Opportunistic and Condition Based Maintenance Modelling for Availability Analysis of Repairable Mechanical System

Project Report submitted in partial fulfilment of the requirements for the award of the degree of

M.Tech
in
Production Engineering
By

Rachna Chawla 2K10/Prd/19

Under the guidance of

Mr. Girish Kumar(Asst. Professor)

Department of Mechanical Engineering

Delhi Technological University, Delhi

July 2013

CERTIFICATE

Date:-_________

This is to certify that report entitled **“Opportunistic andCondition Based Maintenance Modeling for Availability Analysis of a Repairable Mechanical System”** by **Ms. Rachna Chawla** is the requirement of the partial fulfillment for the award of Degree of **Master of Technology (M. Tech.)** in **Production Engineering** at **Delhi Technological University**. This work was completed under my supervision and guidance. She has completed her work with utmost sincerity and diligence. The work embodied in this project has not been submitted for the award of any other degree to the best of my knowledge.

Mr. Girish Kumar

(Assistant Professor)

Department of Mechanical Engg

DTU, Delhi

STUDENT’S DECLARATION

I, Rachna Chawla, hereby certify that the work which is being presented in the major project-II entitled “Opportunistic and Condition Based Maintenance Modelling for Availability Analysis of Repairable Mechanical System”, is submitted, in the partial fulfilment of the requirements for degree of Master of Technology at Delhi Technological University is an authentic record of my own work carried under the supervision of Mr. Girish Kumar. I have not submitted the matter embodied in this seminar for the award of any other degree or diploma also it has not been directly copied from any source without giving its proper reference.

Rachna
M.Tech (PRD)
2K11/Prd/19

Table of Contents

List of Figures

List of Tables

LIST OF SYMBOLS

S.no.	Symbols	Description
1	λ	Denotes the failure rate; the rate at which a system fails
2	μ	24Denotes the rate of repair; the rate at which the system is being repaired.
3	λ_{D1}	Initial state of the system where health is at its best
4	λ_{D2}	The minor degraded state to which our system goes after some use.
5	λ_{D3}	The major degraded state to wehich our system goes after prolonged use.
6	λ_{FR}	The state of Random failure where the system goes after sudden adverse situations viz. mishandling ,natural disasters.
7	λ_{I1}	State of inspection in which the components goes from the state D1
8	λ_{I2}	State of inspection in which the components goes from the state D2
9	λ_{I3}	State of inspection in which the components goes from the state D3
10	λ_{M1}	The state of Minor repair where the system goes from the state D1; if found faulty after inspection
11	λ_{M2}	The state of Minor repair where the system goes from the state D2; if found faulty after inspection
12	λ_{MM2}	The state of Major repair where the system goes from the state D2; if found faulty after inspection state I2
13	λ_{M3}	The state of Minor repair where the system goes from the state D3; if found faulty after inspection

14	λI_{M3}	The state of Imperfect Repair where the system goes from the state D3; if found faiulty after inspection state I3
15	λ_{MM3}	The state of Major repair where the system goes from the state D3; if found faulty after inspection
16	λ_{D1D2}	Denotes the rate at which the system will go from D1 to D2
17	λ_{D2D3}	Denotes the rate at which the system will go from D2 to D3
18	λ_{D1I1}	Denotes the rate at which the system will go from D1 to i1
19	λ_{I1M1}	Denotes the rate at which the system will go from I1 to M1
20	λ_{D2I2}	Denotes the rate at which the system will go from D2 to I2
21	λ_{I2M2}	Denotes the rate at which the system will go from I2 to M2
22	λ_{I2MM2}	Denotes the rate at which the system will go from I2 to MM2
23	λ_{D3I3}	Denotes the rate at which the system will go from D3 to I3
24	λ_{I3M3}	Denotes the rate at which the system will go from I3 to M3
25	λ_{I3MM3}	Denotes the rate at which the system will go from I3 to MM3
26	λ_{I3IM3}	Denotes the rate at which the system will go from I3 to IM3
27	λ_{D1FR}	Denotes the rate at which the system will go from D1 to FR

28	λ_{D2FR}	Denotes the rate at which the system will go from D2 to FR
29	λ_{D3FR}	Denotes the rate at which the system will go from D3 to FR
30	μ_{I1D1}	Denotes the rate at which the system will revert back t ostate D1 after inspection at I1; as no repair work is needed
31	μ_{I2D2}	Denotes the rate at which the system will revert back to state D2 after inspection at I2; as no repair work is needed
31	μ_{I3D3}	Denotes the rate at which the system will revert back to state D3 after inspection at I3; as no repair work is needed
33	μ_{M1D1}	Denotes the rate at which the system will be restored from the repair state M1 to state D1
34	μ_{M2D2}	Denotes the rate at which the system will be restored from the repair state M2 to state D2
35	μ_{MM2D1}	Denotes the rate at which the system will be restored from the repair state MM2 to state D1
36	μ_{M3D3}	Denotes the rate at which the system will be restored from the repair state M3 to state D3
37	μ_{IM3D2}	Denotes the rate at which the system will be restored from the repair state IM3 to state D2
38	μ_{MM3D1}	Denotes the rate at which the system will be restored from the repair state MM3 to state D1
39	μ_{FRD1}	Denotes the rate at which the system will be restored from the state of random failure FR to state D1

ABSTRACT

This project deals with opportunistic and condition based maintenance modeling for availability analysis of repairable mechanical systems using Markov approach. The conventional techniques such as reliability block diagram, fault tree analysis and reliability graphs are no more applicable when repairs and other dependencies are incorporated in the model. Therefore, the Markov approach is selected since it is capable of modeling dependencies. Most of the mechanical systems deteriorate gradually before they fail catastrophically. Availability modeling with binary state doesn't give realistic results. So, it would be more appropriate if multi state degradation is considered.

Opportunistic Maintenance models are developed with corrective maintenance, combined with condition based opportunistic maintenance. Models with three types of repair such as Perfect, Imperfect and Minimal are developed with and without opportunistic Maintenance.

The Markov based condition based model is also developed for availability analysis. Aspects such as multi state degradation, random failures, periodic condition monitoring and repair actions such as 'no repair', 'minimal repair', 'perfect repair' and 'imperfect repair' are considered for modeling.

The solutions of the models are obtained analytically by solving system of ordinary differential equations by Ranga-Kutta method using MATLAB software and validated by Monte Carlo Simulation. The proposed methodology is demonstrated for repairable mechanical systems. The benefits of opportunistic maintenance are quantified in terms of the increased system availability. In condition based maintenance model the condition monitoring interval is determined for maximizing the system availability. The proposed methodology is helpful for maintenance engineers in deciding suitable maintenance and replacement policies.

CHAPTER-1 INTRODUCTION

Modern engineering systems, like process and energy systems, transport systems, offshore structures, bridges, pipelines are designed to ensure successful operation throughout the anticipated service life, in compliance with given safety requirements related to the risk posed to the personnel, the public and the environment. Unfortunately, the threat of deteriorating processes is always present, so that it is necessary to install proper maintenance measures to control the development of deterioration and ensure the performance of the system throughout its service life. This requires decisions on what to inspect and maintain, how to inspect and maintain, and when to inspect and maintain. These decisions are to be taken so as to achieve the maximum benefit from the control of the degradation process while minimizing the impact on the operation of the system and other economical and safety consequences.

Engineers are always on the look out for ways of reducing system down time and increasing availability, without compromising on required level of system reliability. The ultimate objective of any maintenance regime is to maintain the system functionality to the maximum extent possible with optimum tradeoffs between the down times and cost of maintenance, avoiding any hazardous failures. Opportunistic maintenance works out to be the perfect remedy, which utilizes the opportunity of system shutdown or module dismantle to perform any maintenance required in the immediate future and saves a substantial amount of system down-time.

A system of components working in a random environment is subjected to wear and damage over time and may fail unexpectedly. The components are replaced or repaired upon failure, and such unpleasant events of failure are at the same time also considered in practice as opportunities for preventive maintenance on other components.

Opportunistic maintenance basically refers to the scheme in which preventive maintenance is carried out at opportunities, either by choice or based on the physical condition of the system. In this paper, we focus on the situation in which the opportunities for preventive maintenance are generated by the failure epochs of individual components. At each failure epoch, the failed components are correctively repaired and other components that are still operational are also preventively serviced so that all the components are maintained and restored to certain conditions. An advantage of this opportunistic maintenance is that corrective repair combined with preventive repair can be used to save set-up costs. Note that by combining both types of repair, one may not know in advance which repair actions should be taken, and thus sacrifices the plannable feature of preventive maintenance. However, there are many situations in which opportunistic maintenance is effective. For example, when corrective repair on some components requires dismantling of the entire system, a corrective repair on these components combined with preventive repair on other or neighbouring components might be worthwhile. Another instance is when a certain corrective repair on failed components can be delayed until the next scheduled preventive maintenance.

The earliest maintenance technique is basically breakdown maintenance (also called unplanned maintenance, or run-to-failure maintenance), which takes place only at breakdowns. A later maintenance technique is time-based preventive maintenance (also called planned maintenance), which sets a periodic interval to perform preventive maintenance regardless of the health status of a physical asset. With the rapid development of modern technology, products have become more and more complex while better quality and higher reliability are required. This makes the cost of preventive maintenance higher and higher. Eventually, preventive maintenance has become a major expense of many industrial companies. Therefore, more efficient maintenance approaches such as condition-based maintenance (CBM) are being implemented to handle the situation.

CBM is a maintenance program that recommends maintenance actions based on the information collected through condition monitoring. CBM attempts to avoid unnecessary maintenance tasks by taking maintenance actions only when there is evidence of abnormal behaviors of a physical asset. A CBM program, if properly established and effectively implemented, can significantly reduce maintenance cost by reducing the number of unnecessary scheduled preventive maintenance operations.

A CBM program consists of three key steps (see Fig. 1):

- Data acquisition step (information collecting), to obtain data relevant to system health.
- Data processing step (information handling), to handle and analyse the data or signals collected in step 1 for better understanding and interpretation of the data.

- Maintenance decision-making step (decision-making), to recommend efficient maintenance policies.

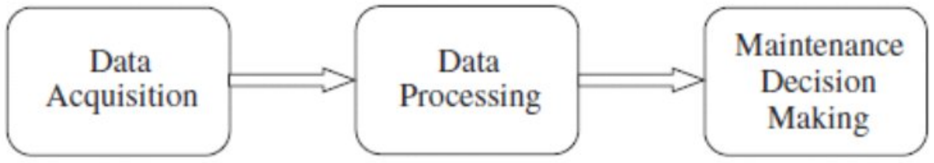


Fig. 1.1. Three steps in a CBM program.

Various conventional techniques are available for reliability and availability analysis like Reliability Block Diagram (RBD), Fault tree Analysis (FTA) and Reliability Graph (RG). RBDs help in clearly understanding the functions of each component, while the Fault tree Analysis technique determines, in a logical way, which failure modes at one level produce critical failures at a higher level in the system. Although these techniques are simple and exact but they are essentially static in nature. Complex systems incorporating repair sequences and non exponential probability distributions cannot be realistically solved with these techniques.

Markov approach is advancement to such techniques as it provides the capability to introduce repair in the system. Markov approach encompasses mainly two concepts. The “state” of the system and the “transitions” in the system from operating to non-operating and vice versa.

Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results; i.e., by running simulations many times over in order to calculate those same probabilities heuristically just like actually playing and recording your results in a real casino situation: hence the name. They are often used in physical and mathematical problems and are most suited to be applied when it is impossible to obtain a closed-form expression or infeasible to apply a deterministic algorithm. Monte Carlo methods are mainly used in three distinct problems: optimization, numerical integration and generation of samples from a probability distribution.

CHAPTER II : Literature Survey

CHAPTER III : Availability modelling techniques

CHAPTER IV: System Modelling

CHAPTER V: Solution of System Model and Result Analysis

CHAPTER VI: Conclusion and Scope for future

CHAPTER 2 - LITERATURE REVIEW

Opportunistic maintenance has been first studied in Radner and Jorgenson 1963, and in McCall 1963. Since then, many extensions of opportunistic maintenance have been introduced and studied in the literature. Berg (1976) studies a system with two identical components with exponential distributed lifetimes, for which the non-failed component as well as the failed component are both replaced by a new one if the age of the non-failed component exceeds a threshold. Zheng and Fard (1991) examine an opportunistic maintenance policy based on failure rate tolerance for a system with k different types of components.

Pham and Wang (2000) propose two new opportunistic maintenance policies for a k-out-of-n system. These and other opportunistic maintenance models have been summarized in Dekker, van der Schouten and Wildeman (1997) and in Wang (2002). All these models, however, address the optimization issues for components operating independently. The thrust of the maintenance model introduced in this study is the general opportunistic repairs implemented at correlated failures for a system of components that are stochastically dependent.

Karin S. de Smidt-Destombes, Matthieu C. van der Heijden, Aart van Harten (2004) published a paper on “On the availability of a k-out-of-N system given limited spares and repair capacity under a condition based maintenance strategy”.
This paper considers a k-out-of-N system with identical, repairable components. Maintenance is initiated when the number of failed components exceeds some critical level. After a possible set-up time, all failed components are replaced by spares. A multi-server repair shop repairs the failed components. The system availability depends on the spare part stock level, the maintenance policy and the repair capacity. They presented a mathematical model supporting the trade-off between these three parameters. Also, they presented both an exact and an approximate approach to analyse our model. In some

numerical experiments, we provide insight on the impact of repair capacity, number of spares and preventive maintenance policy on the availability.

Castanier, Grall, Be´renguer (2005) published a paper on “A condition-based maintenance policy with non-periodic inspections for a two-unit series system”.
This paper considers a condition-based maintenance policy for a two-unit deteriorating system. Each unit is subject to gradual deterioration and is monitored by sequential non-periodic inspections. It can be maintained by good as new preventive or corrective replacements. Every inspection or replacement entails a set-up cost and a component-specific unit cost but if actions on the two components are combined, the set-up cost is charged only once. A parametric maintenance decision framework is proposed to coordinate inspection/replacement of the two components and minimize the long-run maintenance cost of the system. A stochastic model is developed on the basis of the semi-regenerative properties of the maintained system state and the associated cost model is used to assess and optimize the performance of the maintenance model. Numerical experiments emphasize the interest of a control of the operation groupings.

Andrew K.S. Jardine, Daming Lin, Dragan Banjevic (2006) published a paper on “A review on machinery diagnostics and prognostics implementing condition-based maintenance”.
This paper attempts to summarize and review the recent research and developments in diagnostics and prognostics of mechanical systems implementing CBM with emphasis on models, algorithms and technologies for data processing and maintenance decision-making. Diagnostics and prognostics are two important aspects of a CBM program. Research in the CBM area grows rapidly. Realizing the increasing trend of using multiple sensors in condition monitoring, the authors also discuss different techniques for multiple sensor data fusion. The paper concludes with a brief discussion on current practices and possible future trends of CBM.

Romulo I. Zequeira, Jose E. Valdes, Christophe Berenguer (2007) presented the study of the determination of the optimal maintenance policy for a manufacturing facility and the optimal buffer inventory to satisfy the demand during the interruption period due to a maintenance action. We consider the possibility of imperfect production and that opportunities for the fabrication of the buffer inventory and opportunities to carry out a maintenance action to the production facility are random.

Xiaojun Zhou, LifengXi, JayLee (2008) proposed opportunistic preventive maintenance scheduling for a multi-unit series system based on dynamic programming. It is understood that for a multi-unit series system, whenever one of the units stops to perform a preventive maintenance (PM) action, the whole series system must be stopped. At that time PM opportunities arise for the other units in the system. This paper proposes an opportunistic PM scheduling algorithm for the multi-unit series system based on dynamic programming with the integration of the imperfect effect into maintenance actions. An optimal maintenance practice is determined by maximizing the short-term cumulative opportunistic maintenance cost savings for the whole system. Matlab is considered for the optimization which is based on numerical simulation. Numerical examples are given throughout to show how this approach works. Finally, a comparison between the proposed PM model and other models is given.

Radouane Laggounea,*, Alaa Chateauneufb, Djamil Aissania(2009) proposed Opportunistic policy for optimal preventive maintenance of a multi-component system in continuous operating units. A solution procedure based on Monte Carlo simulations with informative search method is proposed and applied to the optimization of preventive maintenance plan for a hydrogen compressor in an oil refinery.

M. S. Samhouri *, A. Al-Ghandoor, R. H. Fouad, S. M. Alhaj Ali (2009) proposed a genetic algorithm approach for an intelligent opportunistic maintenance system. The maintenance regime of complex systems most often consists of a variety of maintenance strategies, like preventive maintenance, corrective maintenance, condition-based maintenance and so on. Opportunistic or opportunity-based maintenance (OM) gives the maintenance staff an opportunity to replace or repair those items, which are found to be defective or need replacement in the immediate future, during the maintenance of a machine or component. This work presents an intelligent method of how to decide whether a particular item requires opportunistic maintenance or not, and if so how cost effective this opportunity-based maintenance will be when compared to a probable future grounding. This maintenance strategy is considered important when dealing with complex systems that contain expensive items with hard lives with condition-based maintenance (CBM) strategies. Genetic algorithms (GA) are employed to decide whether opportunistic maintenance is cost effective or not.

Yongjin(James)Kwon, RichardChiou, LeonardStepanskı (2009) published a paper on "Remote, condition-based maintenance for web-enabled robotic system".
In this paper, mathematical modeling of system availability has been derived in order to account for other failures that might occur in the subsystems of the robot. Compared to the schedule- based maintenance strategies, the proposed approach shows great potential for improving overall production efficiency, while reducing the cost of maintenance. The current trends in industry include an integration of information and knowledge-base network with a manufacturing system, which coined a new term, e-manufacturing. From the perspective of e-manufacturing any production equipment and its control functions do not exist alone, instead becoming a part of the holistic operation system with distant monitoring, remote quality control, and fault diagnostic capabilities.

Ling Wang, Jian Chu, Weijie Mao (2009) published a paper entitled as "A condition-based replacement and spare provisioning policy for deteriorating systems with uncertain deterioration to failure".
A new policy, referred to as the condition-based replacement and spare provisioning policy, is presented for deteriorating systems with a number of identical units. The deterioration level of each unit in the system can be described by a scalar random variable, which is continuous and increasing monotonically. Furthermore, the deterioration level just when the unit failure occurs, termed deterioration to failure, is uncertain. Therefore, the condition-based reliability is proposed in order to characterize various and uncertain deterioration levels when unit failure occurs. A simulation model is

developed for the system operation under the proposed condition-based replacement and spare provisioning policy.

Y.G. Li, P. Nilkitsaranont (2009) published a paper on “Gas turbine performance prognostic for condition-based maintenance”.
This paper describes a prognostic approach to estimate the remaining useful life of gas turbine engines before their next major overhaul based on historical health information. A combined regression techniques, including both linear and quadratic models, is proposed to predict the remaining useful life of gas turbine engines. A statistic ‘‘compatibility check” is used to determine the transition point from a linear regression to a quadratic regression. The developed prognostic approach has been applied to a model gas turbine engine similar to Rolls-Royce industrial gas turbine AVON 1535 implemented with compressor degradation over time.

Fangji Wu, TianyiWang, JayLee (2010) published a paper on “An online adaptive condition-based maintenance method for mechanical systems”.
This paper proposes an online adaptive condition-based maintenance method with pattern discovery and fault learning capabilities for mechanical systems. The method is mainly based on a subtype of neural network techniques called self-organizing map (SOM). It is able to reduce local clusters from the same pattern and optimize the SOM architecture to further decrease the calculation cost in matching patterns in the neuron fitting process. Moreover, distance analysis and statistical pattern recognition (SPR) on neurons of the SOM are combined to establish rules and criteria for conducting and controlling the discovery and learning process so continuous process as purging prototypes on the map can be avoided.

Zhigang Tian, Tongdan Jin, Bairong Wu, Fangfang Ding (2011) published a paper on” Condition based maintenance optimization for wind power generation systems under continuous monitoring”. In this paper, they utilized condition monitoring information collected from wind turbine components, condition based maintenance (CBM) strategy can be used to reduce the operation and maintenance costs of wind power generation systems. The existing CBM methods for wind power generation systems deal with wind turbine components separately, that is, maintenance decisions are made on individual components, rather than the whole system. However, a wind farm generally consists of multiple wind turbines, and each wind turbine has multiple components including main bearing, gearbox, generator, etc. There are economic dependencies among wind turbines and their components. The proposed maintenance policy is defined by two failure probability threshold values at the wind turbine level.

Zhigang Tian, Haitao Liao (2011) published a paper on "Condition based maintenance optimization for multi-component systems using proportional hazards model".
In this paper they presented the objective of condition based maintenance (CBM) is typically to determine an optimal maintenance policy to minimize the overall maintenance cost based on condition monitoring information. The existing work reported in the literature only focuses on determining the optimal CBM policy for a single unit. In this paper, we investigate CBM of multi-component systems, where economic dependency exists among different components subject to condition monitoring. The fixed preventive replacement cost, such as sending a maintenance team to the site, is incurred once a preventive replacement is performed on one component.

Fangfang Ding, Zhigang Tian*(2012) proposed opportunistic maintenance policies which are defined by the component's age threshold values, and different imperfect maintenance thresholds are introduced for failure turbines and working turbines. Three types of preventive maintenance actions are considered, including perfect, imperfect and two-level action. Simulation methods are developed to evaluate the costs of proposed opportunistic maintenance policies. Numerical examples are provided to illustrate the proposed approaches. Comparative study with the widely used corrective maintenance policy demonstrates the advantage of the proposed opportunistic maintenance methods in significantly reducing the maintenance cost.

Sharareh Taghipour ⇑, Dragan Banjevic (2012) proposed two optimization models for the periodic inspection of a system with ''hard-type'' and ''soft-type'' components.

Given that the failures of hard-type components are self-announcing, the component is instantly repaired or replaced, but the failures of soft-type components can only be detected at inspections. A system can operate with a soft failure, but its performance may be reduced. Although a system may be periodically inspected, a hard failure creates an opportunity for additional inspection (opportunistic inspection) of all soft-type components. Two optimization models are discussed in the paper. In the first, soft-type components undergo both periodic and opportunistic inspections to detect possible failures. In the second, hard-type components undergo periodic inspections and are preventively replaced depending on their condition at inspection. Soft-type and hard-type components are either minimally repaired or replaced when they fail. Minimal repair or replacement depends on the state of a component at failure; this, in turn, depends on its age. The paper formulates objective functions for the two models and derives recursive equations for their required expected values. It develops a simulation algorithm to calculate these expected values for a complex model.

Rosmaini Ahmad, Shahrul Kamaruddin (2012) published a paper on "An overview of time-based and condition-based maintenance in industrial application".
This paper presents an overview of two maintenance techniques widely discussed in the literature: time-based maintenance (TBM) and condition-based maintenance (CBM). The paper discusses how the TBM and CBM techniques work toward maintenance decision making. Recent research articles covering the application of each technique are reviewed. The paper then compares the challenges of implementing each technique from a practical point of view, focusing on the issues of required data determination and collection, data analysis/modelling, and decision making.

Cui Yanbin, Cui Bo (2012) published a paper on "The Condition Based Maintenance Evaluation Model on On-post Vacuum Circuit Breaker".
The safe operation of power supply equipments is closely related to the security of electric network. The planned maintenance of existing power equipments cannot meet the needs of development of power system. To solve the problems in maintenance for vacuum circuit breaker, this paper build the equipment condition and risk assessment

index system and bring out the outdoor on-post vacuum circuit breaker condition based maintenance evaluation model which based on Rough Set and Support Vector Machine according to the real condition. To prove the high accuracy of this method, a research which about the data of 100 Box-type sub-station in the distributing network of one power supply company is conducted in this paper.

Qingfeng Wang, Jinji Gao(2012) published a paper on "Research and application of risk and condition based maintenance task optimization technology in an oil transfer station". This paper carries out a research on Risk and Condition Based Maintenance (RCBM) task optimization technology. Utilizing the internet of things (IOT), real-time database, signal-processing, Gray Neural Network, probability statistical analysis and service oriented architecture (SOA) technology, a Risk and Condition Based Indicator Decision-making System (RCBIDS) is built. RCBIDS integrates RCM, condition monitoring system (CMS), key performance management module, file management module, fault and defect management module, maintenance management module together, which aims to realize remote condition monitoring, maintenance technical support services (TSS), quantitative maintenance decision-making, and to ensure the Reliability, Availability, Maintainability and Safety (RAMS).

Chiming Guoa, Wenbin Wang, Bo Guoa, Xiaosheng Sic(2012) published a paper on "A Maintenance Optimization Model for Mission-Oriented Systems Based on Wiener Degradation".
This paper deals with mission-oriented systems subject to gradual degradation modeled by a Wiener stochastic process within the context of CBM. For a mission-oriented system, the mission usually has constraints on availability/reliability, the opportunity for maintenance actions, and the monitoring type (continuous or discrete). Furthermore, in practice, a mission-oriented system may undertake some preventive maintenance (PM) and after such PM, the system may return to an intermediate state between an as-good-as new state and an as-bad-as old state, i.e. the PM is not perfect and only partially restores the system.

