This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

Safety modelling for the time limited dispatch of aircraft

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

© Darren Richard Prescott

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Prescott, Darren R.. 2013. “Safety Modelling for the Time Limited Dispatch of Aircraft”. figshare. https://hdl.handle.net/2134/12235.

Loughborough University

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

Loughborough University

University Library

Author/Filing Title PRESCOTT, D

Class Mark T

Please note that fines are charged on ALL overdue items.

0403604206

Safety Modelling for the Time Limited Dispatch of Aircraft

By

Darren Richard Prescott

A Doctoral Thesis
Submitted in partial fulfilment of the requirements for the award of
Doctor of Philosophy of Loughborough University

September 2007

© by Darren Richard Prescott, 2007

Loughborough University
Pilkington Library
Date Nov '08
Class T
Acc No. 0403604206

ABSTRACT

Time-limited dispatch (TLD) allows aircraft to dispatch with known faults present in the engine control systems. These systems govern the thrust control of engines and, in order for a TLD scheme to be certified, the failure rates to loss of thrust control (LOTC) must lie within or below prescribed bounds. The aim of the work presented in this thesis was to develop a tool that could be used to model the time-limited dispatch (TLD) of aircraft and compare this with currently recommended techniques for modelling TLD.

Currently techniques for modelling TLD require the failure rates to LOTC from various dispatchable system configurations, i.e. system configurations where a number of faults are present. These rates determine how long the system may be dispatched with certain faults present and also a failure rate to LOTC can be computed for the whole system. A number of approximations are used within the models, intended to make them simple to use. The concern is that these approximations might not accurately model the system behaviour and that the results obtained might as a consequence be unreliable.

Monte Carlo simulation (MCS) was identified as an analysis method that could deal with the intricacies and complexities involved in the application of TLD. Software, written in C++, allowed TLD to be modelled using MCS. Full use was made of the flexibility of MCS and many TLD maintenance scenarios were considered as the code was developed. The MCS code was written in such a way that all failure rates to LOTC, i.e. those representing the system LOTC and the dispatchable fault LOTC rates, would be produced as outputs. Results obtained using the developed software and the currently recommended techniques were compared.

Finally, the MCS code was embedded within an optimisation procedure in order to demonstrate how such a tool could be used in the design process for a system. A simple genetic algorithm procedure was employed to carry out this optimisation process.

ACKNOWLEDGEMENTS

There are many people to thank for their support while I have been doing this work. First of these is my supervisor, Professor John Andrews, whom I wish to thank for giving me the opportunity to do this PhD and also for his continued encouragement, guidance and friendship throughout the time of the research. Secondly, I'd like to thank my colleagues in the Department of Aeronautical and Automotive Engineering, and in particular in the Risk and Reliability Research group, for contributing to an enjoyable and productive working environment. Thirdly, my friends and family have given me their support for much longer than the time taken to complete this work and I'd like to thank them all for their continued friendship. Lastly, but most importantly of all, thanks to my wife, Rasa, who has helped me and given me her total support throughout. She's put up with listening to my ideas, good or bad, and also with all of the ups and downs of having a husband work on a PhD while working towards one herself, all the while being a good friend and colleague.

Thank you all!

CONTENTS

NOMENCLATURE **1**

CHAPTER 1 : TIME-LIMITED DISPATCH **2**

1.1 Introduction to Reliability Assessment **2**

1.2 Background of Time-Limited Dispatch **4**

1.3 Introduction to Time-Limited Dispatch (TLD) **5**

1.4 Incorporating TLD in Maintenance Operations **7**

1.4.1 MEL Maintenance *7*

1.4.2 Periodic Inspection/Repair Maintenance *8*

1.5 The Occurrence of Faults **12**

1.5.1 Multiple Faults *13*

1.5.1.1 Example – MEL Maintenance 13

1.5.1.2 Example – PIR Maintenance 14

1.5.2 Combinations of Faults *15*

1.5.3 Ordered Combinations Of Faults *16*

1.5.4 Safety Margins *16*

1.6 Characteristics of Modelling TLD **16**

1.7 Current Recommended Approaches for Modelling TLD **17**

1.8 Research Objectives **18**

CHAPTER 2 : FAULT TREE ANALYSIS **19**

2.1 Background **19**

2.2 Fault Tree Construction **19**

2.3 Qualitative Analysis **21**

2.3.1 Cut Sets and Implicants *21*

2.3.2 Obtaining Minimal Cut Sets *22*

2.3.2.1 Example 23

2.4 Quantitative Analysis **25**

2.4.1 Top Event Probability *25*

2.4.1.1 Upper and Lower Bounds for System Unavailability 26

2.4.1.2 Minimal Cut Set Upper Bound 26

2.4.2 Minimal Cut Set Unconditional Failure Intensity *27*

2.4.3 System Unconditional Failure Intensity *28*

2.4.3.1 Example 28
2.4.3.2 Expected Number of System Failures 29
2.4.3.3 Initiating and Enabling Events 29
2.4.4 Importance Measures *30*
2.4.4.1 Deterministic Measures 31
2.4.4.2 Probabilistic Measures for System Unavailability 31
2.5 Summary of Fault Tree Analysis **33**

CHAPTER 3 : MARKOV ANALYSIS OF SYSTEMS **34**

3.1 Background **34**
3.2 Construction of the Markov Model **35**
3.2.1 Example – A Two-State Repairable Component *35*
3.3 Matrix Representation of the Model **37**
3.3.1 Example – State Transition Matrix for The Repairable Component *38*
3.4 Markov Model Solution **39**
3.4.1 Analytical Solution *39*
3.4.2 Numerical Solution *40*
3.5 Steady-State Probabilities **41**
3.5.1 Example – Repairable Component *41*
3.5.2 Finding the Steady-State Solution of the Markov Model *42*
3.5.3 Example –Two-Component System *43*
3.6 Reduced Markov Models **45**
3.6.1 Example –Two Component System *46*
3.7 Reliability Modelling **47**
3.8 Summary **49**

CHAPTER 4 : MONTE CARLO SIMULATION OF SYSTEMS **50**

4.1 Background **50**
4.2 Uniform Random Numbers **50**
4.2.1 Recursion Formulae *51*
4.3 Direct Simulation **51**
4.3.1 Example – Time-Independent Model *52*
4.4 Using Probability Distributions to Generate Event Times **53**
4.4.1 Exponential Distribution *54*

4.4.2 Weibull Distribution *55*
4.4.3 Normal Distribution *55*
4.4.4 Central Limit Theorem *56*
4.4.5 Example – Simulating a System *56*
4.5 Results of Monte Carlo Simulations **58**

CHAPTER 5 : AEROSPACE RECOMMENDED PRACTICE METHODS OF MODELLING TLD **60**

5.1 Introduction **60**
5.2 A General System Model **61**
5.3 Time-Weighted Averages (TWA) Modelling **61**
5.3.1 The Failure Rate Due to Hydromechanical Faults, λ_{HMC} *62*
5.3.2 The Failure Rate for Unrevealed Faults, λ_{UR} *62*
5.3.3 Average Failure Rates to LOTC, $\lambda_{STD,L}$, $\lambda_{LTD,L}$ *63*
5.3.4 Fractions of Time Spent in Dispatchable States, t_{FU}, t_{STD}, t_{LTD} *64*
5.3.5 LOTC Rate *65*
5.3.6 Failure Rates to LOTC, $\lambda_{i,L}$ *65*
5.3.7 The Failure Rate for Full-Up Electronics, $\lambda_{FU,L}$ *66*
5.4 TWA Approach Using the Balanced Time Coefficients **67**
5.5 Reduced Fault State Markov Approaches **68**
5.5.1 Single Fault State Markov Approach *69*
5.5.2 Calculating the LOTC Rate *70*
5.5.3 First Order Approximation to the LOTC Rate *73*
5.5.4 Simplified Approximations When Dispatch Intervals are Short *75*
5.5.5 Dual Fault State Markov Approach *76*
5.6 Summary **80**
5.6.1 Approximations/Assumptions Involved in the TWA Approach *80*
5.6.2 Approximations/Assumptions Involved in the Single Fault State Markov Approach *81*
5.6.3 Approximations /Assumptions Involved in the Dual Fault State Markov Approach *82*

CHAPTER 6 : TLD SIMULATION MODELLING METHODOLOGY **83**

6.1 Introduction **83**

6.2 TLD MCS Code Outline **85**
6.3 TLD MCS Code Structure **86**
6.4 Event Scheduling **87**
6.4.1 Removing the First Event From the Schedule *89*
6.4.2 Adding an Event to the Schedule *90*
6.4.3 Removing an Event From Any Position In the Schedule *92*
6.4.4 Event Times Added to the Schedule *93*
6.5 Handling System Dispatch Criteria **93**
6.6 Propagating Faults/Repairs Through the Fault Tree Structure **95**
6.7 Implementing Maintenance Within the TLD MCS Code **97**
6.7.1 Repair Strategy 1 – Repair the Last Fault of the Group Causing the Current Maintenance Deadline *98*
6.7.2 Repair Strategy 2 – Repair All Faults in the Group Causing the Current Maintenance Deadline *101*
6.7.3 Repair Strategy 3 – Repair All Faults in the Same Category as the Group Causing the Current Maintenance Deadline *102*
6.7.4 Repair Strategy 4 – Repair All Faults Present in the System *103*
6.7.5 Opportunistic Maintenance *103*
6.7.6 Repairing Associated Faults in the Schedule *104*
6.7.7 Maintenance When the System is Undispatchable *105*
6.8 Adding Maintenance Deadlines to the Schedule in the TLD MCS Code **105**
6.8.1 DND Maintenance Deadlines *106*
6.8.2 MEL Maintenance Deadlines *106*
6.8.3 PIR Maintenance Deadlines *107*
6.8.4 Repair Strategy 1 ..... *108*
6.8.5 Repair Strategy 2 ..... *111*
6.8.6 Repair Strategy 3 ..... *112*
6.8.7 Repair Strategy 4 ..... *114*
6.8.8 Handling Maintenance Deadlines *115*
6.9 TLD MCS Code Output **117**
6.9.1 Results Obtained From the TLD MCS Code *117*
6.9.1.1 The System LOTC Rate 118
6.9.1.2 The Instantaneous LOTC Rates From Dispatchable System Configurations 119

6.9.1.3 Importance Measures 124
6.9.1.4 Updated Dispatch Criteria 125
6.9.2 Results' Output Format *126*
6.10 Summary **127**

CHAPTER 7 : ANALYSIS OF TRANSIENT MARKOV MODELS FOR TLD 128

7.1 Introduction **128**
7.2 TLD MARKOV **128**
7.3 The Transition Rate Matrix **130**
7.4 Calculating the LOTC Rate **131**
7.4.1 The Average System LOTC Rate *132*
7.5 Numerical Integration **132**
7.6 TLD maintenance operations **134**
7.7 Maintenance after LOTC events **138**
7.8 Maintenance Approaches **138**
7.9 Summary **139**

CHAPTER 8 : MODELLING TLD FOR SIMPLE SYSTEMS **140**

8.1 Introduction **140**
8.2 Two Component System Example **140**
8.2.1 TWA Model *141*
8.2.2 Single Fault State Markov Model *143*
8.2.2.1 1st Order Approximation to the Single Fault State Markov LOTC Rate 144
8.2.2.2 Simplified Approximation for Short Dispatch Intervals 144
8.2.3 Transient Markov Model Solutions *144*
8.2.4 Monte Carlo Simulation *145*
8.2.5 Comparison of the Modelling Approaches *146*
8.3 Summary **153**

CHAPTER 9 : AUTOMATICALLY SETTING DISPATCH CRITERIA **154**

9.1 Introduction **154**
9.2 The Methodology **154**
9.3 Eight Component System Example **157**
9.3.1 The Dual Fault State Markov Model *159*

9.3.1.1 Determination of the Instantaneous LOTC Rates 159
9.3.1.2 The Dual Fault State Markov Model 162
9.3.2 The Transient Markov Model 165
9.3.3 Monte Carlo Simulation 166
9.3.4 Comparison of Results Obtained from Alternative Models 166
9.3.5 Further Dual Fault State Markov Model 174
9.3.6 Conclusions 176
9.3.7 Further Results Obtained Using the MCS Code 178
9.3.7.1 PIR Maintenance 178
9.3.7.2 Opportunistic Maintenance 179
9.3.7.3 Varying the System Operational Lifetime 180
9.3.7.4 Importance Measures 181
9.3.7.5 Optional Simulation Output 185
9.4 Summary 187

CHAPTER 10 : LARGE SCALE SYSTEM TLD MODELLING 188

10.1 Introduction 188
10.2 The System to be Modelled 189
10.3 The Average System LOTC Rate 192
10.4 Instantaneous Failure Rates to LOTC 197
10.5 Varying the Operational Lifetime of the System 198
10.5.1 Simulating Major Maintenance Using Reduced Operational Lifetimes ... 201
10.6 Importance Measures for the System 203
10.7 Other Models of the System 206
10.7.1 The Conventional Markov Model 206
10.7.2 The Dual Fault State Markov Model 206
10.8 Summary 208

CHAPTER 11 : TLD SYSTEM OPTIMISATION 211

11.1 Introduction 211
11.2 Genetic Algorithms 211
11.3 An Example System 214
11.3.1 The System Design 214
11.3.2 System Components 216

11.3.3 System Maintenance *218*
11.4 Genetic Algorithm Strings **218**
11.5 Evaluating String Fitness **220**
11.5.1 Calculating String Fitness *225*
11.6 Reproduction Probabilities **226**
11.7 Results **228**
11.8 Summary **232**
CHAPTER 12 : CONCLUSIONS AND FUTURE WORK **233**
12.1 Summary of Work **233**
12.2 Conclusions **237**
12.3 Future Work **239**
REFERENCES **240**
APPENDIX 1 . USER OPTIONS FILE FOR THE TLD MCS CODE **242**
APPENDIX 2 . TLD MCS CODE DETAILS **245**
APPENDIX 3 . TLD MCS CODE INPUT FILES FOR THE 8 COMPONENT SYSTEM **272**
APPENDIX 4 . SAMPLE OUTPUT FILES FOR THE 8 COMPONENT SYSTEM **273**
APPENDIX 5 . TLD MCS CODE INPUT FILES FOR THE LARGE SCALE SYSTEM **280**
APPENDIX 6 . RESULTS FOR THE LARGE SCALE SYSTEM **287**
APPENDIX 7 . THE RESULTS OBTAINED USING THE GENETIC ALGORITHM **290**

NOMENCLATURE

$A_{sys}(t)$	System availability at time t
ARP	Aerospace recommended practice
DND	Do not dispatch
$F(t)$	System unreliability at time t
FAA	Federal Aviation Authority
FADEC	Full Authority Digital Electronic Control
GA	Genetic algorithm
$h(t)$	Hazard rate
HMC	Hydromechanical control
LOTC	Loss of thrust control
LTD	Long time dispatch
MDD	Manufacturer/operator defined dispatch
MEL	Minimum equipment list
PIR	Periodic inspection and repair
$q(t)$	Component unavailability at time t
$Q_{sys}(t)$	System unavailability at time t
$R(t)$	System reliability at time t
SAE	Society of Automotive Engineers
STD	Short time dispatch
TLD	Time-limited dispatch
TWA	Time-weighted averages
$w(t)$	Component unconditional failure intensity
$w_{sys}(t)$	System unconditional failure intensity
$W(t_0, t_1)$	Expected number of failures in (t_0, t_1)

Chapter 1 : Time-Limited Dispatch

1.1 Introduction to Reliability Assessment

Nowadays there exist many complex systems that are required to operate within acceptable levels of safety. Examples of such systems are those used to maintain control in nuclear power stations and the systems used to control aircraft. Such systems must be demonstrated to meet certain levels of safety. The levels of safety demanded from a system can depend on a number of factors, such as the likely danger to human life or the likelihood of an environmental disaster caused by a system failure. It is important when designing any system that the reliability of that system is considered throughout the design process, in order for the system to achieve the required level of reliability in an economical manner. It is important not to progress many steps through the design and building of a system and then discover that the system will not meet the levels of safety required from it.

There are a number of parameters that are useful in investigating system safety. These include availability, reliability and failure rates. In order to calculate any of these parameters for a system the reliability performance of the individual system components must be known. Techniques may then be employed that allow the assessment of system performance. The parameter that is used to express system performance depends on the nature of the system itself.

When system failure can be tolerated and repair of the system is possible, an appropriate performance measure is availability. The availability of a system is defined as the fraction of time that the system is able to perform the function for which it is designed. The opposite of availability is unavailability. This is the quantity which is more commonly quoted as a performance measure. The definition of unavailability is the probability that a system does not work at a given time t. For a system component the unavailability is represented by $q(t)$ and for a system the unavailability is represented by $Q_{SYS}(t)$. The system availability is given by $A_{SYS}(t)$. The relationship between system unavailability and system availability is:

$$Q_{SYS}(t) = 1 - A_{SYS}(t). \qquad 1.1.1$$

For systems which cannot tolerate failure and must operate for specified periods of time without system failure occurring reliability provides the appropriate measure of system performance. Reliability is defined as the probability that a system will operate for a specified time t without a system failure occurring. System reliability is denoted by $R(t)$ and system unreliability is denoted by $F(t)$. The relationship between the two is as follows:

$$R(t) = 1 - F(t). \qquad 1.1.2$$

In the case of non-repairable systems, reliability and availability are equal. This is because, in such systems, if the system still works at time t and it has not been repaired, then it must have worked continuously since time $t = 0$.

