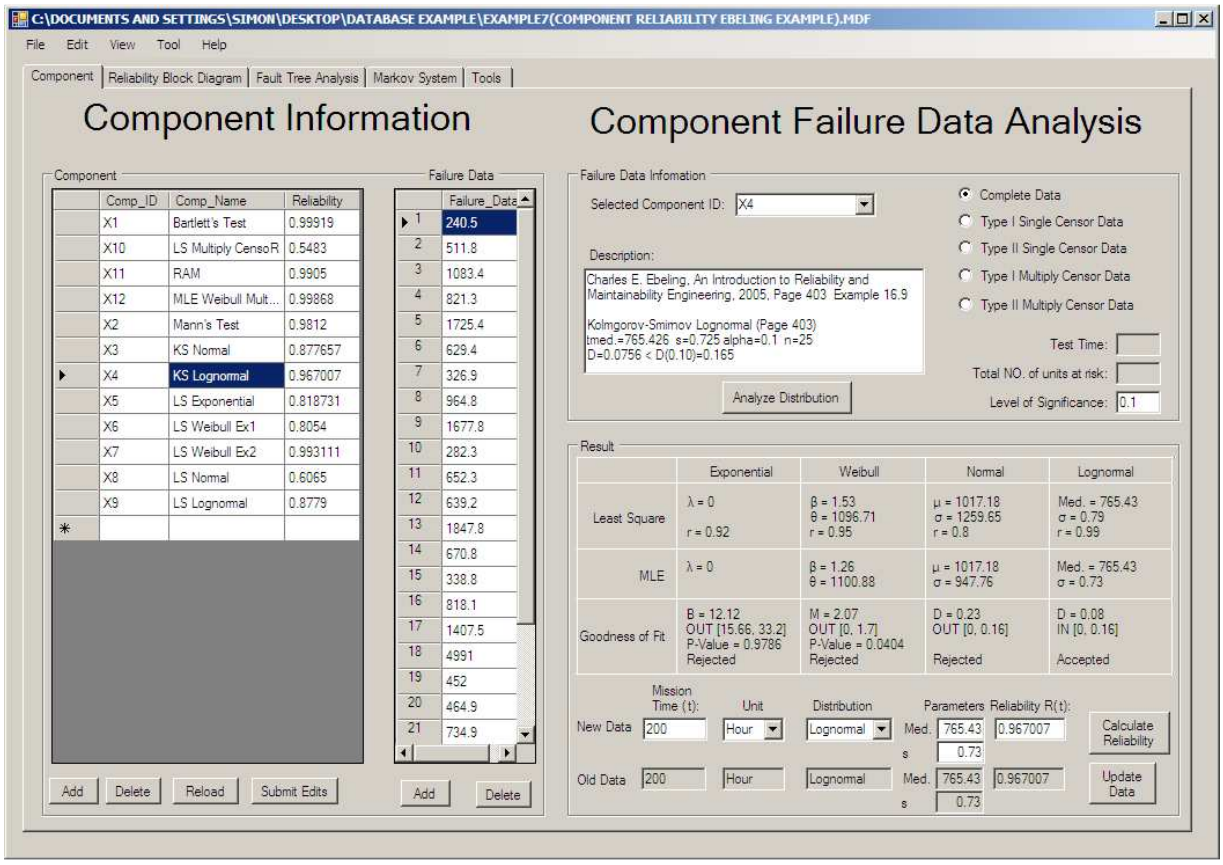


Figure 7.17: Software Lognormal Result

7.4.2 State Independent Systems

7.4.2.1 Series-Parallel systems

The system reliability of the series-parallel system example in Figure 4.5 is 0.97248 which was obtained in section 4.4 . The same result is obtained by the software as shown in Figure 7.18.