CHAPTER III : Availability modelling techniques

CHAPTER IV: System Modelling

CHAPTER V: Solution of System Models and Result Analysis

CHAPTER VI: Conclusion and Scope for future

CHAPTER 3 - AVAILABILTY MODELLING TECHNIQUES

In this chapter reliability and availability modelling and analysis are described. The conventional techniques such as RBD, FTA etc., with their limitations are discussed. Finally the modern technique such as Markov approach is elaborated.

3.1 Conventional techniques for reliability and availability modelling

The conventional methods used in reliability modelling are:-

- Reliability block diagrams
- Fault tree analysis

Other methods are also there which are not discussed here such as Boolean truth table and reliability graphs.

3.1.1 Reliability Block Diagrams

A reliability block diagram shows the interdependencies among all elements (subsystems, equipments, etc.) or functional groups of the item for item success in each service use event. The blocks in the diagram follow a logical order which relates the sequence of events during the prescribed operation of the item. The reliability block diagram is drawn so that each element or function employed in the item can be identified. Each block of the reliability block diagram represents one element of function contained in the item. All blocks are configured in series, parallel, standby, or combinations thereof as appropriate. Refer fig 3.1.

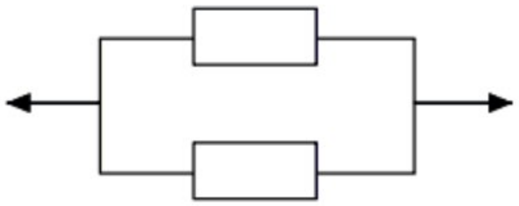


Figure 1 RBD of 2 component parallel.

The following general assumptions apply to reliability block diagrams:

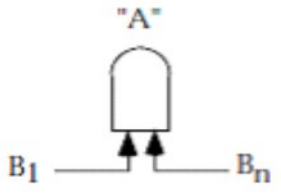
- Blocks denote elements or functions of the items that are considered when evaluating reliability and which have reliability values associated with them.
- Lines connecting blocks have no reliability values.
- All inputs to the item are within specification limits.
- Failure of any element or function denoted by a block in the diagram will cause failure of the entire item, except where alternative modes of operation may be present; i.e., redundant units or paths.
- Each element or function denoted by a block in the diagram is independent with regard to probability of failure from all other blocks.

3.1.2 Fault Tree Analysis

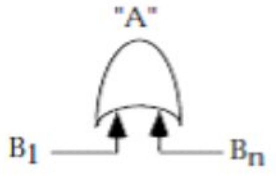
The "fault tree" analysis (FTA) technique is a method for block diagramming constituting lower level elements. It determines, in a logical way, which failure modes at one level produce critical failures at a higher level in the system. The fault tree provides a concise and orderly description of the various combinations of possible occurrences within the system which can result in a predetermined critical output event. Fault tree methods can be applied beginning in the early design phase, and progressively refined and updated to track the probability of an undesirable event as the design evolves. Initial fault tree diagrams might represent functional blocks (e.g., units, equipments, etc.), becoming more definitive at lower levels as the design materializes in the form of specific parts and materials.

Results of the analysis are useful in the following applications:

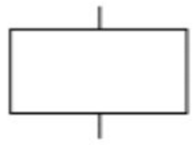
- Allocation of critical failure mode probabilities among lower levels of the system breakdown.
- Comparison of alternative design configurations from a safety point of view.
- Identification of critical fault paths and design weaknesses for corrective action.
- Evaluation of alternative corrective action approaches.
- Development of operational, test, and maintenance procedures to recognize and accommodate unavoidable critical failure modes



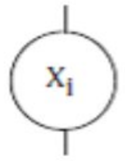
A logical "AND" gate - "A" exists if and only if <u>all</u> of B_1, B_2 . . . . B_n exist simultaneously.



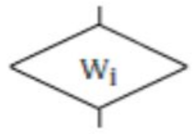
A logical inclusive "OR" gate - "A" exists if <u>any</u> B_1, B_2, . . . B_n or any combination thereof



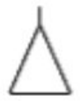
An event--usually the output of (or input to) and "AND" or an "OR" gate



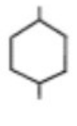
A failure rate of malfunction event-in terms of a specific circuit or component, represented by the symbol X with a numerical subscript



An event not developed further because of lack of information or because of lack of sufficient consequence. Represented by the symbol W with a numerical subscript

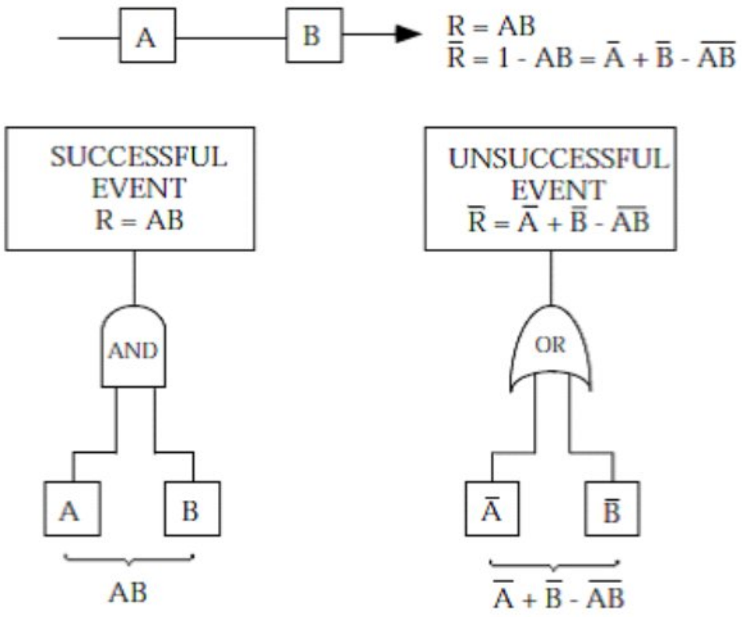


A connecting symbol to another part of the fault tree within the same major branch



An "inhibit" gate, used to describe the relationship between one fault and another. The input fault directly produces the output fault if the indicated conditions is satisfied

Figure 2 Symbols commonly used in diagramming a fault tree analysis



A
B
R = AB
$\overline{R} = 1 - AB = \overline{A} + \overline{B} - \overline{AB}$
SUCCESSFUL EVENT R = AB
UNSUCCESSFUL EVENT $\overline{R} = \overline{A} + \overline{B} - \overline{AB}$
AND
OR
A
B
$\overline{A}$
$\overline{B}$
AB
$\overline{A} + \overline{B} - \overline{AB}$

Figure 3 Transformation of two element series reliability block diagram to Fault tree logic diagram

3.2 Markov approach

A Markov process is a mathematical model that is useful in the study of the availability of complex systems. The basic concepts of the Markov process are those of "state" of the system (e.g., operating, non operating) and state "transition" (from operating to non operating due to failure, or from non operating to operating due to repair).. Any Markov process is defined by a set of probabilities pij which define the probability of transition from any state i to any state j. One of the most important features of any Markov model is that the transition probability pij depends only on states i and j and is completely independent of all past states except the last one, state i; also pij does not change with time.

In system availability modelling utilizing the Markov process approach, the following additional assumptions are made:

- The conditional probability of a failure occurring in time (t, t + dt) is λ dt.
- The conditional probability of a repair occurring in time (t, t + dt) is μ dt.

- Each failure or repair occurrence is independent of all other occurrences.
- λ (failure rate) and μ (repair rate) are constant.

Let us now apply the Markov process approach to the availability analysis of a single unit with failure rate λ and repair rate μ.

3.2.1 Two components (Markov Process Approach)

The Markov graph for two components is shown:

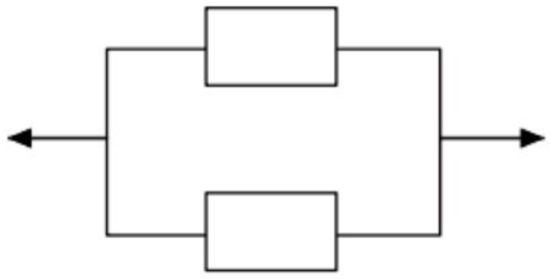


Figure 4 RBD of 2 components in parallel

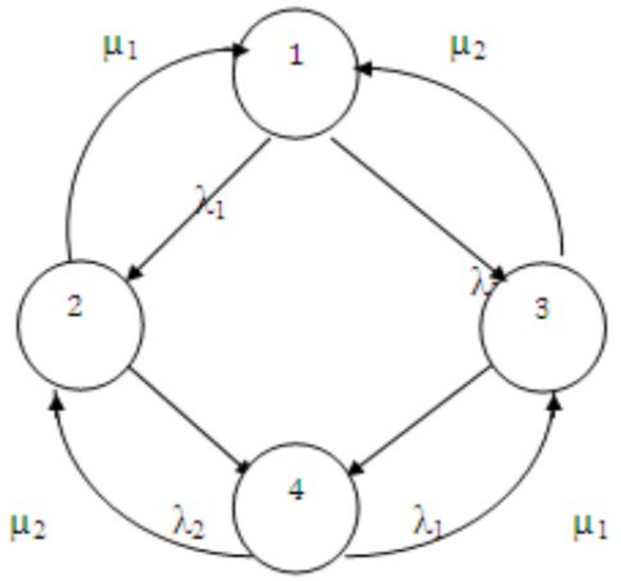


Figure 5 Markov model of two components in parallel

Where:

S_1 = OO = both the components are operating

S_2 = FO = first unit has failed and second is operating

S_3 = OF= first unit is operating and second has failed

S_4= FF= both the components have failed

λ = failure rate

μ = repair rate

The differential equations involved are:-

$-(\lambda_1 + \lambda_2)P_1(t) + \mu_1 P_2(t) + \mu_2 P_3(t)$

$-(\mu_1 + \lambda_2)P_2(t) + \lambda_1 P_4(t) + \mu_2 P_4(t)$

$-(\lambda_1 + \mu_2)P_3(t) + \lambda_2 P_1(t) + \mu_1 P_4(t)$

$-(\mu_1 + \mu_2)P_4(t) + \lambda_1 P_3(t) + \lambda_2 P_2(t)$

Analytically, these differential equations can be solved by Ranga-Kutta method or Laplace transformation. However, we have solved by Ranga-Kutta using MATLAB and the availability of the system is obtained by adding the operating states.

3.2.2 Advantages

Markov models offer significant advantages over other reliability modelling techniques, some of these advantages are:

- Simplistic modelling approach: the models are simple to generate although they do require a more complicated mathematical approach.
- Redundancy management techniques: system reconfiguration required by failure is easily incorporated in the model.
- Coverage: covered and uncovered failures of components are mutually exclusive events. These are not easily modelled using classical techniques, but are readily handled by the Markov mathematics.
- Complex systems: many simplifying techniques exist which allow the modelling of complex systems.
- Sequenced events: often the analyst is interested in computing the probability of an event resulting from a sequence of sub events. while these types of problems do not lend themselves well to classical techniques, they are easily handled using Markov modelling.

3.2.3 Limitations

The major drawback of Markov methods is the explosion of the number of states as the size of the system increases. The resulting diagrams for large systems are generally extremely large and complicated, difficult to construct and computationally extensive.

3.3 Monte Carlo Simulation

Monte Carlo (MC) simulation is a quantitative risk analysis technique in which uncertain inputs in a model (for example an Excel spreadsheet) are represented by probability distributions (instead of by one value such as the most likely value). By letting your computer recalculate your model over and over again (for example 10,000 times) and each time using different randomly selected sets of values from the (input) probability distributions, the computer is using all valid combinations of possible input to simulate all possible outcomes. The results of a MC simulation are distributions of possible outcomes (rather than the one predicted outcome you get from a deterministic model); that is, the range of possible outcomes that could occur and the likelihood of any outcome occurring. This is like running hundreds or thousands of "What-if" analyzes on your model, all in one go, but with the added advantage that the ’what-if’ scenarios are generated with a frequency proportional to the probability we think they have of occurring.

The most important advantages of Monte Carlo include:

- The probability distributions within the model can be easily and flexibly used, without the need to approximate them;
- Correlations and other relations and dependencies (such as ”if” statements) can be modeled without difficulty;
- The level of mathematics required is quite basic;
- The behavior of and changes to the model can be investigated with great ease and speed.

An often claimed disadvantage of MC Simulation is that it is an approximate technique. However, any degree of precision can be achieved (at least in theory) by simply increasing the number of iterations, so the real limitations of MC simulation are:

- The number of random numbers that can be produced from a random number generating algorithm and;

- The time a computer needs to generate the iterations (and the time the risk analyst has).

CHAPTER 4 - SYSTEM MODELLING

In this chapter condition based maintenance model and opportunistic maintenance models are developed for system availability analysis.

4.1 Condition based Maintenance Modeling

For Condition based Modeling aspects such as degradation, Random failure, Periodic inspection and repair actions such as ‘no repair’, ‘minimal repair’, ‘ imperfect repair’, and ‘perfect repair’ are considered.

4.1.1 DEGRADATION

Whenever a system or a model is in working it degrades with time. The degradation is gradual not sudden. We are trying to study a mode that follows this kind of failure.

In degradation modeling we study a system that is prone to degradation and mostly we study the systems where reliability is critical. As shown in the figure is such a system.

There are four stages shown. Fresh component is given the stage D1, then with time it degrades to a stage D2 and so on and finally it goes to a failure state. We will be studying the degradation rate from one stage to the other for all the stages.

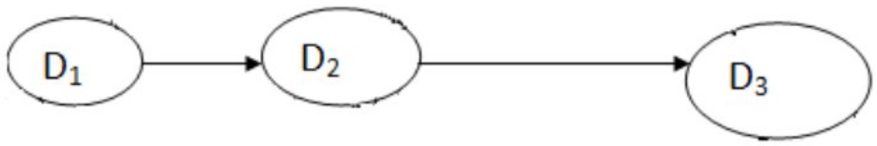


Figure 6 Multi State degradation

4.1.2 INSPECTION

Inspection is a way to see the health of the system and deciding whether the system requires repair/maintenance or not. Now there are two types of inspections:

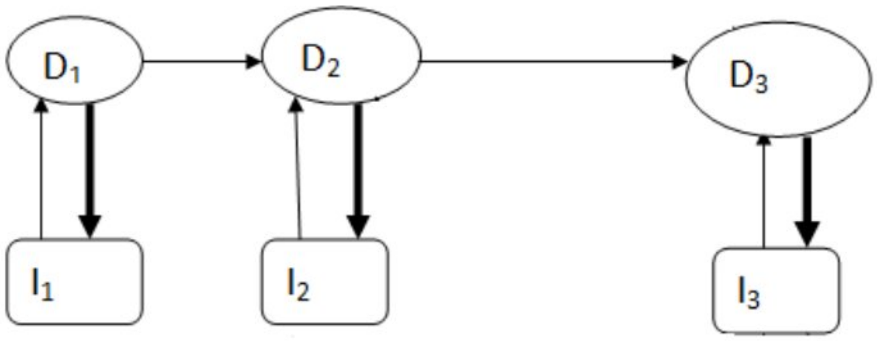


Figure 7 Periodic Inspection at every stage.

Online: In this we need not to stop the system for inspection so the availability of the system is more, and

Offline: In this we need to stop the system for inspection.

As described in the above figure we take the system further and do periodic inspections at each state defined. These inspections help us in maintain the system by doing timely repairs and maintenance.

4.1.3 CONDITION BASED MAINTENANCE

Condition based maintenance (CBM), shortly described, is maintenance when need arises. This maintenance is performed after one or more indicator shows the equipment is going to fail or that equipment performance is deteriorating.

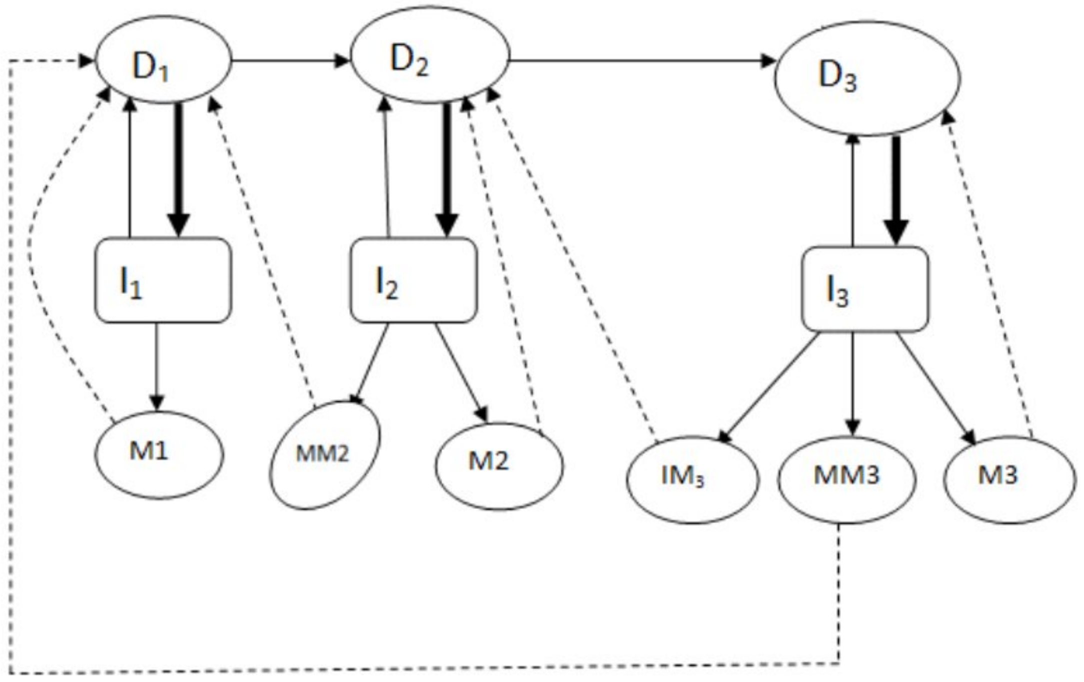


Figure 8 Condition based maintenance

In our system model, we have described three types of maintenance on the basis of the requirement of the system. The three types of maintenance are: Minor maintenance, intermediate maintenance and major maintenance.

In stage D1, our system is new, thus , we need not require much maintenance for it. Therefore we have kept the probability for our system to undergo minor maintenance to be 0.1 and the probability that system would go back to the stage D1 without any maintenance to be 0.9.

Similarly in stage D2 as our system is in continuous working state, it deteriorates and thus its efficiency decreases and the need to repair it or maintain it increases as compared to the system in stage D1. Due to this reason we have decreased the probability that the system would go back to stage D2 without any repair from 0.9 to 0.7 and the probability that the system would require maintenance has been increased from 0.1 to 0.3.

Finally, when our system moves from stage D2 to D3, it deteriorates further giving rise to the need to repair it in order to increase its availability. Therefore the probability is that he system requires minor repair or intermediate repair or major repair or no repair has been altered again the probability that the system would require major maintenance has been changed to 0.2.The probability that the system would require intermediate repair has been changed to 0.4. The probability that the system would require minor maintenance has been change to 0.2 and finally the probability that the system would go back to stage D3 without any repair has been changed to 0.2.

4.1.4 RANDOM FAILURE

Random failure is defined as the situation/Condition in which the system fails due to some random causes. These random causes can be anything from natural calami8ty to human error. Random failures can also occur due to voltage fluctuations, manufacturing defects, problem in system components, etc.

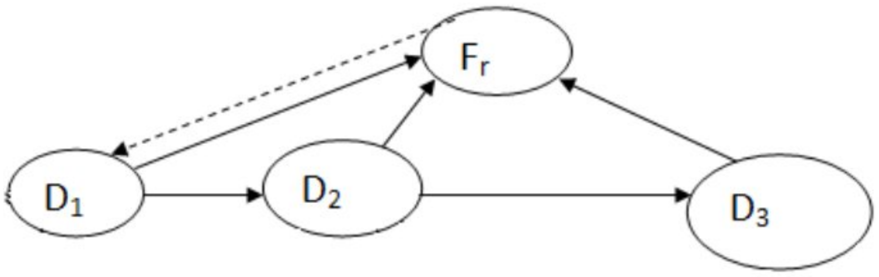


Figure 9 Random failure

Random failure causes the system to go in offline mode thereby bringing its availability to 0.

4.1.5 SYSTEM MODEL

Considering all system modeling aspects, following model is developed:

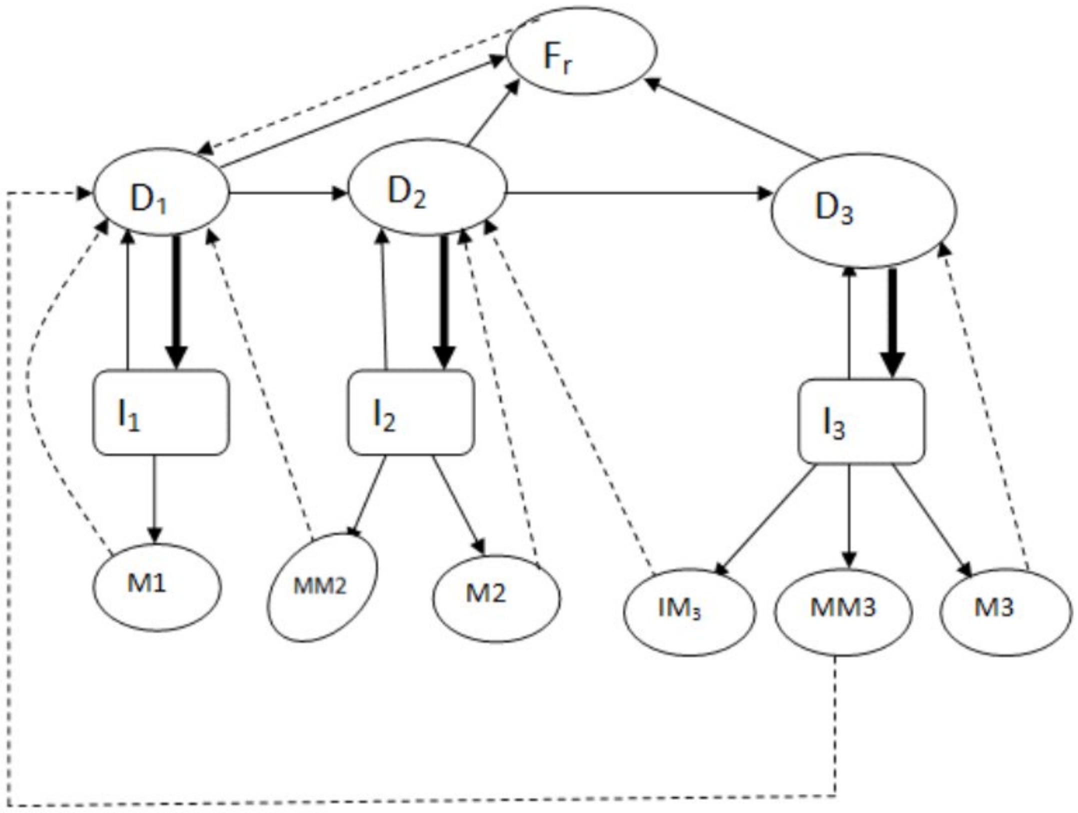


Figure 10 System Model

4.2 System Model for Opportunistic maintenance

We have considered Motor and Pump in series connection. Time to failure and time to repair data are tabulated for Motor and Pump from website (www.baringer.com).

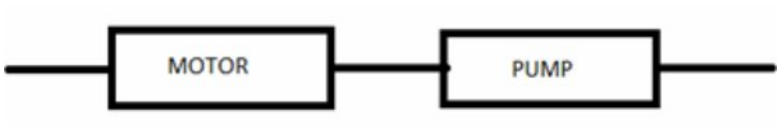


Figure 11 Series system

In multi state degradation, a series of state is assumed in which the machine would function before finally reaching the failed state, these states exist between the new state of the machine and the failed state of the machine.

For our work we have assumed to have three states of working and one failed state. Thus making it in a four state system.

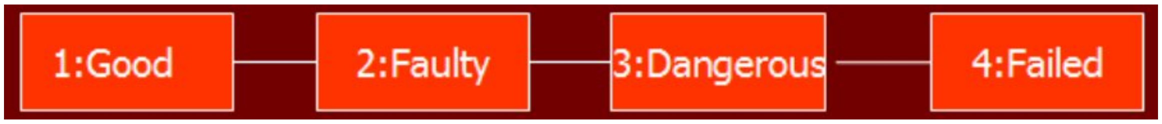


Figure 12 Four State system

4.2.1 System states for two components in series

W: Working;
R: Repair,
O: Operable.

S.No.	State of component ‘A’	State of component ‘B’	System state	
1	1W	1W	${}^{1}A_1B_1$	Good
2	2W	1W	${}^{2}A_2B_1$	Good
3	3W	1W	${}^{3}A_3B_1$	Good
4	4R	1O	${}^{4}A_4B_1$	Failed
5	1W	2W	${}^{5}A_1B_2$	Good
6	2W	2W	${}^{6}A_2B_2$	Good
7	3W	2W	${}^{7}A_3B_2$	Good
8	4R	2O	${}^{8}A_4B_2$	Failed
9	1W	3W	${}^{9}A_1B_3$	Good
10	2W	3W	${}^{10}A_2B_3$	Good
11	3W	3W	${}^{11}A_3B_3$	Good
12	4R	3O	${}^{12}A_4B_3$	Failed
13	1O	4R	${}^{13}A_1B_4$	Failed
14	2O	4R	${}^{14}A_2B_4$	Failed
15	3O	4R	${}^{15}A_3B_4$	Failed

Table 1 System States

4.2.2 PERFECT REPAIR

- To develop the system model with perfect repair, the health state 1,2 and 3 are considered as working state and Health state 4 is considered as repair state.
- With Corrective Maintenance, component is brought from failed health state ‘4’ to good health state ‘1’ i.e. 4-1
- With Corrective as well as Opportunistic Maintenance component is brought from failed health state ‘4’ to good health state ‘1’, from state ‘3’ to good health state ‘1’ and from state ‘2’ to ‘1’ i.e. 4-1, 3-1 and 2-1.