A further measure of component or system performance is the failure rate. This comes in two forms. The first of these, the conditional failure rate, denoted by $h(t)$, also known as the hazard rate, is the probability that the system fails in the time interval $[t, t+dt)$ given that it has not failed in the interval $[0, t)$. The second failure rate is known as the unconditional failure intensity, which is defined as the probability of a system failing in the interval $[t, t+dt)$ given that it was working at time $t = 0$. This is denoted for a component as $w(t)$ and for a system as $w_{SYS}(t)$. The unconditional failure intensity can be used to calculate the expected number of failures over the time interval from t_0 to t_1 by integration, from:

$$W(t_0, t_1) = \int_{t_0}^{t_1} w(t)dt. \qquad 1.1.3$$

Here, $W(t_0, t_1)$ represents the expected number of failures in (t_0, t_1).

Many methods, such as fault tree analysis (FTA), Markov analysis or Monte Carlo simulation, can be used to evaluate the system safety parameters (all of these methods are described later in this thesis). Figure 1.1.1 gives a representation of the properties of these methods. As a general rule the complexity of models increase as one moves from FTA to Markov to MCS. The efficiency of the methods (in terms of processing power required and model construction) generally improves in the opposite direction to complexity. When modelling any system the most efficient method is selected whose complexity enables the system to be effectively modelled. Poor predictions of system

performance will come from using inappropriate or poorly constructed models or inaccurate solution routines.

Figure 1.1.1. A representation of the characteristics of three modelling techniques.

1.2 Background of Time-Limited Dispatch

Time-limited dispatch was first seen after the introduction of Full Authority Digital Electronic Control (FADEC) to commercial transport aircraft. FADEC is an electronic engine control system that uses a digital computer to control the variables required to regulate engine thrust throughout the operating range from the beginning of fuel metering to fuel shutoff. When first introduced on the Boeing 757 FADEC systems replaced the previously-used hydromechanical control (HMC) systems. Indeed, it was the first time that HMC systems would be unavailable to pilots in the event of an electronic system failure [1]. FADEC systems utilise a dual channel control system (two essentially identical channels per engine) that brings with it a level of redundancy. Redundant elements or dual systems exist for all critical loops and functions within the system. At the point of their introduction, due to the relative youth of the technology, there were no existing safety analyses for FADEC systems. No relevant loss of thrust control (LOTC) reliability criteria existed. Thus, in order to attempt to establish the viability of aircraft dispatch, it was initially decided to follow existing methods, which involved considering the consequences of the next failure to the system [2]. However, the complexity of FADEC systems made this a difficult task. Due to this, together with the absence of existing safety analyses and experience of the use of FADEC systems, the dispatch criteria subsequently imposed led to a negative impact on delays and cancellations of aircraft [2]. This negative impact appeared to originate mainly from the fact that independent faults in more than one channel were leading to overly prohibitive

dispatch criteria [1]. In order to overcome the restrictions imposed by these dispatch criteria a fresh approach was required which would reduce the numbers of delays and cancellations to more acceptable levels and yet still assure the desired standard of system reliability. Due to the fact that FADEC systems exhibited higher levels of reliability than established limits for HMC systems the opportunity arose to utilise redundancy within the system in order to allow for dispatch in the presence of faults. Systems are still required to meet the necessary airworthiness standards over the period of the degraded-redundancy dispatch [2]. In addition to substantially reducing the delays experienced by aircraft operators, the new approach also allows for the better planning and scheduling of maintenance operations. This new approach is known as **time-limited dispatch (TLD).**

Although the application of TLD to areas other than FADEC systems may be possible no guidelines currently exist for such uses. It is for this reason that the discussions in the following sections refer to FADEC systems.

1.3 Introduction to Time-Limited Dispatch (TLD)

TLD is a method allowing the dispatch of aircraft for a limited time in the presence of one or more faults whilst assuring a certain level of system reliability. It takes advantage of any available redundancy within the system, thus reducing the number of delays and cancellations, as well as the magnitude of their effects. In addition to this TLD enables more efficient scheduling of maintenance activities than would be possible without its use. According to the significance of the faults present within the system the aircraft may be dispatched for differing lengths of time or **dispatch intervals**. The dispatch interval is the maximum time interval that is approved for dispatching aircraft with known faults before maintenance must be undertaken to remedy those faults. Dispatch intervals can fall into any one of four categories (as defined by the FAA [2]),

- Do Not Dispatch (DND),
- Short Time Dispatch (STD),
- Long Time Dispatch (LTD),
- Manufacturer/Operator Defined Dispatch (MDD),

the names of which are reasonably self-explanatory. If applied, the DND category of dispatch means an aircraft may not be dispatched because of the faults present and corrective maintenance must be carried out immediately. The STD category of dispatch means the aircraft may be dispatched in the short-term with faults present in the system before suitable maintenance, while the LTD category permits a longer possible period of dispatch before the appropriate repairs are implemented. The MDD category of dispatch is defined to be for faults that do not fall into any of the other three categories or do not affect the LOTC rate of the system [2]. It is possible that the faults may or may not appear in the analysis for the LOTC rate but they will appear in the TLD analysis with a repair interval agreed between the engine and aircraft manufacturers.

Faults causing the application of the DND criteria are those that leave the system particularly vulnerable to the occurrence of further individual faults, which explains the urgency of corrective maintenance. Faults demanding the application of the STD and LTD criteria will leave the system less susceptible to further faults, indeed the longer the dispatch time, the more secure the system must be in the event of an additional solitary fault. The FAA gives an upper limit for the computed LOTC rate of 100 events per million flight hours for dispatchable configurations of the system. Indeed, if the computed LOTC rates lie within certain intervals then the dispatch criteria applied to the system must fall into specific categories [2]. The intervals for each category are given in Table 1.3.1.

Dispatch Category	DND	STD	LTD
LOTC rate, r	$r > 100$	$75 < r < 100$	$0 < r < 75$

Table 1.3.1. Computed LOTC rate for the different dispatch criteria (number of events per 10^6 flight hours).

The dispatch interval allowed for each category of dispatch may vary according to the maturity of the system. The FAA prescribe upper limits for the dispatch intervals for 'entry level' systems (see Table 1.3.2), which it defines as systems that have undergone less than 250,000 hours of operation in a particular installation or its equivalent [2]. Upon reaching maturity after this length of service these limits may be altered if

sufficient evidence is provided to support the changes, given that the FADEC system performs according to the analysis upon which the TLD approval is based.

Dispatch Category	DND	STD	LTD
Max. Dispatch Interval	0	125	250

Table 1.3.2. Maximum allowable dispatch intervals for TLD operations of 'entry level' systems (flight hours).

Since the requirement that FADEC systems should exhibit levels of reliability equal to or better than that displayed by earlier HMC systems, the average LOTC rate due to FADEC systems should be less than or equal to 10 events per 10^6 flight hours [1],[2]. Note that this is an average LOTC rate and should not be confused with the maximum permitted LOTC rate of 100 events per 10^6 flight hours previously quoted for TLD operations of FADEC systems.

1.4 Incorporating TLD in Maintenance Operations

Two maintenance strategies exist that allow the incorporation of the dispatch intervals associated with TLD, **MEL maintenance** and **periodic inspection/repair (PIR) maintenance**, which are described below. When implementing TLD it is possible to use either one of these strategies to address all faults from each dispatch category or some combination of the two strategies [2]. DND category faults must always be reported in the flight deck of the aircraft so that the aircraft is not dispatched with these faults. In order to ensure this DND faults are reported in the flight deck using essential equipment, i.e. equipment that is always available as an aircraft is dispatched. Thus the means of reporting the DND faults should always be available before dispatch.

1.4.1 MEL Maintenance

MEL maintenance is a time-since-fault repair strategy. The time of occurrence of a fault repaired using this strategy must be known and maintenance to rectify that fault must be implemented within the dispatch interval for the category of dispatch into which the

fault falls. Generally, the presence of the fault condition must be reported in the flight deck using essential equipment, as is the case with DND faults. The essential equipment must always be available to report the faults. MEL maintenance receives its name from the **minimum equipment list (MEL)**. The minimum equipment list details the aircraft equipment that must be operable in order for the aircraft to be airworthy. Before a flight, if a fault is indicated, the flight crew check the MEL to see if the aircraft can be flown with that fault. The reason for the MEL giving its name to this maintenance strategy is that the flight crew must check the MEL if a fault is present in order to check that dispatch is allowed and the MEL provides the length of the allowable dispatch interval.

MEL maintenance is principally used to repair faults that fall into the STD category, although this needn't be the case. The FAA states that this maintenance strategy isn't normally used to repair faults falling into categories of dispatch with intervals greater than 300 hours and also that it is an acceptable practice to "start the clock" at midnight on the day that the fault indication occurred [2].

Figure 1.4.1 contains a diagrammatic representation of the MEL maintenance strategy. If a fault occurs at time t_f a dispatch interval is initiated, during which time the aircraft may be dispatched. This assumes knowledge of the time of occurrence of the fault. If the fault has not been repaired by time t_r further dispatch of the aircraft is prohibited.

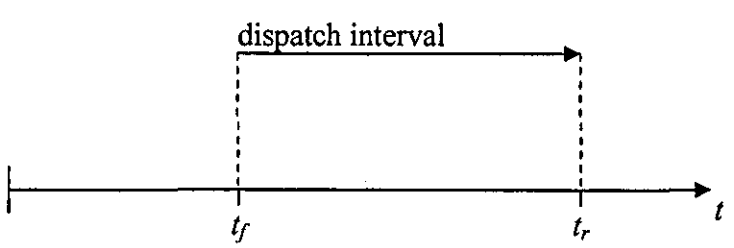


Figure 1.4.1. Timeline illustrating MEL maintenance.

1.4.2 Periodic Inspection/Repair Maintenance

The periodic inspection/repair maintenance strategy (PIR), as its name suggests, involves checking the system for faults at regular intervals. It is usually used to carry out maintenance on faults that fall into the LTD category, but could also be used to

maintain STD category faults. In contrast to the MEL maintenance approach there is no requirement for flight deck indications of fault occurrences, since the exact time of occurrence of the faults need not be known. Instead faults discovered at the periodic system inspections must be repaired within a certain time which ensures that the average exposure of the system to a fault doesn't exceed the maximum allowed by its dispatch category [2]. The exposure time of the system is the amount of time the system is required to operate in the presence of faults, before the faults are repaired.

When applying this maintenance strategy, because the time of occurrence of the fault is not known, it is assumed upon its discovery that the fault occurred halfway through the periodic inspection interval. This is the average time at which the fault will occur and is a reasonable assumption when the failure rate is constant with time. The inspection interval can be adjusted in order to satisfy this condition. If the failure rates of faults aren't constant the average exposure time should be adjusted accordingly [2].

There is, in practice, clearly going to be a maximum limit for the periodic inspection interval for each category of TLD. This will depend upon the length of the dispatch interval. Figure 1.4.2 gives an illustration of the PIR maintenance strategy and can be used to show how this maximum limit is derived. I_n and I_{n+1} represent two consecutive system inspections. A fault, A, occurs at some time between the inspections, but the exact time is not known. When the fault is discovered during inspection I_{n+1} it is assumed that the fault occurred halfway between the two inspections at point M. If TLD is to be used the aircraft may be dispatched for a further time, t_d, after which repairs must be implemented. This time t_d will be equal to the dispatch interval minus half of the periodic inspection interval, the reason for this being that the dispatch interval is initiated at the assumed time of occurrence of the fault, i.e. at M. It would be possible to set the inspection interval such that t_d equals zero in order to have the dispatch interval end as inspection I_{n+1} takes place. This gives a maximum size of the periodic inspection interval of twice the dispatch interval. In practice this would normally be undesirable since this allows no flexibility within the maintenance programme as to when the repair is carried out.

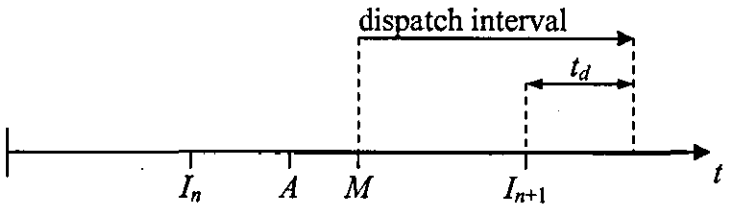


Figure 1.4.2. Timeline illustrating PIR maintenance.

The above description of the PIR maintenance process is perhaps better illustrated using specific values for the periodic inspection interval and the dispatch interval. Consider a fault occurring at some point between two consecutive inspection intervals that carries with it a dispatch interval of 300 hours. If the fault is assumed to have occurred at the midpoint of the inspection interval the maximum length of the inspection interval will be 600 hours. However, if this is the case, there is no scope for dispatching the aircraft after the fault has been discovered. In such a situation it may be more prudent to set the inspection interval to be 500 hours, say, so that if the fault is assumed to have occurred at the midpoint of the interval the exposure to this fault will be 250 hours, thus giving an extra 50 hours in which the aircraft may be dispatched. This allows some leeway for the fault to be repaired at a more convenient time.

The different approaches to maintenance taken in the MEL and PIR strategies mean that the maximum possible exposure time of a system to faults differs for each strategy despite the fact that the dispatch intervals are the same. If the maximum possible limit for the periodic inspection interval is twice the maximum possible dispatch interval, as described above, then a fault occurring immediately after an inspection will be present in the system until the next inspection for faults of that dispatch category. Table 1.4.1 shows these maximum values for the STD and LTD dispatch categories for an 'entry level' system (defined in Table 1.3.2).

Dispatch Category	STD	LTD
Max. Allowable Dispatch Interval	125	250
Max. Possible Exposure Time – MEL	125	250
Max. Possible Exposure Time – PIR	250	500

Table 1.4.1. Comparison of maximum allowable dispatch intervals and exposure times for MEL and PIR maintenance.

If the maintenance of more than one category of fault is to be addressed using a periodic inspection/repair maintenance strategy it is possible that during an inspection for faults of one dispatch category a fault related to another dispatch category could also be discovered. In situations such as this procedures are usually in place to acknowledge the fault at the current inspection but it is then addressed as though discovered at the next inspection for faults of its own dispatch category [2]. In order to illustrate this consider the timeline shown in Figure 1.4.3, on which the periodic inspections and faults relating to the STD fault category lie below the timeline and the periodic inspections and faults relating to the LTD fault category lie above the time line. I_1 and I_2 represent consecutive periodic inspections for faults covered by the STD category of dispatch, while I_A and I_B represent consecutive periodic inspections for faults covered by the LTD category of dispatch. Now assume that an STD fault, F_S, occurs at some time between I_1 and I_2 and an LTD fault, F_L, occurs at some time between I_A and I_B. These faults will be discovered at I_2 and I_B respectively. However, when F_S is discovered at I_2 the fault F_L may also be discovered, despite the fact that the inspection isn't for faults in its category. Even if F_L is discovered at this time there is no need for a dispatch interval to be initiated for it at this point. Its existence is acknowledged and it may be treated as usual when 'discovered' at I_B.

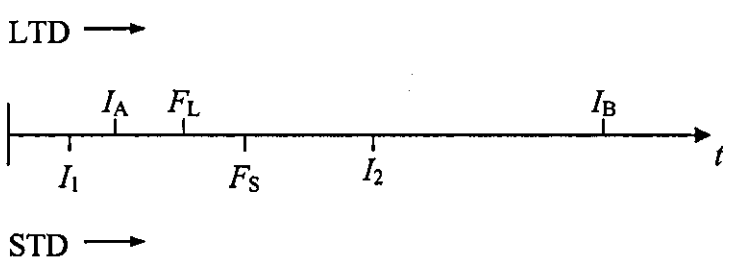


Figure 1.4.3. PIR maintenance for STD and LTD faults.

When the periodic inspection/repair maintenance strategy is implemented there exists the possibility to carry out all inspections and repairs at uniformly spaced intervals. This is achieved by setting the inspection/repair interval to be ⅔ of the length of the average exposure of the system to faults [2]. This situation is illustrated using the timeline in Figure 1.4.4, on which an inspection/repair process is shown with four consecutive inspections, $I_1,\ldots,I_4$, and the occurrence of two faults, A and B. As is customary, when A and B are discovered at I_2 and I_3 respectively, they are assumed to occur at the midpoint of the periodic inspection interval in which they occurred, that is at A' and B'.

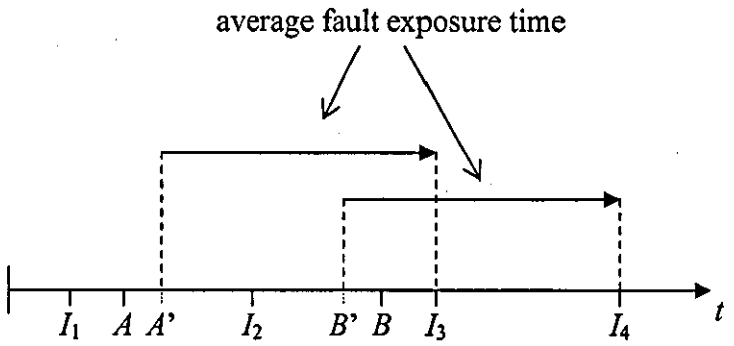


Figure 1.4.4. PIR maintenance allowing uniformly spaced repairs.