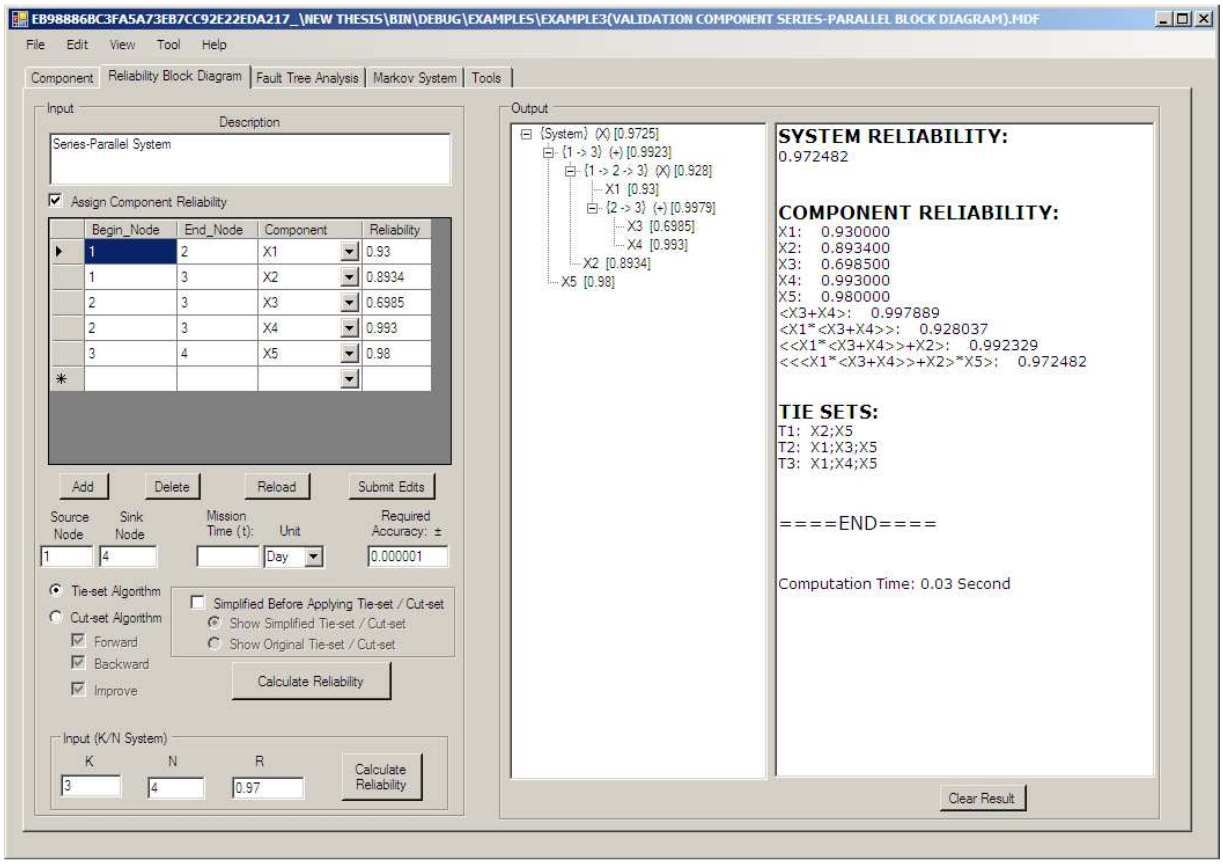


Figure 7.18: Software Series-Parallel systems Result

7.4.2.2 K/N Systems

A K/N system example shown in section 4.5 is recalculated by the software and the same result is obtained as shown in Figure 7.19.

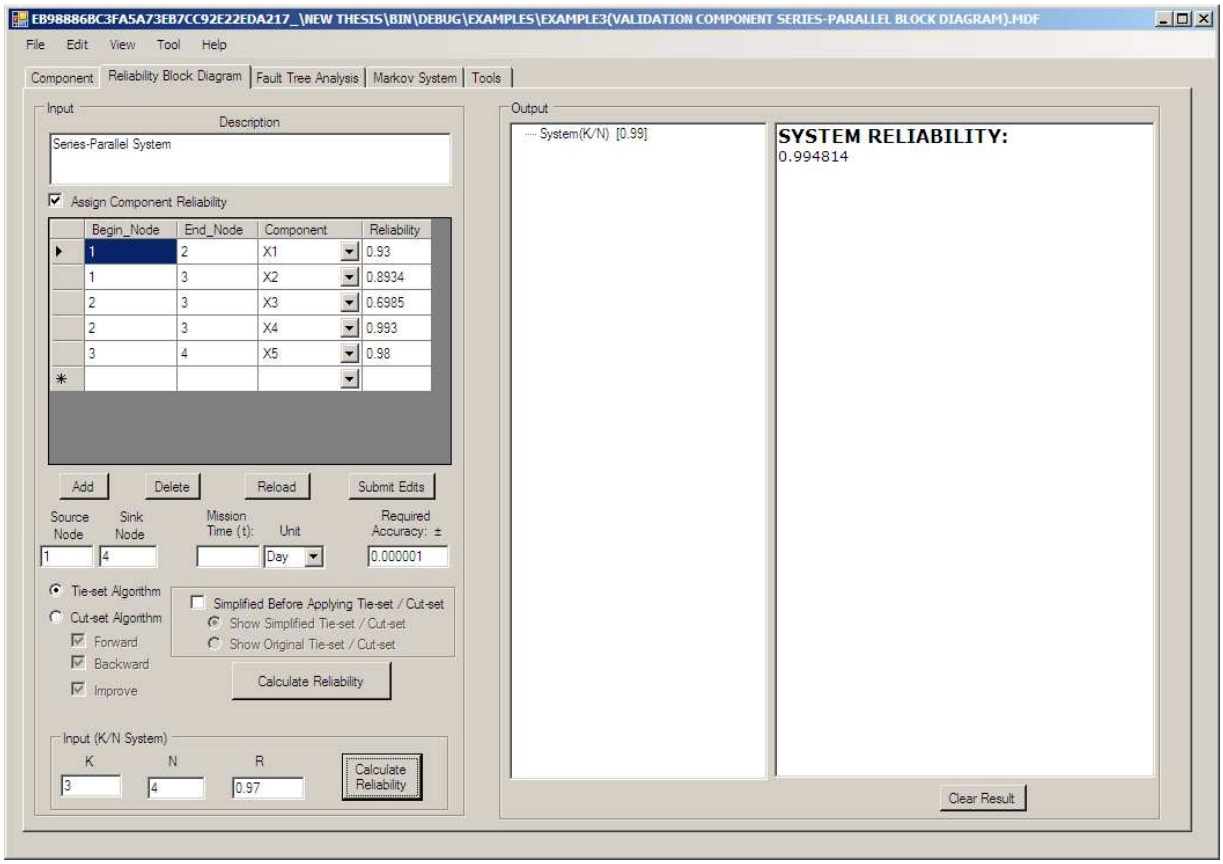


Figure 7.19: Software K/N Systems Result

7.4.2.3 Complex Systems

This section uses a system shown in Figure 4.14 from Nelson et al. [2] to test the software. System has 55 tie-sets and 10 cut-sets. All of the tie-sets and tie-sets are listed in Table 7.10 and Table 7.11 respectively.