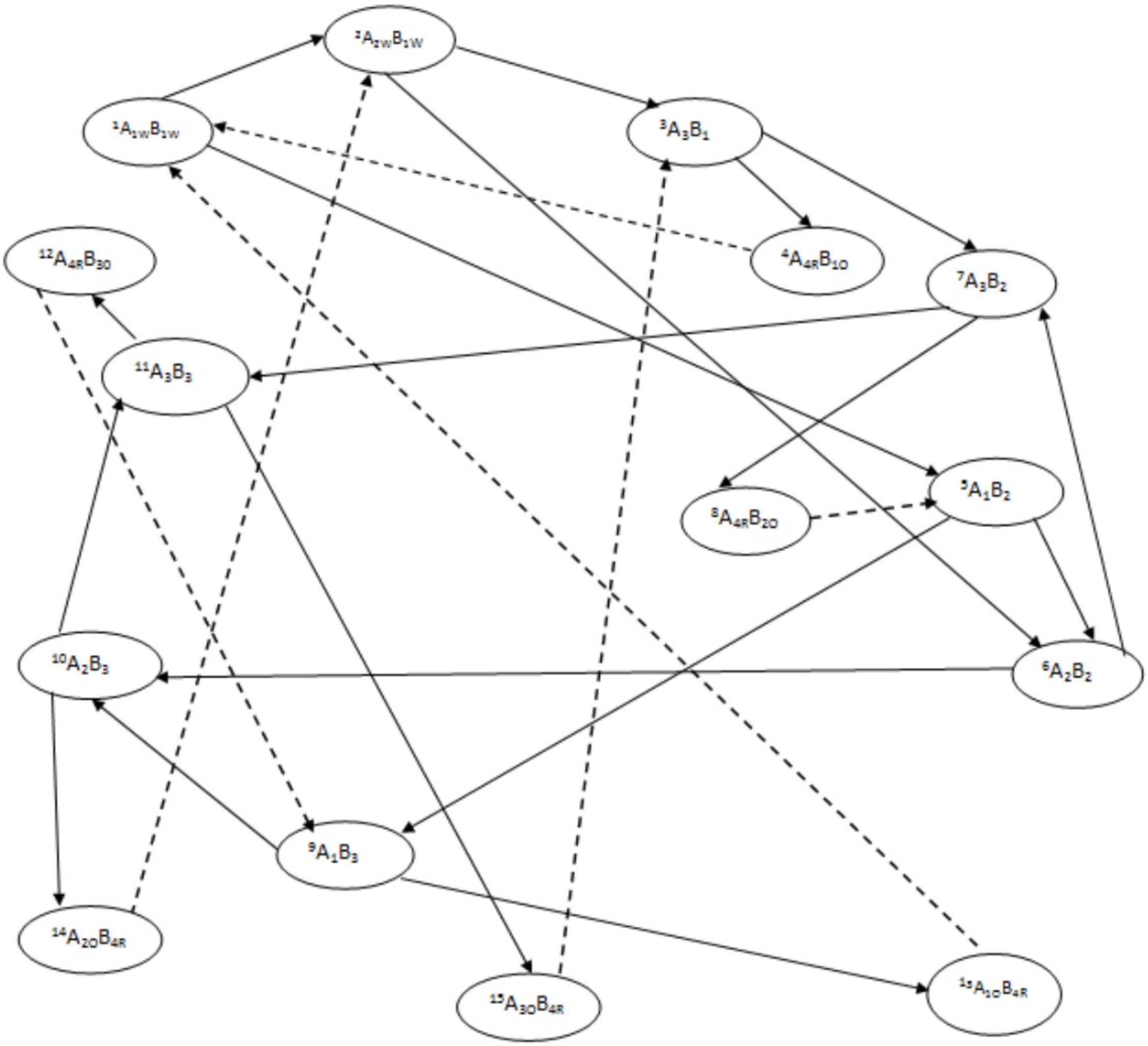


Figure 13 Corrective Maintenance (Perfect repair) without opportunistic maintenance model

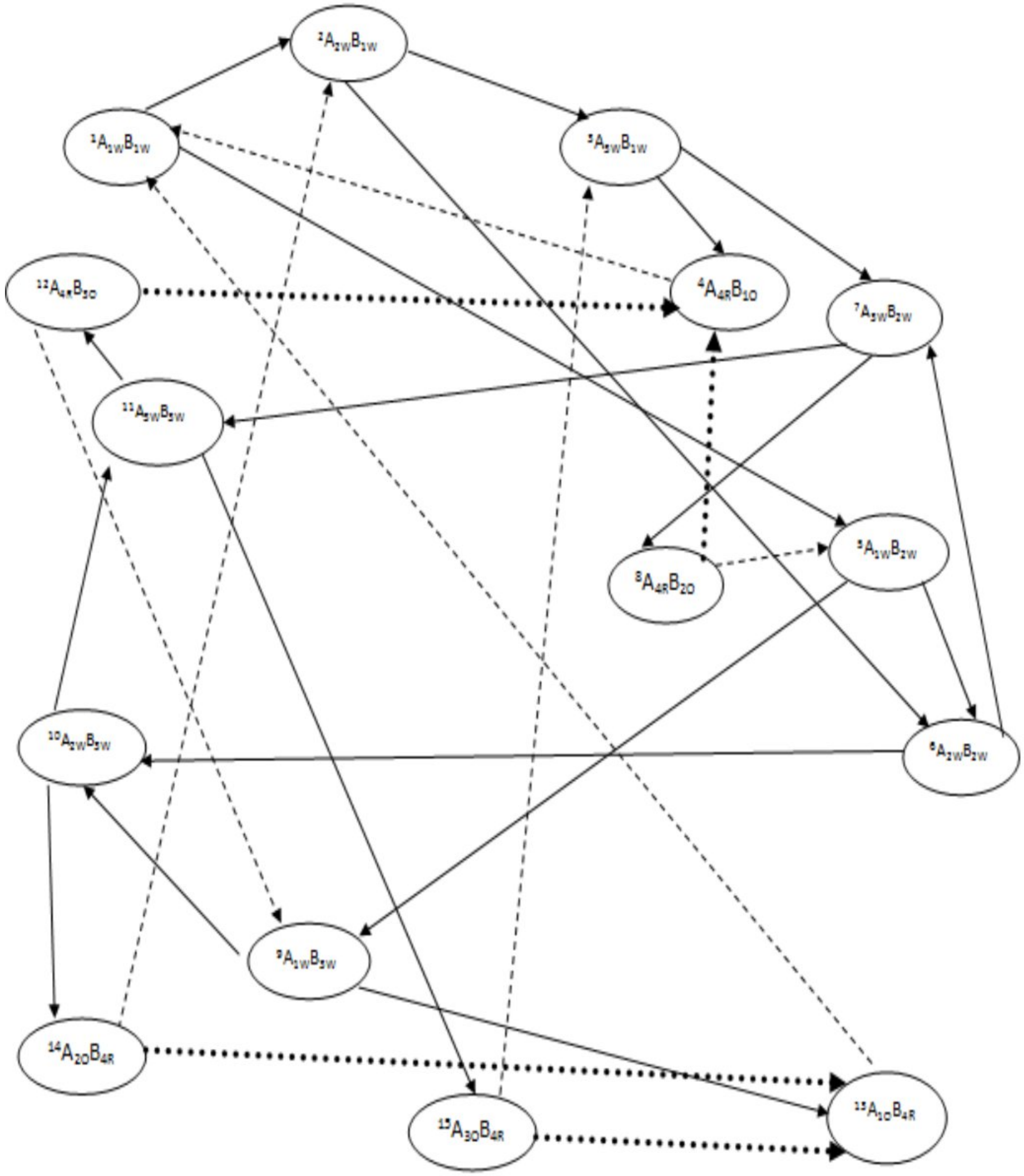


Figure 14 Corrective maintenance (Perfect repair) with opportunistic maintenance model

4.2.3 Imperfect Repair

- In this model, it is assumed that component is brought from failed health state ‘4’ to state ‘2’ with corrective maintenance.
- With Corrective as well as Opportunistic Maintenance component is brought from failed health state ‘4’ to good health state ‘2’, from state ‘3’ to good health state ‘1’ and from state ‘2’ to ‘1’ i.e. 4-2, 3-1 and 2-1.

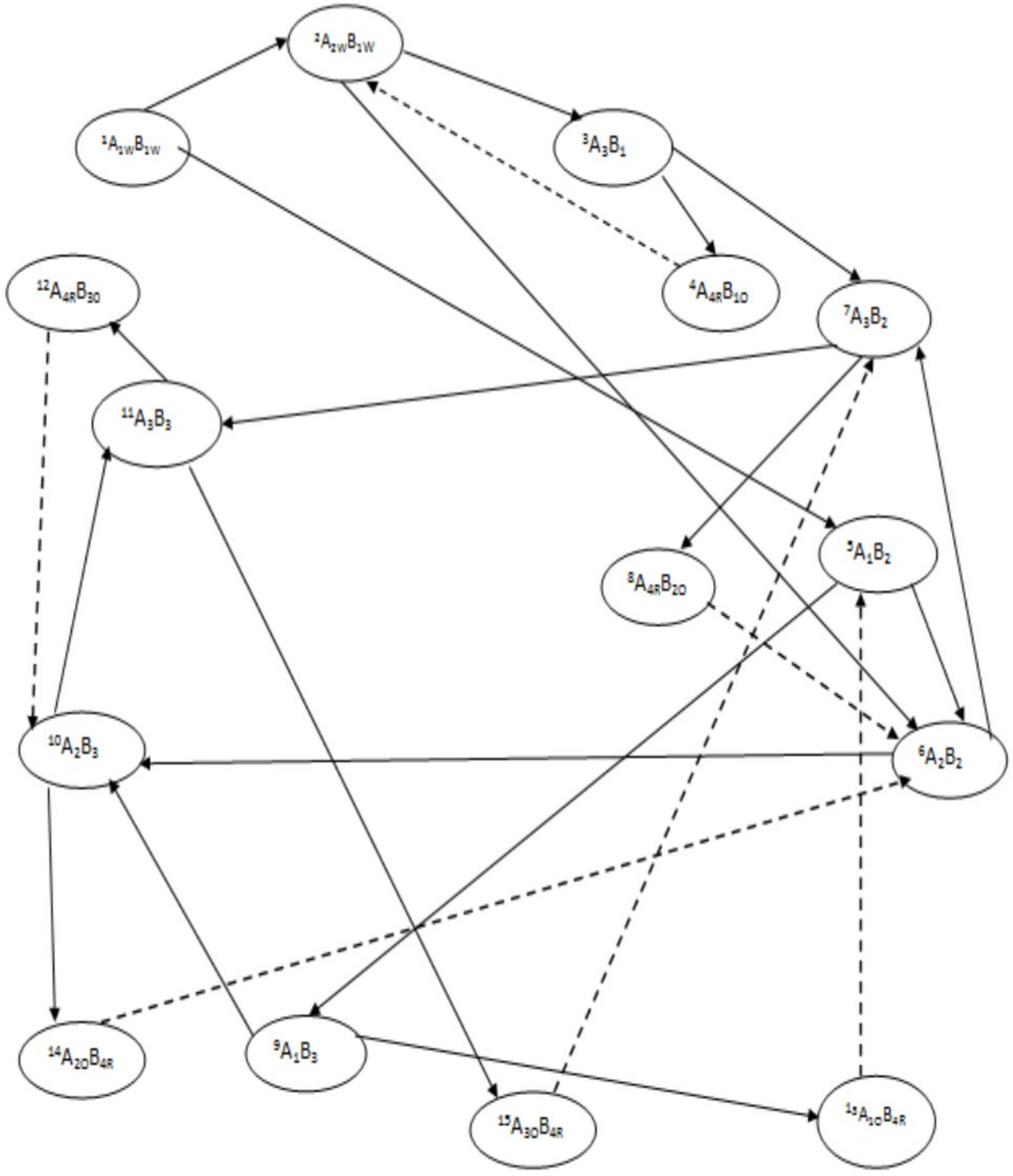


Figure 15 Corrective maintenance (Imperfect repair) without opportunistic maintenance model

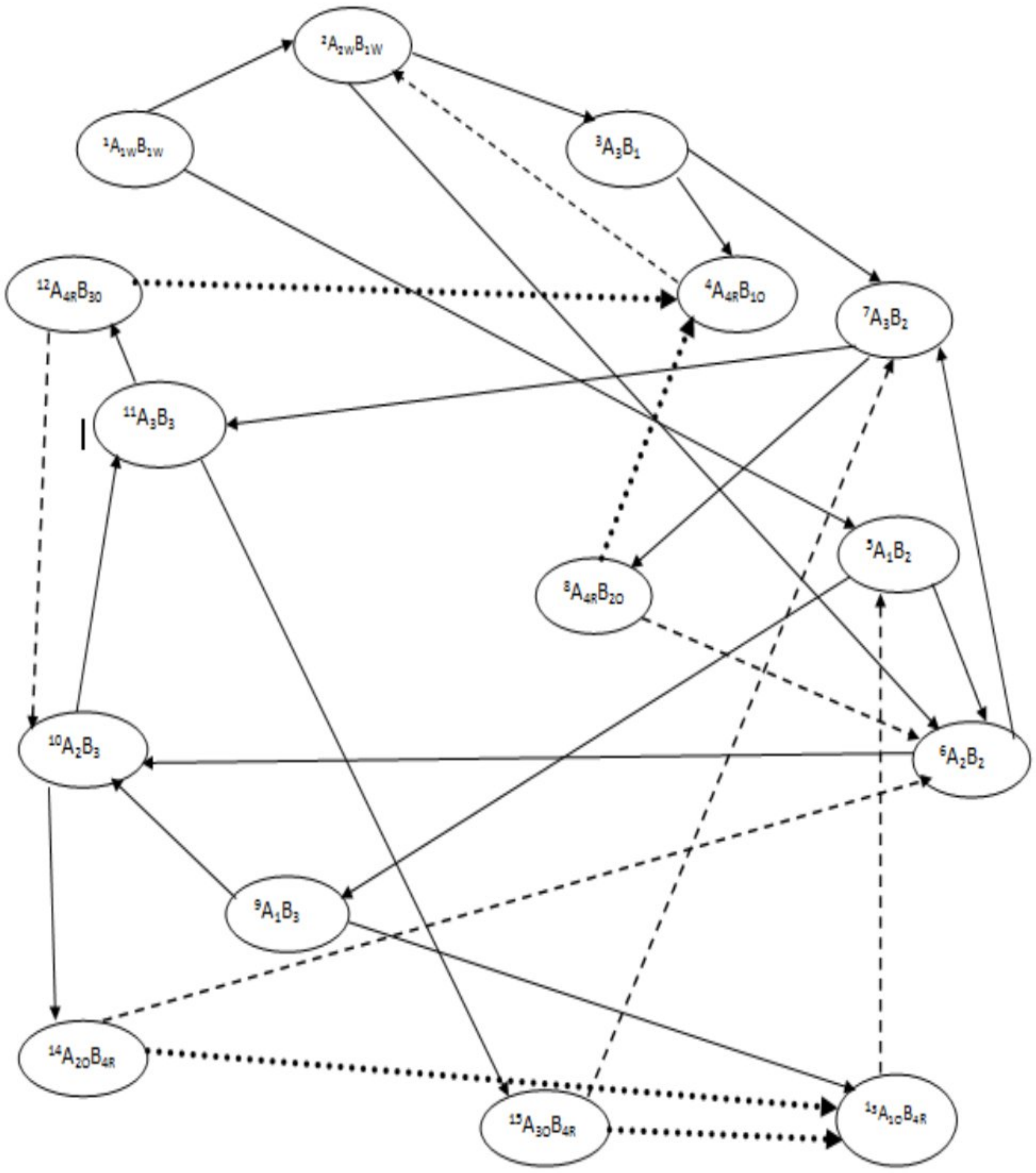


Figure 16 Corrective Maintenance (Imperfect repair) with opportunistic maintenance model

4.2.4 Minimal Repair

- In this model, it is assumed that component is brought from failed health state ‘4’ to state ‘3’ with corrective maintenance.
- With Corrective as well as Opportunistic Maintenance component is brought from failed health state ‘4’ to good health state ‘2’, from state ‘3’ to good health state ‘1’ and from state ‘2’ to ‘1’ i.e. 4-2, 3-2 and 2-1.

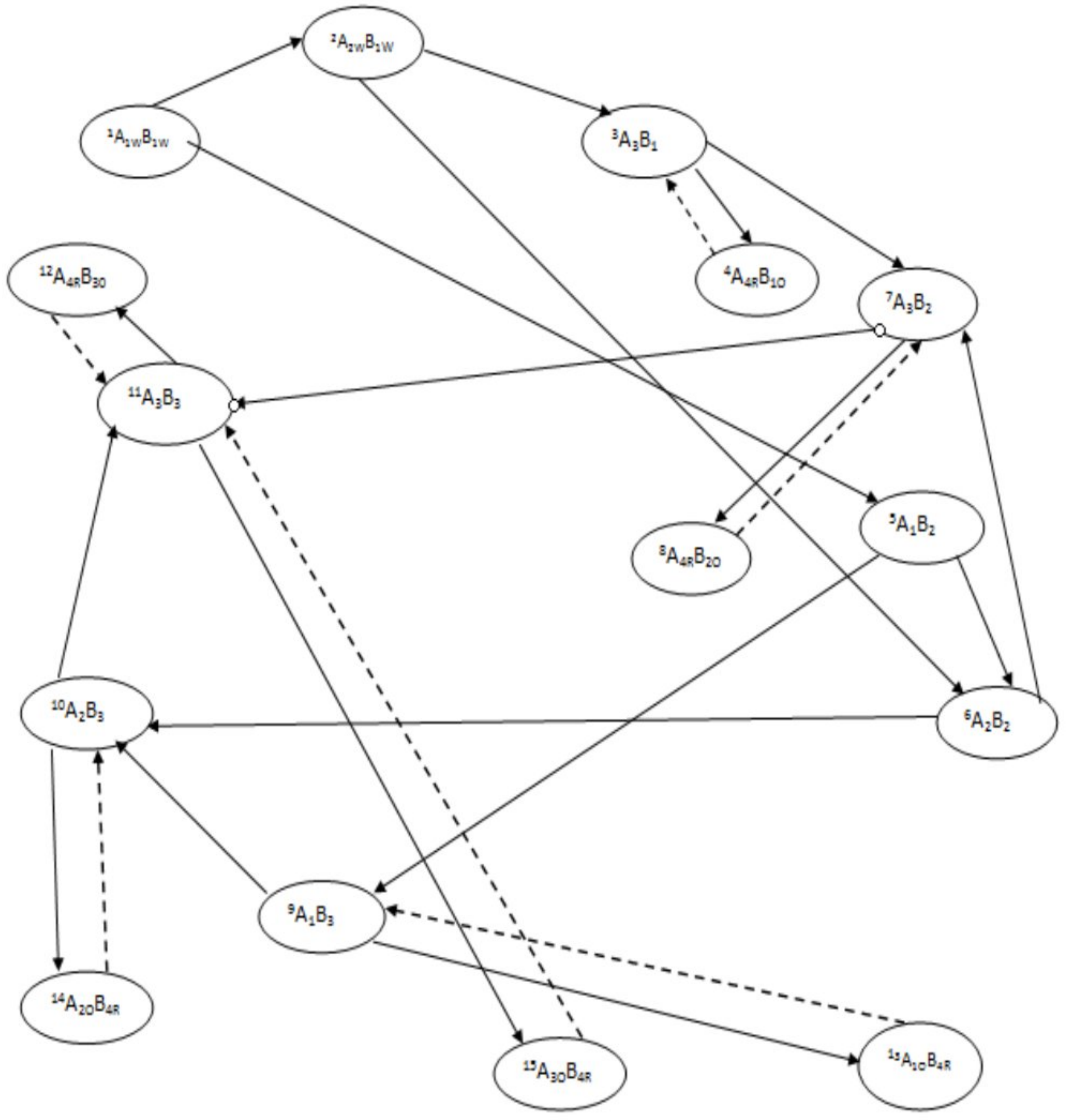


Figure 17 Corrective maintenance (Minimal repair) without opportunistic maintenance model

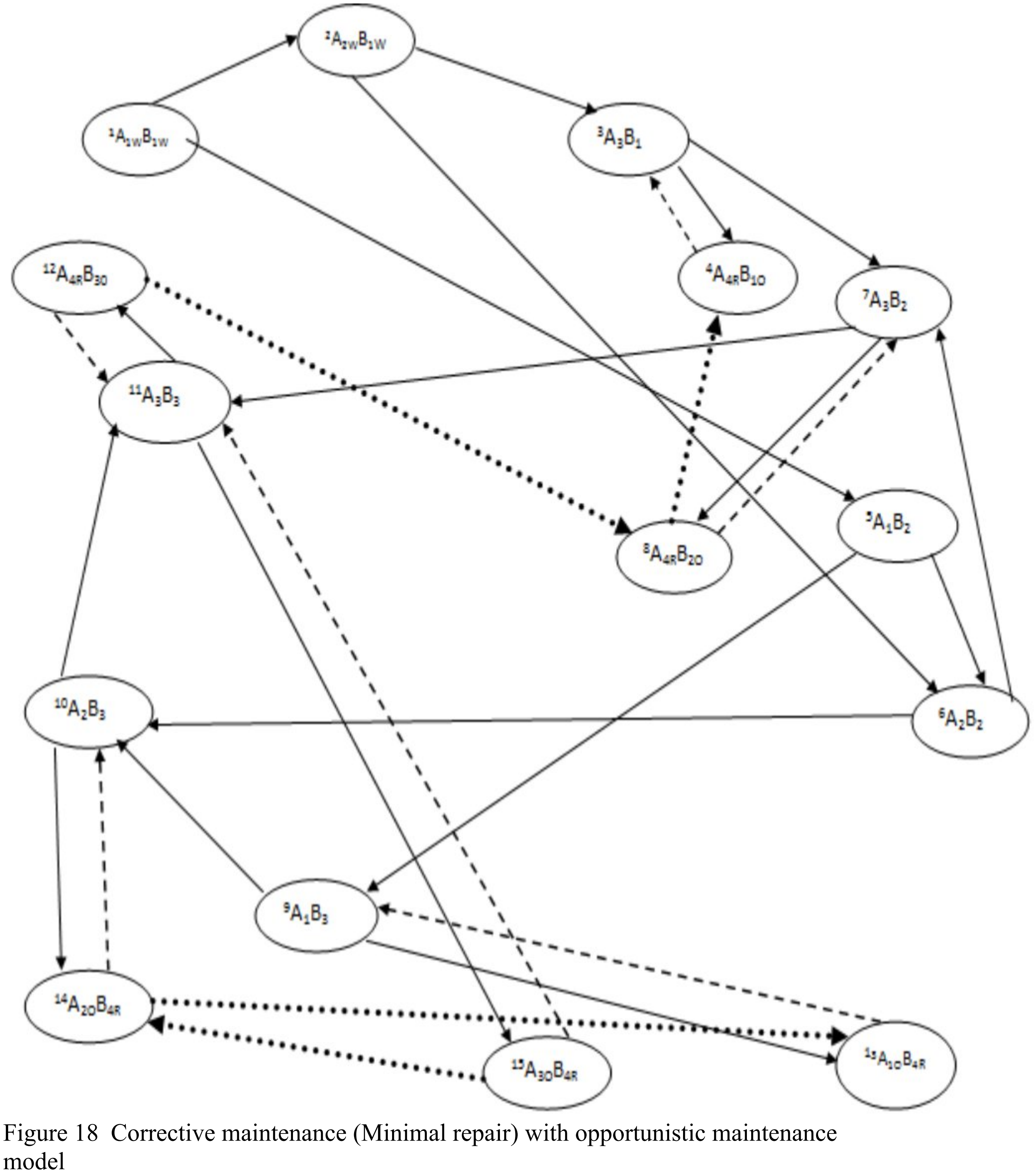


Figure 18 Corrective maintenance (Minimal repair) with opportunistic maintenance model

CHAPTER-5 SOLUTION OF SYSTEM MODEL AND RESULT ANALYSIS

After developing the model as above, we will now obtain the solution using analytical approach (Markov Analysis).