If the inspection interval is set according to the definition above, i.e. ⅔ of the average exposure time of the system to faults, the dispatch interval initiated upon the discovery of fault A will allow dispatch until inspection I_3 due to the fact that the average exposure of the system to the fault will begin at A' and end at I_3. Similarly, the discovery of fault B at I_3 will result in the assumed time of occurrence of the fault being B' and dispatch will be permitted until the next periodic inspection, I_4. Implementing periodic inspection/repair maintenance in this manner, with repairs set to occur at the time of inspections, could, for instance, allow relevant parts to be ordered for the repair of faults present at the current inspection in time for the next inspection. An interesting result of the implementation of this particular configuration of the inspections is that it would be theoretically possible for the system to always operate in the presence of faults and yet still meet all of the safety requirements asked of it.

Note that, when implementing the periodic inspection/repair maintenance strategy, all faults pertaining to a certain category of TLD must be repaired at the same time if there is no indication of the exact timing of these faults. This would be the case in practice when implementing this maintenance strategy since in general the warning in the flight deck of the system faults will be a generic one which merely indicates the presence of such faults.

1.5 The Occurrence of Faults

There are a number of ways in which the occurrence of faults within the system can affect the application of TLD to that system. Some of these are detailed below.

1.5.1 Multiple Faults

Despite the high reliability of FADEC systems the possibility exists for one or more faults to occur within the dispatch interval associated with another fault [2]. In situations such as these, when there are multiple faults within a system and a maintenance deadline is reached there will be a number of different possible repair options, in terms of the faults that must be cleared from the system. Whether DND, STD or LTD faults are present, or MEL or PIR maintenance is being implemented, it is important that the system is not dispatched with any fault for more than the allowed time for that fault. There are many possible scenarios to consider when implementing TLD when multiple faults arise. Advantage could be taken of the fact that not all faults will require immediate repair in order to allow maintenance to be conveniently scheduled or to wait until appropriate parts are available. Simple examples of multiple faults are illustrated below for MEL and PIR maintenance.

1.5.1.1 Example – MEL Maintenance

Consider the situation shown in Figure 1.5.1, which shows a timeline where A and B represent the occurrence of two faults that initiate dispatch intervals ending at t_1 and t_2 respectively. At the end of the dispatch interval relating to fault A this fault must be repaired, if this has not already happened. The repair of fault B may also be implemented at this time, thus clearing both faults from the system and returning it to a full-up configuration. However, assuming the exact time of occurrence of fault B is known then the aircraft may subsequently be dispatched further until time t_2 when the dispatch interval for this fault ends. Fault B could then be repaired at t_2.

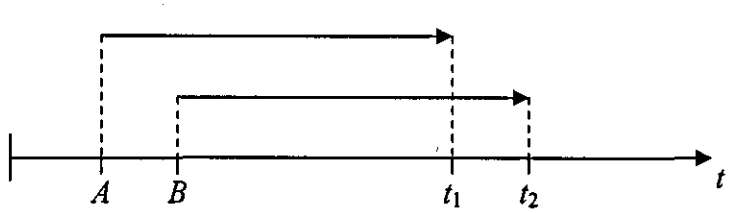


Figure 1.5.1. Two faults and their dispatch intervals.

1.5.1.2 Example – PIR Maintenance

If PIR maintenance is being implemented and multiple faults occur between two consecutive inspections then all of the faults will be repaired at the next maintenance deadline for faults of the relevant category. Another situation that may arise when multiple faults occur and PIR maintenance is being used is that one or more faults may occur in the time after an inspection but before the maintenance deadline initiated at that inspection. If this is the case the faults discovered at the inspection must be repaired as the deadline is reached. The faults occurring after the inspection could also be repaired at this time, if discovered, or they could be left to be 'discovered' at the next inspection at which point a maintenance deadline for their repair will be set. In order to illustrate this consider the timeline in Figure 1.5.2, which depicts such a scenario for two faults, A and B, which, when discovered at the following inspection, will be assumed to occur at A' and B' respectively. When fault A is discovered at inspection I_2 the maximum average fault exposure time of the system leads to a maintenance deadline ending at t_1. When this deadline is reached fault B may be discovered to have occurred in the time between the inspection and the deadline. If this is the case fault B could be repaired along with fault A at the deadline, or it could be allowed to remain in the system, then be 'discovered' at inspection I_3, at which point the appropriate maintenance deadline would be set. Note that in the scenario described above it is assumed that the faults A and B fall in the same dispatch category.

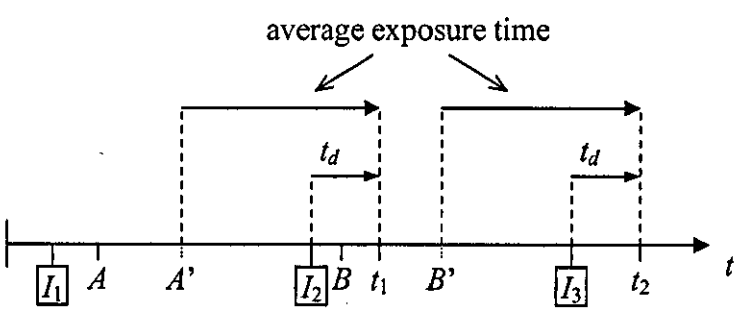


Figure 1.5.2. Two faults maintained using PIR, falling between different pairs of inspections.

1.5.2 Combinations of Faults

When using TLD in maintenance operations some faults may, if occurring in certain combinations, cause a reduction in the dispatch interval that is to be implemented. To illustrate this consider the example shown in Figure 1.5.3. In this example the occurrence of either fault A or fault B alone would initiate an LTD dispatch interval, the dispatch interval for A ending at t_2 and the dispatch interval for B ending at t_3. However, in combination, A and B cause the initiation of an STD dispatch interval ending at t_1, which is implemented as soon as both faults are present. Upon reaching the maintenance deadline at the end of the shortened dispatch interval (in Figure 1.5.3 this would be at time t_1) there are two maintenance options. The first of these is to repair both of the faults, thus leading to dispatch of the aircraft with no faults, and the second option is to repair only one of the faults, thus allowing the dispatch of the aircraft up to the point at which the maintenance deadline would occur for the remaining fault. In this way, in the example in Figure 1.5.3, once time t_1 is reached repairs to the aircraft are necessary. If only fault A is repaired the aircraft may then be dispatched until such a time that the LTD dispatch interval ends for fault B, i.e. t_3. Similarly, if only fault B is repaired the aircraft may be dispatched until the LTD dispatch interval for fault A ends, i.e. t_2.

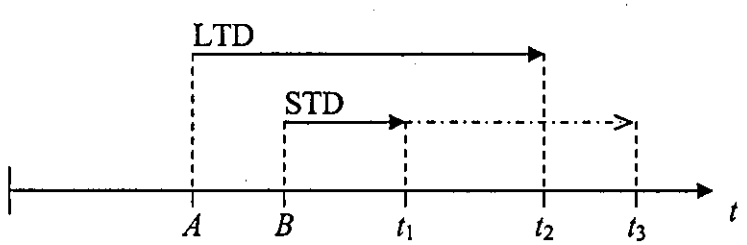


Figure 1.5.3. A fault combination causing a reduction in the dispatch interval.

The fact that combinations of faults may be addressed in this way again allows for greater flexibility and convenience when scheduling and implementing repairs. This situation could conceivably occur with a number of faults greater than two, with a number of faults leading to a reduction in the dispatch interval, but the process of repairing these faults could be generalised from that described above for two faults. Of course, this situation needn't only occur with STD and LTD dispatch intervals. A combination of faults may cause a DND criteria to be applied to the aircraft, so that it can't be dispatched until maintenance of some kind has been implemented. In such a

situation it may prove particularly useful to repair a small number of faults in order to allow dispatch of the aircraft, for example so that further maintenance could be carried out at a more suitable location or when parts are more readily available.

1.5.3 Ordered Combinations Of Faults

When faults work in combination to shorten the dispatch time it is possible that the faults may need to occur in a particular order for this to happen. For instance, consider again the example shown in Figure 1.5.3. Suppose that for the faults *A* and *B* to combine and shorten the dispatch time the mere presence of the faults was insufficient and that fault *A* must occur first and be followed by fault *B*, as was depicted in Figure 1.5.3. If *A* is not followed by *B* their individual dispatch intervals are applied as usual.

1.5.4 Safety Margins

For 'entry level' systems a 2:1 safety margin is introduced by the FAA to dispatch intervals when implementing TLD [2]. This means that, in order to approve the dispatch times for an 'entry level' system, the predicted average LOTC rate for the system should be no greater than the level of 10 events per million flight hours, discussed in Section 1.3, for dispatch intervals that are twice the length of those to be used. In order to illustrate this point, consider a case where a FADEC system was shown to meet the 10 events per million flight hours requirement for an STD dispatch interval of 200 hours. For an entry level system this means the allowed STD dispatch interval would initially be 100 hours.

1.6 Characteristics of Modelling TLD

TLD has a number of characteristics and features that must be accounted for in any process that is used to model it. These lead to a number of requirements of a TLD modelling tool:

- The systems to which TLD is applied involve dependencies, e.g. operating procedures brought about by TLD bring dependencies, as do maintenance and repair operations. The modelling tool chosen for TLD must be able to handle dependencies.

- The systems to which TLD is applied are large, meaning that the tool used to model them must be able to deal effectively and accurately with large systems.
- TLD brings with it flexibility in the maintenance options, e.g. MEL or PIR maintenance are possible and various ways of addressing faults at maintenance are possible. A good model should be able to deal with these different maintenance possibilities.
- The dispatch criteria that are applied to a system must conform to regulations regarding the instantaneous failure rates to LOTC from the dispatchable configurations, as must the overall system LOTC rate. It must be possible to show that all requirements regarding the LOTC rates are met.

1.7 Current Recommended Approaches for Modelling TLD

There are currently two approaches recommended for modelling TLD [2], a time-weighted average (TWA) approach and a reduced fault state Markov approach. The TWA approach is stated to be a fault tree based approach but the two models use similar data and are essentially variations on a similar approach. Conventional fault tree analysis is unsuitable for modelling TLD since it cannot deal with the dependencies introduced. Conventional Markov analysis will be unsuitable because of a state space explosion that takes place because of the large system size.

The recommended approaches do not appear wholly suitable for modelling TLD for a number of reasons:

- The models are simplistic and as such could be incapable of capturing the detailed behaviour of large, complex aircraft systems.
- The models do not have the capacity to deal with the different possible maintenance and repair options available while implementing TLD.
- The instantaneous LOTC rates are not calculated using the models and must be estimated and included in the models. If these rates are incorrectly approximated the combination of this with the potentially poor model structure could lead to inaccurate results.

1.8 Research Objectives

It is proposed here that Monte Carlo simulation (MCS) is well-suited to modelling TLD. It offers the capability to deal with complex systems, dependencies and different repair and maintenance strategies and produce accurate, auditable results.

The objectives of this research are to:

- Develop a MCS methodology for modelling TLD that will encompass the different options available when modelling TLD, such as different maintenance and repair options and system information such as operating lifetime and flight lengths.
- Develop the MCS methodology to give an integrated approach to setting dispatch criteria for a system and ensuring that all certification requirements are met.
- Develop importance measures, using the MCS approach, for systems to which TLD is applied.
- Compare the recommended approaches with the MCS approach and identify and explain differences in the results.
- Demonstrate how an integrated approach to modelling TLD could be incorporated at the design stage of a system and used to obtain more information about the system lifetime.

Chapter 2 : Fault Tree Analysis

2.1 Background

Developed by H. A. Watson in the early 1960's, fault tree analysis (FTA) is the most widely used tool in safety and reliability assessment [3]. After the introduction of computer-based analysis techniques the use of fault tree analysis became increasingly common [4]. Fault trees provide a method of visually representing the causes of a particular system failure mode in a structured and logical way. All causes of the system failure mode are described within the fault tree in terms of component failure modes and operator actions/human errors. Any weak points in the system may subsequently be identified, allowing the rectification of any potential problems.

2.2 Fault Tree Construction

The construction of fault trees begins by considering a particular system failure mode, known as the **top event**, and identifying the events that cause it. These events are then developed by determining their immediate, necessary and sufficient causes and the process continues until all component operating and failure modes, termed **basic events**, are encountered and included in the fault tree. These basic events must be independent. This process of fault tree construction and subsequent analysis is discussed in detail in [4], [5] and [6]. A fault tree is concerned with only one system failure mode. For this reason the assessment of a system may require the construction of a number of fault trees.

A fault tree diagram contains two basic elements, **gates** and **events**. Events are classified as either intermediate or basic. Intermediate events, which can be further developed in terms of other events, are represented in fault trees by rectangles. Basic events, which cannot be developed any further, are represented by circles. These symbols are shown in Table 2.2.1. Gates allow or inhibit the flow of logic through the fault tree and define the causal relationship between lower level events needed for the occurrence of a higher level event. The symbols used to represent these causal

relationships are shown in Table 2.2.2. The three main gate types used during the construction of a fault tree are the **OR** gate, the **AND** gate and the **NOT** gate. These allow the combination of events in exactly the same way as the Boolean operations of **union, intersection** and **complementation** respectively. Other common gate types are the **priority AND** gate, whose output will occur if all of the inputs occur in order from left to right and the ***k/n* voting** gate, for which the output occurs if at least k out of the n inputs occur.

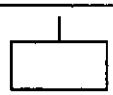
Event symbol	Meaning of symbol
	Intermediate event further developed by a gate
	Basic event

Table 2.2.1. Event symbols.

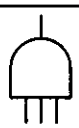
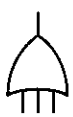
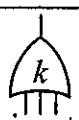
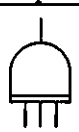
Gate symbol	Gate Name	Causal Relation
	**AND** gate	Output event occurs if all inputs occur simultaneously.
	**OR** gate	Output event occurs if at least one of the input events occurs.
	NOT gate	Output event occurs if the input event does not occur.
 *n* inputs	***k/n* voting** gate	Output event occurs if at least k of the n inputs occur.
	**priority AND** gate	Output event occurs if the input events occur in order from left to right.

Table 2.2.2. Common gate types and their symbols.

In a **coherent** fault tree the constituent gates are restricted to only AND and OR gates, which means the system's failure modes are described purely in terms of component failures. A **non-coherent** fault tree will also contain, in addition to the gates present in

coherent fault trees, inverse gates such as NOT gates. This means that a non-coherent fault tree will have a system failure mode described not only by component failure modes, but also by working states.

Analysis of a fault tree involves determining information about the system being analysed by utilising information about probabilities of basic events within the fault tree. In this way system information is obtained through an analysis of component information. Two types of analysis may be performed on the fault tree, **qualitative** and **quantitative**, discussions of which follow in sections 2.3 and 2.4.

2.3 Qualitative Analysis

2.3.1 Cut Sets and Implicants

The aim of qualitative fault tree analysis is to identify the sets of basic events that, if they occur in combination, will bring about the occurrence of the top event, i.e. cause system failure. These sets of basic events are the **cut sets** of the fault tree. A cut set is defined as a group of basic events such that if they all occur then so does the top event.

The fault tree analysis of industrial engineering systems can lead to large numbers of cut sets due to the complex nature and size of the system being analysed. However, it is not necessarily the case that all basic events within a cut set are required to fail in order to produce system failure. In order to illustrate this, consider a cut set of basic events $\{A, B, C\}$, which represent the failure within a system of components A, B and C. Because $\{A, B, C\}$ forms a cut set, if all three components fail then the top event will occur and hence the system fails. However, if the failure of components A and B alone are capable of failing the system then the state of component C is irrelevant. Whether C is failed or not, the system will fail if both A and B fail. This leads to the concept of a **minimal cut set,** which is defined as a cut set such that if any basic event is removed from it the top event will no longer occur.

The above definitions of a cut set and a minimal cut set apply only to coherent fault trees. In the case of non-coherent fault trees the combinations of component failed and working states that cause system failure are called **implicants** (the non-coherent

equivalent of cut sets) and the minimal sets of these implicants are called **prime implicants**.

In general, lower order minimal cut sets contribute more to system failure than higher order minimal cut sets because fewer components are required to fail in order for the system to fail. Because of this it is usual for greater effort to be directed towards eliminating lower order minimal cut sets than higher order minimal cut sets. In particular, first order minimal cut sets should be eliminated in preference to any others, since these represent system failure due to the failure of a single component. Fault trees of particular system failure modes are not unique. Two fault trees constructed using different methods are said to be **logically equivalent** if their minimal cut sets are identical.

2.3.2 Obtaining Minimal Cut Sets

Either of two methods can be used in order to extract the minimal cut sets from a fault tree. Both require the expansion of the fault tree's Boolean expression into a disjunctive normal form. These two methods are the **top-down** approach and the **bottom-up** approach [5]. The difference between the approaches, as the names suggest, is which end of the tree is used to initiate the expansion process. The top-down approach is described below.