Cut-Sets:

Table 7.10: Complex System Cut-Set Result

NO.	Cut-Sets	NO.	Cut-Sets
C1	X1,X2,X3	C6	X11,X12,X13,X14
C2	X3,X4,X5	C7	X7,X8,X10,X14
C3	X1,X2,X6	C8	X9,X10,X15,X16
C4	X4,X5,X6	C9	X11,X12,X13,X15,X16
C5	X10,X14,X9	C10	X7,X8,X10,X15,X16

Tie-Sets:

Table 7.11: Complex System Tie-Set Result

NO.	Tie-Set	NO.	Tie-Set	NO.	Tie-Set
T1	X1,X4,X10,X11	T20	X2,X5,X14,X16	T39	X2,X4,X7,X9,X12
T2	X1,X4,X10,X12	T21	X3,X6,X10,X11	T40	X2,X4,X7,X9,X13
T3	X1,X4,X10,X13	T22	X3,X6,X10,X12	T41	X2,X4,X8,X9,X11
T4	X1,X4,X14,X15	T23	X3,X6,X10,X13	T42	X2,X4,X8,X9,X12
T5	X1,X4,X14,X16	T24	X3,X6,X14,X15	T43	X2,X4,X8,X9,X13
T6	X1,X5,X10,X11	T25	X3,X6,X14,X16	T44	X2,X5,X7,X9,X11
T7	X1,X5,X10,X12	T26	X1,X4,X7,X9,X11	T45	X2,X5,X7,X9,X12
T8	X1,X5,X10,X13	T27	X1,X4,X7,X9,X12	T46	X2,X5,X7,X9,X13
T9	X1,X5,X14,X15	T28	X1,X4,X7,X9,X13	T47	X2,X5,X8,X9,X11
T10	X1,X5,X14,X16	T29	X1,X4,X8,X9,X11	T48	X2,X5,X8,X9,X12
T11	X2,X4,X10,X11	T30	X1,X4,X8,X9,X12	T49	X2,X5,X8,X9,X13
T12	X2,X4,X10,X12	T31	X1,X4,X8,X9,X13	T50	X3,X6,X7,X9,X11
T13	X2,X4,X10,X13	T32	X1,X5,X7,X9,X11	T51	X3,X6,X7,X9,X12
T14	X2,X4,X14,X15	T33	X1,X5,X7,X9,X12	T52	X3,X6,X7,X9,X13
T15	X2,X4,X14,X16	T34	X1,X5,X7,X9,X13	T53	X3,X6,X8,X9,X11
T16	X2,X5,X10,X11	T35	X1,X5,X8,X9,X11	T54	X3,X6,X8,X9,X12
T17	X2,X5,X10,X12	T36	X1,X5,X8,X9,X12	T55	X3,X6,X8,X9,X13
T18	X2,X5,X10,X13	T37	X1,X5,X8,X9,X13		
T19	X2,X5,X14,X15	T38	X2,X4,X7,X9,X11		

System reliability was found to be 0.972302, which is same as that stated in the paper, by Nelson, Batts and Beadles. Figure 7.20 shows the results obtained by running the software.