5.1 Equation for Markov Analysis of CBM Model

1. ${}^{d}P_{d1}/dt$ = $-\lambda_{d1d2}P_{d1}(t) - \lambda_{d1i1}P_{d1}(t) - \lambda d_{1fr}P_{d1}(t) + \mu_{i1d1}P_{i1}(t) + \mu_{m1d1}P_{m1}(t) + \mu_{mm2d1}P_{mm2}(t) + \mu_{mm3}P_{mm3}(t) + \mu_{frd1}P_{fr}(t)$
2. ${}^{d}P_{d2}/dt$ = $\lambda_{d1d1}P_{d1}(t) - \lambda_{d2d3}P_{d2}(t) - \lambda_{d2fr}P_{d2}(t) - \lambda_{d2i2}P_{d2}(t) + \mu_{i2d2}P_{i2}(t) + \mu_{m2d2}P_{m2}(t) + \mu_{im3d2}P_{im3}(t)$
3. ${}^{d}P_{d3}/dt$ = $\lambda_{d2d3}P_{d2}(t) - \lambda_{d3i3}P_{d3}(t) - \lambda_{d3fr}P_{d3}(t) + \mu_{m3d3}P_{m3}(t) + \mu_{i3d3}P_{i3}(t)$
4. ${}^{d}P_{I1}/dt$ = $\lambda_{d1i1}P_{d1}(t) - \lambda_{i1m1}P_{i1}(t) - \mu_{i1d1}P_{i1}(t)$.
5. ${}^{d}P_{I2}/dt$ = $\lambda_{2i2}P_{d2}(t) - \lambda_{i2m2}P_{i2}(t) - \lambda_{i2mm2}P_{i2}(t) - \mu_{i2d2}P_{i2}(t)$
6. ${}^{d}P_{I3}/dt$ = $\lambda_{d3i3}P_{d3}(t) - \lambda_{i3m3}P_{i3}(t) - \lambda_{i3mm3}P_{i3}(t) - \lambda_{i3m3}P_{i3}(t) - \mu_{i3d3}P_{i3}(t)$
7. ${}^{d}P_{Fr}/dt$ = $\lambda_{d1fr}P_{d1}(t) + \ddot{e}_{dfr}P_{d2}(t) + \ddot{e}_{d3fr}P_{d3}(t) - \mu_{frd1}P_{fr}(t)$
8. ${}^{d}P_{m1}/dt$ = $\lambda_{i1m1}P_{i1}(t) - \mu_{m1d1}P_{m1}(t)$
9. ${}^{d}P_{m2}/dt$ = $\lambda_{i2m2}P_{i2}(t) - \mu_{m2d2}P_{m2}(t)$
10. ${}^{d}P_{mm2}/dt$ = $\lambda_{i2mm2}P_{i2}(t) - \mu_{mm2d1}P_{mm2}(t)$
11. ${}^{d}P_{m3}/dt$ = $\lambda_{i3m3}P_{i3}(t) - \mu_{m3d3}P_{m3}(t)$
12. $dPim3/dt$ = $\lambda_{i3m3}P_{i3}(t) - \mu_{im3}P_{i3}(t)$
13. ${}^{d}P_{mm3}/dt$ = $\lambda_{i3mm3}P_{i3}(t) - \mu_{mm3d1}P_{mm3}(t)$

Table 2 CBM Distribution parameters for failure/Repair/Inspection Interval Transition of Centrifugal Pump(Source of Data – Baringer.com)

S.no.	Transition	PARAMETER	VALUE
1	D_1D_2	λ_{D1D2}	0.00025
2	D2D3	λ_{D2D3}	0.00067
3	D1I1	λ_{D1I1}	0.004
4	I1M1	Λ_{I1M1}	0.5
5	D2I2	λ_{D2I2}	0.00595
6	I2M2	λ_{I2M2}	0.25
7	I2MM2	λ_{I2MM2}	0.25
8	D3I3	λ_{D3I3}	0.01
9	I3M3	λ_{I3M3}	0.125
10	I3MM3	λ_{I3MM3}	0.125
11	I3IM3	λ_{I3IM3}	0.125
12	D1Fr	λ_{D1Fr}	0.00002
13	D2Fr	λ_{D2Fr}	0.00002
14	D3Fr	λ_{D3Fr}	0.00002
15	I1D1	μ_{I1D1}	0.005
16	I2D2	μ_{I2D2}	0.025
17	I3D3	μ_{I3D3}	0.0125
18	M1D1	μ_{M1D1}	0.05
19	M3D2	μ_{M2D2}	0.025
20	MM2D1	μ_{MM2D1}	0.0125
21	M3D3	μ_{M3D3}	0.016
22	IM3D2	μ_{IM3D2}	0.01
23	MM3D1	μ_{MM3D1}	0.0625
24	FrD1	μ_{FrD1}	0.02

5.2 Equation for Markov Analysis of OM Model

5.2.1 Perfect repair without OM

1. $dP_1/dt = -(\lambda_{1\text{-}5} + \lambda_{1\text{-}2})*P_1(t) + \mu_{4\text{-}1}*P_4(t) + \mu_{13\text{-}1}*P_{13}(t)$

2. $dP_2/dt = \lambda_{1\text{-}2}*P_1(t) - (\lambda_{2\text{-}3} + \lambda_{2\text{-}6})*P_2(t) + \mu_{14\text{-}2}*P_{14}(t)$

3. $dP_3/dt = \lambda_{2\text{-}3}*P_2(t) - (\lambda_{3\text{-}4} + \lambda_{3\text{-}7})*P_3(t) + \mu_{15\text{-}3}*P_{15}(t)$

4. $dP_4/dt = \lambda_{3\text{-}4}*P_3(t) - \mu_{4\text{-}1}*P_4(t)$

5. $dP_5/dt = \mu_{8\text{-}5}*P_8(t) - (\lambda_{5\text{-}6} + \lambda_{5\text{-}9})*P_5(t) + \lambda_{1\text{-}5}*P_1(t)$

6. $dP_6/dt = \lambda_{5\text{-}6}*P_5(t) + \lambda_{2\text{-}6}*P_2(t) - (\lambda_{6\text{-}7} + \lambda_{6\text{-}10})*P_6(t)$

7. $dP_7/dt = \lambda_{3\text{-}7}*P_3(t) + \lambda_{6\text{-}7}*P_6(t) - (\lambda_{7\text{-}11} + \lambda_{7\text{-}8})*P_7(t)$

8. $dP_8/dt = \lambda_{7\text{-}8}*P_7(t) - \mu_{8\text{-}5}*P_8(t)$

9. $dP_9/dt = \lambda_{5\text{-}9}*P_5(t) + \mu_{12\text{-}9}*P_{12}(t) - (\lambda_{9\text{-}10} + \lambda_{9\text{-}13})*P_9(t)$

10. $dP_{10}/dt = \lambda_{6\text{-}10}*P_6(t) + \lambda_{9\text{-}10}*P_9(t) - (\lambda_{10\text{-}11} + \lambda_{10\text{-}14})*P_{10}(t)$

11. $dP_{11}/dt = \lambda_{10\text{-}11}*P_{10}(t) + \lambda_{7\text{-}11}*P_7(t) - (\lambda_{11\text{-}15} + \lambda_{11\text{-}12})*P_{11}(t)$

12. $dP_{12}/dt = \lambda_{11\text{-}12}*P_{11}(t) - \mu_{12\text{-}9}*P_{12}(t)$

13. $dP_{13}/dt = \lambda_{9\text{-}13}*P_9(t) - \mu_{13\text{-}1}*P_{13}(t)$

14. $dP_{14}/dt = \lambda_{10\text{-}14}*P_{10}(t) - \mu_{14\text{-}2}*P_{14}(t)$

15. $dP_{15}/dt = \lambda_{11\text{-}15}*P_{11}(t) - \mu_{15\text{-}3}*P_{15}(t)$

5.2.2 Perfect repair with OM

1. $dP_1/dt = -(\lambda_{1\text{-}5}+ \lambda_{1\text{-}2})*P_1(t)+\mu_{4\text{-}1}*P_4(t)+ \mu_{13\text{-}1}*P_{13}(t)$

2. $dP_2/dt = \lambda_{1\text{-}2}*P_1(t)-(\lambda_{2\text{-}3}+ \lambda_{2\text{-}6})*P_2(t)+ \mu_{14\text{-}2}*P_{14}(t)$

3. $dP_3/dt = \lambda_{2\text{-}3}*P_2(t)-(\lambda_{3\text{-}4}+ \lambda_{3\text{-}7})*P_3(t)+ \mu_{15\text{-}3}*P_{15}(t)$

4. $dP_4/dt = \lambda_{3\text{-}4}*P_3(t)- \mu_{4\text{-}1}*P_4(t) + \mu_{12\text{-}4}*P_{12}(t)+ \mu_{8\text{-}4}*P_8(t)$

5. $dP_5/dt = \mu_{8\text{-}5}*P_8(t)-(\lambda_{5\text{-}6}+ \lambda_{5\text{-}9})*P_5(t)+ \lambda_{1\text{-}5}*P_1(t)$

6. $dP_6/dt = \lambda_{5\text{-}6}*P_5(t)+ \lambda_{2\text{-}6}*P_2(t) - (\lambda_{6\text{-}7}+ \lambda_{6\text{-}10})*P_6(t)$

7. $dP_7/dt = \lambda_{3\text{-}7}*P_3(t)+ \lambda_{6\text{-}7}*P_6(t)-(\lambda_{7\text{-}11}+ \lambda_{7\text{-}8})*P_7(t)$

8. $dP_8/dt = \lambda_{7\text{-}8}*P_7(t)- \mu_{8\text{-}5}*P_8(t) - \mu_{8\text{-}4}*P_8(t)$

9. $dP_9/dt = \lambda_{5\text{-}9}*P_5(t)+ \mu_{12\text{-}9}*P_{12}(t)-(\lambda_{9\text{-}10}+ \lambda_{9\text{-}13})*P_9(t)$

10. $dP_{10}/dt = \lambda_{6\text{-}10}*P_6(t) + \lambda_{9\text{-}10}*P_9(t) - (\lambda_{10\text{-}11}+ \lambda_{10\text{-}14})*P_{10}(t)$

11. $dP_{11}/dt = \lambda_{10\text{-}11}*P_{10}(t) + \lambda_{7\text{-}11}*P_7(t) - (\lambda_{11\text{-}15}+ \lambda_{11\text{-}12})*P_{11}(t)$

12. $dP_{12}/dt = \lambda_{11\text{-}12}*P_{11}(t)- \mu_{12\text{-}9}*P_{12}(t)- \mu_{12\text{-}4}*P_{12}(t)$

13. $dP_{13}/dt = \lambda_{9\text{-}13}*P_9(t) - \mu_{13\text{-}1}*P_{13}(t) + \mu_{14\text{-}13}*P_{14}(t)+ \mu_{15\text{-}13}*P_{15}(t)$

14. $dP_{14}/dt = \lambda_{10\text{-}14}*P_{10}(t) - \mu_{14\text{-}2}*P_{14}(t) - \mu_{14\text{-}13}*P_{14}(t)$

15. $dP_{15}/dt = \lambda_{11-15}*P_{11}(t) - \mu_{15-3}*P_{15}(t) - \mu_{15-13}*P_{15}(t)$

5.2.3 Imperfect repair without OM

1. $dP_1/dt = -(\lambda_{1-5}+ \lambda_{1-2})*P_1(t)$

2. $dP_2/dt = -(\lambda_{2-3}+ \lambda_{2-6}) *P_2(t)+ \mu_{4-2}*P_4(t)+ \lambda_{1-2}*P_1(t)$

3. $dP_3/dt = -(\lambda_{3-4}+ \lambda_{3-7})*P_3(t)+ \lambda_{2-3}*P_2(t)$

4. $dP_4/dt = \lambda_{3-4}*P_3(t)- \mu_{4-2}*P_4(t)$

5. $dP_5/dt = -(\lambda_{5-6}+ \lambda_{5-9}) *P_5(t)+ \lambda_{1-5}*P_1(t)+ \mu_{13-5}*P_{13}(t)$

6. $dP_6/dt = \lambda_{5-6}*P_5(t)+\lambda_{2-6}*P_2(t)-(\lambda_{6-7}+\lambda_{6-10})*P_6(t)+\mu_{8-6}*P_8(t)+\mu_{14-6}*P_{14}(t)$

7. $dP_7/dt = \mu_{15-7}*P_{15}(t)+ \lambda_{6-7}*P_6(t)+ \lambda_{3-7}*P_3(t) - (\lambda_{7-11}+ \lambda_{7-8})*P_7(t)$

8. $dP_8/dt = \lambda_{7-8}*P_7(t)- \mu_{8-6}* P_8(t)$

9. $dP_9/dt = \lambda_{5-9}* P_5(t)-(\lambda_{9-10}+ \lambda_{9-13})* P_9(t)$

10. $dP_{10}/dt = \lambda_{6-10}*P_6(t)+ \lambda_{9-10}*P_9(t)-(\lambda_{10-11}+\lambda_{10-14})*P_{10}(t)+ \mu_{12-10}*P_{12}(t)$

11. $dP_{11}/dt = \lambda_{10-11}* P_{10}(t)+ \lambda_{7-11}* P_7(t)-(\lambda_{11-15}+ \lambda_{11-12})* P_{11}(t)$

12. $dP_{12}/dt = \lambda_{11-12}*P_{11}(t) - \mu_{12-10}*P_{12}(t)$

13. $dP_{13}/dt = \lambda_{9-13}*P_9(t)- \mu_{13-5}*P_{13}(t);$

14. $dP_{14}/dt = \lambda_{10-14}*P_{10}(t)- \mu_{14-6}*P_{14}(t)$

15. $dP_{15}/dt = \lambda_{11-15} * P_{11}(t) - \mu_{15-7} * P_{15}(t)$

5.2.4 Imperfect repair with OM

1. $dP_1/dt = -(\lambda_{1-5} + \lambda_{1-2}) * P_1(t)$

2. $dP_2/dt = -(\lambda_{2-3} + \lambda_{2-6}) * P_2(t) + \mu_{4-2} * P_4(t) + \lambda_{1-2} * P_1(t)$

3. $dP_3/dt = -(\lambda_{3-4} + \lambda_{3-7}) * P_3(t) + \lambda_{2-3} * P_2(t)$

4. $dP_4/dt = \lambda_{3-4} * P_3(t) - \mu_{4-2} * P_4(t) + \mu_{12-4} * P_{12}(t) + \mu_{8-4} * P_4(t)$

5. $dP_5/dt = -(\lambda_{5-6} + \lambda_{5-9}) * P_5(t) + \lambda_{1-5} * P_1(t) + \mu_{13-5} * P_{13}(t)$

6. $dP_6/dt = \lambda_{5-6} * P_5(t) + \lambda_{2-6} * P_2(t) - (\lambda_{6-7} + \lambda_{6-10}) * P_6(t) + \mu_{8-6} * P_8(t) + \mu_{14-6} * P_{14}(t)$

7. $dP_7/dt = \mu_{15-7} * P_{15}(t) + \lambda_{6-7} * P_6(t) + \lambda_{3-7} * P_3(t) - (\lambda_{7-11} + \lambda_{7-8}) * P_7(t)$

8. $dP_8/dt = \lambda_{7-8} * P_7(t) - \mu_{8-6} * P_8(t) - \mu_{8-4} * P_4(t)$

9. $dP_9/dt = \lambda_{5-9} * P_5(t) - (\lambda_{9-10} + \lambda_{9-13}) * P_9(t)$

10. $dP_{10}/dt = \lambda_{6-10} * P_6(t) + \lambda_{9-10} * P_9(t) - (\lambda_{10-11} + \lambda_{10-14}) * P_{10}(t) + \mu_{12-10} * P_{12}(t)$

11. $dP_{11}/dt = \lambda_{10-11} * P_{10}(t) + \lambda_{7-11} * P_7(t) - (\lambda_{11-15} + \lambda_{11-12}) * P_{11}(t)$

12. $dP_{12}/dt = \lambda_{11-12} * P_{11}(t) - \mu_{12-10} * P_{12}(t) - \mu_{12-4} * P_{12}(t)$

13. $dP_{13}/dt = \lambda_{9-13} * P_9(t) - \mu_{13-5} * P_{13}(t) + \mu_{14-13} * P_{14}(t) + \mu_{15-13} * P_{15}(t)$

14. $dP_{14}/dt = \lambda_{10-14} * P_{10}(t) - \mu_{14-6} * P_{14}(t) - \mu_{14-13} * P_{14}(t)$

15.$dP_{15}/dt = \lambda_{11-15} * P_{11}(t) - \mu_{15-7} * P_{15}(t) - \mu_{15-13} * P_{15}(t)$

5.2.5 Minimal repair without OM

1 $dP_1/dt = -(\lambda_{1-5} + \lambda_{1-2}) * P_1(t)$

2 $dP_2/dt = -(\lambda_{2-3} + \lambda_{2-6}) * P_2(t)$

3 $dP_3/dt = -(\lambda_{3-4} + \lambda_{3-7}) * P_3(t) + \mu_{4-3} * P_4(t) + \lambda_{2-3} * P_2(t)$

4 $dP_4/dt = \lambda_{3-4} * P_3(t) - \mu_{4-3} * P_4(t)$

5 $dP_5/dt = -(\lambda_{5-6} + \lambda_{5-9}) * P_5(t) + \lambda_{1-5} * P_1(t)$

6 $dP_6/dt = \lambda_{5-6} * P_5(t) + \lambda_{2-6} * P_2(t) - (\lambda_{6-7} + \lambda_{6-10}) * P_6(t)$

7 $dP_7/dt = \mu_{8-7} * P_8(t) + \lambda_{6-7} * P_6(t) - (\lambda_{7-11} + \lambda_{7-8}) * P_7(t) + \lambda_{3-7} * P_3(t)$

8 $dP_8/dt = \lambda_{7-8} * P_7(t) - \mu_{8-7} * P_8(t)$

9 $dP_9/dt = \lambda_{5-9} * P_5(t) + \mu_{13-9} * P_{13}(t) - (\lambda_{9-10} + \lambda_{9-13}) * P_9(t)$

10 $dP_{10}/dt = \lambda_{6-10} * P_6(t) + \lambda_{9-10} * P_9(t) - (\lambda_{10-11} + \lambda_{10-14}) * P_{10}(t) + \mu_{14-10} * P_{14}(t)$

11 $dP_{11}/dt = \lambda_{10-11} * P_{10}(t) + \lambda_{7-11} * P_7(t) - (\lambda_{11-15} + \lambda_{11-12}) * P_{11}(t) + \mu_{15-11} * P_{15}(t) +$

$\mu_{12-11} * P_{12}(t)$

12 $dP_{12}/dt = \lambda_{11-12} * P_{11}(t) - \mu_{12-11} * P_{12}(t)$

13 $dP_{13}/dt = \lambda_{9-13} * P_9(t) - \mu_{13-9} * P_{13}(t)$

14 $dP_{14}/dt = \lambda_{10\text{-}14} * P_{10}(t) - \mu_{14\text{-}10} * P_{14}(t)$

15 $dP_{15}/dt = \lambda_{11\text{-}15} * P_{11}(t) - \mu_{15\text{-}11} * P_{15}(t)$

5.2.6 Minimal repair with OM

1. $dP_1/dt = -(\lambda_{1\text{-}5} + \lambda_{1\text{-}2}) * P_1(t)$

2. $dP_2/dt = -(\lambda_{2\text{-}3} + \lambda_{2\text{-}6}) * P_2(t)$

3. $dP_3/dt = -(\lambda_{3\text{-}4} + \lambda_{3\text{-}7}) * P_3(t) + \mu_{4\text{-}3} * P_4(t) + \lambda_{2\text{-}3} * P_2(t)$

4. $dP_4/dt = \lambda_{3\text{-}4} * P_3(t) - \mu_{4\text{-}3} * P_4(t) + \mu_{8\text{-}4} * P_8(t)$

5. $dP_5/dt = -(\lambda_{5\text{-}6} + \lambda_{5\text{-}9}) * P_5(t) + \lambda_{1\text{-}5} * P_1(t)$

6. $dP_6/dt = \lambda_{5\text{-}6} * P_5(t) + \lambda_{2\text{-}6} * P_2(t) - (\lambda_{6\text{-}7} + \lambda_{6\text{-}10}) * P_6(t)$

7. $dP_7/dt = \mu_{8\text{-}7} * P_8(t) + \lambda_{6\text{-}7} * P_6(t) - (\lambda_{7\text{-}11} + \lambda_{7\text{-}8}) * P_7(t) + \lambda_{3\text{-}7} * P_3(t)$

8. $dP_8/dt = \lambda_{7\text{-}8} * P_7(t) - \mu_{8\text{-}7} * P_8(t) + \mu_{12\text{-}8} * P_{12}(t) - \mu_{8\text{-}4} * P_8(t)$

9. $dP_9/dt = \lambda_{5\text{-}9} * P_5(t) + \mu_{13\text{-}9} * P_{13}(t) - (\lambda_{9\text{-}10} + \lambda_{9\text{-}13}) * P_9(t)$

10. $dP_{10}/dt = \lambda_{6\text{-}10} * P_6(t) + \lambda_{9\text{-}10} * P_9(t) - (\lambda_{10\text{-}11} + \lambda_{10\text{-}14}) * P_{10}(t) + \mu_{14\text{-}10} * P_{14}(t)$

11. $dP_{11}/dt = \lambda_{10\text{-}11} * P_{10}(t) + \lambda_{7\text{-}11} * P_7(t) - (\lambda_{11\text{-}15} + \lambda_{11\text{-}12}) * P_{11}(t) + \mu_{15\text{-}11} * P_{15}(t) + \mu_{12\text{-}11} * P_{12}(t)$

12. $dP_{12}/dt = \lambda_{11\text{-}12} * P_{11}(t) - \mu_{12\text{-}11} * P_{12}(t) - \mu_{12\text{-}8} * P_{12}(t)$

13. $dP_{13}/dt = \lambda_{9\text{-}13} * P_9(t) - \mu_{13\text{-}9} * P_{13}(t) + \mu_{14\text{-}13} * P_{14}(t)$

14. $dP_{14}/dt = \lambda_{10\text{-}14} * P_{10}(t) - \mu_{14\text{-}10} * P_{14}(t) + \mu_{15\text{-}14} * P_{15}(t) - \mu_{14\text{-}13} * P_{14}(t)$

15. $dP_{15}/dt = \lambda_{11\text{-}15} * P_{11}(t) - \mu_{15\text{-}11} * P_{15}(t) - \mu_{15\text{-}14} * P_{15}(t)$

5.3 Simulation Algorithm of CBM Model

To verify the solution obtained analytically, we use the simulation techniques. Here we are using Monte Carlo Simulation method. It helps in obtaining the result by generating random numbers. With the use of Matlab software for generating random numbers, we proceed to achieve the solution of our model systematically. This step wise procedure is as follows:

We start observing the working of our component from beginning. It is in perfect condition and is brand new. We designate this as stage one, D_1 .

Now we make an algorithm follow these steps:

Step 1.

We start with stage one, D_1. Now using Matlab we generate three random numbers between 0 and 1; as from here it can go to three stages namely I_1, Fr and D_2.

These random numbers are used to calculate the time values for each of the three stages. We proceed the stage for which minimum time is obtained.

Suppose the least time is obtained for Fr.

Step 2.

This means the component goes towards the random failure. Here we need to undertake substantial work and replacement work on it and bring it back to stage D_1.

Now we proceed with step 1 again.

Suppose the least time is obtained for I_1.

Step 3.

This means we undertake inspection of the component. Now, we generate random variable between 0 and 1.

It its value is [0.9,1]; we send it to minor repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_1.

Now we proceed with step 1 again.

Suppose the least time is obtained for D_2.

Step 4.

This means the component has arrived at the deteriorated level, D_2. Now we generate 3 random variables between 0 and 1; as from here it can go to stages namely I_2, Fr, and D_2.

Time values for these three states are calculated and we proceed to the stage for which the minimum time is obtained.

Suppose the least time is obtained for Fr.

Step 5.

This means the component goes towards the random failure. Here we need to undertake substantial repair and replacement work on it and bring it to stage D_1.

Now we proceed with step 1 again.

Suppose the least time is obtained for I_2.

Step 6.

This means we undertake inspection of the component. Now, we generate random variable between 0 and 1.

It its value is [0,0.7]; we approve the condition as okay and send it back to stage D_2. Now the step 4 is repeated again.

It its value is [0.7,0.9]; we send it to minor repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_2. Now the step 4 is repeated again.

It its value is [0.9,1]; we send it to major repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_1. Now the step 1 is repeated again.

Suppose the least time is obtained for D_3.

Step 7.

This means the component has arrived at the deteriorated level, D_2. Now we generate 2 random variables between 0 and 1; as from here it can go to stages namely I_3 and Fr.

Time values for these two states are calculated and we proceed to the stage for which the minimum time is obtained.

Suppose the least time is obtained for Fr.

Step 8.

This means we undertake inspection of the component. Now, we generate random variable between 0 and 1.

It its value is [0,0.2]; we approve the condition as okay and send it back to stage D_3. Now the step 7 is repeated again.

It its value is [0.2,0.4]; we send it to minor repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_3. Now the step 7 is repeated again.

It its value is [0.4,0.8]; we send it to imperfect repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_2. Now the step 4 is repeated again.

It its value is [0.8,1]; we send it to major repair work. Now we again generate random variable to determine the time of repair work done. After that the component is again sent back to stage D_1. Now the step 1 is repeated again.

The above mentioned steps are followed in continuation forming a cycle. Here we note down each time value and run our system till the specified cycle time.

We notice that the results obtained by both the approaches are same.

5.4 Simulation Algorithm of OM Model

Initially both the components (A and B) of the system are in state 1 and 1, i.e. **1, 1** henceforth we will use the same assignment for A and B state.

We have used various random variables for each stage for deciding randomly where the system should go.

Time coefficient are considered when the stage moves from one state to another.