The initial step is to express the top event of the fault tree in terms of its inputs using Boolean algebra. Next, the laws of Boolean algebra, given in Table 2.3.1, are used to expand and simplify the expression, if this is possible. This process of expansion and simplification continues down the tree, substituting the Boolean expression for each gate in the tree until the full expansion contains only basic component failures. The top-down approach gets its name from the fact that the process begins at the top of the fault tree and continues down through the fault tree. As its name suggests, the bottom-up approach involves starting at the bottom of the fault tree and working up to the top event. The same expansion and simplification techniques are applied. The product, '.', is used to represent AND gates in the logic equations and the sum, '+', is used to represent OR gates.

1. Commutative Laws	$A+B=B+A$ $A.B=B.A$
2. Associative Laws	$(A+B)+C=A+(B+C)$ $(A.B).C=A.(B.C)$
3. Distributive Laws	$A+(B.C)=(A+B).(A+C)$ $A.(B+C)=(A.B)+(A.C)$
4. Identities	$A+0=A$ $\quad$ $A.0=0$ $A.1=A$ $\quad$ $A+1=1$
5. Idempotent Laws	$A+A=A$ $A.A=A$
6. Absorption Laws	$A+A.B=A$ $A.(A+B)=A$
7. Complementation	$\overline{A}=1-A$ $\quad$ $\overline{(\overline{A})}=A$ $A.\overline{A}=0$
8. De Morgan's Laws	$\overline{(A+B)}=\overline{A}.\overline{B}$ $\overline{(A.B)}=\overline{A}+\overline{B}$

Table 2.3.1. The laws of Boolean algebra.

2.3.2.1 Example

The top-down approach for finding the minimal cut sets of a fault tree is now demonstrated on the fault tree shown in Figure 2.3.1.

Using Boolean algebra to express the top event in terms of its inputs gives

$$\text{TOP} = \text{G1} + A + \text{G2},$$

which cannot be simplified in any way using the Boolean laws. Next the Boolean expressions for G1 and G2 are substituted into this equation to give

$$\text{TOP} = \text{G3}.B + A + C.\text{G4},$$

which again cannot be simplified. Similarly the Boolean expressions for G3 and G4 are substituted into the equation to give

$$\text{TOP} = (A + C).B + A + C.(A.B)$$

which, after use of an associative law, a distributive law and the commutative laws, yields the following equation for the top event:

$$\text{TOP} = A + A.B + B.C + B.C.A$$

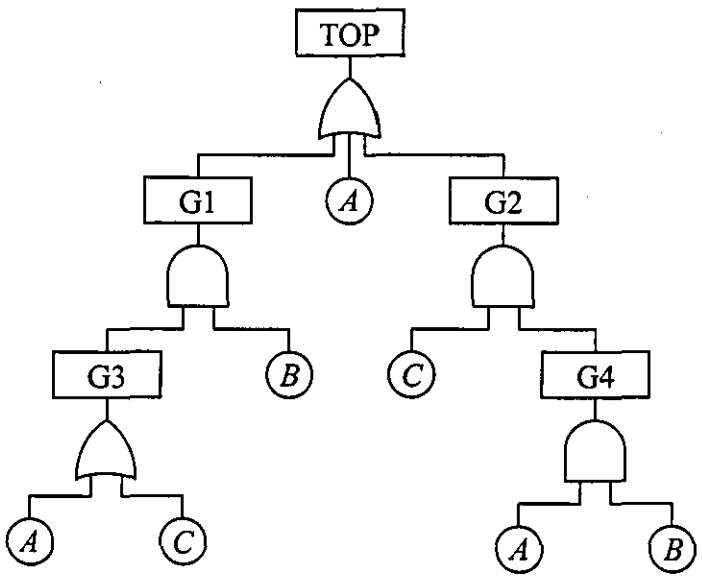


Figure 2.3.1. Example fault tree.

This expression gives the cut sets in their disjunctive normal form. Redundancies may then be removed using the absorption law. This gives

$$\text{TOP} = A + B.C,$$

which is the minimal disjunctive form of the logic equation, the terms of which are the minimal cut sets of the system. Thus there are two minimal cut sets for this system, one first order and one second order. These are $\{A\}$ and $\{B, C\}$ respectively.

Finding the minimal cut sets of a larger fault tree produced from a more complex system can be a time consuming task. In practice algorithms exist which enable the calculation of minimal cut sets to be carried out on computer. An example of one of these algorithms is MOCUS [7], the mechanics of which are described in [4] and [5]. Despite the low complexity of the algorithms the process can be extremely time-consuming. For this reason high order cut sets may be removed during the calculation process, in order that the number of computations required to determine the minimal cut sets is reduced. Approximations such as this will clearly lead to a reduction in the

accuracy of the final minimal cut sets obtained and this will subsequently have an effect on the results obtained from any ensuing quantitative analysis.

2.4 Quantitative Analysis

Quantitative analysis of the fault tree enables the calculation of a wide range of system parameters and performance measures, including the top event probability and frequency. These may be used alongside quantities such as the expected number of occurrences of the top event and event and cut set importance measures in order to gain a better knowledge and understanding of the system and how it could be improved.

2.4.1 Top Event Probability

The exact top event probability (i.e. system unavailability) may be determined by using the minimal cut sets obtained during the qualitative analysis of the fault tree. The method implemented in order to do this is the **inclusion-exclusion expansion** [5]. This is a general method that gives the correct results irrespective of whether or not there is repetition of basic events within the fault tree. However, it will only ever give the correct results if the assumption of basic event independence is true.

Consider a fault tree with N_C minimal cut sets C_i, $i = 1,\ldots, N_C$. The top event of this fault tree exists if at least one minimal cut set exists, i.e.

$$\begin{aligned} TOP &= C_1 + C_2 + \ldots + C_{N_C} \\ &= \bigcup_{i=1}^{N_C} C_i, \end{aligned} \qquad 2.4.1$$

and therefore the top event probability is given by

$$Q_{SYS}(t) = P\left(\bigcup_{i=1}^{N_C} C_i\right). \qquad 2.4.2$$

Expanding this gives

$$Q_{SYS}(t) = \sum_{i=1}^{N_C} P(C_i) - \sum_{i=2}^{N_C}\sum_{j=1}^{i-1} P(C_i \cap C_j) + \cdots + (-1)^{N_C-1} P(C_1 \cap C_2 \cap \cdots \cap C_{N_C}), \qquad 2.4.3$$

where $P(C_i)$ is the probability that minimal cut set i exists. This expansion is the inclusion-exclusion expansion. This name derives from the fact that the terms within it are alternately added and subtracted.

Due to the nature of the inclusion-exclusion expansion the calculation of the top event probability is an arduous task if a fault tree contains a large number of minimal cut sets, even with the aid of a powerful computer. For this reason the inclusion-exclusion expansion is used to obtain upper and lower bounds for the system unavailability [5].

2.4.1.1 Upper and Lower Bounds for System Unavailability

As can be seen from Equation 2.4.3, the odd-numbered terms from the expansion are always added, and the even-numbered terms are always subtracted, when calculating the system unavailability. Each term produces a numerically less significant contribution to the final result than the one that preceded it. These properties may be utilised to calculate upper and lower bounds for the system unavailability. Considering only the first two terms in the expansion the following inequality can be derived [5]:

$$\underbrace{\sum_{i=1}^{N_C} P(C_i) - \sum_{i=2}^{N_C}\sum_{j=1}^{i-1} P(C_i \cap C_j)}_{\text{lower bound}} \leq \underbrace{Q_{SYS}(t)}_{\text{exact}} \leq \underbrace{\sum_{i=1}^{N_C} P(C_i)}_{\text{upper bound}}, \qquad 2.4.4$$

the left hand side of which contains the first two terms of the inclusion-exclusion expansion and the right hand side of which contains just the first term. The upper bound is known as the **rare event approximation,** Q_{RE}, due to the fact that it is accurate if component failures are rare.

$$Q_{RE} = \sum_{i=1}^{N_C} P(C_i). \qquad 2.4.5$$

2.4.1.2 Minimal Cut Set Upper Bound

A more accurate approximation for the top event unavailability is the **minimal cut set upper bound** [5], which is derived below. Firstly,

$$\begin{aligned} P(\text{system failure}) &= P(\text{at least one minimal cut set occurs}) \\ &= 1 - P(\text{no minimal cut set occurs}). \end{aligned} \qquad 2.4.6$$

Also,

$$P(\text{no minimal cut set occurs}) \geq \prod_{i=1}^{N_C} P(\text{min. cut set } i \text{ doesn't occur}) \qquad 2.4.7$$

(equality occurs when no event appears in more than one minimal cut set). Substituting this equation into Equation 2.4.6 gives

$$P(\text{system failure}) \leq 1 - \prod_{i=1}^{N_C} P(\text{min. cut set } i \text{ doesn't occur}), \qquad 2.4.8$$

or

$$Q_{SYS}(t) \leq 1 - \prod_{i=1}^{N_C} [1 - P(C_i)], \qquad 2.4.9$$

the right hand side of which is the **minimal cut set upper bound,** Q_{MCSU}, which is hence given by

$$Q_{MCSU} = 1 - \prod_{i=1}^{N_C} [1 - P(C_i)], \qquad 2.4.10$$

It can also be shown that

$$\underset{\text{exact}}{Q_{SYS}(t)} \leq \underset{\text{min. cut set upper bound}}{1 - \prod_{i=1}^{N_C} [1 - P(C_i)]} \leq \underset{\text{rare event approximation}}{\sum_{i=1}^{N_C} P(C_i)}. \qquad 2.4.11$$

2.4.2 Minimal Cut Set Unconditional Failure Intensity

The **minimal cut set unconditional failure intensity,** $w_{C_i}(t)$, of a minimal cut set containing n_i components is given by:

$$w_{C_i}(t) = \sum_{\substack{j=1 \\ j \in C_i}}^{n_i} \left(w_j(t) \prod_{\substack{k=1 \\ k \neq j \\ k \in C_i}}^{n_i} q_k(t) \right), \qquad 2.4.12$$

where n_i is the number of components in minimal cut set i. It is defined as the probability that the minimal cut set will occur at time t per unit time [5]. Thus the probability that the minimal cut set occurs in time $[t, t + dt)$ is given by $w_{C_i}(t)dt$.

2.4.3 System Unconditional Failure Intensity

The **system unconditional failure intensity** or **top event frequency**, $w_{SYS}(t)$, of a system of n components is given by:

$$w_{SYS}(t) = \sum_{i=1}^{n} G_i(q(t))w_i(t), \qquad 2.4.13$$

and is defined as the probability that the top event occurs at t per unit time [5]. Thus the probability that the top event occurs in the interval $[t, t + dt)$ is given by $w_{SYS}(t)dt$. The first term in the summation,

$$G_i(q(t)) = Q_{SYS}(1_i, q(t)) - Q_{SYS}(0_i, q(t)), \qquad 2.4.14$$

is the criticality function, where $Q_{SYS}(t)$ is the system unavailability and

$$(1_i, q(t)) = (q_1, \ldots, q_{i-1}, 1, q_{i+1}, \ldots q_n),$$
$$(0_i, q(t)) = (q_1, \ldots, q_{i-1}, 0, q_{i+1}, \ldots q_n).$$

The criticality function gives the probability that the system will go from a working to a failed state due to the failure of component i, i.e. that the system is in a critical state for i. This is because, as can be seen from 2.4.14, the probability that the system fails with i working is subtracted from the probability that the system fails with i failed.

2.4.3.1 Example

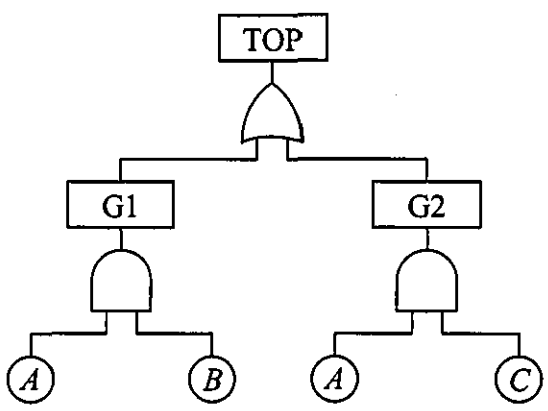


Figure 2.4.1. A simple system fault tree.

Consider the simple system fault tree shown in Figure 2.4.1, with three basic events, A, B and C, which have failure probabilities of q_A, q_B and q_C and unconditional failure intensities w_A, w_B and w_C respectively. When finding the system unconditional failure intensity the first step is to calculate the probability that the system fails, Q_{SYS}. This could be obtained by considering the structure function of the system and then using

pivotal decomposition or by using the inclusion-exclusion expansion, both of which yield the following:

$$Q_{SYS} = q_A q_B + q_A q_C - q_A q_B q_C.$$

This can now be used together with 2.4.14 to find the criticality function for each component. For component A the criticality function will therefore be:

$$\begin{aligned} G_A &= Q_{SYS}(1_A, q) - Q_{SYS}(0_A, q) \\ &= [q_B + q_C - q_B q_C] - [0] \\ G_A &= q_B + q_C - q_B q_C. \end{aligned}$$

In a similar way the criticality functions for components B and C can be shown to be:

$$\begin{aligned} G_B &= q_A, \\ G_C &= q_A. \end{aligned}$$

Now from 2.4.13 the unconditional failure intensity will be given by:

$$w_{SYS} = G_A w_A + G_B w_B + G_C w_C,$$

which, upon substitution of the criticality functions, G_A, G_B and G_C, becomes:

$$w_{SYS} = (q_B + q_C - q_B q_C) w_A + q_A w_B + q_A w_C.$$

2.4.3.2 Expected Number of System Failures

The expected number of system failures in the time interval t_0 to t_1, $W(t_0, t_1)$, is found by integrating the system unconditional failure intensity over the interval t_0 to t_1.

$$W(t_0, t_1) = \int_{t_0}^{t_1} w_{SYS}(u) du. \qquad 2.4.15$$

2.4.3.3 Initiating and Enabling Events

The previous calculation of the system failure intensity assumes that the order in which basic events occur in any minimal cut set is unimportant. However, in some cases, this assumption is wrong. This is especially true in the case of safety protection systems. For example, consider a hazardous event that has an associated protection system. If the protection system fails before the hazardous event then system failure will result. However, if the hazardous event occurs first, the protection system will come into

action and prevent system failure. This type of situation is modelled by considering failures as either **initiating** or **enabling** events [5]. In this case the protection system failure is the enabling event and the hazardous failure is the initiating event. An initiating event is one that perturbs system variables and places a demand on control/system variables to respond. An enabling event is an inactive control/protection system that permits initiating events to cause the top event.

To calculate the system unconditional failure intensity in the case when initiating and enabling events are present within the system consider 2.4.13. The enabling events put the system in a critical state for the initiating events, which will then fail and consequently fail the system. Thus the summation in 2.4.13 is taken over the initiating events only which means that, when considering initiating and enabling events, the system unconditional failure intensity is given by:

$$w_{SYS}(t) = \sum_{\substack{i \\ initiator}} G_i(q(t))w_i(t). \qquad 2.4.16$$

To illustrate the use of this failure, consider again the system shown in Figure 2.4.1. If B and C are enabling events and A is an initiating event the unconditional failure intensity of the system will be given by:

$$\begin{aligned} w_{SYS} &= G_A w_A \\ &= (q_B + q_C - q_B q_C) w_A. \end{aligned}$$

2.4.4 Importance Measures

When analysing a system it is important to identify where the weak points of a system lie and where effort should be directed in any attempt to improve the performance of the system. Importance measures provide a means to find the system weak points by ranking the system's components or cut sets according to the extent of their role in the failure of the system. Importance measures are of particular use during system design processes when it is possible to identify and attempt to rectify system weaknesses.

Importance measures fall into either of two categories, **deterministic** or **probabilistic.** The probabilistic measures can be split into two groups, those which deal with system

unavailability assessment (top event probability) and those which deal with system unreliability assessment (expected number of top event occurrences).

2.4.4.1 Deterministic Measures

Deterministic measures give a measure of the importance of the component to the system without considering the component's probability of failure. One such deterministic measure is the structural measure of importance.

Structural Measure of Importance

The structural measure of importance for component i is

$$I = \frac{\text{number of critical system states for component } i}{\text{total number of states for the } (n-1) \text{ remaining components}}, \qquad 2.4.17$$

where a **critical system state** for component i is a state for the remaining $n - 1$ components such that failure of component i causes the system to switch from a working to a failed state.

2.4.4.2 Probabilistic Measures for System Unavailability

Probabilistic measures are generally of more use than deterministic measures due to the fact that the component failure probabilities are taken into account.

Birnbaum's Measure of Importance

This is also known as the criticality function, see 2.4.14, which is defined as the probability that the system is in a critical state for component i. Two expressions exist for the criticality function. The first of these is given in 2.4.14 and is:

$$G_i(q(t)) = Q_{SYS}(1_i, q(t)) - Q_{SYS}(0_i, q(t)), \qquad 2.4.18$$

where $Q_{SYS}(t)$ is the probability that the system fails and

$$(1_i, q(t)) = (q_1, \ldots, q_{i-1}, 1, q_{i+1}, \ldots q_n),$$
$$(0_i, q(t)) = (q_1, \ldots, q_{i-1}, 0, q_{i+1}, \ldots q_n).$$

The criticality function is thus the probability that the system fails with component i failed minus the probability that the system fails with component i working, and results in the probability that the system fails only if component i fails.