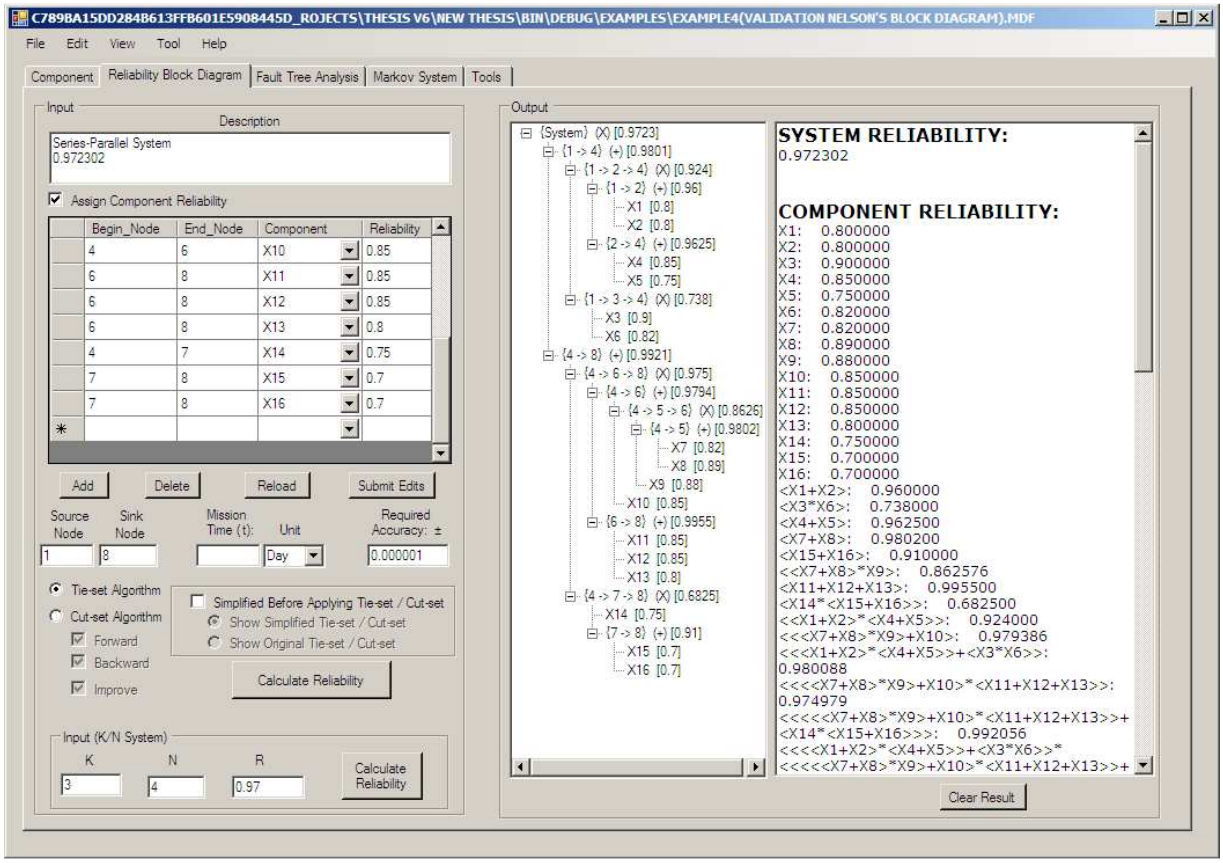


Figure 7.20: Software Complex System Result

7.4.3 State Dependent Systems

The state dependent system used to test the software is a Primary/Backup System with Internal/External Fault Monitoring which is equipped in airplane.

The system shown schematically in the figure below consists of a primary unit (Unit 1) with continuous internal fault monitoring, a backup unit (Unit 2) with no self-monitoring, and an external monitoring unit (Unit 3) whose function is to monitor the health of the backup unit.

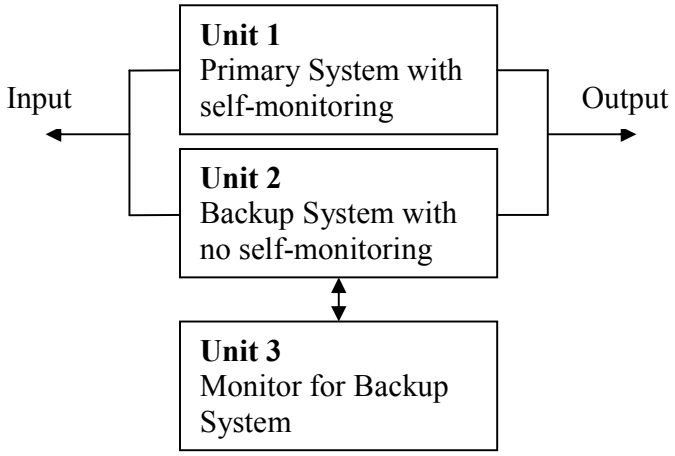


Figure 7.21: Diagram of Active System with self-monitoring and Back-up System with an Independent Monitor

The failure rate of Unit 1 is λ_1 = 5.0E-03per hour. The full time self-monitoring of this unit enables it's functionality to be verified prior to every flight. (The duration of each flight is assumed to be 5 hours.) If it is found to be faulty or inoperative, it is repaired before dispatch.

The failure rate of Unit 2 is λ_2 = 2.5E-03per hour. The backup system has no self-monitoring, but is monitored continuously by an independent monitor (Unit 3). If the backup system fails and the monitor is working, the backup is repaired before the next dispatch. If the monitor is not working, the backup can fail latently, but it is checked every 10 flights (50 hours). If the backup unit is found faulty at one of these 50-hour checks, with no indication of backup system failure from the monitor, it is assumed that the monitor system is also failed, so both Units 2 and 3 are repaired prior to the next flight.

The external monitor (Unit 3) has a failure rate of λ_3 = 2.5E-03per hour. If it fails, it can be repaired in one of two ways. First, as noted above, if the backup system is found to have failed at its periodic 50-hour inspection and there was no monitor indication of a backup system failure, then the monitor is repaired along with the backup system prior to the next flight. Second, a

periodic check of the monitor is performed every 100 flights (500 hours), and if the monitor is found to have failed, it is repaired prior to the next flight.

The MTBFs of the individual units are 20,000 and 40,000 hours, whereas the periodic inspection intervals are only 5, 50, and 500 hours, all of which are orders of magnitude smaller than the MTBFs. Also, most of the states being repaired are first-order states, i.e., they are just one failure removed from the full-up state, so there is no appreciable loss of accuracy in modeling these repairs as continuous transitions with constant rates given by m = 2/*T* for the respective intervals. The exception to this is the state in which both the monitor and the backup system are failed. The 50-hour periodic inspection/repair of this state will actually have an effective repair rate somewhat greater than 2/*T*, but it is conservative to use 2/*T*, so for convenience we will use this expression for all the repair rates. Thus we set *m1* = 2/5, *m2* = 2/50, and *m3* = 2/500, and we can construct the Markov model for the overall system as shown below.