Repair coefficients are considered whenever one of the states reaches failed state (i.e. state 4).

Perfect repair without Opportunistic Maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B (12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied and system is bought to 1st state.

Step5: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 6: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (rt) using standard formulae.

Perfect repair with opportunistic maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B (12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied

Step 5: When any of the component goes to failed state, we do opportunistic maintenance

Step 6: In opportunistic maintenance we try to bring the other component back to better stage (two stage improvement) if the time to repair from failed state to state1 takes more time than the opportunistic maintenance of the other component , otherwise simple repair is done.

Step7: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 8: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (rt) using standard formulae.

Imperfect repair without Opportunistic Maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B (12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied and system is bought to 2nd state.

Step5: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 6: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (rt) using standard formulae.

Imperfect repair with opportunistic maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B (12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied

Step 5: When any of the component goes to failed state, we do opportunistic maintenance

Step 6: In opportunistic maintenance we try to bring the other component back to better stage (two stage improvement) if the time to repair from failed state to state 2 takes more time than the opportunistic maintenance of the other component, otherwise simple repair is done.

Step7: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 8: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (rt) using standard formulae.

Minimal repair without Opportunistic Maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B(12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied and system is bought to 3rd state.

Step5: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 6: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (rt) using standard formulae.

Minimal repair with opportunistic maintenance

Step 1: User inputs mission time, time for which he/she wants to see the availability of the system.

Step2: System initially is in 11 State.

If condition applied on random variable decides whether the system should change the state of A (21) or B (12). And subsequently it is decided with the help of random variable which state comes next using switch case.

Step3: Time (ct) is added for each passing state using standard formulae and is compared with mission time to see if it exceeds, if exceeds then the while loop breaks.

Step4: When any one of the state of A or B reaches failed state, repair mechanism is applied

Step 5: When any of the component goes to failed state, we do opportunistic maintenance

Step 6: In opportunistic maintenance we try to bring the other component back to better stage (one stage improvement) if the time to repair from failed state to state 3 takes more time than the opportunistic maintenance of the other component otherwise simple repair is done.

Step7: Repair time (rt) is calculated using standard formulae and system state is bought back to perfect state if the mission time you have entered hasn't crossed else the loop will keep iterating.

Step 8: Availability is calculated using time added for traversal from each state (ct) and repair time whenever the state fails (r) using standard formulae.

5.5 Sensitivity Analysis of CBM Model

Sensitivity refers to the change in the result obtained when one or more independent parameters considered in the calculations are varied. Sensitivity Analysis is a technique to check the sensitivity of the solution obtained. For that, keeping other factors constant, one of the parameters is varied.

VARYING THE INSPECTION INTERVAL:

In the beginning we change the periodic inspection time at I1 keeping those at I2 and I3 constant. We observe that as we decrease the periodic time, the availability of the component decreases. This is so because in the beginning the component is new and the frequent inspection lead to time wastage and increases the possibility of minor repair work on the component. Thus decrease its availability. As shown in the table below:

Table 3 Sensitivity analysis for system availability varying inspection interval for degradation stage 1.

S.No.	I1(hrs)	I2(hrs)	I3(hrs)	Availability	
				Markov	MCS
1.	50	150	100	0.8807	0.8809
2.	100	150	100	0.9194	0.9198
3.	200	150	100	0.9607	0.9609
4.	300	150	100	0.9731	0.9738
5.	400	150	100	0.9792	0.9791
6.	600	150	100	0.9859	0.9855
7.	800	150	100	0.9893	0.9895
8.	1000	150	100	0.9991	0.9997
9.	1100	150	100	0.9922	0.9923
10.	1200	150	100	0.9931	0.9933
11.	1300	150	100	0.9934	0.9935
12.	1500	150	100	0.9942	0.9945

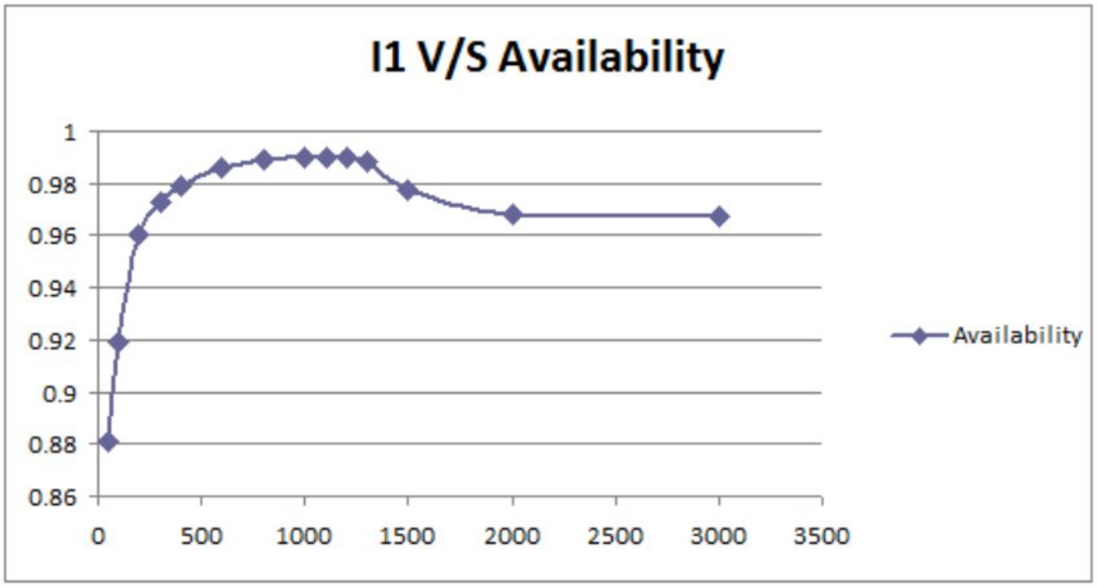


Figure 19 Inspection Interval for I1.

Next, we change the periodic inspection time at I2 keeping those at I1 and I3 constant. We observe that when we increase the periodic inspection time there is very slight increase in availability of the component. This is so , because the system has degraded to an extent that it needs frequent inspection to increase the availability of the component.

Table 4 Sensitivity analysis for system availability varying inspection interval for degradation stage 2.

S.No.	I1(hrs)	I2(hrs)	I3(hrs)	Availability	
				Markov	MCS
1.	250	50	100	0.9667	0.9669
2.	250	100	100	0.9695	0.9675
3.	250	150	100	0.9677	0.9679
4.	250	200	100	0.9682	0.9683
5.	250	400	100	0.9689	0.9688
6.	250	600	100	0.9692	0.9695
7.	250	800	100	0.9694	0.9696
8.	250	1000	100	0.9694	0.9696
9.	250	1500	100	0.9696	0.9697
10.	250	2000	100	0.9697	0.9697
11.	250	10000	100	0.9700	0.9701
12.	250	20000	100	0.9697	0.9698

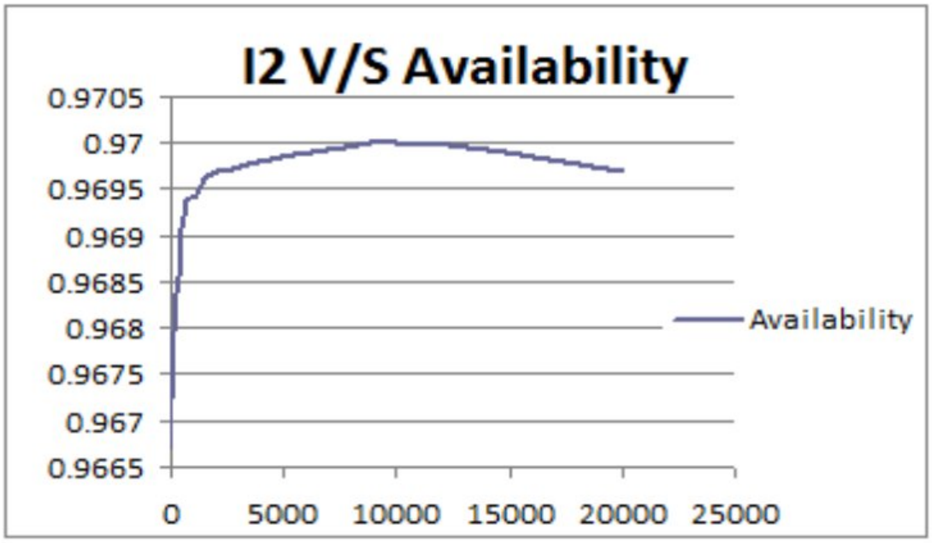


Figure 20 Inspection Interval for I2.

Next, we change the periodic inspection time at I3 keeping those at I1 and I2 constant. We observe that, as we increase the periodic inspection time the availability of the component merely increases. This is so, because the component has degraded to a higher level and need frequent inspection.

Table 5 Sensitivity analysis for system availability varying inspection interval for degradation stage 3.

S.No.	I1(hrs)	I2(hrs)	I3(hrs)	Availability	
				Markov	MCS
1.	250	150	25	0.9676	0.9675
2.	250	150	50	0.9676	0.9675
3.	250	150	100	0.9677	0.9676
4.	250	150	150	0.9677	0.9676
5.	250	150	200	0.9677	0.9676
6.	250	150	300	0.9678	0.9677
7.	250	150	400	0.9678	0.9677
8.	250	150	500	0.9678	0.9678

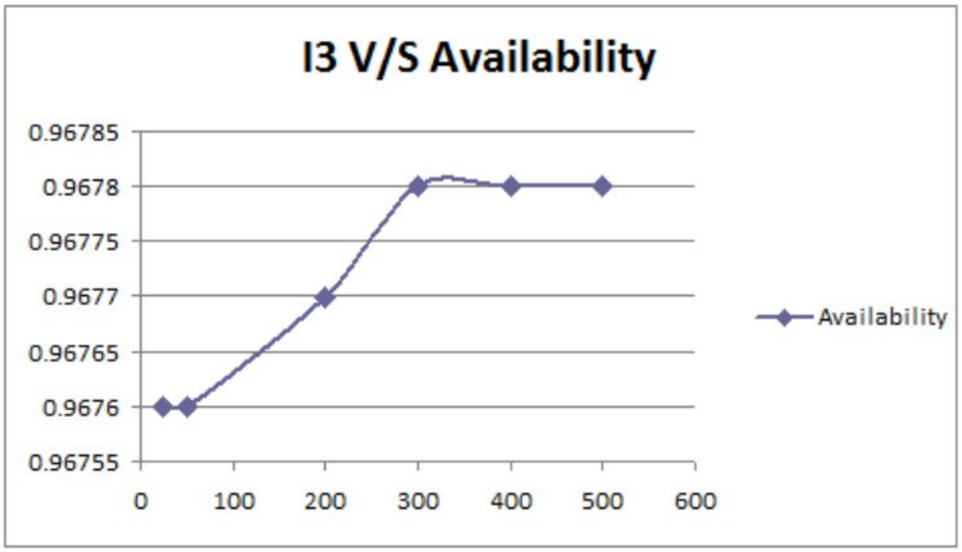


Figure 21 Inspection Interval for I3.

5.6 Comparison of Result for OM Model

In this section, we compare the various results obtained by the simulation of the program codes in MATLAB for all the three types of maintenance policies. The source of data is baringer.com. The results include the availability of the multi state system, found out by using both Markov analysis and the Monte Carlo Simulation.

The following pages represent the tabulated result with their corresponding graphical presentation, with all the three methods compared graphically in the end.

Table 6 Results for availability

	MISSION TIME					
	5000		10000		15000	
	Markov Analysis	MCS	Markov Analysis	MCS	Markov Analysis	MCS
Perfect without OM	0.9986	0.9985	0.9972	0.9972	0.9965	0.9968
Perfect with OM	0.9989	0.9989	0.9975	0.9974	0.9972	0.9971
Imperfect without OM	0.9980	0.9981	0.9955	0.9951	0.9915	0.9912
Imperfect with OM	0.9989	0.9987	0.9965	0.9969	0.9932	0.9929
Minimal without OM	0.9975	0.9976	0.9955	0.9756	0.9244	0.9245
Minimal with OM	0.9977	0.9979	0.9767	0.9761	0.9955	0.9254

Figure 22 Availability for Perfect Repair without OM

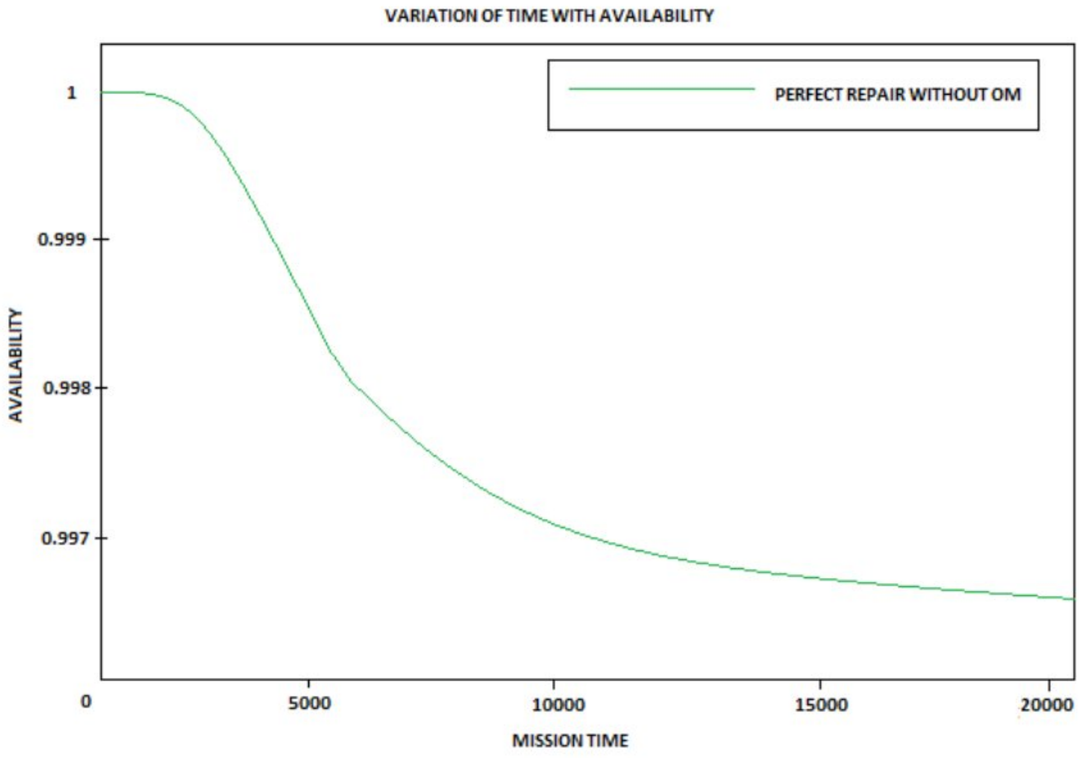


Figure 23 Availability for Perfect Repair with OM

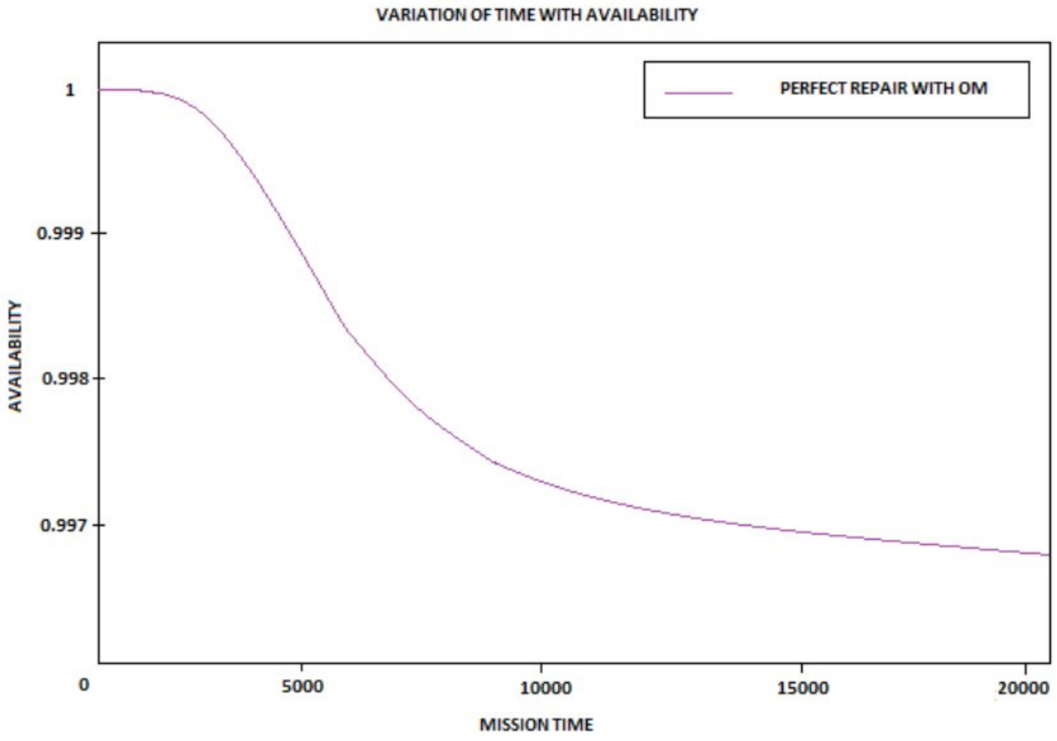


Figure 24 Availability for Imperfect Repair without OM

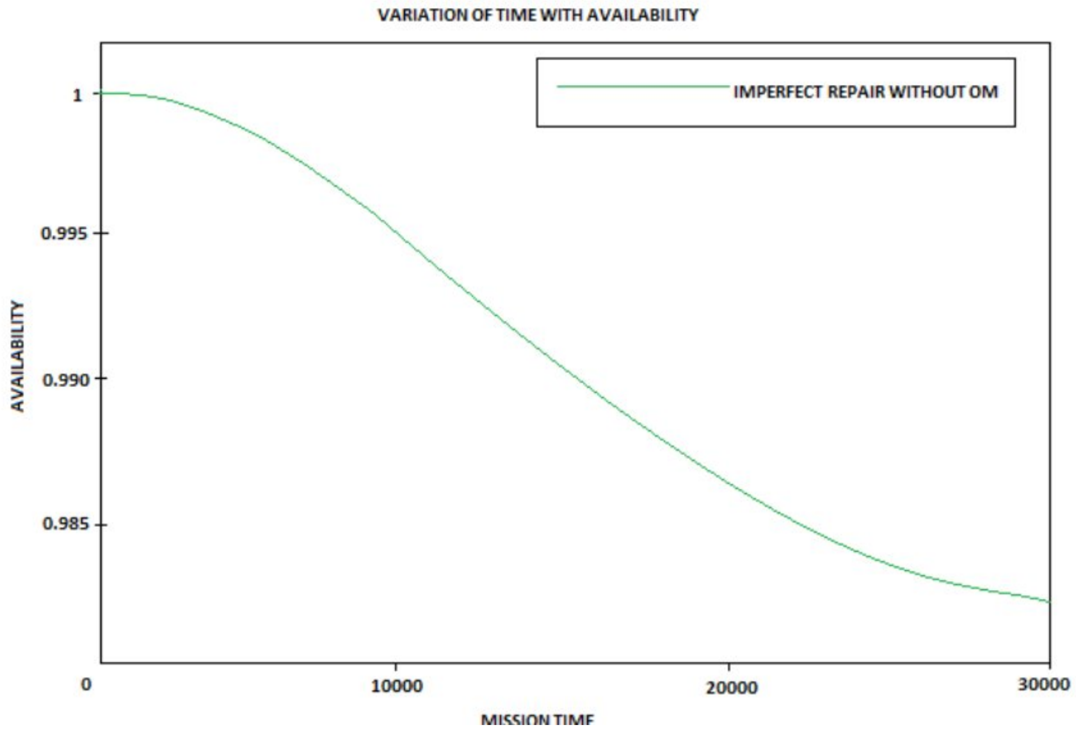


Figure 25 Availability for Imperfect Repair with OM

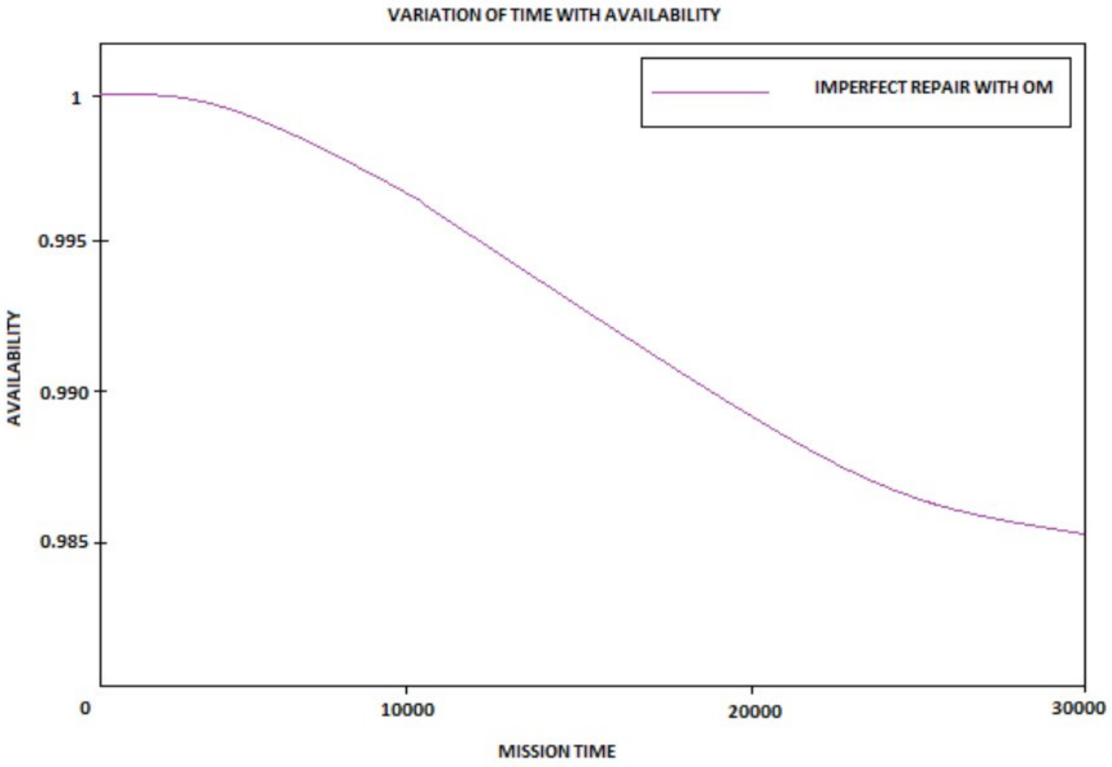


Figure 26 Availability for Minimal Repair without OM

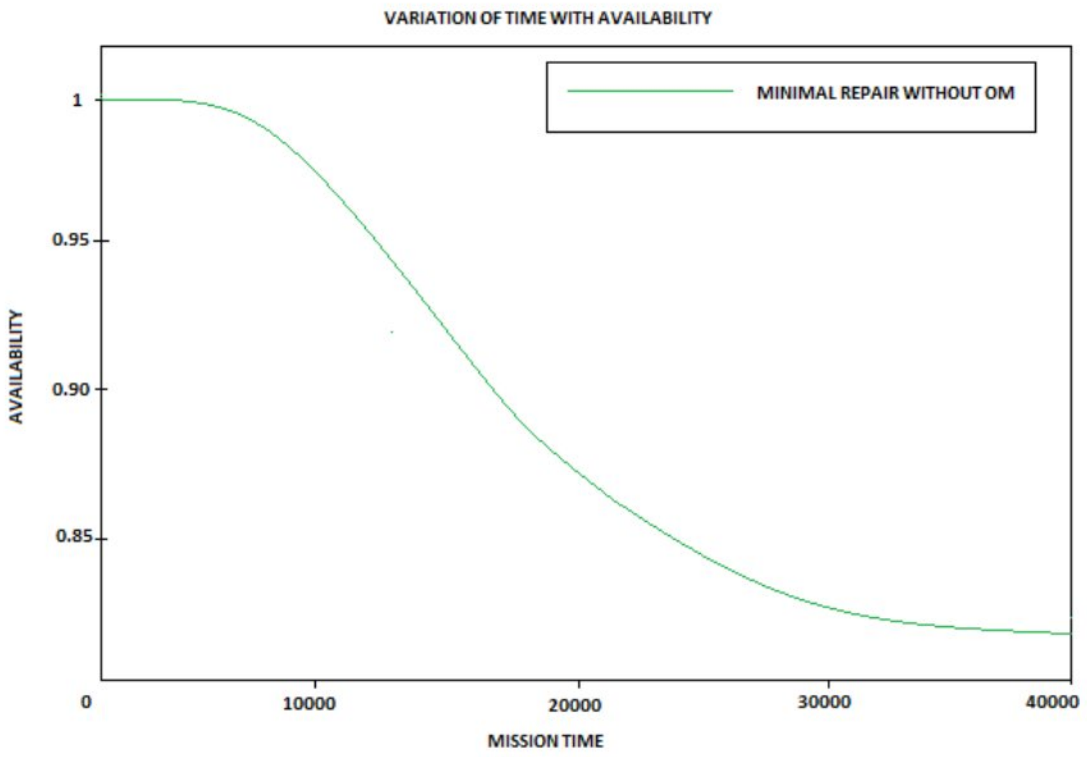


Figure 27 Availability for Minimal Repair with OM

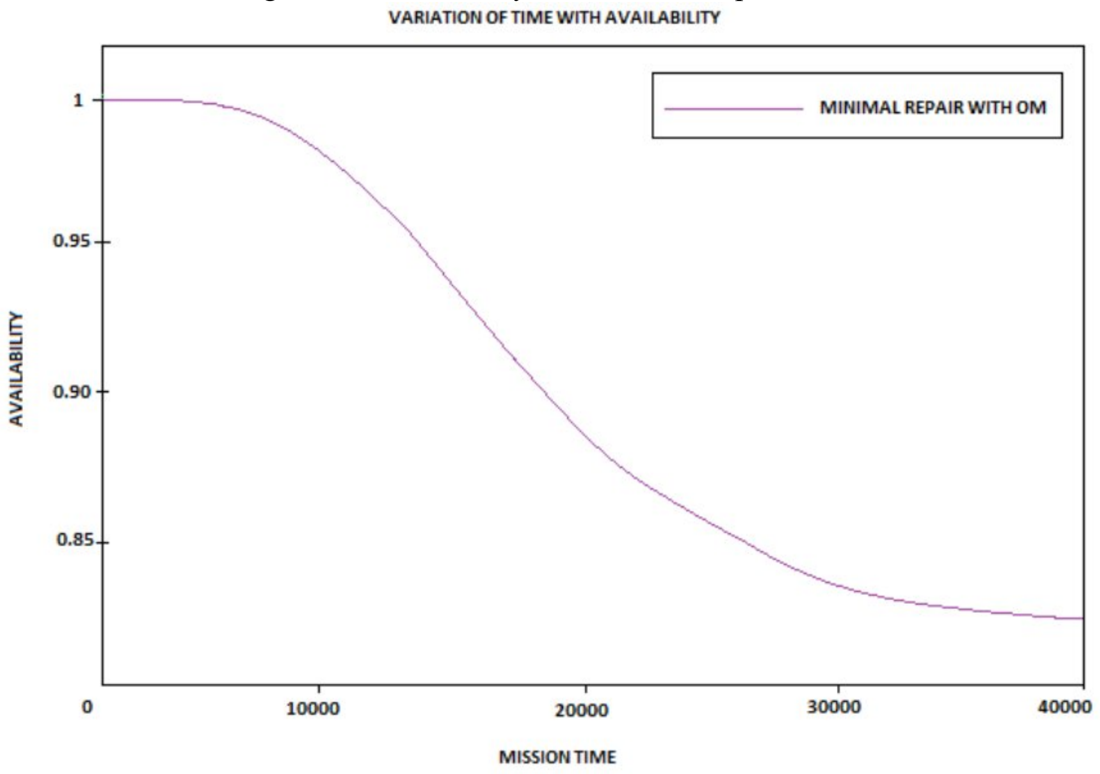


Figure 28 Comparison of Availability Values for Perfect Repair with and without OM