The second expression for the criticality function is

$$G_i(q(t)) = \frac{\partial Q_{SYS}(q(t))}{\partial q_i}, \qquad 2.4.19$$

which is equal to the expression in 2.4.18 since Q_{SYS} is linear in each $q_i(t)$ and so:

$$\frac{\partial Q_{SYS}(q(t))}{\partial q_i} = \frac{Q_{SYS}(1_i, q(t)) - Q_{SYS}(0_i, q(t))}{1-0}. \qquad 2.4.20$$

The criticality function is used to define many other importance measures.

Criticality Measure of Importance

This calculates the probability that the system is in a critical state for component i and i has failed. It is given by

$$I_i = \frac{G_i(q(t))q_i(t)}{Q_{SYS}(q(t))}. \qquad 2.4.21$$

Fussell-Vesely Measure of Importance

This importance measure calculates the probability that component i contributes to system failure by appearing in one or more of the minimal cut sets [5]. It is defined as the probability of the union of the minimal cut sets, k, containing i, given that the system has failed, i.e.

$$I_i = \frac{P\left(\bigcup_{k|i\in k} C_k\right)}{Q_{SYS}(q(t))}. \qquad 2.4.22$$

The rankings produced by this method relate closely to the rankings given by the criticality measure.

Fussell-Vesely Measure of Minimal Cut Set Importance

In the same way that the performance measures discussed so far rank the component failures in terms of their contribution to failure of the system, this measure ranks the importance of the minimal cut sets [5]. It is defined as the probability of occurrence of minimal cut set *i* given that the system has failed,

$$I_{C_i} = \frac{P(K_i)}{Q_{SYS}(q(t))}. \qquad 2.4.23$$

2.5 Summary of Fault Tree Analysis

Fault trees are a clear and concise way of representing the failure logic of a system and methods exist to analyse the fault tree both qualitatively and quantitatively. However, there are two key drawbacks to the analysis of fault trees, both of which would impact on the potential fault tree analysis of the application of TLD to a system. The first of these drawbacks is the system size in the analysis of fault trees, since both quantitative and qualitative methods can require a substantial amount of computational time to complete as system size increases. This then introduces the need to use approximations, which, in turn lead to some loss of accuracy in the analysis. The second drawback in applying fault tree analysis to systems is the requirement that the basic events within the fault tree must be statistically independent. Amongst other things, when one considers a system to which TLD is applied, aircraft operating procedures may introduce some dependencies. Maintenance and repair introduce more dependencies and, as such make fault tree analysis unsuitable. The next chapter discusses a technique that, unlike fault tree analysis, is capable of dealing with dependencies within a system. This technique is Markov analysis.

Chapter 3 : Markov Analysis of Systems

3.1 Background

The basic concepts of Markov Analysis were developed by the Russian mathematician, A. A. Markov, in the early 20th century [9]. The approach may be applied to stochastic processes, i.e. systems that vary, discretely or continuously, in a random way with respect to time. These processes are known as **Markov processes**. Markov processes for which time varies discretely, rather than continuously, are known as **Markov chains** [9], [10] and will not be discussed in this chapter. Problems involving the reliability and availability of systems are generally discrete in space and continuous in time [11]. This means that the systems (and their components) may exist in one of a number of discrete states, e.g. failed or working, and that transitions between these states may occur at any point in time. The basic requirements of the system that must be met in order to be able to apply the Markov technique [5] are:

- **Lack of memory in the system**; future states of the system are independent of all previous system states except the immediately preceding one.
- **Stationary system**; the system must behave in the same way irrespective of the current time, i.e. the probability of the system making the transition from one state to another is constant at all points in time.
- **Identifiable system states**; there should be mutually exclusive discrete states that the system can assume, all of which can be identified.

Due to these requirements the Markov approach is applicable to systems whose behaviour is modelled using exponential distributions characterised by constant failure and repair rates. If the system is not stationary, i.e. the transition probabilities are functions of time, then the system is known as **non-Markovian** [5].

Crucially, the technique of Markov analysis does not require the condition of independence between component failures, which gives an advantage over a combinatorial technique such as fault tree analysis.

3.2 Construction of the Markov Model

Markov models assume that the time taken for transitions between states is instantaneous and that an element of time, dt, is sufficiently small such that only one transition may occur in dt. The Markov model is then constructed as follows:

- A list of all possible mutually exclusive system states is composed.
- The transition rates between system states are determined. These are usually failure and repair rates of system components.
- A state transition diagram is drawn. This takes the form of a directed graph within which each node represents one of the discrete system states and transition rates and directions are represented on arrows drawn on the edges between the states.

3.2.1 Example – A Two-State Repairable Component

Consider a repairable component that may exist in either one of two states – working or failed. The transition diagram for this component is shown in Figure 3.2.1. If the component is in the working state, 1, transition to the failed state, 2, occurs with constant rate λ and if it is in state 2, transition to state 1 occurs with constant rate ν. Therefore the probability of the component being in the failed state at $t + dt$ given that it worked at t is λdt. Similarly, the probability of the component working at $t + dt$ given that it was failed at t is νdt. The probability of remaining in state 1 is $1 - \lambda dt$, and the

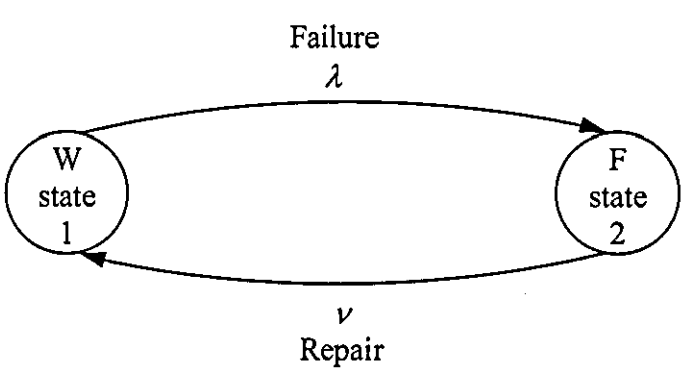


Figure 3.2.1. Transition diagram for a two-state repairable component.

probability of remaining in state 2 is $1 - \nu dt$. As previously stated, dt is sufficiently small that only one transition may occur in the dt. Now define

$$P_w(t) = \text{probability that the component is working at time } t, \qquad 3.2.1$$

and

$$P_f(t) = \text{probability that the component is failed at time } t, \qquad 3.2.2$$

and consider the probability that the component exists in the failed state (i.e. the unavailability) of the component after time increment dt,

$$P_f(t+dt) = P\begin{bmatrix} \text{the component works at time } t \text{ and fails in } dt \\ \text{OR} \\ \text{the component is failed at time } t \text{ and remains failed in } dt \end{bmatrix} \qquad 3.2.3$$

$$= P_w(t)\lambda dt + P_f(t)[1 - \nu dt],$$

which, upon rearrangement, gives

$$\frac{P_f(t+dt) - P_f(t)}{dt} = P_w(t)\lambda - P_f(t)\nu. \qquad 3.2.4$$

Now, in the limit as $dt \to 0$,

$$\left.\frac{P_f(t+dt) - P_f(t)}{dt}\right|_{dt\to 0} = \frac{dP_f(t)}{dt}, \qquad 3.2.5$$

and substituting this into 3.2.4 gives

$$\frac{dP_f(t)}{dt} = P_w(t)\lambda - P_f(t)\nu, \qquad 3.2.6$$

and, since $P_w(t) + P_f(t) = 1$, this becomes

$$\frac{dP_f(t)}{dt} = \lambda - (\lambda + \nu)P_f(t), \qquad 3.2.7$$

which may be solved with the initial condition that the component is working, i.e. $P_f(0) = 0$, to yield the unavailability, or the probability that the component is in a failed state at time t:

$$P_f(t) = \frac{\lambda}{\lambda+\nu} - \frac{\lambda}{\lambda+\nu}e^{-(\lambda+\nu)t}. \qquad 3.2.8$$

In a similar way it may be shown that

$$\frac{dP_w(t)}{dt} = P_f(t)\nu - P_w(t)\lambda, \qquad 3.2.9$$

and since $P_w(t) + P_f(t) = 1$, this becomes

$$\frac{dP_w(t)}{dt} = \nu - (\lambda + \nu)P_w(t), \qquad 3.2.10$$

which is solved with the same initial condition, $P_w(0) = 1$, to give the availability, or the probability that the component is in a working state at time t,

$$P_w(t) = \frac{\nu}{\lambda+\nu} - \frac{\lambda}{\lambda+\nu} e^{-(\lambda+\nu)t}. \qquad 3.2.11$$

3.3 Matrix Representation of the Model

Consider a general system for which the Markov model will have n system states. If these are numbered from 1 to n then the first m, numbered 1 to m, represent the system working states and the states $m + 1$ to n represent the system failed states. a_{ij} is defined as the transition rate from i to j, thus:

$$P(\text{system in state } j \text{ at } t+dt \mid \text{system in state } i \text{ at } t) = a_{ij}dt. \qquad 3.3.1$$

If $Q_i(t)$ denotes the system being in failed state i at time t the above scenarios can be represented as follows:

$$\begin{aligned} Q_i(t+dt) &= P(\text{no transition from state } i \text{ after } dt) \\ &\quad + P(\text{transition from state } j \text{ to state } i \text{ after } dt) \\ &= Q_i(t)\left[1 - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}dt\right] + \sum_{\substack{j=1 \\ j\neq i}}^{n} Q_j(t)a_{ji}dt. \end{aligned} \qquad 3.3.2$$

Rearranging this gives

$$\frac{Q_i(t+dt) - Q_i(t)}{dt} = \sum_{\substack{j=1 \\ j\neq i}}^{n} Q_j(t)a_{ji} - \sum_{\substack{j=1 \\ j\neq i}}^{n} Q_i(t)a_{ij}. \qquad 3.3.3$$

In the limit as $dt \to 0$ equation 3.3.3 becomes

$$\frac{dQ_i(t)}{dt} = \sum_{\substack{j=1 \\ j\neq i}}^{n} Q_j(t)a_{ji} - \sum_{\substack{j=1 \\ j\neq i}}^{n} Q_i(t)a_{ij}. \qquad 3.3.4$$

Let

$$a_{ii} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}, \qquad 3.3.5$$

and substitute this into 3.3.4 to give

$$\frac{dQ_i(t)}{dt} = \sum_{j=1}^{n} Q_j(t)a_{ji}dt. \qquad 3.3.6$$

This may be written in matrix form as:

$$\dot{\mathbf{Q}} = \mathbf{QA}, \qquad 3.3.7$$

where

$$\begin{aligned} \mathbf{Q} &= [Q_1(t), Q_2(t), \ldots, Q_n(t)], \\ \dot{\mathbf{Q}} &= [\dot{Q}_1(t), \dot{Q}_2(t), \ldots, \dot{Q}_n(t)], \end{aligned} \qquad 3.3.8$$

and

$$\mathbf{A} = \begin{cases} a_{ij}, \text{ the transition rate from } i \text{ to } j, \\ a_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}. \end{cases} \qquad 3.3.9$$

The square matrix **A** is called the **state transition matrix** and may be directly obtained from the state transition diagram for the model. When it is constructed there are three rules that should be followed:

- The dimension of matrix **A** is equal to the number of states in the system.
- All of the rows should sum to zero.
- Off-diagonal elements in row i, column j represent the rate of transition from state i to state j.

The equations given in 3.3.7 are dependent, which means that before they can be solved further information is needed. This information comes from the fact that the probability of being in any of the system states must add up to 1, i.e.

$$\sum_{i=1}^{n} Q_i(t) = 1. \qquad 3.3.10$$

3.3.1 Example – State Transition Matrix for The Repairable Component

Applying the above rules to the two-state repairable component depicted in Figure 3.2.1 the state transition matrix is found to be:

$$\mathbf{A} = \begin{bmatrix} -\lambda & \lambda \\ \nu & -\nu \end{bmatrix}, \qquad 3.3.11$$

which means that:

$$[\dot{P}_w(t) \quad \dot{P}_f(t)] = [P_w(t) \quad P_f(t)] \begin{bmatrix} -\lambda & \lambda \\ \nu & -\nu \end{bmatrix}, \qquad 3.3.12$$

which is equivalent to equations 3.2.6 and 3.2.9.

3.4 Markov Model Solution

It can be seen from the previous sections that a Markov model will produce a system of homogeneous linear ordinary differential equations where the system may be represented in matrix form as:

$$\dot{\mathbf{Q}} = \mathbf{QA},$$

Such systems of differential equations may be solved using a number of different methods, which may be either analytical or numerical.

Once the probability of residing in each system state is known, finding the system availability, $A(t)$, involves summing the probabilities of all the individual states representing the system being in a working state, i.e.

$$A(t) = \sum_{i=1}^{m} Q_i(t), \qquad 3.4.1$$

and the system unavailability, $Q(t)$, involves summing the probabilities of all the individual states that represent the system being in a failed state, i.e.

$$Q(t) = \sum_{i=m+1}^{n} Q_i(t). \qquad 3.4.2$$

3.4.1 Analytical Solution

A number of analytical methods exist that may be used to find a solution to a set of ordinary differential equations. One such method uses Laplace transforms [5], [12], which are used to convert the system of differential equations into a system of algebraic equations. These algebraic equations may then be solved. Once the solution is found the inverse Laplace transform of it is taken in order to obtain the solution of the original system of differential equations. This procedure is outlined below.

The system of differential equations may be written as:

$$\dot{\mathbf{Q}}^T = \mathbf{A}^T\mathbf{Q}^T, \qquad 3.4.3$$

or

$$\begin{bmatrix} \dot{Q}_1(t) \\ \dot{Q}_2(t) \\ \vdots \\ \dot{Q}_n(t) \end{bmatrix} = \mathbf{A}^T \begin{bmatrix} Q_1(t) \\ Q_2(t) \\ \vdots \\ Q_n(t) \end{bmatrix}. \qquad 3.4.4$$

If the Laplace transform of Q_i is

$$\mathcal{L}\left[Q_i(t)\right] = Q_i^*, \qquad 3.4.5$$

then the Laplace transform of $\dot{Q}_i(t)$ is

$$\mathcal{L}\left[\dot{Q}_i(t)\right] = sQ_i^* - Q_i(0), \qquad 3.4.6$$

and the Laplace transform of equation 3.4.4 becomes:

$$s\begin{bmatrix} Q_1^* \\ Q_2^* \\ \vdots \\ Q_n^* \end{bmatrix} - \begin{bmatrix} Q_1(0) \\ Q_2(0) \\ \vdots \\ Q_n(0) \end{bmatrix} = \mathbf{A}^T \begin{bmatrix} Q_1^* \\ Q_2^* \\ \vdots \\ Q_n^* \end{bmatrix}. \qquad 3.4.7$$

Then the Laplace transform of equation 3.3.10,

$$s\sum_{i=1}^{n} Q_i^* = 1, \qquad 3.4.8$$

is substituted into equation 3.4.7, allowing the solution of the equations and their subsequent inverse Laplace transformation.

Another method that may be used to find the analytical solution involves determining the eigenvalues of the state transition matrix **A**. However, if the system being modelled is very reliable the precision offered by this method of solution can be dissatisfactory [13].

3.4.2 Numerical Solution

In addition to finding the solution of Markov models analytically it is also possible to solve the differential equations that describe the system numerically. Many techniques exist, such as Euler's method and its variations, predictor-corrector methods or Runge-Kutta methods [14]. These methods are of particular use when one wishes to examine the transient behaviour of the system. Numerical methods of solution may also offer a greater flexibility than analytical methods. One situation where they offer an advantage is when periodic inspection is used to find component failures. These failures lie

dormant within the system and are only revealed as the system is inspected. Numerical methods of solution are discussed in Chapter 7. A problem that may occur when numerically solving systems of differential equations such as these is that of stiffness. When solving Markov models stiffness can arise because of the difference in magnitude between the failure rates and the repair rates [5], [14], [15].

3.5 Steady-State Probabilities

A process whose limiting probabilities are independent of its initial conditions is known as **ergodic** [5]. For this to be true it must be possible to reach every state in the system from every other state in the system. This need not be directly from one state to another, it could be that states are reached indirectly through other, intermediate, states. An ergodic Markov process, such as the one in the previous example of a two state component, will have non-zero limiting probabilities [5], [11].

3.5.1 Example – Repairable Component

Consider again the previous example of a two-state repairable component depicted in Figure 3.2.1. The probability of this component being in the failed state was given in equation 3.2.8. By allowing time to become large in this equation, i.e. $t \to \infty$, the limiting value for the unavailability becomes

$$P_f = \frac{\lambda}{\lambda + \nu}, \qquad 3.5.1$$

and in the same way, using equation 3.2.11, the limiting value for the availability becomes

$$P_w = \frac{\nu}{\lambda + \nu}. \qquad 3.5.2$$

These steady-state values could have been obtained directly from the differential equations 3.2.6 and 3.2.9 by utilising the fact that at steady-state the rate of change of the probabilities will be zero, i.e.

$$\frac{dP_w(t)}{dt} = \frac{dP_f(t)}{dt} = 0. \qquad 3.5.3$$

Substituting 3.5.3 into 3.2.6 and 3.2.9 gives

$$-\lambda P_w(t)+\nu P_f(t)=0, \qquad 3.5.4$$

and

$$\lambda P_w(t)-\nu P_f(t)=0, \qquad 3.5.5$$

which are, in fact, exactly the same equation. The fact that $P_w(t) + P_f(t) = 1$ is used in order to solve the system of equations and obtain the solutions 3.5.1 and 3.5.2 above.