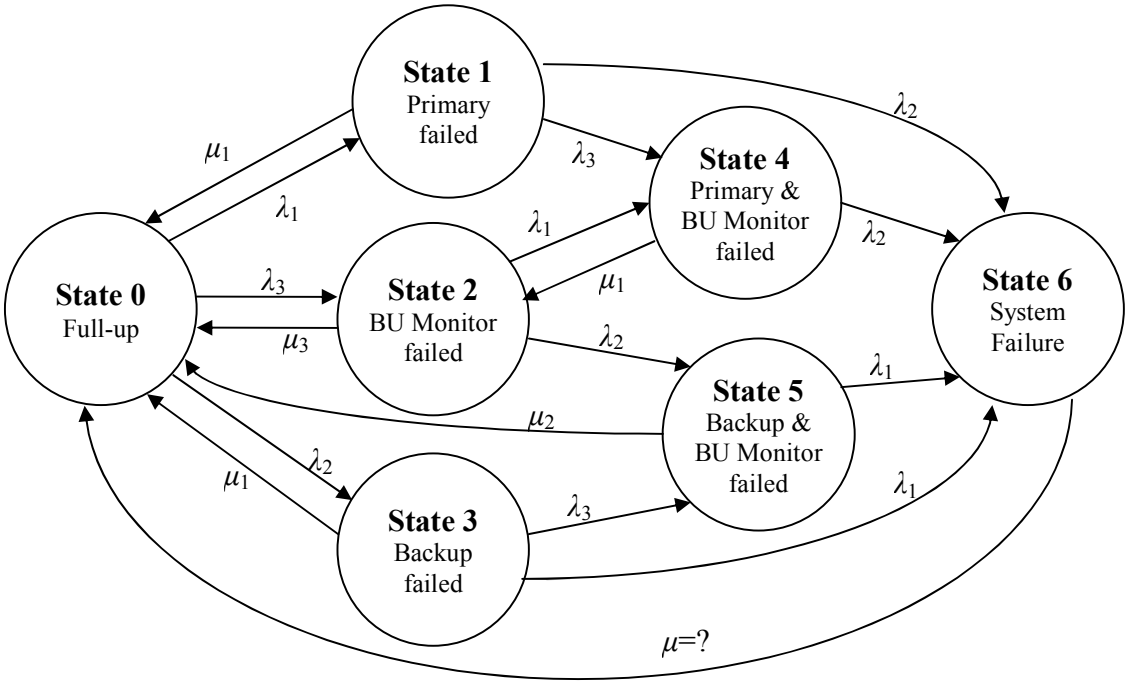


Figure 7.22: Markov Example

As usual, we set the repair rate on the total system failure state (State 6) to infinity, which effectively eliminates that state from the system equations. The system failure rate is simply the rate of entry into that state, i.e., $\lambda_{sys} = (P_1 + P_4)\,\lambda_2 + (P_3 + P_5)\,\lambda_1$. Also, since the probabilities of the remaining states must sum to 1, we can disregard one of them, so we need only consider the steady-state equations for the state 1 through 5, as listed below.

$$\lambda_1 P_0 - (\lambda_2 + \lambda_3 + \mu_1)P_1 = 0$$

$$\lambda_3 P_0 - (\lambda_1 + \lambda_2 + \mu_3)P_2 + \mu_1 P_4 = 0$$

$$\lambda_2 P_0 - (\lambda_1 + \lambda_3 + \mu_1)P_3 = 0$$

$$\lambda_3 P_1 + \lambda_1 P_2 - (\lambda_2 + \mu_1)P_4 = 0$$

$$\lambda_2 P_2 + \lambda_3 P_3 - (\lambda_1 + \mu_2)P_5 = 0$$

Combining these with the conservation equation $P_0 + P_1 + P_2 + P_3 + P_4 + P_5 = 1$, we have six equations in six unknowns. In terms of the matrix notation of Section 2.3, the average system failure rate for this example is

$$\lambda_{sys} = LC^{-1}U \tag{7.1}$$

where

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \lambda_1 & -(\lambda_2 + \lambda_3 + \mu_4) & 0 & 0 & 0 & 0 \\ \lambda_3 & 0 & -(\lambda_1 + \lambda_2 + \mu_3) & 0 & \mu_1 & 0 \\ \lambda_2 & 0 & 0 & -(\lambda_1 + \lambda_3 + \mu_4) & 0 & 0 \\ 0 & \lambda_3 & \lambda_1 & 0 & -(\lambda_2 + \mu_1) & 0 \\ 0 & 0 & \lambda_2 & \lambda_3 & 0 & -(\lambda_1 + \mu_2) \end{bmatrix}$$

$$U = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and L is the row vector L = [0 λ_2 0 λ_1 λ_2 λ_1]. Inserting the values of the failure and repair rates, this gives the result λ_{sys} = 0.0001278 per hour. Reliability for mission time of 2 hours is $R(2) = \exp(-\lambda_{sys} * t) = 0.99997$

Solve the above example with software:

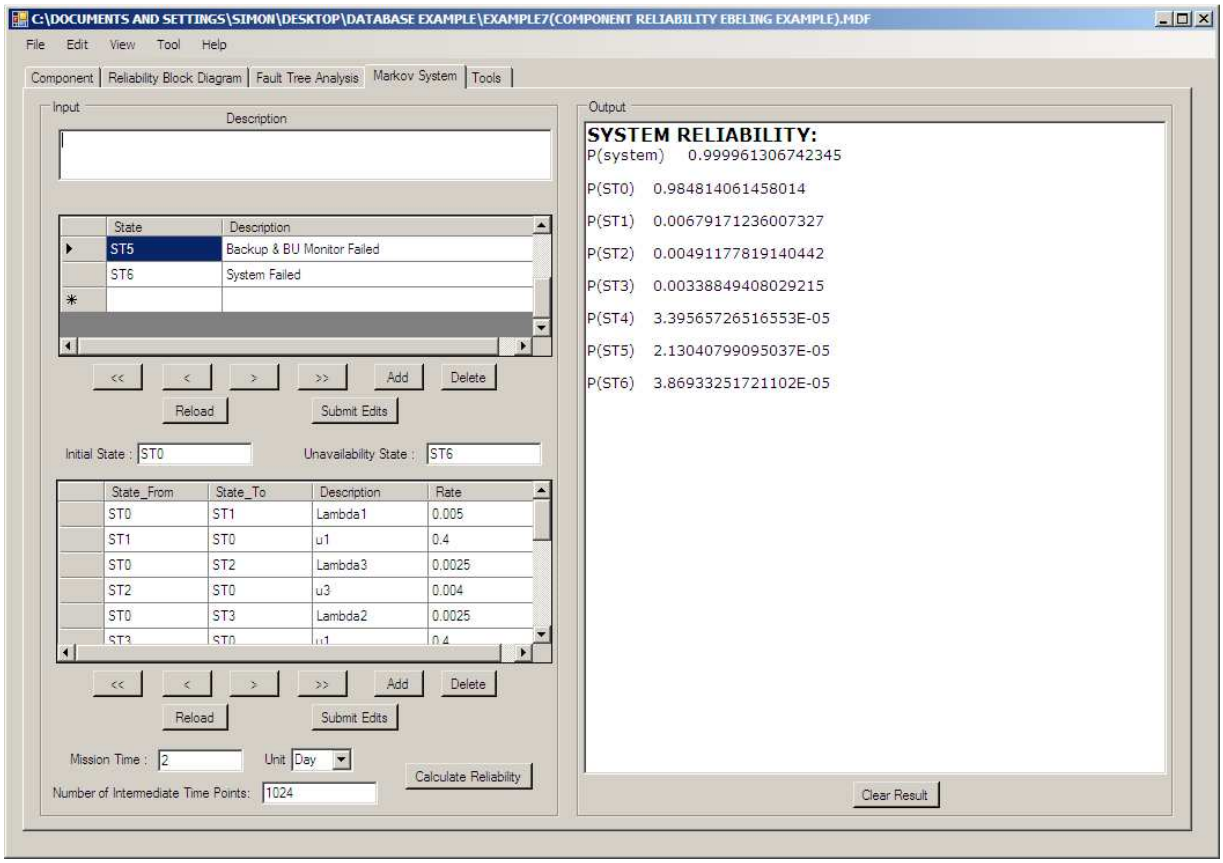


Figure 7.23: Software Markov Model Result

7.4.4 Fault Tree Analysis

In this section we will solve the fault tree example in section 6.2 by software and compare obtained result with the theory result. In that example, the minimal cut-sets are {X3, X4, X5}, {X2, X3}, {X1, X3}, and {X1, X2}. Assuming the unreliability of components X1, X2, X3, X4, X5 are R_1'=0.1, R_2'=0.2, R_3'=0.3, R_4'=0.4, R_5'=0.5 respectively, the system's reliability will be 0.85880 which was obtained in section 4.6.6 .

The same result is obtained when it is solved by the software (Figure 7.24).