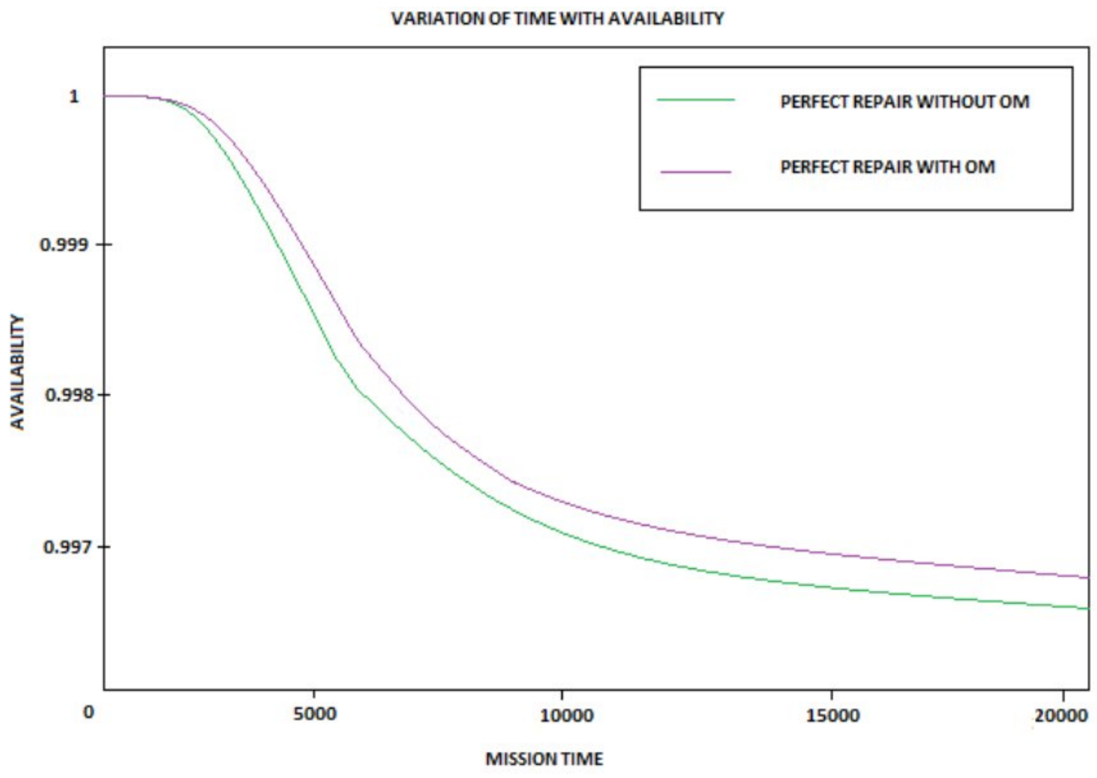


Figure 29 Comparison of Availability Values for Imperfect Repair with and without OM

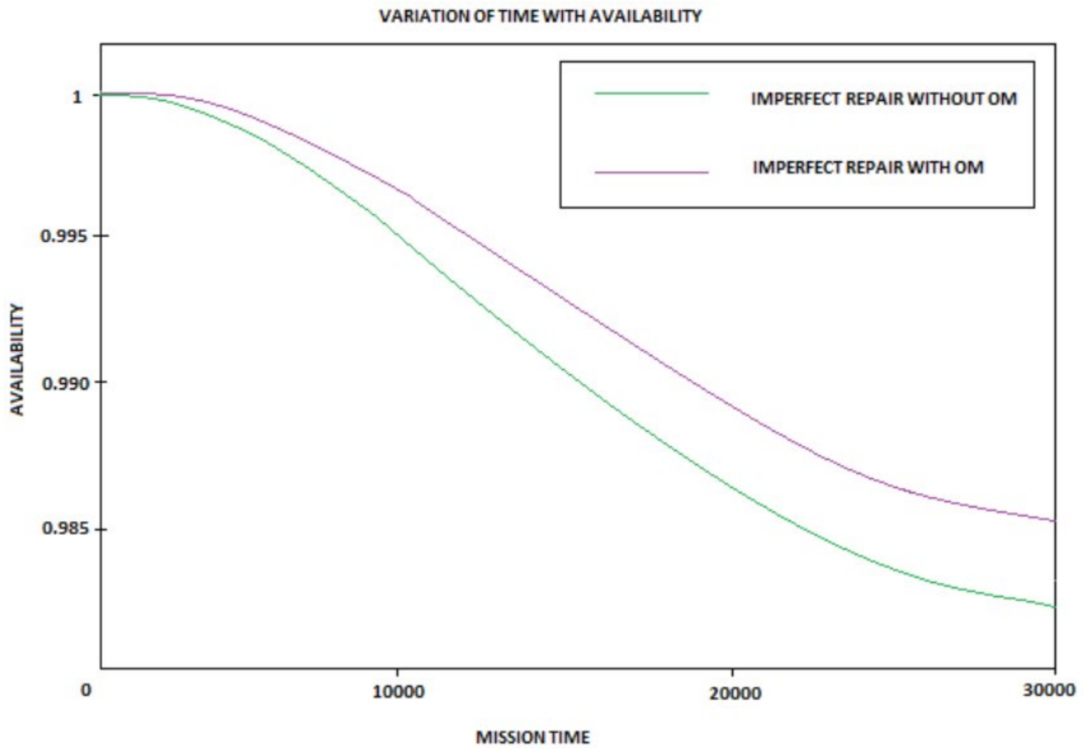
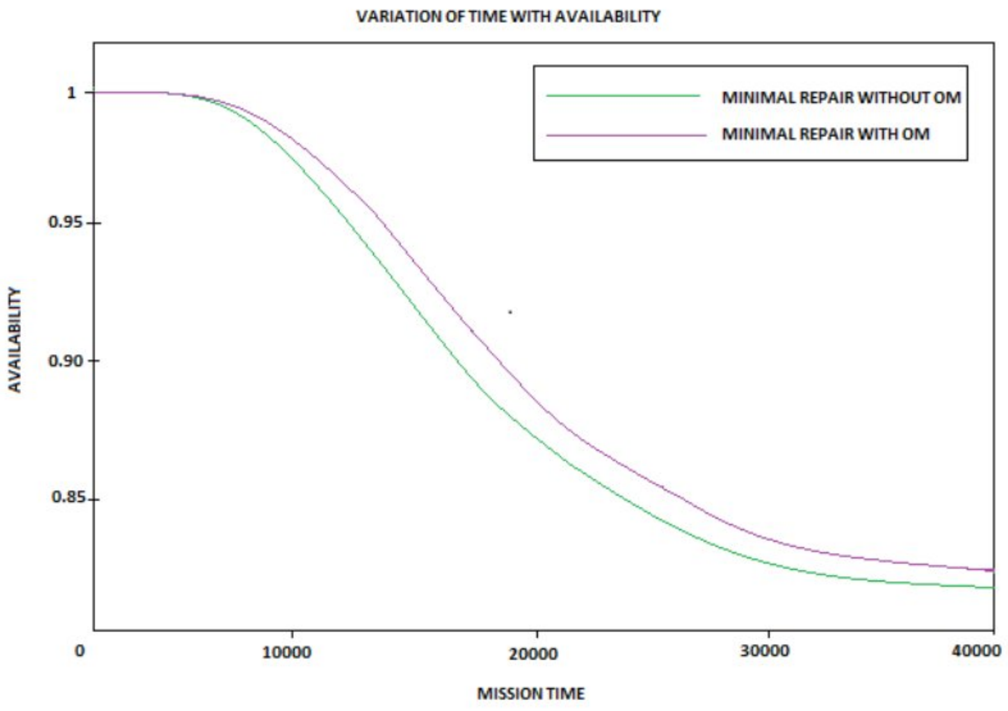


Figure 30 Comparison of Availability Values for Minimal Repair with and without OM



5.6.1 RESULT DISCUSSION

As the results obtained provide a definite indication of the trend in the availability for different maintenance policies, these numeric results can be analyzed quantitatively to compare the relative improvement in the performance of the system in the different scenarios. For the same mission time moving from perfect repair to minimal repair, the availability shows the decreasing trend. The percentage of decrease in the availability from perfect to imperfect repair is less but the percentage of decrease in availability from imperfect to minimal is very high. This clearly establishes that the minimal repair policy is extremely inefficient and should be seldom used unless cost of maintenance is the only dictating factor.

The availability of the system increases when opportunistic maintenance is done. When the system is new, the effect of opportunistic maintenance on availability is very less and the effect increases with mission time till the steady state is reached . The system shows the slight availability increase in all the types of repair work. When opportunistic maintenance is considered, the system shows the same decreasing trend as it was showing in the system with corrective maintenance only.

Thus analysis of availability of repairable mechanical systems under different scenarios is a vital tool in creating a system/policy for a definite application to maximize its performance and the availability increases with the opportunistic maintenance.

CHAPTER-6 CONCLUSION AND SCOPE FOR FUTURE WORK

The final chapter of this project contains the conclusion of the project and the scope for improvement in this project.

6.1 CONCLUSION

System availability model considering multi stage degradation, periodic inspection, condition based maintenance and random failure is developed. The system model is solved analytically by MARKOV approach and verified by Monte-Carlo simulation. And the results by both the methods are almost same.

A sensitivity analysis is conducted to see the effect of variation in probability for various maintenance decision, variation of inspection interval and final degraded states with and without failure.

- As far as frequency of inspection is concerned at stage D1, less frequent inspection should be done as the health of the component is very good and unnecessary inspection will only lead to time wastage and reducing our component availability .
- At stage D2, the inspection should be done too frequently either as here too the health of the component is fairly good.
- At stage D3, inspection work should be done quite frequently as the health of the component has deteriorated and frequent inspection would readily provide us information about its degradation so we can undertake necessary repair actions.

The system models with opportunistic maintenance are developed for motor-pump system. Each based on different maintenance policy.

- There is gain in availability when opportunistic maintenance is done with corrective maintenance.
- The maximum gain in availability hours is observed in perfect repair when opportunistic maintenance is done with corrective maintenance.

- The minimum gain in availability hours is observed in minimal repair as expected.

6.2 FUTURE SCOPE OF WORK

In this project, the failure and the repair behavior are modeled with exponential distribution so that Markov approach can be applied. But in real life the exponential distribution is not appropriate for mechanical systems. So, there is a need to develop a model with non-exponential distributions such as weibull for failure and log normal for repair which are realistic for a mechanical system.

The costs in the repair are not considered in this project and there is a scope to study availability gains and the amount of resource spent on the repairs.

The other maintenance such as preventive and reliability centered maintenance cab be analyzed for availability by considering multi-state degradation.

In many industries, still, not much attention is paid to the above considered factors, this analysis shows how the individual parameters can contribute significantly to the enhanced availability. Hence, it can be an initiation in this regard for many firms to analyze the parameters discussed here and improve the availability of the component(s) and thereby, that of the overall system.

REFERENCES

1. Martin KF,1994,” A review by discussion of condition monitoring and fault diagnosis in machine tools”, International Journal of Machine Tools Manufacturing, 34(4):527-551.

2. Savic, G. Walters, J. Knezevic, 1995, "Optimal, opportunistic maintenance policy using genetic algorithms, 2: analysis". Journal of Quality in Maintenance Engineering, Vol. 1, No. 3, 25-34.

3. Fricks RM., Trivedi, KS.,1997, “Modelling failure dependencies in reliability analysis using stochastic Petri nets, in ESM”, Proceedings 11th European Simulation Multiconf., Istanbul, Turkey, SCS Europe, 1-22.

4. Chen A, Guo RS, Yang A, Tseng CL,1998, “An integrated approach to semiconductor equipment monitoring”, Journal of Chinese Society of Mechanical Engineering, 19(6):581-591.

5. O. Mohamed-Salah, A-K. Daoud, G. Ali, "A simulation model for opportunistic maintenance strategies". 7th IEEE International Conference on Energy Technologies and Factory Automation, Vol. 1, 1999, Barcelona, Spain, 703 – 708.

6. J. Crocker, U. Kumar, 2000, "Age related maintenance versus reliability centered maintenance: a case study on aero-engines". Journal of Reliability Engineering and Systems Safety, Vol. 67, No. 2, 113-118.

7. Grall A, Berenguer C, Dieulle L.,2002, “A condition-based maintenance policy for stochastically deteriorating systems” Reliability Engineering and System Safety ,76:167-180.

8. Amari SV.,2004, “Optimal design of a condition- based maintenance model. Annual Reliability and Maintainability Symposium”, 528-533.

9. Karin S. de Smidt-Destombes, Matthieu C. van der Heijden, Aart van Harten, 2004, “On the availability of a k-out-of-N system given limited spares and repair capacity under a condition based maintenance strategy”. Journal of Reliability Engineering and System Safety 83, 287–300.

10. Castanier, Grall, Be´renguer, 2005, “A condition-based maintenance policy with non-periodic inspections for a two-unit series system”, Journal of Reliability Engineering and System Safety 87, 109–120.

11. Chen D, Trivedi KS.,2005,"Optimization for condition-based maintenance with Semi Markov decision process" Reliability Engineering and System Safety, 90(1): 25-29.

12. Xie W, Hong Y, Trivedi K.,2005, "Analysis of a two-level software rejuvenation policy", Reliability Engineering and System Safety, 87(1):13-22.

13. Andrew K.S. Jardine, Daming Lin, Dragan Banjevic, 2006, "A review on machinery diagnostics and prognostics implementing condition-based maintenance", Journal of Mechanical systems and signal processing 20, 1483–1510.

14. Endrenyi J, Anders GJ.,2006, "Aging, maintenance, and reliability", IEEE Power and Energy Magazine, 59-67.

15. Jardine AKS, Lin D, Banjevic D,2006,"A review on machinery diagnostic and prognostics implementing condition-based maintenance", Mechanical Systems and Signal Processing, 20:1483-1510.

16. X. Zhaou, L. Xi, J. Lee, 2006, "A dynamic opportunistic maintenance policy for continuously monitored systems". Trans. J. of Quality Management Engineering, Vol. 12, No. 3, 294 – 305.

17. L. Cui, H. Li, 2006, "Opportunistic maintenance for multi-component shock models". Journal of Mathematical Methods of Operations Research, Vol. 63, No. 3, 180 – 191.

18. Chen A, Wu GS, 2007," Real-time health prognosis and dynamic preventive maintenance policy for equipment under aging Markovian deterioration", International Journal of Production Research ,45(15):3351-3379.

19. Romulo I. Zequeiraa, Jose E. Valdesb, Christophe Berenguer, 2008, "Optimal buffer inventory and opportunistic preventive maintenance under random production capacity availability", Int. J. Production Economics 111, 686–696.

20. M. S. Samhouri , A. Al-Ghandoor, R. H. Fouad, S. M. Alhaj Ali, 2009, "An Intelligent Opportunistic Maintenance (OM) System: A Genetic Algorithm Approach", Jordan Journal of Mechanical and Industrial Engineering, Volume 3, Number 4,Pages 246 – 251.

21. Xiaojun Zhou, LifengXi, JayLee, 2009, "Opportunistic preventive maintenance scheduling for a multi-unit series system based on dynamic programming", Int. J. Production Economics 33, 361–366.

22. Radouane Laggoune, Alaa Chateauneuf, Djamil Aissani, 2009," Opportunistic policy for optimal preventive maintenance of a multi-component system in continuous operating units", Journal of Computers and Chemical Engineering 111, 1499–1510.

23. Ling Wang, Jian Chu, Weijie Mao, 2009, "A condition-based replacement and spare provisioning policy for deteriorating systems with uncertain deterioration to failure", European Journal of Operational Research 194, 184–205.

24. Pandian A, Ali A.,2009,"A review of recent trends in machine diagnosis and prognosis algorithm", World Congress on Nature & Biologically Inspired Computing, 1731–1736.

25. Gorjian N, Ma L, Mittinty M, Yarlagadda P, Sun Y.,2009,"A review on degradation models in reliability analysis", Proceedings of the 4th World Congress on Engineering Asset Management, Athens, Greece, 1-16.

26. Yongjin(James)Kwon, RichardChiou, LeonardStepanskı, 2009, "Remote, condition-based maintenance for web-enabled robotic system", Journal of Robotics and Computer-Integrated Manufacturing 25 , 552– 559.

27. Radouane Laggoune, AlaaChateauneuf, DjamilAissani, 2010, "Impact of few failure data on the opportunistic replacement policy for multi-component systems", Journal of Reliability Engineering and System Safety 95, 108–119.

28. Qiang H, Zhihua D, Xiao Z.,2010, "Application of HSMM on NC machine's state recognition", International conference on E-health Networking, Digital Ecosystems and Technologies, 189-191.

29. Taghipour S, Banjevic D, Jardine AKS,2010, "Periodic inspection optimization model for a complex repairable system", Reliability Engineering and System Safety, 95(9):944 – 952.

30. Wang N, Sun S, Li S, Si S.,2010,"Modelling and optimization of deteriorating equipment with predictive maintenance and inspection", IEEE 17th International conference on Industrial Engineering and Engineering Management , 942-946.

31. Fangji Wu, TianyiWang, JayLee, 2010, "An online adaptive condition-based maintenance method for mechanical systems".Journal of Mechanical Systems and Signal Processing 24 , 2985–2995.

32. Marquez AC, Heguedas AS,2010, "Models for maintenance optimization: A study for repairable systems and finite time periods", Reliability Engineering and System Safety ,75:367-377.

33. El-Damcese MA Temraz NS,2011, “Availability and reliability measures for multi-state system by using Markov reward model”.,Reliability:Theory and Application , 2:68-85.

34. Zhigang Tian, Tongdan Jin, Bairong Wu, Fangfang Ding, 2011,” Condition based maintenance optimization for wind power generation systems under continuous monitoring”.Journal of Renewable Energy 36, 1502-1509.

35. Zhigang Tian, Haitao Liao, 2011, “Condition based maintenance optimization for multi-component systems using proportional hazards model”. Journal of Reliability Engineering and System Safety 96, 581–589.

36. Majid MAA, Nasir M.,2011, “Multi-state system availability model of electricity generation for a cogeneration district cooling plant”, Asian Journal of Applied Sciences, 4(4):431-438.

37. Chryssaphinou O, Liminios N, Malefaki S.,2011, “Multi-state reliability systems under discrete time Semi-Markovian hypothesis”, IEEE Trans. Reliability, 60(1):80-87.

38. Sharareh Taghipour, Dragan Banjevic, 2012, “Optimal inspection of a complex system subject to periodic and opportunistic inspections and preventive replacements”, European Journal of Operational Research 220, 649–660.

39. Fangfang Ding, Zhigang Tian, 2012, “Opportunistic maintenance for wind farms considering multi-level imperfect maintenance thresholds”. Journal of Renewable Energy 45, 175-182.

40. Rosmaini Ahmad, Shahrul Kamaruddin, 2012, “An overview of time-based and condition-based maintenance in industrial application”. Journal of Computers & Industrial Engineering 63 , 135–149.

41. Cui Yanbin, Cui Bo, 2012, “The Condition Based Maintenance Evaluation Model on On-post Vacuum Circuit Breaker”, Journal of Systems Engineering Procedia 4 , 182 – 188.

42. Qingfeng Wang, Jinji Gao, 2012, “Research and application of risk and condition based maintenance task optimization technology in an oil transfer station”. Journal of Loss Prevention in the Process Industries 25, 1018-1027.

43. Chiming Guoa, Wenbin Wang, Bo Guoa, Xiaosheng Si, 2012, “A Maintenance Optimization Model for Mission-Oriented Systems Based on Wiener Degradation”. Journal of Reliability Engineering and System Safety 90, 1856-1874.