3.5.2 Finding the Steady-State Solution of the Markov Model

A system's limiting steady-state (or asymptotic) probabilities are determined by utilising the fact that the rate of change of the probabilities will be zero at the limit, i.e.

$$\dot{\mathbf{Q}} = \mathbf{0}, \qquad 3.5.6$$

where **0** is the null vector. Substituting this into equation 3.3.7 yields the dependent set of simultaneous linear equations given by:

$$\mathbf{QA} = \mathbf{0}. \qquad 3.5.7$$

Because of the dependency of these equations 3.3.10 is again used in order to allow a solution to be found. A technique such as Gaussian elimination may be used to determine the exact solution of the system of equations [5], [14]. However, it can be shown [13] that the steady-state asymptotic probabilities may also be obtained from:

$$Q_i(\infty)=\frac{\begin{vmatrix} a_{11} & \cdots & a_{1,n-1} & 0 \\ \vdots & & \vdots & 0 \\ a_{i,1} & \cdots & a_{i,n-1} & 1 \\ \vdots & & \vdots & 0 \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{vmatrix}}{\begin{vmatrix} a_{11} & \cdots & a_{1,n-1} & 1 \\ \vdots & & \vdots & \vdots \\ a_{i,1} & \cdots & a_{i,n-1} & 1 \\ \vdots & & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 1 \end{vmatrix}} \qquad 3.5.8$$

The numerator is the determinant of the state transition matrix, **A**, with the nth column replaced by zeroes except in the ith row, where unity is inserted. The denominator is the determinant of **A** with the last column entirely replaced by unity. Thus, using a similar method to that used when obtaining 3.4.1 and 3.4.2, the steady-state availability is

$$A(\infty)=\sum_{i=1}^{m} Q_i(\infty), \qquad 3.5.9$$

and the steady-state unavailability is

$$Q(\infty) = \sum_{i=m+1}^{n} Q_i(\infty). \qquad 3.5.10$$

The system unavailability may also be obtained using

$$Q(\infty) = \frac{\begin{vmatrix} a_{11} & \cdots & a_{1,n-1} & 0 \\ \vdots & & \vdots & 0 \\ a_{m,1} & \cdots & a_{m,n-1} & 0 \\ a_{m+1,1} & \cdots & a_{m+1,n-1} & 1 \\ \vdots & & \vdots & 1 \\ a_{n,1} & \cdots & a_{n,n-1} & 1 \end{vmatrix}}{\begin{vmatrix} a_{11} & \cdots & a_{1,n-1} & 1 \\ \vdots & & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 1 \end{vmatrix}} \qquad 3.5.11$$

3.5.3 Example –Two-Component System

Consider a system containing two components, A and B, which have failure rates λ_A and λ_B respectively, and repair rates ν_A and ν_B respectively. The Markov model for this system is depicted in Figure 3.5.1. As can be seen from the diagram the system has a total of four possible states, categorised as follows; A and B work (state 1), A is failed and B works (state 2), A works and B is failed (state 3) and A and B are both failed (state 4). Using equation 3.3.9 the transition matrix **A** can be written as:

$$\mathbf{A} = \begin{bmatrix} -(\lambda_A + \lambda_B) & \lambda_A & \lambda_B & 0 \\ \nu_A & -(\nu_A + \lambda_B) & 0 & \lambda_B \\ \nu_B & 0 & -(\lambda_A + \nu_B) & \lambda_A \\ 0 & \nu_B & \nu_A & -(\nu_A + \nu_B) \end{bmatrix}. \qquad 3.5.12$$

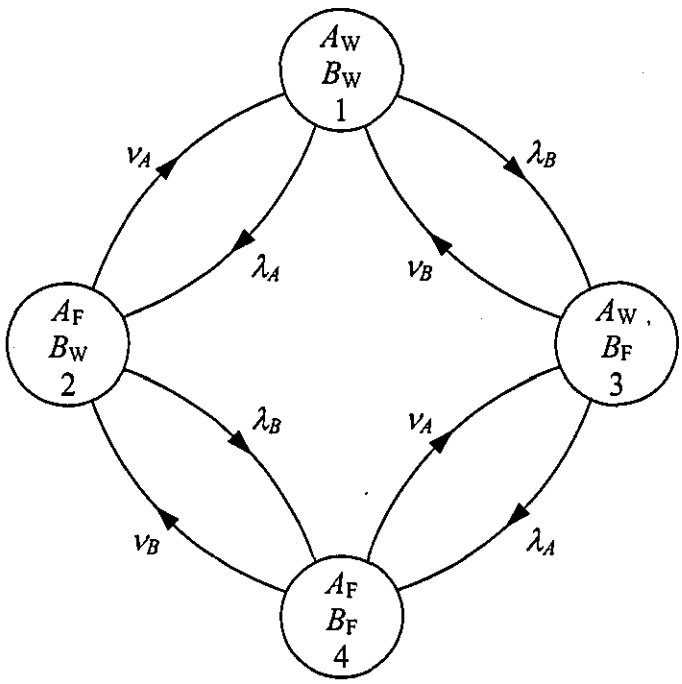


Figure 3.5.1. Two component Markov model.

In order to find the steady-state probability of residing in each of the system states the determinant method, equation 3.5.8, may be used. Take, for example, state 3. The application of equation 3.5.8 leads to the following:

$$Q_3(\infty)=\frac{\begin{vmatrix} -(\lambda_A+\lambda_B) & \lambda_A & \lambda_B & 0 \\ \nu_A & -(\nu_A+\lambda_B) & 0 & 0 \\ \nu_B & 0 & -(\lambda_A+\nu_B) & 1 \\ 0 & \nu_B & \nu_A & 0 \end{vmatrix}}{\begin{vmatrix} -(\lambda_A+\lambda_B) & \lambda_A & \lambda_B & 1 \\ \nu_A & -(\nu_A+\lambda_B) & 0 & 1 \\ \nu_B & 0 & -(\lambda_A+\nu_B) & 1 \\ 0 & \nu_B & \nu_A & 1 \end{vmatrix}}, \qquad 3.5.13$$

which is equivalent to

$$\begin{aligned} Q_3(\infty) &= \frac{-\lambda_B\nu_A(\lambda_A+\lambda_B+\nu_A+\nu_B)}{-(\lambda_A+\nu_A)(\lambda_B+\nu_B)(\lambda_A+\lambda_B+\nu_A+\nu_B)} \\ &= \frac{\lambda_B\nu_A}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)}. \end{aligned} \qquad 3.5.14$$

In a similar way the steady-state probabilities for the other system states can be determined and thus the complete set of steady-state probabilities are:

$$Q_1(\infty)=\frac{\nu_A\nu_B}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)},$$
$$Q_2(\infty)=\frac{\lambda_A\nu_B}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)},$$
$$Q_3(\infty)=\frac{\lambda_B\nu_A}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)},$$
$$Q_4(\infty)=\frac{\lambda_A\lambda_B}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)}. \qquad 3.5.15$$

In order to find the steady-state probability of system failure the probabilities of being in states where the system is failed are added. For a two-component system that is in a series configuration the system will be in a failed state if A is failed, if B is failed or if both are failed. Therefore the steady-state probability of system failure is obtained by adding Q_2, Q_3 and Q_4 and is:

$$\frac{\lambda_A\nu_B+\lambda_B\nu_A+\lambda_A\lambda_B}{(\lambda_A+\nu_A)(\lambda_B+\nu_B)}. \qquad 3.5.16$$

If the system is in a parallel configuration the steady-state probability of system failure is again obtained by adding the probabilities of being in states where the system is failed. However, in this case the system will only fail if both components fail. Therefore the steady-state probability of system failure is given by Q_4:

3.6 Reduced Markov Models

Perhaps the main drawback of using Markov methods to model system behaviour is the rapid explosion in the number of system states as the number of components in the system increases [5], [15]. For a general system with n components, each of which can exist in either one of two states, working or failed, the Markov model will have 2^n system states. Clearly, as n becomes large, the differential equations produced during the formation of the model will become increasingly difficult to solve. For this reason it is to the modeller's advantage to reduce the number of states within the system, if at all possible. If identical components exist within the system it may be possible to perform reductions on the Markov model.

3.6.1 Example –Two Component System

Consider again the two component system depicted in Figure 3.5.1. Let us assume that the two components, A and B, are arranged in a parallel configuration and that their failure and repair rates are identical. That is, assume $\lambda_A = \lambda_B = \lambda$, and $\nu_A = \nu_B = \nu$. In the original example four system states were listed. However, the system can be satisfactorily described using three states, which are:

1. 2 components work,
2. 1 component works, 1 component is failed,
3. Both components are failed.

The reduced Markov diagram for this system is shown in Figure 3.6.1. The transition rate from state 1 to state 2 is determined by considering that either component A or component B can fail to take the system from the fully-functioning state (state 1) to the state where one component is failed and one component still works (state 2). Thus the transition rate will be the sum of the failure rates of these two components, i.e. 2λ. In a similar way the transition rate from state 3 to state 2 is determined to be 2ν. The transitions from state 2 to state 3 and from state 2 to state 1 are respectively caused by the failure and repair of a single component. Thus they have transition rates λ and ν respectively.

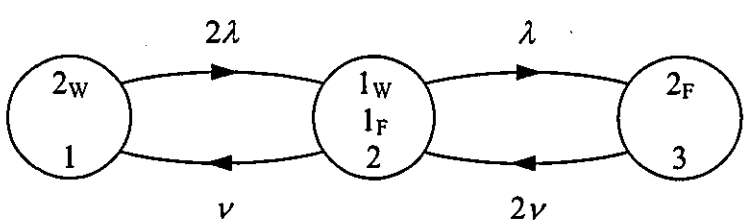


Figure 3.6.1. Three-state Markov model for the two component system.

The transition matrix, **A**, for this system may then be shown to be:

$$\mathbf{A} = \begin{bmatrix} -2\lambda & 2\lambda & 0 \\ \nu & -(\lambda+\nu) & \lambda \\ 0 & 2\nu & -2\nu \end{bmatrix}. \qquad 3.6.1$$

For this simple system the benefits of reducing the number of states in the Markov model are not very great, with only one state being removed from the full Markov

model. However, if the system is large this type of reduction may prove extremely useful in reducing the complexity and scale of the model.

3.7 Reliability Modelling

In systems such as those discussed thus far the processes involved were ergodic, i.e. every system state could be reached, either directly or indirectly from every other system state. In the case of systems that may suffer catastrophic failure, such as aircraft, the processes involved are not ergodic, since the system contains a state that, once entered, cannot be left until the system begins a new mission. These states are known as **absorbing states** [5]. In these non-ergodic systems the steady-state availability or unavailability will hold no real significance, since the steady-state availability will always be zero and the steady-state unavailability will always be unity. It is more appropriate to investigate the reliability of such systems. In order to represent a system reliability model its Markov diagram is constructed by deleting from it any transitions that correspond to a change from a failed to a working state. As an example of this consider again the two component system depicted in Figure 3.5.1. The Markov diagram representing the reliability model for this system is shown in Figure 3.7.1.

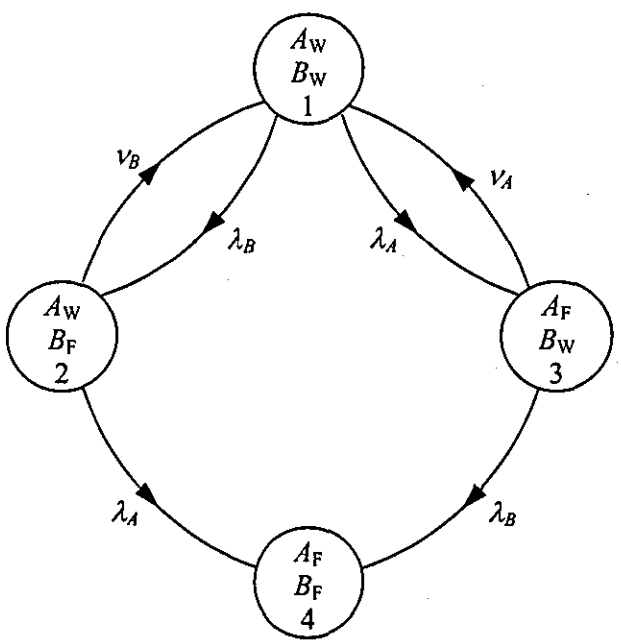


Figure 3.7.1. Reliability model for a two component system.

The equations for a reliability model are derived in a similar way to those for the ergodic processes described previously [5], [13]. Consider a system that has a total of n states and, as before, number the states that represent the system working from 1 to m and the states that represent the system being failed from $m + 1$ to n. Since all of the failed states are absorbing the transitions representing changes from these states back to a working state will be zero, i.e.

$$a_{ij} = 0 \quad \text{for} \quad i > m, j \leq m. \tag{3.7.1}$$

Representing the probability of residing in state i at time t by $P_i(t)$ allows a group of first order linear differential equations to be obtained, which are as follows:

$$\frac{dP_i(t)}{dt} = \sum_{j=1}^{n} P_j(t) a_{ji}. \tag{3.7.2}$$

The system of n differential equations may therefore be represented as:

$$\left[\dot{P}_1, \dot{P}_2, \ldots, \dot{P}_n\right] = \left[P_1, P_2, \ldots, P_n\right]\mathbf{A}', \tag{3.7.3}$$

where $\mathbf{A}'$ is the $n \times n$ transition rate matrix where:

$$\mathbf{A}' = \begin{cases} a_{ij}, & \text{the transition rate from } i \text{ to } j, \\ a_{ij} = 0, & \text{if } i > m, j \leq m \\ a_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}. & \end{cases} \tag{3.7.4}$$

The **reduced transition rate matrix,** $\mathbf{A}'_m$, is formed from the first m rows and columns of $\mathbf{A}'$. Therefore

$$\left[\dot{P}_1, \dot{P}_2, \ldots, \dot{P}_m\right] = \left[P_1, P_2, \ldots, P_m\right]\mathbf{A}'_m. \tag{3.7.5}$$

This system of equations may be solved given a set of initial conditions, $P_i(0), i = 1, \ldots, m$. The system reliability, $R(t)$, will be the sum of the probabilities of being in each of the m system working states, i.e.

$$R(t) = \sum_{i=1}^{m} P_i(t). \tag{3.7.6}$$

This may then be used to calculate the mean time to first failure (MTFF) of the system since:

$$\text{MTFF} = \int_0^\infty R(t)dt. \qquad 3.7.7$$

Equations 3.7.5, 3.7.6 and 3.7.7 can then be solved [13] to give:

$$\text{MTFF} = \frac{1}{|\mathbf{A}'_m|}\begin{vmatrix} 0 & P_1(0) & \cdots & P_m(0) \\ 1 & a_{11} & \cdots & a_{1m} \\ \vdots & \vdots & & \vdots \\ 1 & a_{m1} & \cdots & a_{mm} \end{vmatrix}, \qquad 3.7.8$$

which includes the initial conditions of the system. This result means that the MTFF is dependent on the initial conditions, in contrast to the asymptotic availability of the system, c.f. 3.5.8. The asymptotic failure rate may be deduced from 3.7.8, see [13], and is given by:

$$\text{Asymptotic failure rate} = \frac{1}{\text{MTFF}}. \qquad 3.7.9$$

3.8 Summary

Markov analysis provides a means of modelling systems for which dependencies exist due to maintenance and repair processes. This gives it the edge over fault tree analysis in terms of suitability for modelling the application of TLD. However, Markov analysis has features that make it unsuitable for use in modelling TLD. The size of Markov models increases considerably for larger systems. This means that for large aircraft systems of the type to which TLD will be applied the size of the Markov model will become too large to be used. This would also be true, even if the methods of reducing the size of the Markov model, described earlier in this chapter, were applied. This alone makes Markov modelling impractical for TLD. In addition, the requirement that the system be stationary, i.e. that the probability of the system making the transition from one state to another is constant at all points in time, will not necessarily be true for all components in a system. For example, some components could have failure distributions governed by the Weibull distribution.

Chapter 4 : Monte Carlo Simulation of Systems

4.1 Background

In some cases conventional fault tree analysis or Markov Methods may not be a suitable means to model a system. For example, components or subsystems may not be independent, meaning that fault tree analysis is an inappropriate choice of method, or the components' failure and repair rates may not be constant which will prevent the use of Markov methods [5]. The system may simply be too complex to solve using analytical methods. In situations such as these it is appropriate to simulate the system using Monte Carlo methods.

In order to perform a Monte Carlo simulation of a system a computer model of the system is generated. This is based on a logical representation of the system under investigation. The model must contain a set of rules that governs the response of the system to events that can occur, such as component failures, component repairs or particular sequences of failures. The computer model then simulates the operation of the system for a suitable period of time, perhaps the length of a mission or the lifetime of the system. In order for the model to be a good representation of reality the events occurring must do so according to the same distributions as the corresponding events in the real system [16]. To accomplish this, whilst the simulation is in progress, random failures and repairs occur to the components in the system model, according to their failure and repair time distributions. Over the course of the simulation the time spent by the system in any of its states that are of interest is logged. Using this information parameters such as the system reliability or availability can be obtained. Many simulations are performed so that the probability of any particular outcome of interest may be calculated.