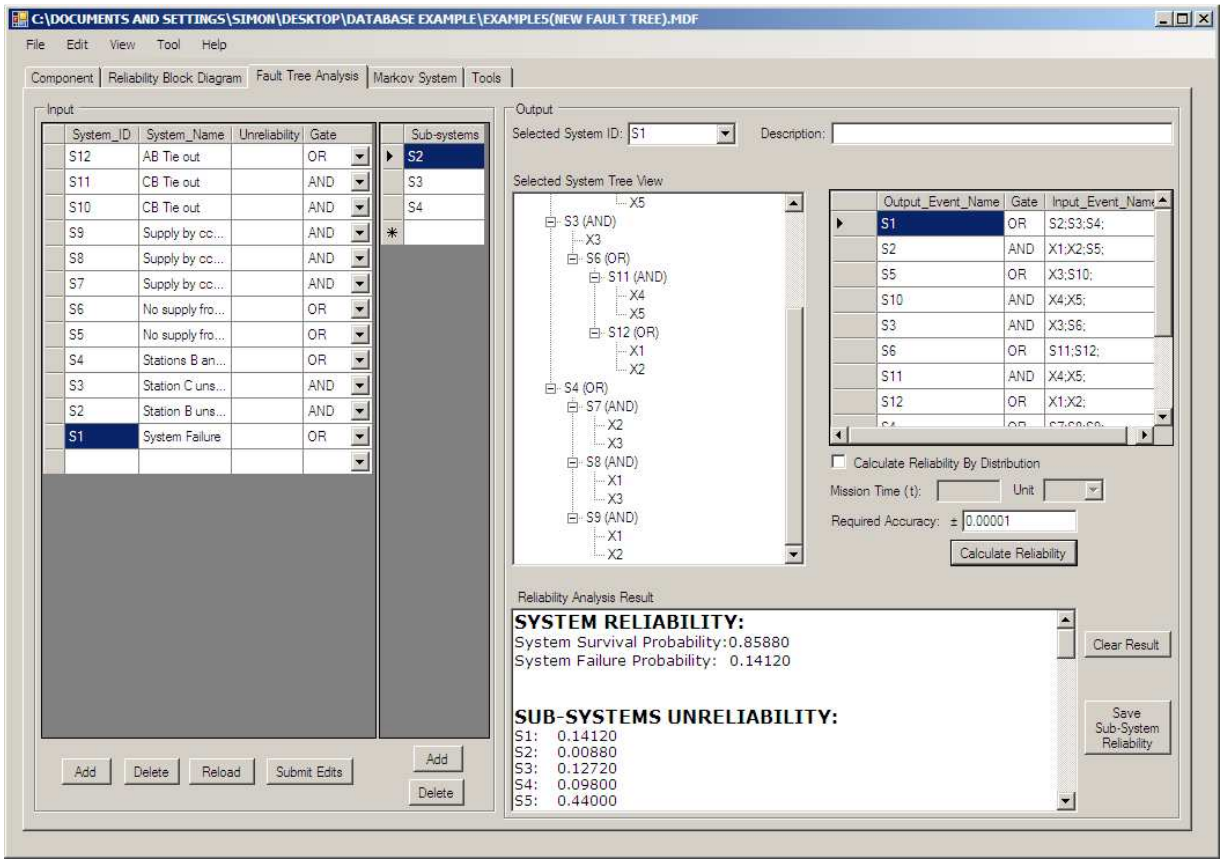


Figure 7.24: Software Fault Tree Analysis Result

CHAPTER 8

CONCLUSION

8.1 Contributions

The software is able to estimate the component reliability by analyzing failure data and determine system reliability by RBD, fault tree, and Markov approach. The integration of all these functions makes it different from commercial reliability software and easier for user to analyze component reliability and manage failure data. To make the software more efficient this research proposed an approach containing a revised connection matrix and a simplification method for large simple and complex network system. The simplification method can simplify the system by identifying and combining the series and parallel sub-system until a pure complex system is attained. After simplification the number of minimal path and cut-set are reduced, so using the simplification method before applying minimal path or cut-set technique can improve the efficiency of the identification of paths and cuts, and save time from reliability calculation. This research also makes an improvement on the element substitution algorithm. In the improved method, whenever a potential cut-set is deleted, all levels of its successors will also be eliminated. In this way, potential cut-set array will contain less non-minimal cut-sets and its size will be decreased leading to reduce computation time. Comparison of these improved methods with current available methods is listed in Table 8.1.

Table 8.1: Reliability Method and Contribution

Available Methods/Techniques	This Research's Proposed/ Improved Methods	Improvements
No failure data distribution analysis function in commercial software	Integration of failure data analysis functions with other reliability analysis functions	Integration of failure data enable users conveniently manage failure data and analyze component reliability.
Traditional connection matrix	Revised connection matrix	Revised connection matrix is more concise and efficient than traditional connection matrix in tie-set and cut-set determination and reliability calculation.
Determine tie-set or cut-set directly from network	Network simplification before the application of tie-set or cut-set algorithm	Network simplification method can reduce tie-set and cut-set, hence it reduces computation time.
Element substitution to determine cut-set	Improved element substitution	Improved element substitution generates less non-minimal cut-set, so it is more efficient.

8.2 Conclusion

The aim of this research was to develop a software tool to efficiently estimate component and system reliability including state dependent and independent system. The software tool was successfully built to:

a. Analyze failure data (component reliability)

b. Improve computation efficiency by

    1) Introducing revised connection matrix

    2) Using simplification method

    3) Improving cut-set algorithm

c. Calculate reliability of independent system (system reliability using block diagram and fault tree)
d. Calculate the reliability of dependent system (system reliability using Markov Model)

The software provides a user-friendly environment. Data entry, data update and data retrieval can be performed in a short period of time. And all functions of the software were validated by different examples with known solutions.

8.3 Future Work

The software is successfully developed to analyze both component reliability and system reliability. However, it can be enhanced in following areas:

- Ability to handle degraded network systems
- Improvement to the tie-set and cut-set algorithm
- Reliability estimation by using other distributions
- Developing a web based tool
- Developing a Graphical User Interface which will allow the user to build a reliability block diagram to show the network of components

REFERENCE

[1] Parekh, M. B. *Development of Decision Support System for Reliability.* West Virginia University. Morgantown, WV, 1999. MS Thesis.

[2] Nelson A. C., Batts J. R., and Beadles R. L. A Computer Program Approximating. *IEEE Trans. Reliability.* 1970, Vols. R-19, pp. 61-65.

[3] Dhillon, B.S. *Robot reliability and safety.* New York : Springer-Verlag, 1991.

[4] Schlager, N. *When Technology Fails.* Detroit : Gale Group, 1994.

[5] Fotuhi-Firuzabad, M., Billinton, R., Munian, T.S., Vinayagam, B. A Novel Approach to Determine Minimal Tie-Sets of Complex Network. *IEEE TRANSACTIONS ON RELIABILITY.* March 2004, Vol. 53, 1, pp. 61-70.

[6] Ramirez-Marquez, J.E., Coit, D. and Tortorella, M. A generalized multistate based path vector approach for multistate two-terminal reliability. *IIE Transactions.* June 2006, Vol. 38, 6, pp. 477-488.

[7] O'Connor, P. D. T., Newton, D. and Bromley, R. *Practical Reliability Engineering.* 4th. s.l. : Wiley, 2002.