APPENDIX

Program for Markov Analysis of CBM model

```
function dydt = CBM
(t,y,ld1d2,ld1i1,ld1fr,ld2d3,ld2fr,ld2i2,ld3i3,ld3fr,li1m1,li3m3,li2m2,li2mm2,li3mm3,li3im3,mui1d1,mum
1d1,mumm2d1,mumm3d1,mufrd1,mui2d2,mum2d2,muim3d2,mum3d3,mui3d3)
ld1d2=0.00025;ld1i1=0.004;ld1fr=0.00002;ld2fr=0.00002;
ld2i2=0.006667;ld2d3=0.00067;
ld3i3=0.01;ld3fr=0.00002;li1m1=0.5;li3m3=0.125;
li2m2=0.25;li2mm2=.25;li3mm3=.125;li3im3=.125;
mui1d1=0.005,mum1d1=0.05;
mumm2d1=0.0125;mumm3d1=0.0625;mufrd1=0.02;
mui2d2=0.025;mum2d2=0.025;muim3d2=0.01;
mum3d3=0.016;mui3d3=0.0125;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(ld1d2+ld1i1+ld1fr)*y(1)+mum1d1*y(8)+mumm2d1*y(10)+mumm3d1*y(13)+mufrd1*y(7));
    (-(ld2d3+ld2fr+ld2i2)*y(2)+ld1d2*y(1)+mui2d2*y(5)+mum2d2*y(9)+muim3d2*y(12));
    (-(ld3i3+ld3fr)*y(3)+ld2d3*y(2)+mum3d3*y(11)+mui3d3*y(6));
    (ld1i1*y(1)-li1m1*y(4)-mui1d1*y(4));
    (ld2i2*y(2)-(li2m2+li2mm2)*y(5)-mui2d2*y(5));
    (-(li3m3+li3mm3+li3im3)*y(6)+ld3i3*y(3)-mui3d3*y(6));
    (ld1fr*y(1)+ld2fr*y(2)+ld3fr*y(3)-mufrd1*y(7));
    (li1m1*y(4)-mum1d1*y(8));
    (li2m2*y(5)-mum2d2*y(9));
    (li2mm2*y(5)-mumm2d1*y(10));
    (li3m3*y(6)-mum3d3*y(11));
    (li3im3*y(6)-muim3d2*y(12));
    (li3mm3*y(6)-mumm3d1*y(13));];

End
```

Program for Markov Analysis of Perfect Repair System (without OM)

```
function dydt =
withoutopp1(t,y,lemda12,lemda15,lemda26,lemda37,lemda56,lemda910,lemda23,lemda59,lemda67,le
mda610,lemda711,lemda1011,lemda34,lemda78,lemda913,lemda1014,lemda1112,lemda1115,mu41,m
u153,mu142,mu131,mu129,mu85)
lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
mu129=0.0016667;mu85=0.0016667;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(lemda15+lemda12)*y(1)+mu41*y(4)+mu131*y(13));
    (lemda12)*y(1)-(lemda23+lemda26)*y(2)+mu142*y(14);
    (lemda23)*y(2)-(lemda34+lemda37)*y(3)+mu153*y(15);
    lemda34*y(3)-mu41*y(4);
    mu85*y(8)-(lemda56+lemda59)*y(5)+lemda15*y(1);
    lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6);
    lemda37*y(3)+lemda67*y(6)-(lemda711+lemda78)*y(7);
    lemda78*y(7)-mu85*y(8);
    lemda59*y(5)+mu129*y(12)-(lemda910+lemda913)*y(9);
    lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10);
    lemda1011*y(10)+lemda711*y(7)-(lemda1115+lemda1112)*y(11);
    lemda1112*y(11)-mu129*y(12);
    lemda913*y(9)-mu131*y(13);
    lemda1014*y(10)-mu142*y(14);
    lemda1115*y(11)-mu153*y(15);];

end
```

Program for Markov Analysis of Perfect Repair System(with OM)

```
function dydt =withopp1
(t,y,lemda12,lemda15,lemda26,lemda37,lemda56,lemda910,lemda23,lemda59,lemda67,lemda610,lem
da711,lemda1011,lemda34,lemda78,lemda913,lemda1014,lemda1112,lemda1115,mu41,mu153,mu142
,mu131,mu129,mu85,mu1513,mu124,mu1413,mu84)
lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
mu129=0.0016667;mu85=0.0016667;
mu1413=0.001;mu84=0.01333;
mu1513=0.003334;mu124=0.005;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(lemda15+lemda12)*y(1)+mu41*y(4)+mu131*y(13));
    (lemda12)*y(1)-(lemda23+lemda26)*y(2)+mu142*y(14);
    (lemda23)*y(2)-(lemda34+lemda37)*y(3)+mu153*y(15);
    lemda34*y(3)-mu41*y(4)+mu124*y(12)+mu84*y(8);
    mu85*y(8)-(lemda56+lemda59)*y(5)+lemda15*y(1);
    lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6);
    lemda37*y(3)+lemda67*y(6)-(lemda711+lemda78)*y(7);
    lemda78*y(7)-mu85*y(8)-mu84*y(8);
    lemda59*y(5)+mu129*y(12)-(lemda910+lemda913)*y(9);
    lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10);
    lemda1011*y(10)+lemda711*y(7)-(lemda1115+lemda1112)*y(11);
    lemda1112*y(11)-mu129*y(12)-mu124*y(12);
    lemda913*y(9)-mu131*y(13)+mu1413*y(14)+mu1513*y(15);
    lemda1014*y(10)-mu142*y(14)-mu1413*y(14);
    lemda1115*y(11)-mu153*y(15)-mu1513*y(15);];

end
```

Program for Markov Analysis of Imperfect Repair System(without OM)

```
function dydt =withoutopp3
(t,y,lemda12,lemda15,lemda26,lemda37,lemda56,lemda910,lemda23,lemda59,lemda67,lemda610,lem
da711,mu1210,mu157,lemda1011,lemda34,lemda78,lemda913,lemda1014,lemda1112,lemda1115,mu4
1,mu42,mu153,mu86,mu146,mu135,mu142,mu131,mu129,mu85,mu1513,mu124,mu1413,mu84)
lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
mu129=0.0016667;mu85=0.0016667;
mu1210=0.0025;
mu1413=0.001;mu84=0.1333;mu42=0.0025;mu135=0.003334;
mu86=0.0025;mu146=0.003334;mu157=0.003334;
mu1513=0.001;mu124=0.005;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(lemda15+lemda12)*y(1));
    (-(lemda23+lemda26)*y(2)+mu42*y(4)+lemda12*y(1));
    (-(lemda34+lemda37)*y(3))+lemda23*y(2);
    lemda34*y(3)-mu42*y(4);
    -(lemda56+lemda59)*y(5)+lemda15*y(1)+mu135*y(13);
    lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6)+mu86*y(8)+mu146*y(14);
    mu157*y(15)+lemda67*y(6)+lemda37*y(3)-(lemda711+lemda78)*y(7);
    lemda78*y(7)-mu86*y(8);
    lemda59*y(5)-(lemda910+lemda913)*y(9);
    lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10)+mu1210*y(12);
    lemda1011*y(10)+lemda711*y(7)-(lemda1115+lemda1112)*y(11);
    lemda1112*y(11)-mu1210*y(12);
    lemda913*y(9)-mu135*y(13);
    lemda1014*y(10)-mu146*y(14);
    lemda1115*y(11)-mu157*y(15);];
end
```

Program for Markov Analysis of Imperfect Repair System(with OM)

```
    function dydt =withopp3
    (t,y,lemda12,lemda15,lemda26,lemda37,lemda56,lemda910,lemda23,lemda59,lemda67,lemda610,lem
    da711,mu1210,mu157,lemda1011,lemda34,lemda78,lemda913,lemda1014,lemda1112,lemda1115,mu4
    1,mu42,mu153,mu86,mu146,mu135,mu142,mu131,mu129,mu85,mu1513,mu124,mu1413,mu84)
    lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
    lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
    lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
    lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
    lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
    lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
    mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
    mu129=0.0016667;mu85=0.0016667;mu1210=0.0025;
    mu1413=0.001;mu84=0.01333;mu42=0.0025;mu135=0.003334;
    mu86=0.0025;mu146=0.003334;mu157=0.003334;
    mu1513=0.003334;mu124=0.005;
    y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
    dydt = [(-(lemda15+lemda12)*y(1));
        (-(lemda23+lemda26)*y(2)+mu42*y(4)+lemda12*y(1));
        (-(lemda34+lemda37)*y(3))+lemda23*y(2);
        lemda34*y(3)-mu42*y(4)+mu124*y(12)+mu84*y(4);
        -(lemda56+lemda59)*y(5)+lemda15*y(1)+mu135*y(13);
        lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6)+mu86*y(8)+mu146*y(14);
        mu157*y(15)+lemda67*y(6)+lemda37*y(3)-(lemda711+lemda78)*y(7);
        lemda78*y(7)-mu86*y(8)-mu84*y(4);
        lemda59*y(5)-(lemda910+lemda913)*y(9);
        lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10)+mu1210*y(12);
        lemda1011*y(10)+lemda711*y(7)-(lemda1115+lemda1112)*y(11);
        lemda1112*y(11)-mu1210*y(12)-mu124*y(12);
        lemda913*y(9)-mu135*y(13)+mu1413*y(14)+mu1513*y(15);
        lemda1014*y(10)-mu146*y(14)-mu1413*y(14);
        lemda1115*y(11)-mu157*y(15)-mu1513*y(15);];

end
```

Program for Markov Analysis of Minimal Repair System (without OM)

```
function dydt =withoutopp2
(t,y,lemda12,lemda15,mu43,mu87,mu1511,mu1410,mu1211,lemda26,lemda37,lemda56,lemda910,le
mda23,lemda59,lemda67,lemda610,lemda711,lemda1011,lemda34,lemda78,lemda913,lemda1014,lem
da1112,lemda1115,mu41,mu153,mu142,mu131,mu139,mu85)
lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
mu1511=0.001;mu1410=0.001;mu1211=0.001;
mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
mu129=0.0016667;mu85=0.0016667;
mu139=0.001;mu43=0.001;mu87=0.001;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(lemda15+lemda12)*y(1));
    (-(lemda23+lemda26)*y(2));
    (-(lemda34+lemda37)*y(3))+mu43*y(4)+lemda23*y(2);
    lemda34*y(3)-mu43*y(4);
    -(lemda56+lemda59)*y(5)+lemda15*y(1);
    lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6);
    mu87*y(8)+lemda67*y(6)-(lemda711+lemda78)*y(7)+lemda37*y(3);
    lemda78*y(7)-mu87*y(8);
    lemda59*y(5)+mu139*y(13)-(lemda910+lemda913)*y(9);
    lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10)+mu1410*y(14);
    lemda1011*y(10)+lemda711*y(7)-
(lemda1115+lemda1112)*y(11)+mu1511*y(15)+mu1211*y(12);
    lemda1112*y(11)-mu1211*y(12);
    lemda913*y(9)-mu139*y(13);
    lemda1014*y(10)-mu1410*y(14);
    lemda1115*y(11)-mu1511*y(15);];
end
```

Program for Markov Analysis of Minimal Repair System(with OM)

```
function dydt =withopp2(t,y,lemda12,lemda15,mu43,mu87,mu1511,mu1410,mu1514,mu1413,mu128,
mu1211,lemda26,lemda37,lemda56,lemda910,lemda23,lemda59,lemda67,lemda610,lemda711,lemda1
011,lemda34,lemda78,lemda913,lemda1014,lemda1112,lemda1115,mu41,mu153,mu142,mu131,mu13
9,mu85)
lemda12=0.00004107;lemda15=0.00001;lemda26=0.00001;
lemda37=0.00001;lemda56=0.00004107;lemda910=0.00004107;
lemda23=0.00004107;lemda59=0.00001;lemda67=0.00004107;
lemda610=0.00001;lemda711=0.00001;lemda1011=0.00004107;
lemda34=0.00004107;lemda78=0.00004107;lemda913=0.00001;
lemda1014=0.00001;lemda1112=0.00004107;lemda1115=0.00001;
mu1511=0.001;mu1410=0.001;mu1211=0.001;
mu41=0.0016667;mu153=0.002;mu142=0.002;mu131=0.002;
mu129=0.0016667;mu85=0.0016667;mu139=0.001;
mu43=0.001;mu87=0.001;
mu1413=0.001;mu84=0.01333;
mu1514=0.005;mu128=0.006667;
y0=[1;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
dydt = [(-(lemda15+lemda12)*y(1));
    (-(lemda23+lemda26)*y(2));
    (-(lemda34+lemda37)*y(3))+mu43*y(4)+lemda23*y(2);
    lemda34*y(3)-mu43*y(4)+mu84*y(8);
    -(lemda56+lemda59)*y(5)+lemda15*y(1);
    lemda56*y(5)+lemda26*y(2)-(lemda67+lemda610)*y(6);
    mu87*y(8)+lemda67*y(6)-(lemda711+lemda78)*y(7)+lemda37*y(3);
    lemda78*y(7)-mu87*y(8)+mu128*y(12)-mu84*y(8);
    lemda59*y(5)+mu139*y(13)-(lemda910+lemda913)*y(9);
    lemda610*y(6)+lemda910*y(9)-(lemda1011+lemda1014)*y(10)+mu1410*y(14);
    lemda1011*y(10)+lemda711*y(7)-
(lemda1115+lemda1112)*y(11)+mu1511*y(15)+mu1211*y(12);
    lemda1112*y(11)-mu1211*y(12)-mu128*y(12);
    lemda913*y(9)-mu139*y(13)+mu1413*y(14);
    lemda1014*y(10)-mu1410*y(14)+mu1514*y(15)-mu1413*y(14);
    lemda1115*y(11)-mu1511*y(15)-mu1514*y(15);];
end
```

Program for MCS of CBM

```
function t= monteCBM(missiontime, ld1d2, ld1fr, ld1i1, ld2d3, ld2fr, ld3fr, ld2i2, ld3i3, mufrd1, mui1d1,
mum3d3, muim3d2, mumm3d1, mumm2d1, mum2d2, mui2d2, mui3d3, td1d2, td1fr, td1i1, td2d3, td2i2,
td2fr, tdd, tdr, td1, td2, td3, ct, rt, trt, rfrt, trfrt, it, tit, state, i, a, p, avail)
%ld1d2=1/2500;     %failre rate from d1 to d2
%ld1fr=1/50000;
%ld1i1=1/240;       %d1 to inspection rate
%ld2d3=1/1500;
%ld2fr=1/50000;
%ld3fr=1/50000;
%ld2i2=1/168;
%ld3i3=1/96;
%mufrd1=1/50;       %repaire rate from random failure to d1
%mum1d1=1/2;
%mui1d1=1/2;
%mui2d2=1/4;
%mui3d3=1/8;
%mumm2d1=1/8;
%mumm3d1=1/16;
%muim3d2=1/10;
%mum3d3=1/6;
%mum2d2=1/4;
ld1d2=0.00025;ld1i1=0.004;ld1fr=0.00002;ld2fr=0.00002;
ld2i2=0.006667;ld2d3=0.00067;
ld3i3=0.01;ld3fr=0.00002;
mui1d1=0.005,mum1d1=0.05;mumm2d1=0.0125;mumm3d1=0.0625;mufrd1=0.02;
mui2d2=0.025;mum2d2=0.025;muim3d2=0.01;mum3d3=0.016;mui3d3=0.0125;
a=zeros(1,1000);
p=zeros(1,1000);
avail=0;
availability=0;
missiontime=input('enter the mission time');
for i=1:1:1000
   ct=0;
   rt=0;           %repair time
   trt=0;          %total repair time
   rfrt=0;         % random failure repair time
   trfrt=0;        %total random failure repair time
   it=0;           %inspection time
   tit=0;          %total inspection time
   state=1;
   while ct<missiontime
                      %so that user doesnt input less than 0
                      %value
      switch state
         case 1
            yd1d2=random('unif', 0,1);     % for randomly calling value between 0 and 1
            yd1fr=random('unif', 0,1);
            yd1i1=random('unif', 0,1);
            td1d2=-1/ld1d2*log(yd1d2);     % time take from d1 to d2
            td1fr=-1/ld1fr*log(yd1fr);
            td1i1=-1/ld1i1*log(yd1i1);
            tdd=min(td1d2,td1fr);
```

```
    tdr=min(td1d2,td1i1);
    td1=min(tdd,tdr);
    if td1==td1d2
        ct=ct+td1d2;
        state=2;
    elseif td1==td1fr
        ct=ct+td1fr;
        yfrd1=random('unif',0,1);
        rfrt=-1/mufrd1*log(yfrd1);
        trfrt=trfrt+rfrt;
        ct=ct+rfrt;
        state=1;
    else ct=ct+td1i1;
        yi1=random('unif',0,1);
        if(yi1>0)&(yi1<.1)
            ym1d1=random('unif',0,1);
            yi1d1=random('unif',0,1);
            rt=-1/mum1d1*log(ym1d1);    %maintenance to d1
            trt=rt+trt;
            it=-1/mui1d1*log(yi1d1);    % inspection to d1
            tit=tit+it;
            ct=ct+rt+it;
            state=1;
        else
            yi1d1=random('unif',0,1);
            it=-1/mui1d1*log(yi1d1);
            tit=tit+it;
            ct=ct+it;
            state=1;
        end
    end
case 2
    yd2d3=random('unif',0,1);
    yd2fr=random('unif',0,1);
    yd2i2=random('unif',0,1);
    td2d3=-1/ld2d3*log(yd2d3);
    td2fr=-1/ld2fr*log(yd2fr);
    td2i2=-1/ld2i2*log(yd2i2);
    tdd=min(td2d3,td2fr);
    tdr=min(td2d3,td2i2);
    td2=min(tdd,tdr);
    if td2==td2d3
        ct=ct+td2d3;
        state=3;
    elseif td2==td2fr
        ct=ct+td2fr;
        yfrd1=random('unif',0,1);
        rfrt=-1/mufrd1*log(yfrd1);
        trfrt=trfrt+rfrt;
        ct=ct+rfrt;
        state=1;
    else ct=ct+td2i2;
        yi2=random('unif',0,1);
        if(yi2>0.1)&(yi2<0.3)
            ym2d2=random('unif',0,1);
            yi2d2=random('unif',0,1);
            rt=-1/mum2d2*log(ym2d2);
            trt=trt+rt;
            it=-1/mui2d2*log(yi2d2);
            tit=tit+it;
            ct=ct+it+rt;
            state=2;
        elseif (yi2>0)&(yi2<.1)
            ymm2d1=random('unif',0,1);
            yi2d2=random('unif',0,1);
            rt=-1/mumm2d1*log(ymm2d1);
            trt=trt+rt;
            it=-1/mui2d2*log(yi2d2);
            tit=tit+it;
            ct=ct+it+rt;
            state=1;
        else
            yi2d2=random('unif',0,1);
            it=-1/mui2d2*log(yi2d2);
            tit=tit+it;
            ct=ct+it;
            state=2;
        end
    end
case 3
    yd3fr=random('unif',0,1);
    yd3i3=random('unif',0,1);
    td3fr=-1/ld3fr*log(yd3fr);
    td3i3=-1/ld3i3*log(yd3i3);
    td3=min(td3fr,td3i3);
    if td3==td3fr;
        ct=ct+td3fr;
        yfrd1=random('unif',0,1);
        rfrt=-1/mufrd1*log(yfrd1);
        trfrt=trfrt+rfrt;
        ct=ct+rfrt;
        state=1;
    else ct=ct+td3i3;
        yi3=random('unif',0,1);
        if (yi3>.2)&(yi3<.4)
            ym3d3=random('unif',0,1);
            yi3d3=random('unif',0,1);
            rt=-1/mum3d3*log(ym3d3);
            trt=trt+rt;
            it=-1/mui3d3*log(yi3d3);
            tit=tit+it;
            ct=ct+it+rt;
            state=3;
        elseif(yi3>.4)&(yi3<.8);
            yim3d2=random('unif',0,1);
            yi3d3=random('unif',0,1);
            rt=-1/muim3d2*log(yim3d2);
            trt=trt+rt;
            it=-1/mui3d3*log(yi3d3);
            tit=tit+it;
            ct=ct+it+rt;
            state=2;
```

```
                elseif(yi3>.8)&(yi3<1);
                    ymm3d1=random('unif',0,1);
                    yi3d3=random('unif',0,1);
                    rt=-1/mumm3d1*log(ymm3d1);
                    trt=trt+rt;
                    it=-1/mui3d3*log(yi3d3);
                    tit=tit+it;
                    ct=ct+it+rt;
                    state=1;
                else
                     yi3d3=random('unif',0,1);
                     it=-1/mui3d3*log(yi3d3);
                     tit=tit+it;
                     ct=ct+it;
                     state=3;
                end
            end
        end
    end
    a(1,i)=(ct-trt-trfrt)/ct;
    avail=avail+a(1,i);
    a(1,i)=avail/i;
    p(1,i)=i;
end
a(1,:)
plot(p(1,:),a(1,:))
end
```

Program for MCS of Perfect Repair without OM

```
function t= perfectwithoutOM(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41, mu1411, mu3431,
mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;
aa=0;


l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;


mu1411=0.002;
mu3431=0.002;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;

mu2414=0.001;   %k
mu4241=0.01333;
mu3414=0.003334;  %k
mu4341=0.005;

random11=0;
```

```
 random12=0;
 random13=0;
 random14=0;
 random23=0;
 random33=0;
 random22=0;
 random32=0;
 random21=0;
 random31=0;

 %get the mission time from user and compare everytime for the condition of
%time

   a=zeros(1,1000);            % matrix of 1*1000
   p=zeros(1,1000);            % matrix of 1*1000

   missiontime=input('enter the mission time');

 for i=1:1:1000


   state=11;
    ct=0;
   rt=0;
   avail=0;

   %availability=0;



 while ct<missiontime

 random11=random('unif', 0,1);
 random12=random('unif', 0,1);
 random13=random('unif', 0,1);
 random14=random('unif', 0,1);
 random23=random('unif', 0,1);
 random33=random('unif', 0,1);
 random22=random('unif', 0,1);
 random32=random('unif', 0,1);
 random21=random('unif', 0,1);
 random31=random('unif', 0,1);
 switch state


 case 11
    testStrings={'11 state'}
        if(random11 < 0.5)
        state = 12;

        else
        state = 21;
        end
```

```
case 12
   testStrings={'your system is in 12 state'}
   t12=-1/l12*log(random11)
   ct=ct+t12
   if(random12<.5)
   state = 13;
   else
   state = 22;
   end

case 13
   testStrings = {'your system is in 13 state'}
   %add time formulae
   t13=-1/l13*log(random12);
   ct=ct+t13
   if(random13<.5)
   state = 14;
   else
   state = 23;
   end

case 14

   testStrings = {'your system is in 14 state'}   % have to apply for rpair method
   t14=-1/l14*log(random13);
   ct=ct+t14
   rt=rt + (-1/mu1411*log(random13));
   fprintf('rt is %d',rt)
   state = 11;


              case 23
              testStrings = {'your system is in 23 state'}
              t23=-1/l23*log(random13);
              ct=ct+t23
              if(random23<.5)
              state = 24;
              else
              state = 33;
              end

              case 24
                 testStrings = {'your system is in 24 state'}
                 %add time formulae
                 t24=-1/l23*log(random23);
                 rt=rt + (-1/mu2421*log(random23));
                 fprintf('rt is %d',rt)
                 ct=ct+t24
                state = 21;

             case 33
                   testStrings = {'your system is in 33 state'}
                   t33=-1/l33*log(random23);
                   ct=ct+t33
                   if(random33<.51)
                      state = 34;
```

```
        else
            state = 43;
        end

case 34
            testStrings = {'your system is in 34 state'}              % have to apply for rpair method
            %add time formulae
            t34=-1/l34*log(random33);
            rt=rt + (-1/mu3431*log(random33));
            fprintf('rt is %d',rt)
            ct=ct+t34
            state = 31;

 case 43
            testStrings = {'your system is in 43 state'}
            t43=-1/l43*log(random33);
            rt=rt + (-1/mu4313*log(random33));
            fprintf('rt is %d',rt)
            ct=ct+t43
            state = 13;




  case 22
             testStrings = {'your system is in 22 state'}
             t22=-1/l22*log(random12);
             ct=ct+t22
             if(random22<.5)
                 state = 23;
             else
                 state = 32;
             end


   case 32
            testStrings = {'your system is in 32 state'}
            t32=-1/l32*log(random22);
            ct=ct+t32
            if(random32<.5)
                state = 33;
            else
                state = 42;
            end



    case 42
              testStrings = {'your system is in 42 state'}
              t42=-1/l42*log(random32);
              rt=rt+ (-1/mu4212*log(random32));
              fprintf('rt is %d',rt)
              ct=ct+t42
              state = 12;
```

```
                    case 21
                         testStrings = {'your system is in 21 state'}
                         t21=-1/l21*log(random11);
                         ct=ct+t21
                         if(random21<.5)
                             state = 22;
                         else
                             state = 31;
                         end


   case 31
   testStrings = {'your system is in 31 state'}
   t31=-1/l31*log(random21);
          ct=ct+t31
          if(random31<0.5)
              state = 32;
          else
              state = 41;
          end



          case 41
     testStrings = {'your system is in 41 state'}
     t41=-1/l41*log(random31);
     rt=rt+ (-1/mu4111*log(random31));
     fprintf('rt is %d',rt)
     ct=ct+t41
     state = 11;


 end
 end
 %x=x+1;



    a(1,i)=(ct-rt)/ct;
    aa=(ct-rt)/ct;

    fprintf('availability is %d',aa);
    p(1,i)=i;
 end
       M=mean(a)
       fprintf('mean value over 1000 cases for perfect repair  is%d',M);
       %a(1,x)=avail/x;
     %avail=avail+a(1,x)
     %a(1,i)
    plot(p(1,:),a(1,:))

end
```