4.2 Uniform Random Numbers

An essential element of a Monte Carlo simulation is the generation of random numbers. Some of the simplest ways to generate random numbers involve using the outcomes of

an experiment such as tossing a coin or rolling a die or using random number tables to read off and generate numbers [5]. These methods are only ever of any real value if carrying out the simulations by hand or demonstrating the Monte Carlo simulation techniques. However, in the simulation of real systems these methods are not generally appropriate. This is because of the potentially large number of random numbers required to simulate large systems and/or carry out many simulations [9]. For this reason the random numbers used in Monte Carlo simulation computer programs are generated by the computer. Note that the random numbers generated by a digital computer aren't necessarily random; they may also be pseudo-random. Pseudo-random numbers are generated using algorithms based on deterministic rules, such as recursion formulae, described below. These pseudo-random numbers have a uniform distribution and can be tested mathematically to prove the properties required of random numbers. Despite the fact they are not actually random, pseudo-random numbers are used because they can be obtained quickly [5].

4.2.1 Recursion Formulae

The recursion formulae most commonly used in the generation of uniform distributions of pseudo-random numbers take the form

$$x_{n+1} = (ax_n + b)(\text{mod}\, c), \qquad 4.2.1$$

where a, b and c are positive integers [5]. In order to use recursion formulae a number must be specified to begin the generation of the sequence of pseudo-random numbers. This number is known as the **seed**. If the seed specified at the start of the process is repeated the same number sequence will be produced. For this reason a, b and c in 4.2.1 are chosen such that a large sequence of numbers is produced before the sequence reproduces the seed value and repeats itself.

4.3 Direct Simulation

Simulation involves using statistical information about the components of the system in combination with its logic in order to investigate the system's behaviour. This combination of information changes variables within the model and hence describes the way that the system moves between distinct states [4], [5]. Most simulations are

programmed in such a way that they will move directly from one system state to another, e.g. after the failure of a component. This chronological ordering of events is one of the more difficult challenges when performing simulations of this type.

It is possible, when carrying out Monte Carlo simulations, to model some systems in a time-independent way. A model such as this is useful in developing an understanding of some of the principles and techniques involved in using simulation. However, as was mentioned above, many models involve the advancement of simulation time from one system state to another. In these time-dependent cases the information to be obtained from the simulation may be something such as the time duration in certain states [11]. The following example demonstrates the use of Monte Carlo simulation on a time-independent model of a two-component parallel system. Time-dependent cases involving the chronological movement of the system from one state to another are discussed later.

4.3.1 Example – Time-Independent Model

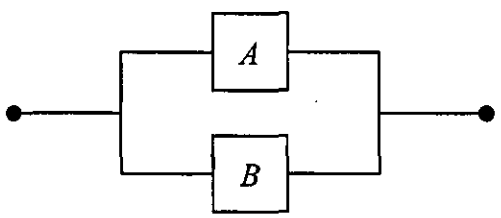


Figure 4.3.1. Two component parallel system.

Consider the two-component parallel system shown in Figure 4.3.1, which will fail if both component A and component B fail. Suppose that the system is to be simulated in order to predict its unavailability and A and B have failure probabilities of 0.1 and 0.2 respectively. The simulation is implemented by generating a random number between 0 and 1 for each component in the system. If the random number is less than the failure probability of the component then that component is assumed to fail. Conversely, if the random number exceeds the failure probability of the component then the component is assumed to remain in a working state. This procedure is illustrated in Figure 4.3.2.

In order to run a single system simulation two random numbers are generated, one for each of the components.

For A Rand. No. = 0.728 > 0.1, component A works

For B Rand. No. = 0.193 < 0.2, component B fails

Therefore component A works and component B fails. After the application of the system logic it is found that the system works in this case. In order to obtain a measure for the system availability many such simulations will be needed.

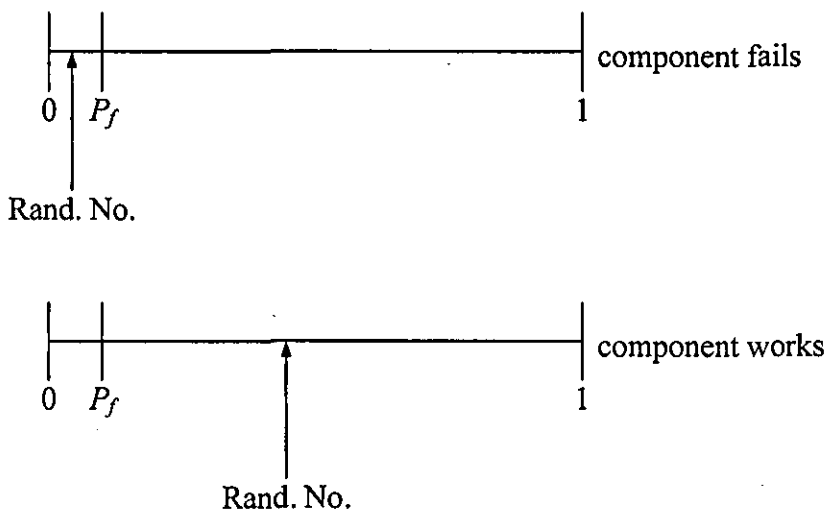


Figure 4.3.2. Direct sampling.

The direct method of sampling described above is a reasonable tool in the simulation of systems whose probability of failure is relatively high. However, if the probability of system failure is low many simulations need to be performed in order to obtain statistically significant results [5]. In order to illustrate this consider a general system of k components. If that system had a probability of failure of 10^{-4} it would be expected to fail once for every $10000k$ random numbers generated (since the completion of n simulations of this system requires the generation of $k \times n$ random numbers).

4.4 Using Probability Distributions to Generate Event Times

A general method for obtaining event times from continuous probability distributions is to integrate the probability density function to give the cumulative distribution function, then to transpose this to give the time to failure/repair [5]. The process of the method is illustrated in Figure 4.4.1, where $F(t)$ represents the cumulative probability distribution for either failure or repair times. Because $F(t)$ has the same range and properties as a

uniform [0,1] distribution of random numbers, a random number, X, is generated and equated to $F(t)$. This then leads to the production of the random event time, T.

This method is demonstrated below for both the exponential and Weibull distributions; however, it is not possible to use this method for the normal distribution so an alternative method is described. Note that it is assumed that the distributions represent failure time distributions, hence an expression for generating random time to failure is obtained. Clearly, this technique is not restricted to failure distributions – the same techniques also apply to repair distributions.

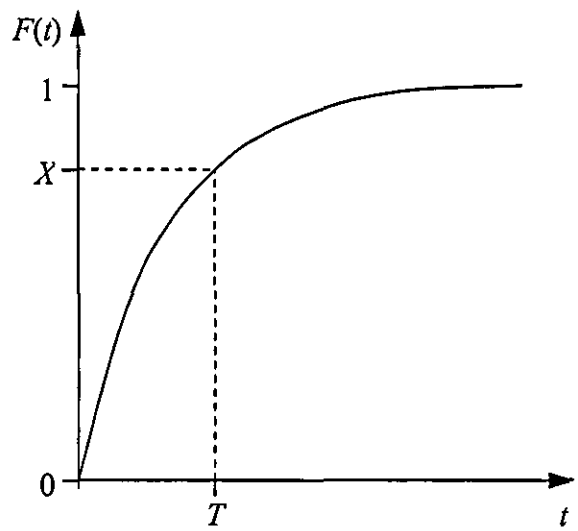


Figure 4.4.1. Generating an event time from the cumulative probability distribution.

4.4.1 Exponential Distribution

The failure density function for the exponential distribution with mean μ is

$$f(t) = \frac{1}{\mu} e^{-t/\mu}, \qquad 4.4.1$$

from which random samples can be obtained by first integrating to give the cumulative failure distribution $F(t)$:

$$F(t) = \int_0^t f(u)du = 1 - e^{-t/\mu}. \qquad 4.4.2$$

$F(t)$ will have the same range and properties as a distribution of random numbers between 0 and 1. Thus, in order to take a random sample, a random number X is generated and equated to $F(t)$ ($0 \le F(t) \le 1$). This gives

$$X = 1 - e^{-t/\mu}, \qquad 4.4.3$$

which is rearranged to give

$$t = \mu \ln(1 - X). \qquad 4.4.4$$

Since X is uniform over [0,1] so is $1 - X$ and equation 4.4.4 may be simplified, yielding

$$t = -\mu \ln X. \qquad 4.4.5$$

Given this formula a random number X ($0 \leq X \leq 1$) can be used to obtain a random time to failure for a component with an exponential failure distribution.

4.4.2 Weibull Distribution

The failure density function for the Weibull distribution with parameters β and η is

$$f(t) = \beta \frac{t^{\beta-1}}{\eta^{\beta}} \exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right], \qquad 4.4.6$$

from which random samples are again obtained by first integrating to give the cumulative failure distribution $F(t)$:

$$F(t) = 1 - \exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right]. \qquad 4.4.7$$

As was done in the case of the exponential distribution, a random number X is generated and equated to $F(t)$ ($0 \leq F(t) \leq 1$). From this we get:

$$t = \eta(-\ln X)^{1/\beta}, \qquad 4.4.8$$

which, upon generation of a random number X ($0 \leq X \leq 1$) gives a random time to failure of a component with a Weibull failure distribution.

4.4.3 Normal Distribution

The failure density function for the normal distribution with mean μ and standard deviation σ is

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^{2}\right]. \qquad 4.4.9$$

This cannot be integrated to give a formula for $F(t)$ that can be transposed to give t in terms of $F(t)$. A method that may be used to obtain random samples from uniformly distributed variables is the central limit theorem [5].

4.4.4 Central Limit Theorem

Consider the n independent identically distributed random variables $X_1, X_2, \ldots, X_n$, which have mean μ and variance σ^2. Then if $S_n = X_1 + X_2 + \ldots + X_n$, the random variable $(S_n - n\mu)/(\sigma\sqrt{n})$ is asymptotically normally distributed with mean 0 and standard deviation 1. Random numbers from the uniform distribution $U(0,1)$ are independent and identically distributed. These may be used to form S_n. In reality when using the central limit theorem the choice of n will mean that the resulting S_n is only approximately normal. A choice of $n = 2$ yields a triangular distribution which is unsuitable. Choosing $n = 3$ gives a bell-shaped distribution, similar in appearance to a normal distribution. This means that in practice a choice of $n \geq 3$ is suitable and in fact $n = 12$ is a good choice from a mathematical point of view (since X_i has $\mu = 0.5$ and $\sigma^2 = 1/12$, hence S_n is $N(6,1)$).

Obtaining a random sample from the normal distribution will hence involve the generation of 12 random numbers $X_1, X_2, \ldots, X_n$ from the uniform distribution $U(0,1)$. These are then added, giving

$$X = \sum_{i=1}^{12} X_i. \qquad 4.4.10$$

Now, by the central limit theorem X is normally distributed with mean 6 and standard deviation 1 and thus

$$t = (X - 6)\sigma + \mu \qquad 4.4.11$$

is normally distributed with mean μ and standard deviation σ.

4.4.5 Example – Simulating a System

Consider the simple system in Figure 4.4.2. Now suppose that the components A, B and C have failure and repair distributions as shown in Table 4.4.1.

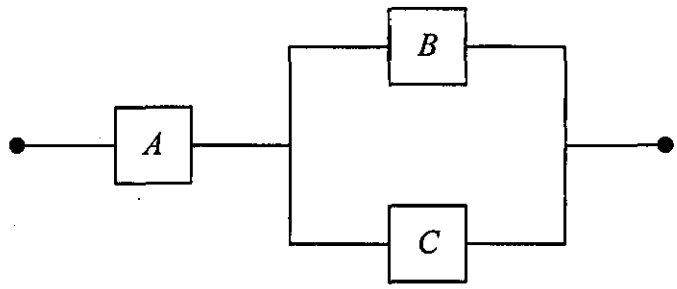


Figure 4.4.2. A simple system.

C is initially on cold standby for *B*. This means *C* is only called into action when *B* fails and *C* can't fail when it is in standby. If *B* is repaired while *C* is working *B* is then in standby for *C*. If both components are repaired *C* is again put on standby for *B*.

Component	Failure Distribution	Repair Distribution
A	exp(1000)	N(10,1.5)
B	exp(600)	exp(1)
C	exp(500)	exp(2.5)

Table 4.4.1. Failure and repair distributions.

Maintenance checks are carried out on the system after every 750hrs of system use. The checks involve finding failed components and repairing them. 10hrs are allocated for maintenance checks, any time longer than this is undesirable. Upon system failure repairs are carried out immediately to all failed components. It is required to know the percentage total downtime of the system in its first 2000hrs of use. The downtime doesn't include the 10hr periods allocated for maintenance.

A typical simulation is shown in Table 4.5.1. Firstly, failure times are generated for *A* and *B* and the time of the first maintenance check is noted. The simulation then advances to the first event chronologically, which is a maintenance operation. As there haven't been any failures the next maintenance is scheduled, then the simulation advances to the next event, the failure of component *B*. This means a failure time must be generated for *C* before the simulation continues again, since *C* is on cold standby for *B*. The next event to be removed from the schedule is the failure of *A*, at which point the system fails and unscheduled maintenance must be carried out. Repair times are generated for the failed components, then a new failure time is generated for *A*. The

process of removing the next event chronologically from the schedule of events to happen continues until the simulation time is over.

The total downtime of the system during the simulation is 8hrs. Therefore the total percentage downtime is given as

$$\frac{8}{2500} \times 100 = 0.32, \qquad 4.4.12$$

meaning the system is inoperable (outside the realms of scheduled maintenance) for 0.32% of the time.

The procedure carried out forms just one sample and in order to obtain a more accurate indicator of the expected system performance many more simulations should be implemented. As more simulations are carried out the results obtained are averaged and the data gathered will converge to the correct value. The results must be checked for convergence in order that a sufficient number of simulations can be carried out. This would be done in practice by monitoring the results obtained as more and more simulations are performed. Once the results after a number of successive simulations match to the required accuracy the simulation procedure can be brought to a halt.

4.5 Results of Monte Carlo Simulations

The previous example goes some way to demonstrate the power of Monte Carlo simulation when modelling system behaviour. It can be seen from Table 4.5.1 that detailed information about the behaviour of the system can be gleaned from what is, in effect a relatively simple method. For instance, it is not only the reliability or availability of the system that may be obtained. Other information, such as the number of times a component fails and whether the failure was revealed or not, or the availability or otherwise of maintenance personnel, could also be incorporated into the model and logged. In fact details of almost any other occurrence one would care to think of could be recorded for later analysis. The attainment of parameters such as these may be extremely difficult, if not impossible, using normal analytic methods. Indeed, the only real restrictions to the Monte Carlo simulation method are the computing power and time available and, of course, the imagination and skill of the programmer.

Simulation Time (hrs)	Event	Computations
0.0	Simulation starts	generate failure times for *A* and *B*: *A*: Rand. no. = 0.241 $t_f = -1000\ln(0.241) = \mathbf{1423.0}$
		B: Rand. no. = 0.129 $t_f = -600\ln(0.129) = \mathbf{1228.8}$
		Maintenance at **750.0**
750.0	Maintenance	No failures to repair Next maintenance at **1500.0**
1228.8	*B* fails	generate failure time for *C*: *C*: Rand. no. = 0.511 $t_f = -500\ln(0.511) = 335.7$ *C* will fail at time 1228.8 + 335.7 = **1564.5**
1423.0	*A* fails SYSTEM fails Unscheduled maintenance	generate repair times for *A* and *B*: *A*: 12 Rand. nos. 0.612 0.717 0.422 0.124 0.096 0.872 0.693 0.642 0.981 0.451 0.249 0.726 using 4.4.10 and 4.4.11 $t_r = 6.8$
		B: Rand. no. = 0.297 $t_r = -1 \times \ln(0.297) = 1.2$
		TOTAL REPAIR TIME = 6.8 + 1.2 = 8.0
		generate new failure time for *A*: *A*: Rand. no. = 0.276 $t_f = -1000\ln(0.276) = 1287.4$ *A* will fail at time 1423.0 + 1287.4 = **2710.4**
1500.0	Maintenance	No failures to repair Next maintenance at **2250.0**
1564.5	*C* fails	generate failure time for *B*: *B*: Rand. no. = 0.425 $t_f = -600\ln(0.425) = 855.7$ *B* will fail at time 1564.5 + 855.7 = **2420.2**
2250.0	Maintenance	The total simulation time exceeds 2000hrs, therefore the simulation ends

Table 4.5.1. An example simulation.

Chapter 5 : Aerospace Recommended Practice Methods of Modelling TLD

5.1 Introduction

Before TLD can be safely applied to any real FADEC system a reliability analysis must be performed. This reliability analysis is used to ensure that the system will conform with the appropriate certification requirements and safety regulations. TLD was introduced to utilise redundancy within the FADEC systems and the relatively high reliability of their individual components in order to improve the efficiency of aircraft dispatch. In improving this efficiency the maintenance of the FADEC systems could be planned in such a way that any delays and cancellations of aircraft, which would be undesirable for travellers, airports, operators and manufacturers alike, would be minimised. In order to achieve these objectives immediately after the certification of a system, as it comes into active service, the methods of analysis used to gain certification must be as accurate as possible. However, it is important that the model used to demonstrate system safety is at worst pessimistic and the levels of safety or reliability predicted by the model are underestimated. A good TLD analysis will demonstrate system safety, but it may also allow the setting of dispatch intervals in order to permit as long a period of dispatch as possible. This would achieve a well-planned maintenance strategy whilst demonstrating confidence in the system reliability. In this way, the dispatchability of the FADEC system is the best it can be as the system comes into service.