[8] Rudner, L.M. and Shafer, W.D. *Reliability. ERIC Digest.* University of Maryland. College Park, MD : ERIC Clearinghouse on Assessment and Evaluation, 2001.

[9] Bellmore, M. and Jensen, P.A. An implicit enumeration scheme for proper cut generation. *Technometrics.* 1970, Vol. 12, pp. 775-788.

[10] Rai, S. and Aggarwal, K.K. On complementation of path sets and cutsets. *IEEE Transactions on Reliability.* 1980, Vols. R-29, pp. 139-140.

[11] Yeh, W.C. A Revised layered-network algorithm to search for all d-minpaths of a limited-flow acyclic network. *IEEE Transactions on Reliability.* 1998, Vol. 47, pp. 436-442.

[12] Lin, H.-Y., Kuo, S.-Y. and Yeh, F.-M. Minimal Cutset Enumeration and Network Reliability Evaluation by Recursive Merge and BDD. 2003, Vol. 2, pp. 1341-6.

[13] Yeh, W.-C. A simple algorithm t search for all MCs in networks. *European Journal of Operational Research.* 2005, Vol. 174, pp. 1694-1705.

[14] Jasmon, G.B. and Foong, K.W. A method for evaluating all the minmal cuts of a graph. 1987, Vols. R-26, pp. 538-545.

[15] Younes, A. and Girgis, M. R. A Tool for computing computer network reliability. 2005, Vol. 82, 12, pp. 1455-1465.

[16] Gebre, B. A and Ramirez-Marquez, J. E. Element substitution algorithm for general two-terminal network reliability analyses. *IIE Transactions.* 2007, Vol. 39, pp. 265-275.

[17] Ebeling, C. E. *An Introduction to Reliability and Maintainability Engineering.* Long Grove, IL : Waveland Pr Inc, 2005.

[18] Ahluwalia, R. S. *Quality and Reliability Engineering.* West Virginia : West Virginia University, 2007.

[19] Walpole, R.E., Myers, R. H., Myers, S. L., Ye, K. *Probability & Statistics for Engineers & Scientists.* NJ : Upper Saddle River, 2002.

[20] Mann, N. R., Schafer, R. E. and Singpurwalla, N. D. *Methods for Statistical Analysis of Reliability and Life Data.* New York : John Wiley & Sons, Inc, 1974. p. 341.

[21] DeMercado, J., Spyratos, N. and Bowen, B. A. A Method for Calculation of Network. *IEEE Trans. Reliability.* June 1976, Vols. R-25, pp. 71-77.

[22] Billinton, R. and Allan, R. *Reliability Evaluation of Engineering Systems:.* s.l. : Plenum Press, 1994.

[23] Singh, C. and Billinton, R. *System Reliability Modeling and Evaluation.* London, England : Hutchinson Educational, 1977.

[24] Nahman, J. M. Enumeration of minimal paths of modified networks. *Journal of Microelectronics and Reliability.* 1994, Vol. 34, pp. 475-484 .

[25] Samad, M. A. An efficient method for terminal and multi-terminal path set enumeration. *Journal of Microelectronics and Reliability.* 1978, Vol. 27, pp. 443-446.

[26] Ramirez-Marquez, J. E. and Coit, D. W. A Monte-Carlo simulation appraoch for approximating multi-state two-terminal. *Reliability Engineering and System Safety.* 2005, Vol. 87, pp. 253-264.

[27] Rausand, M. and Høyland, A. *System Reliability Theory : models, statistical methods, and applications.* 2nd. Hoboken, N.J. : Wiley-Interscience, 2004.

[28] Fussell, J. B., Henry, E. B. and Marshall, N. H. MOCUS-A Computer To Obtain Minimal Sets From Fault Trees. 1974, Vols. ANCR-1156.

[29] W.E.Vesely. *Analysis Of Fault Trees By Kinetic Tree Theory.* s.l. : Idaho Nuclear Corp, 1969. IN-1330.

[30] Pande, P. K., Spector, M. E. and Chatterjee, P. *Computerized Fault Tree Analysis: TREEL and MICSUP.* Berkeley Operations Research Center, California University. California : Univ. of California, 1975. ORC-75-3.

[31] Semanderes, S. N. ELRAFT: A Computer Program For The Efficient Logic Reduction Analysis Of Fault Tree. *IEEE Transactions On Nuclear Science.* 1971, Vols. NS-18.

[32] McCalley, J. Analysis of Non Series/Parallel Systems of Non-Repairable Components. *Power Learn Electric Power Engineering Education.* 2005, Vol. Module PE.PAS.U15.5.

[33] Isograph. Isograph. *Isograph.* [Online] Isograph Ltd . [Cited: June 9, 2009.] http://www.isograph-software.com/index.htm.

[34] Relex. Relex. *Relex.* [Online] Relex Software Corporation. [Cited: June 9, 2009.] http://www.relex.com/.

[35] ITEM. ITEM software. *ITEM software.* [Online] ITEM Software, Inc. [Cited: June 9, 2009.] http://www.itemsoft.com/index.shtml.

[36] ReliaSoft. ReliaSoft. *ReliaSoft.* [Online] ReliaSoft Corporation. [Cited: June 9, 2009.] http://www.reliasoft.com/.

[37] Dhillon, B.S. *Reliability, quality, and safety for engineers.* Boca Raton, FL : CRC Press, 2005.