Program for MCS of Perfect Repair with OM

```
function t= perfectwithopportunisticmaintenance(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41,
mu1411, mu3431, mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;
aa=0;


l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;

mu1411=0.002;
mu3431=0.002;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;

mu2414=0.001;   %k
mu1411
mu4241=0.01333;
mu3414=0.003334;  %k
mu4341=0.005;

random11=0;
 random12=0;
```

```
random13=0;
random14=0;
random23=0;
random33=0;
random22=0;
random32=0;
random21=0;
random31=0;

%get the mission time from user and compare everytime for the condition of
%time
   a=zeros(1,1000);           % matrix of 1*1000
   p=zeros(1,1000);           % matrix of 1*1000

   missiontime=input('enter the mission time');


for i=1:1:1000


   state=11;
    ct=0;
   rt=0;
   avail=0;

   %availability=0;

while ct<missiontime

random11=random('unif', 0,1);
random12=random('unif', 0,1);
random13=random('unif', 0,1);
random14=random('unif', 0,1);
random23=random('unif', 0,1);
random33=random('unif', 0,1);
random22=random('unif', 0,1);
random32=random('unif', 0,1);
random21=random('unif', 0,1);
random31=random('unif', 0,1);
switch state


case 11
    testStrings={'11 state'}
        if(random11 < 0.5)
        state = 12;

        else
        state = 21;
        end


case 12
    testStrings={'your system is in 12 state'}
    t12=-1/l12*log(random11)
    ct=ct+t12
    if(random12<.5)
    state = 13;
    else
    state = 22;
    end

case 13
    testStrings = {'your system is in 13 state'}
    %add time formulae
    t13=-1/l13*log(random12);
    ct=ct+t13
    if(random13<.5)
    state = 14;
    else
    state = 23;
    end

case 14

    testStrings = {'your system is in 14 state'}   % have to apply for rpair method
    t14=-1/l14*log(random13);
    ct=ct+t14
    rt=rt + (-1/mu1411*log(random13));
    fprintf('rt is %d',rt)
    state = 11;


                case 23
                testStrings = {'your system is in 23 state'}
                t23=-1/l23*log(random13);
                ct=ct+t23
                if(random23<.5)
                state = 24;
                else
                state = 33;
                end

                case 24
                    testStrings = {'your system is in 24 state'}
                    %add time formulae
                    t24=-1/l23*log(random23);

                    rt2414=-1/mu2414*log(random23);
                    rt2421=-1/mu2421*log(random23);

                    if(rt2414<rt2421)
                    %rt=rt + rt2414;
                    rt=rt + rt2421;
                    ct=ct+t24
                    state = 11;

                    else

                    rt=rt + rt2421;
                    %fprintf('rt is %d',rt)
                    ct=ct+t24
    state = 21;
    end

case 33
      testStrings = {'your system is in 33 state'}
      t33=-1/l33*log(random23);
      ct=ct+t33
      if(random33<.51)
         state = 34;
      else
         state = 43;
      end

case 34
          testStrings = {'your system is in 34 state'}              % have to apply for rpair method
          %add time formulae
          t34=-1/l34*log(random33);

          rt3414=-1/mu3414*log(random33);
          rt3431=-1/mu3431*log(random33);

          if(rt3414<rt3431)
          %rt=rt + rt3414;
          rt=rt + rt3431;
          ct=ct+t34
          state = 11;

          else

          rt=rt + (-1/mu3431*log(random33));
          %fprintf('rt is %d',rt)
          ct=ct+t34
          state = 31;
          end
 case 43
          testStrings = {'your system is in 43 state'}
          t43=-1/l43*log(random33);

          rt4341=-1/mu4341*log(random33);
          rt4313=-1/mu4313*log(random33);

          if(rt4341<rt4313)
          rt=rt + rt4313;
          ct=ct+t43
          state = 11;

          else
          rt=rt + rt4313;
          ct=ct+t43
          state = 13;
          end
```

```
case 22
        testStrings = {'your system is in 22 state'}
        t22=-1/l22*log(random12);
        ct=ct+t22
        if(random22<.5)
           state = 23;
        else
           state = 32;
        end


 case 32
        testStrings = {'your system is in 32 state'}
        t32=-1/l32*log(random22);
        ct=ct+t32
        if(random32<.5)
           state = 33;
        else
           state = 42;
        end



  case 42
         testStrings = {'your system is in 42 state'}
         t42=-1/l42*log(random32);

         rt4241= -1/mu4241*log(random32);
         rt4212= -1/mu4212*log(random32);

         if(rt4241<rt4212)
         rt=rt+rt4212;
         ct=ct+t42
         state = 11;

         else
         rt=rt+rt4212;
         ct=ct+t42
         state = 12;
         end


   case 21
        testStrings = {'your system is in 21 state'}
        t21=-1/l21*log(random11);
        ct=ct+t21
        if(random21<.5)
           state = 22;
        else
           state = 31;
        end
        case 31
    testStrings = {'your system is in 31 state'}
    t31=-1/l31*log(random21);
        ct=ct+t31
        if(random31<0.5)
            state = 32;
        else
            state = 41;
        end



        case 41
      testStrings = {'your system is in 41 state'}
      t41=-1/l41*log(random31);
      rt=rt+ (-1/mu4111*log(random31));
      fprintf('rt is %d',rt)
      ct=ct+t41
      state = 11;


 end
 end
 %x=x+1;



     a(1,i)=(ct-rt)/ct;
     aa=(ct-rt)/ct;

     fprintf('availability is %d',aa);
     p(1,i)=i;

 end
        M=mean(a)
        fprintf('mean value over 1000 cases for perfect repair  is%d',M);
      %a(1,x)=avail/x;
      %avail=avail+a(1,x)
      %a(1,i)
     plot(p(1,:),a(1,:))

end
```

Program for MCS of Imperfect Repair without OM

```
function t= perfectrepairwithoutOM(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41, mu1411,
mu3431, mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;
aa=0;


l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;

mu1411=0.002;
mu3431=0.002;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;
random11=0;
 random12=0;
 random13=0;
 random14=0;
 random23=0;
 random33=0;
 random22=0;
 random32=0;
 random21=0;
 random31=0;
 %get the mission time from user and compare everytime for the condition of
%time



   a=zeros(1,1000);            % matrix of 1*1000
   p=zeros(1,1000);            % matrix of 1*1000

   missiontime=input('enter the mission time');



 for i=1:1:1000


   state=11;
    ct=0;
   rt=0;
   avail=0;

   %availability=0;



 while ct<missiontime

 random11=random('unif', 0,1);
 random12=random('unif', 0,1);
 random13=random('unif', 0,1);
 random14=random('unif', 0,1);
 random23=random('unif', 0,1);
 random33=random('unif', 0,1);
 random22=random('unif', 0,1);
 random32=random('unif', 0,1);
 random21=random('unif', 0,1);
 random31=random('unif', 0,1);
 switch state


 case 11
    testStrings={'11 state'}
        if(random11 < 0.5)
        state = 12;

        else
        state = 21;
        end
```

```
case 12
    testStrings={'your system is in 12 state'}
    t12=-1/l12*log(random11)
    ct=ct+t12
    if(random12<.5)
    state = 13;
    else
    state = 22;
    end

case 13
    testStrings = {'your system is in 13 state'}
    %add time formulae
    t13=-1/l13*log(random12);
    ct=ct+t13
    if(random13<.5)
    state = 14;
    else
    state = 23;
    end

case 14

    testStrings = {'your system is in 14 state'}   % have to apply for rpair method
    t14=-1/l14*log(random13);
    ct=ct+t14
    rt=rt + (-1/mu1411*log(random13));
    fprintf('rt is %d',rt)
    state = 12;


                case 23
                testStrings = {'your system is in 23 state'}
                t23=-1/l23*log(random13);
                ct=ct+t23
                if(random23<.5)
                state = 24;
                else
                state = 33;
                end

                case 24
                    testStrings = {'your system is in 24 state'}
                    %add time formulae
                    t24=-1/l23*log(random23);
                    rt=rt + (-1/mu2421*log(random23));
                    fprintf('rt is %d',rt)
                    ct=ct+t24
                   state = 22;

               case 33
                       testStrings = {'your system is in 33 state'}
                       t33=-1/l33*log(random23);
                       ct=ct+t33
                       if(random33<.51)
```

```
        state = 34;
    else
        state = 43;
    end

case 34
        testStrings = {'your system is in 34 state'}            % have to apply for rpair method
        %add time formulae
        t34=-1/l34*log(random33);
        rt=rt + (-1/mu3431*log(random33));
        fprintf('rt is %d',rt)
        ct=ct+t34
        state = 32;

 case 43
        testStrings = {'your system is in 43 state'}
        t43=-1/l43*log(random33);
        rt=rt + (-1/mu4313*log(random33));
        fprintf('rt is %d',rt)
        ct=ct+t43
        state = 23;




  case 22
         testStrings = {'your system is in 22 state'}
         t22=-1/l22*log(random12);
         ct=ct+t22
         if(random22<.5)
             state = 23;
         else
             state = 32;
         end


   case 32
        testStrings = {'your system is in 32 state'}
        t32=-1/l32*log(random22);
        ct=ct+t32
        if(random32<.5)
            state = 33;
        else
            state = 42;
        end



    case 42
          testStrings = {'your system is in 42 state'}
          t42=-1/l42*log(random32);
          rt=rt+ (-1/mu4212*log(random32));
          fprintf('rt is %d',rt)
          ct=ct+t42
          state = 22;
```

```
                    case 21
                         testStrings = {'your system is in 21 state'}
                         t21=-1/l21*log(random11);
                         ct=ct+t21
                         if(random21<.5)
                            state = 22;
                         else
                            state = 31;
                         end



          case 31
   testStrings = {'your system is in 31 state'}
   t31=-1/l31*log(random21);
          ct=ct+t31
          if(random31<0.5)
             state = 32;
          else
             state = 41;
          end


          case 41
     testStrings = {'your system is in 41 state'}
     t41=-1/l41*log(random31);
     rt=rt+ (-1/mu4111*log(random31));
     fprintf('rt is %d',rt)
     ct=ct+t41
     state = 21;


 end
 end
 %x=x+1;



    a(1,i)=(ct-rt)/ct;
    aa=(ct-rt)/ct;

    fprintf('availability is %d',aa);
    p(1,i)=i;

 end

      M=mean(a)
      fprintf('mean value over 1000 cases for perfect repair  is%d',M);
     %a(1,x)=avail/x;
    %avail=avail+a(1,x)
    %a(1,i)
   plot(p(1,:),a(1,:))

end
```

Program for MCS of Imperfect Repair with OM

```
function t= imperfectwithopportunisticmaintenance(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41,
mu1411, mu3431, mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;



l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;



mu1411=0.002;
mu3431=0.002;
mu3432=.003334;
mu2422=.003334;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;

mu2414=0.001;   %k
mu4241=0.01333;
mu3414=0.003334;  %k
mu4341=0.005;
```

```
mu4323=0.0025;

random11=0;
 random12=0;
 random13=0;
 random14=0;
 random23=0;
 random33=0;
 random22=0;
 random32=0;
 random21=0;
 random31=0;




 %get the mission time from user and compare everytime for the condition of
%time




    a=zeros(1,1000);            % matrix of 1*1000
    p=zeros(1,1000);            % matrix of 1*1000

    missiontime=input('enter the mission time');




 for i=1:1:1000


    state=11;
     ct=0;
    rt=0;
    avail=0;

    %availability=0;



 while ct<missiontime

 random11=random('unif', 0,1);
 random12=random('unif', 0,1);
 random13=random('unif', 0,1);
 random14=random('unif', 0,1);
 random23=random('unif', 0,1);
 random33=random('unif', 0,1);
 random22=random('unif', 0,1);
 random32=random('unif', 0,1);
 random21=random('unif', 0,1);
 random31=random('unif', 0,1);
 switch state
```

```
case 11
   testStrings={'11 state'}
      if(random11 < 0.5)
      state = 12;

      else
      state = 21;
      end



case 12
   testStrings={'your system is in 12 state'}
   t12 =-1/l12*log(random11)
   ct =ct+t12
   if(random12<.5)
   state = 13;
   else
   state = 22;
   end

case 13
   testStrings = {'your system is in 13 state'}
   %add time formulae
   t13=-1/l13*log(random12);
   ct=ct+t13
   if(random13<.5)
   state = 14;
   else
   state = 23;
   end

case 14

   testStrings = {'your system is in 14 state'}   % have to apply for rpair method
   t14=-1/l14*log(random13);
   ct=ct+t14
   rt=rt + (-1/mu1411*log(random13));
   fprintf('rt is %d',rt)
   state = 12;


              case 23
              testStrings = {'your system is in 23 state'}
              t23=-1/l23*log(random13);
              ct=ct+t23
              if(random23<.5)
              state = 24;
              else
              state = 33;
              end

              case 24
                 testStrings = {'your system is in 24 state'}
```

```
     %add time formulae
     t24=-1/l23*log(random23);

     rt2414=-1/mu2414*log(random23);
     rt2422=-1/mu2422*log(random23);
     if(rt2414<rt2422)
     rt=rt + rt2422;
     ct=ct+t24
     state = 12;

     else
     rt=rt + rt2422;
     ct=ct+t24
     state = 22;
     end

case 33
       testStrings = {'your system is in 33 state'}
       t33=-1/l33*log(random23);
       ct=ct+t33
       if(random33<.51)
          state = 34;
       else
          state = 43;
       end

case 34
            testStrings = {'your system is in 34 state'}             % have to apply for rpair method
            %add time formulae
            t34=-1/l34*log(random33);
            rt3414=-1/mu3414*log(random33);
            rt3432=-1/mu3432*log(random33);
            if(rt3414<3432)
               rt=rt+rt3432;
               ct=ct+t34
               state = 12;
            else

            rt=rt + rt3432;
            ct=ct+t34
            state = 32;
            end

 case 43
            testStrings = {'your system is in 43 state'}
            t43=-1/l43*log(random33);

            rt4341=-1/mu4341*log(random33);
            rt4313=-1/mu4323*log(random33);
            if(rt4341<rt4313)
            rt=rt + rt4313;
            ct=ct+t43
            state = 21;

            else
            rt=rt + rt4313;
```

```
         ct=ct+t43
         state = 23;
         end




case 22
          testStrings = {'your system is in 22 state'}
          t22=-1/l22*log(random12);
          ct=ct+t22
          if(random22<.5)
             state = 23;
          else
             state = 32;
          end


 case 32
        testStrings = {'your system is in 32 state'}
        t32=-1/l32*log(random22);
        ct=ct+t32
        if(random32<.5)
           state = 33;
        else
           state = 42;
        end



  case 42
           testStrings = {'your system is in 42 state'}
           t42=-1/l42*log(random32);

           rt4241= -1/mu4241*log(random32);
           rt4222= -1/mu4212*log(random32);
           if(rt4241<rt4222)

           rt=rt+ rt4222;
           ct=ct+t42
           state = 21;
           else
           rt=rt+ rt4222;
           ct=ct+t42
           state = 22;
           end


   case 21
          testStrings = {'your system is in 21 state'}
          t21=-1/l21*log(random11);
          ct=ct+t21
          if(random21<.5)
             state = 22;
          else
             state = 31;
```

```
                        end




         case 31
   testStrings = {'your system is in 31 state'}
   t31=-1/l31*log(random21);
         ct=ct+t31
         if(random31<0.5)
             state = 32;
         else
             state = 41;
         end



         case 41
     testStrings = {'your system is in 41 state'}
     t41=-1/l41*log(random31);
     rt=rt+ (-1/mu4111*log(random31));
     fprintf('rt is %d',rt)
     ct=ct+t41
     state = 21;


 end
 end
 %x=x+1;



    a(1,i)=(ct-rt)/ct;
    aa=(ct-rt)/ct;

    fprintf('availability is %d',aa);
    p(1,i)=i;
 end

      M=mean(a)
      fprintf('mean value over 1000 cases for perfect repair  is%d',M);
     %a(1,x)=avail/x;
    %avail=avail+a(1,x)
    %a(1,i)
    plot(p(1,:),a(1,:))

end
```

<u>Program for MCS of Minimal Repair without OM</u>

```
function t= minimalrepairwithoutOM(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41, mu1411,
mu3431, mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;
aa=0;


l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;

mu1411=0.002;
mu3431=0.002;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;
random11=0;
 random12=0;
 random13=0;
 random14=0;
 random23=0;
 random33=0;
 random22=0;
 random32=0;
 random21=0;
 random31=0;
 %get the mission time from user and compare everytime for the condition of
%time



   a=zeros(1,1000);            % matrix of 1*1000
   p=zeros(1,1000);            % matrix of 1*1000

   missiontime=input('enter the mission time');



 for i=1:1:1000


   state=11;
    ct=0;
   rt=0;
   avail=0;

   %availability=0;



 while ct<missiontime

 random11=random('unif', 0,1);
 random12=random('unif', 0,1);
 random13=random('unif', 0,1);
 random14=random('unif', 0,1);
 random23=random('unif', 0,1);
 random33=random('unif', 0,1);
 random22=random('unif', 0,1);
 random32=random('unif', 0,1);
 random21=random('unif', 0,1);
 random31=random('unif', 0,1);
 switch state


 case 11
    testStrings={'11 state'}
        if(random11 < 0.5)
        state = 12;

        else
        state = 21;
        end
```

```
case 12
   testStrings={'your system is in 12 state'}
   t12=-1/l12*log(random11)
   ct=ct+t12
   if(random12<.5)
   state = 13;
   else
   state = 22;
   end

case 13
   testStrings = {'your system is in 13 state'}
   %add time formulae
   t13=-1/l13*log(random12);
   ct=ct+t13
   if(random13<.5)
   state = 14;
   else
   state = 23;
   end

case 14

   testStrings = {'your system is in 14 state'}   % have to apply for rpair method
   t14=-1/l14*log(random13);
   ct=ct+t14
   rt=rt + (-1/mu1411*log(random13));
   fprintf('rt is %d',rt)
   state = 13;


               case 23
               testStrings = {'your system is in 23 state'}
               t23=-1/l23*log(random13);
               ct=ct+t23
               if(random23<.5)
               state = 24;
               else
               state = 33;
               end

               case 24
                  testStrings = {'your system is in 24 state'}
                  %add time formulae
                  t24=-1/l23*log(random23);
                  rt=rt + (-1/mu2421*log(random23));
                  fprintf('rt is %d',rt)
                  ct=ct+t24
                 state = 23;

              case 33
                    testStrings = {'your system is in 33 state'}
                    t33=-1/l33*log(random23);
                    ct=ct+t33
                    if(random33<.51)
```

```
            state = 34;
        else
            state = 43;
        end

case 34
            testStrings = {'your system is in 34 state'}              % have to apply for rpair method
            %add time formulae
            t34=-1/l34*log(random33);
            rt=rt + (-1/mu3431*log(random33));
            fprintf('rt is %d',rt)
            ct=ct+t34
            state = 33;

 case 43
            testStrings = {'your system is in 43 state'}
            t43=-1/l43*log(random33);
            rt=rt + (-1/mu4313*log(random33));
            fprintf('rt is %d',rt)
            ct=ct+t43
            state = 33;




  case 22
             testStrings = {'your system is in 22 state'}
             t22=-1/l22*log(random12);
             ct=ct+t22
             if(random22<.5)
                 state = 23;
             else
                 state = 32;
             end


   case 32
            testStrings = {'your system is in 32 state'}
            t32=-1/l32*log(random22);
            ct=ct+t32
            if(random32<.5)
                state = 33;
            else
                state = 42;
            end



    case 42
              testStrings = {'your system is in 42 state'}
              t42=-1/l42*log(random32);
              rt=rt+ (-1/mu4212*log(random32));
              fprintf('rt is %d',rt)
              ct=ct+t42
              state = 32;
```

```
                    case 21
                          testStrings = {'your system is in 21 state'}
                          t21=-1/l21*log(random11);
                          ct=ct+t21
                          if(random21<.5)
                              state = 22;
                          else
                              state = 31;
                          end




            case 31
      testStrings = {'your system is in 31 state'}
      t31=-1/l31*log(random21);
            ct=ct+t31
            if(random31<0.5)
                state = 32;
            else
                state = 41;
            end



            case 41
       testStrings = {'your system is in 41 state'}
       t41=-1/l41*log(random31);
       rt=rt+ (-1/mu4111*log(random31));
       fprintf('rt is %d',rt)
       ct=ct+t41
       state = 31;


 end
 end
 %x=x+1;

    a(1,i)=(ct-rt)/ct;
    aa=(ct-rt)/ct;

    fprintf('availability is %d',aa);
    p(1,i)=i;

 end
      M=mean(a)
      fprintf('mean value over 1000 cases for perfect repair  is%d',M);
      %a(1,x)=avail/x;
     %avail=avail+a(1,x)
     %a(1,i)
    plot(p(1,:),a(1,:))

end
```

Program for MCS of Minimal Repair with OM

```
function t= minimalwithopportunisticmaintenance(l12,l13,l14,l23,l22,l33,l34,l43,l24,l32,l42,l21,l31,l41,
mu1411, mu3431, mu4313, mu2421, mu4212, mu4111,
random11,random12,random13,random14,random23,random33,random22,random32,random21,random31,
ct, rt, state, i, a, p, avail)
%l11  is stage when components are in 1,1 state1=good,2=faulty,3=dangerours,4=faulty
%m1411 is the repairing of faulty state 14 to 11

%to include  repair time with every fault
%total time calculation
%how to call a function

% time taken by system A to degrade from one state to another is .00004107
% time taken by system B to degrade from one state to another is .00001

% time taken by system A to repair is .001667
% time taken by system B to repair is .002
l12=0.00001;
l13=0.00001;
l14=0.00001;
l22=0.00001;
l34=0.00001;
l24=0.00001;
l32=0.00001;
aa=0;


l23=0.00004107;
l33=0.00004107;
l43=0.00004107;
l33=0.00004107;
l42=0.00004107;
l21=0.00004107;
l31=0.00004107;
l41=0.00004107;

mu2423=0.001;
mu3424=0.005;
mu3433=0.001;

mu1411=0.002;
mu3431=0.002;
mu2421=0.002;


mu4313=0.001667;
mu4212=0.001667;
mu4111=0.001667;
mu2414=0.001;   %k
mu4241=0.01333;
mu3414=0.003334;  %k
mu4341=0.005;
```

```
mu4342=0.00667;
mu4333=0.001;
mu4232=0.001;
random11=0;
 random12=0;
 random13=0;
 random14=0;
 random23=0;
 random33=0;
 random22=0;
 random32=0;
 random21=0;
 random31=0;




 %get the mission time from user and compare everytime for the condition of
%time



   a=zeros(1,1000);            % matrix of 1*1000
   p=zeros(1,1000);            % matrix of 1*1000

   missiontime=input('enter the mission time');



 for i=1:1:1000


   state=11;
    ct=0;
   rt=0;
   avail=0;

   %availability=0;



 while ct<missiontime

 random11=random('unif', 0,1);
 random12=random('unif', 0,1);
 random13=random('unif', 0,1);
 random14=random('unif', 0,1);
 random23=random('unif', 0,1);
 random33=random('unif', 0,1);
 random22=random('unif', 0,1);
 random32=random('unif', 0,1);
 random21=random('unif', 0,1);
 random31=random('unif', 0,1);
```

```
switch state


case 11
   testStrings={'11 state'}
      if(random11 < 0.5)
      state = 12;

      else
      state = 21;
      end



case 12
   testStrings={'your system is in 12 state'}
   t12=-1/l12*log(random11)
   ct=ct+t12
   if(random12<.5)
   state = 13;
   else
   state = 22;
   end

case 13
   testStrings = {'your system is in 13 state'}
   %add time formulae
   t13=-1/l13*log(random12);
   ct=ct+t13
   if(random13<.5)
   state = 14;
   else
   state = 23;
   end

case 14

   testStrings = {'your system is in 14 state'}   % have to apply for rpair method
   t14=-1/l14*log(random13);
   ct=ct+t14
   rt=rt + (-1/mu1411*log(random13));
   fprintf('rt is %d',rt)
   state = 13;


               case 23
               testStrings = {'your system is in 23 state'}
               t23=-1/l23*log(random13);
               ct=ct+t23
               if(random23<.5)
               state = 24;
               else
               state = 33;
               end

               case 24
```

```
    testStrings = {'your system is in 24 state'}
    %add time formulae
    t24=-1/l23*log(random23);

    rt2414=-1/mu2414*log(random23);
    rt2423=-1/mu2423*log(random23);
    if(rt2414<2423)
       rt=rt+rt2423;
       ct=ct+t24
       state = 13;

    else
       rt=rt + rt2423;
       ct=ct+t24
       state = 23;
    end

case 33
      testStrings = {'your system is in 33 state'}
      t33=-1/l33*log(random23);
      ct=ct+t33
      if(random33<.51)
         state = 34;
      else
         state = 43;
      end

case 34
          testStrings = {'your system is in 34 state'}          % have to apply for rpair method
          %add time formulae
          t34=-1/l34*log(random33);

          rt3424=-1/mu3424*log(random33);
          rt3433=-1/mu3433*log(random33);
          if(rt3424<3433)

             rt=rt+rt3433;
             ct=ct+t34
             state = 23;
          else

          rt=rt + rt3433;
          ct=ct+t34
          state = 33;
          end

 case 43
          testStrings = {'your system is in 43 state'}
          t43=-1/l43*log(random33);

          rt4342=-1/mu4342*log(random33);
          rt4333=-1/mu4333*log(random33);
          if(rt4342<rt4333)
             rt=rt+rt4333;
             ct=ct+t43
             state = 32;
          else
              rt=rt+rt4333;
              ct=ct+t43
              state = 33;
          end


 case 22
           testStrings = {'your system is in 22 state'}
           t22=-1/l22*log(random12);
           ct=ct+t22
           if(random22<.5)
              state = 23;
           else
              state = 32;
           end


  case 32
          testStrings = {'your system is in 32 state'}
          t32=-1/l32*log(random22);
          ct=ct+t32
          if(random32<.5)
             state = 33;
          else
             state = 42;
          end



   case 42
             testStrings = {'your system is in 42 state'}
             t42=-1/l42*log(random32);

             rt4241= -1/mu4241*log(random32);
             rt4232= -1/mu4232*log(random32);
             if(rt4241<4232)
                rt=rt+rt4232;
                ct=ct+t42
                state = 31;
             else
                rt=rt+rt4232;
                ct=ct+t42
                state = 32;
             end


    case 21
           testStrings = {'your system is in 21 state'}
           t21=-1/l21*log(random11);
           ct=ct+t21
           if(random21<.5)
              state = 22;
           else
              state = 31;
```

```
                        end




          case 31
      testStrings = {'your system is in 31 state'}
      t31=-1/l31*log(random21);
          ct=ct+t31
          if(random31<0.5)
             state = 32;
          else
             state = 41;
          end



          case 41
        testStrings = {'your system is in 41 state'}
        t41=-1/l41*log(random31);
        rt=rt+ (-1/mu4111*log(random31));
        fprintf('rt is %d',rt)
        ct=ct+t41
        state = 31;


  end
  end
  %x=x+1;



     a(1,i)=(ct-rt)/ct;
     aa=(ct-rt)/ct;

     fprintf('availability is %d',aa);
     p(1,i)=i;
  end
        M=mean(a)
        fprintf('mean value over 1000 cases for perfect repair  is%d',M);
        %a(1,x)=avail/x;
      %avail=avail+a(1,x)
      %a(1,i)
     plot(p(1,:),a(1,:))

 end
```