A general approach to the analysis of TLD is described in SAE ARP5107 [17], [18]. The reliability of the system is analysed in the full-up state and any allowed dispatchable fault configurations. In SAE ARP5107 two techniques of analysis are recommended. Faults included in the analyses are those that affect the LOTC rate of the system – if a fault does not affect the LOTC rate of the system it may be omitted from the analysis. The following sections outline the methods used in applying each of these approaches to a general system.

5.2 A General System Model

Consider a system that has n dispatchable system configurations, numbered from 1 to n. Amongst these dispatchable configurations there will be a full-up dispatchable system state and a number of STD and LTD dispatchable system states. Let state $i = 1$ represent the full-up system configuration, states $i = 2,\ldots,m$ represent the STD dispatchable system states and states numbered $i = m+1,\ldots,n$ the LTD dispatchable system configurations. Two modelling techniques, time-weighted average and reduced fault state Markov methods are recommended by SAE ARP 5107 as being appropriate to assess TLD. These are discussed in the following sections.

5.3 Time-Weighted Averages (TWA) Modelling

This is a relatively simple method of calculating the overall system LOTC rate [1], [17], [18], which is obtained by adding the contributions from the following:

1. the failure rates to LOTC of individual faults from the mechanical and hydromechanical portions of the FADEC system,
2. the failure rates to LOTC of any unrevealed faults in the electrical and electronic portions of the FADEC system,
3. a time-weighted average (TWA) of the failure rates to LOTC of the system from each of its dispatchable system configurations.

Thus the TWA failure rate of a simple system, λ_{TWA}, is given by:

$$\lambda_{TWA} = \lambda_{HMC} + \lambda_{UR} + t_{FU}\lambda_{FU,L} + t_{STD}\lambda_{STD,L} + t_{LTD}\lambda_{LTD,L} \qquad 5.3.1$$

The first two terms, λ_{HMC} and λ_{UR}, represent the sum of the failure rates due to single mechanical and hydromechanical faults, and the sum of the failure rates due to unrevealed electrical and electronic faults respectively. Unrevealed faults are included in the analysis in this way because if they are all assumed to lead to LOTC events the subsequent TWA failure rate will be an overestimate with respect to these faults. The final three terms in equation 5.3.1 represent the TWA of the failure rate of the system from each of the dispatchable configurations. t_{FU} represents the fraction of time that the system spends in the full-up system state. t_{STD} and t_{LTD} represent the fractions of time spent in the STD and LTD dispatchable system states respectively. Effectively this is the probability that the system will reside in a particular dispatchable configuration. The

final elements of the TWA LOTC rate are the failure rates, $\lambda_{FU,L}$, $\lambda_{STD,L}$ and $\lambda_{LTD,L}$. The first of these, $\lambda_{FU,L}$, represents the average failure rate from the full-up system configuration to the LOTC state caused by failures in the redundant portions of the FADEC in any one flight. $\lambda_{STD,L}$ represents the average LOTC rate of the system when a second electronic or electrical fault occurs in addition to the STD fault that is already present when operating in an STD dispatch configuration. Similarly, $\lambda_{LTD,L}$ represents the average LOTC rate of the system when a second electronic or electrical fault occurs in addition to the LTD fault that is present when operating in an LTD dispatch configuration.

What now follow are definitions of the above quantities that may be used in the TWA failure rate given in equation 5.3.1. These are obtained from the original (1997) [17] and revised (2004) versions [18] of SAE ARP5107.

5.3.1 The Failure Rate Due to Hydromechanical Faults, λ_{HMC}

In both the original SAE ARP5107 document and the revised version this failure rate is approximated as the sum of the failure rates due to all mechanical/hydromechanical faults.

5.3.2 The Failure Rate for Unrevealed Faults, λ_{UR}

The definition of this quantity is essentially unchanged from the original release of SAE ARP5107 [17] to its revision in 2004 [18]. It is stated to be generally expressed as being the product of some constant, say X, with the sum of the failure rates "of all electrical/electronic hardware associated with LOTC critical elements," i.e.

$$\lambda_{UR} = X \times \{\sum \lambda \text{ of all electrical/electronic hardware associated with LOTC critical elements}\} \qquad 5.3.2$$

The definition of the constant X will be discussed later. However, the second quantity may be approximated by:

$$\lambda_{STD} + \lambda_{LTD}, \qquad 5.3.3$$

where λ_{STD} is defined as the sum of the failure rates of the STD faults in both channels and λ_{LTD} is defined as the sum of the failure rates of the LTD faults in both channels, i.e.

$$\lambda_{STD} = \sum_{i=2}^{m} \lambda_i,$$
$$\lambda_{LTD} = \sum_{i=m+1}^{n} \lambda_i. \qquad 5.3.4$$

Thus the expression for the unrevealed failure rate may be approximated by:

$$\lambda_{UR} = X \times \sum_{i=2}^{n} \lambda_i. \qquad 5.3.5$$

The quantity X is rather more difficult to define. It is stated to be "a number generally between 0 and 0.05." It will differ according to the system being modelled and is an indication of the general trend that the unrevealed fault rate of a channel will lie somewhere between 0 and 5% of the total LOTC rate of that channel. Equation 5.3.5 is said to be a conservative estimate for the unrevealed fault rate since it assumes that the failure of any unrevealed fault in any given flight will lead to a LOTC event and this will most likely not be the case.

5.3.3 Average Failure Rates to LOTC, $\lambda_{STD,L}$, $\lambda_{LTD,L}$

If the TWA LOTC rate is to be calculated using equation 5.3.1 then the average failure rates to LOTC from the STD and LTD dispatchable system configurations are defined as follows:

$$\lambda_{STD,L} = \frac{\sum_{i=2}^{m} \lambda_i \lambda_{i,L}}{\lambda_{STD}} = \frac{\sum_{i=2}^{m} \lambda_i \lambda_{i,L}}{\sum_{i=2}^{m} \lambda_i}, \qquad 5.3.6$$

and

$$\lambda_{LTD,L} = \frac{\sum_{i=m+1}^{n} \lambda_i \lambda_{i,L}}{\lambda_{LTD}} = \frac{\sum_{i=m+1}^{n} \lambda_i \lambda_{i,L}}{\sum_{i=m+1}^{n} \lambda_i}, \qquad 5.3.7$$

where $\lambda_{i,L}$ is the failure rate to LOTC for the ith dispatchable system configuration.

5.3.4 Fractions of Time Spent in Dispatchable States, t_{FU}, t_{STD}, t_{LTD}

The definitions of the fractions of time spent in the STD and LTD dispatchable system states, t_{STD} and t_{LTD}, changed slightly from the original SAE ARP5107 to the revised version. However, in both cases, once these two quantities are known the fraction of time spent in the full-up dispatchable system configuration is calculated using the fact that the total of all these fractions must be unity, i.e.:

$$t_{FU} = 1 - t_{STD} - t_{LTD}, \qquad 5.3.8$$

Given that the dispatch interval for STD faults is given by T_{STD} and that the dispatch interval for LTD faults is given by T_{LTD}, the fractions of time spent in the STD and LTD dispatchable system configurations are approximated by:

$$t_{STD} = \lambda_{STD} T_{STD}, \qquad 5.3.9$$

and

$$t_{LTD} = \lambda_{LTD} T_{LTD}. \qquad 5.3.10$$

These approximations, given in equations 5.3.9 and 5.3.10, assume that the fractions of time are much less than 1 [17]. In [18] the fractions of time spent in the STD and LTD dispatchable system configurations are approximated by:

$$t_{STD} = t_{FU}(\lambda_{STD} T_{STD}), \qquad 5.3.11$$

and

$$t_{LTD} = t_{FU}(\lambda_{LTD} T_{LTD}). \qquad 5.3.12$$

These latter approximations, given in equations 5.3.11 and 5.3.12, are similar to those given in equations 5.3.9 and 5.3.10, but for the fact that the quantity on the RHS is multiplied by t_{FU}. This change was made to the technique in order to better balance the actual portion of time spent in the various system states [18]. It is stated in [18] that the first approximations, from [17], assume that the system is in the full-up state for all of the time, while the second ones, from [18], take into account that the system is in the full-up state for the fraction of time t_{FU}, and, as such, are more accurate. This is demonstrated mathematically in Section 5.4. In the case of the revised method of finding the time fractions, there are 3 simultaneous equations to be solved for t_{STD}, t_{LTD} and t_{FU}, these being equations 5.3.8, 5.3.11 and 5.3.12.

Thus there are two different possible values for each of t_{STD}, t_{LTD} and t_{FU}. These are listed below. The time fractions obtained from the original SAE ARP5107 document

will henceforth be known as the *fractional coefficients* and, as in the revised SAE ARP5107 document, the time fractions obtained from that document will be known as the *balanced fractional coefficients*. Thus the fractional coefficients are:

$$\begin{aligned} t_{FU} &= 1 - t_{STD} - t_{LTD}, \\ t_{STD} &= \lambda_{STD} T_{STD}, \\ t_{LTD} &= \lambda_{LTD} T_{LTD}, \end{aligned} \qquad 5.3.13$$

and the balanced fractional coefficients are:

$$\begin{aligned} t_{FU} &= \frac{1}{1 + \lambda_{STD} T_{STD} + \lambda_{LTD} T_{LTD}}, \\ t_{STD} &= \frac{\lambda_{STD} T_{STD}}{1 + \lambda_{STD} T_{STD} + \lambda_{LTD} T_{LTD}}, \\ t_{LTD} &= \frac{\lambda_{LTD} T_{LTD}}{1 + \lambda_{STD} T_{STD} + \lambda_{LTD} T_{LTD}}. \end{aligned} \qquad 5.3.14$$

5.3.5 LOTC Rate

It is now possible, using the definitions and equations derived above, to calculate the TWA LOTC rate using both the fractional coefficients and the balanced fractional coefficients. These will now be referred to as $\lambda_{TWA,frac}$ and $\lambda_{TWA,bal}$. Substituting equation 5.3.13 into equation 5.3.1 yields:

$$\lambda_{TWA,frac} = \lambda_{HMC} + \lambda_{UR} + \lambda_{FU,L} + (\lambda_{STD}\lambda_{STD,L} - \lambda_{STD}\lambda_{FU,L})T_{STD} + (\lambda_{LTD}\lambda_{LTD,L} - \lambda_{LTD}\lambda_{FU,L})T_{LTD}. \qquad 5.3.15$$

Similarly, substituting equation 5.3.14 into equation 5.3.1 gives:

$$\lambda_{TWA,bal} = \lambda_{HMC} + \lambda_{UR} + \frac{\lambda_{FU,L} + (\lambda_{STD}\lambda_{STD,L})T_{STD} + (\lambda_{LTD}\lambda_{LTD,L})T_{LTD}}{1 + \lambda_{STD}T_{STD} + \lambda_{LTD}T_{LTD}}. \qquad 5.3.16$$

5.3.6 *Failure Rates to LOTC, $\lambda_{i,L}$*

A method of calculating the failure rates to LOTC from the ith dispatchable system configuration is given in [1]. The process involves taking the failure probability (of LOTC) for the relevant system configurations and dividing this by a suitable time period, the average flight time is suggested, in order to obtain a probability per flight hour. This quantity is then equated to the failure rate to LOTC over the time period used. When the flight time is used the LOTC rate from dispatchable configuration i is calculated to be:

$$\lambda_{i,L} = \frac{Q_{i,L}}{t_{flt}}, \qquad 5.3.17$$

where $Q_{i,L}$ is the failure probability (of LOTC) for dispatchable system configuration i.

5.3.7 The Failure Rate for Full-Up Electronics, $\lambda_{FU,L}$

In [17], [18] the LOTC rate for full-up electronics, $\lambda_{FU,L}$, is approximated using a conservative upper bound, given by:

$$\begin{aligned} \lambda_{FU,L} &\approx \left(\tfrac{1}{2}\left(\lambda_{STD} + \lambda_{LTD}\right)\right)^2 \times t_{flt} \\ &= \left(\tfrac{1}{2}\sum_{i=2}^{n}\lambda_i\right)^2 \times t_{flt}, \end{aligned} \qquad 5.3.18$$

where λ_{STD} and λ_{LTD} are as defined in equation 5.3.4. The constant t_{flt} represents the length of a flight in hours. This approximation is conservative since it assumes all faults in a channel result in the loss of that channel. In order to see where the approximation given in equation 5.3.18 comes from consider this conservative assumption in which all channel faults result in the loss of function of that channel.

The occurrence of a LOTC event will thus require the occurrence, in a single flight, of two independent faults, one in each channel. Therefore the probability of a LOTC event occurring after a full-up dispatch, $P_{FU,L}$, is given by:

$$\begin{aligned} P_{FU,L} &= P(\text{channel 1 fails AND channel 2 fails}) \\ &\leq P_{CHAN1} P_{CHAN2}, \end{aligned} \qquad 5.3.19$$

where P_{CHAN1} and P_{CHAN2} are respectively the probabilities that channel 1 and channel 2 fail in a flight given a full-up dispatch at the beginning of that flight. Note that the upper bound given above is analogous to the rare event approximation for the system unavailability detailed in Section 2.4.1.1. Now, using the method shown in Section 5.3.6 to calculate a failure rate by dividing the failure probability by the average flight time, P_{CHAN1} and P_{CHAN2} can be expressed as follows:

$$\begin{aligned} P_{CHAN1} &= \lambda_{CHAN1} t_{flt}, \\ P_{CHAN2} &= \lambda_{CHAN2} t_{flt}, \end{aligned} \qquad 5.3.20$$

where λ_{CHAN1} and λ_{CHAN2} represent the failure rates of channel 1 and channel 2 in a flight given full-up dispatch. Substituting equations 5.3.20 into equation 5.3.19 yields

the following upper bound for the probability of a LOTC event in a flight after a full-up dispatch:

$$P_{FU,L} \leq \lambda_{CHAN1}\lambda_{CHAN2}t_{flt}^{\ 2}. \qquad 5.3.21$$

Again approximating a failure rate by a failure probability divided by a flight time for this equation 5.3.21 gives the following upper bound for the LOTC rate for full-up electronics:

$$\lambda_{FU,L} \leq \lambda_{CHAN1}\lambda_{CHAN2}t_{flt}. \qquad 5.3.22$$

Now consider again the conservative assumption that all dispatchable faults occurring within a channel result in the loss of that channel. The LOTC rate for the single channel given full-up dispatch can thus be estimated by half of the sum of the failure rates of all dispatchable faults in both channels, giving:

$$\lambda_{CHAN1} = \lambda_{CHAN2} = \tfrac{1}{2}\sum_{i=2}^{n}\lambda_i. \qquad 5.3.23$$

Equation 5.3.23 can now be substituted into equation 5.3.22 to yield equation 5.3.18, the upper bound used to approximate $\lambda_{FU,L}$.

5.4 TWA Approach Using the Balanced Time Coefficients

Consider equation 5.3.1, the original definition of the TWA failure rate. In order to obtain the TWA failure rate using the balanced time coefficients, $\lambda_{TWA,bal}$, equations 5.3.6, 5.3.7, 5.3.11 and 5.3.12 are substituted into equation 5.3.1 to give:

$$\lambda_{TWA,bal} = \lambda_{HMC} + \lambda_{UR} + t_{FU}\lambda_{FU,L} + t_{FU}\lambda_{STD}T_{STD}\frac{\sum_{i=2}^{m}\lambda_i\lambda_{i,L}}{\sum_{i=2}^{m}\lambda_i} + t_{FU}\lambda_{LTD}T_{LTD}\frac{\sum_{i=m+1}^{n}\lambda_i\lambda_{i,L}}{\sum_{i=m+1}^{n}\lambda_i}. \qquad 5.4.1$$

Substitution of λ_{STD} and λ_{LTD} from equation 5.3.4 gives:

$$\lambda_{TWA,bal} = \lambda_{HMC} + \lambda_{UR} + t_{FU}\lambda_{FU,L} + t_{FU}T_{STD}\sum_{i=2}^{m}\lambda_i\lambda_{i,L} + t_{FU}T_{LTD}\sum_{i=m+1}^{n}\lambda_i\lambda_{i,L}. \qquad 5.4.2$$

Now define a quantity T_i which will represent the dispatch interval for each dispatchable system configuration i except the full-up state such that:

$$T_i = \begin{cases} T_{STD}, & \text{if } i = 2,\ldots,m, \\ T_{LTD}, & \text{if } i = m+1,\ldots,n. \end{cases} \qquad 5.4.3$$

Substituting equation 5.4.3 into equation 5.4.2 leads to:

$$\lambda_{TWA,bal} = \lambda_{HMC} + \lambda_{UR} + t_{FU}\lambda_{FU,L} + t_{FU}\sum_{i=2}^{n} T_i\lambda_i\lambda_{i,L}. \qquad 5.4.4$$

Defining the time fraction in dispatchable configuration i to be:

$$t_i = t_{FU}T_i\lambda_i, \qquad 5.4.5$$

means that equations 5.3.11 and 5.3.12 are satisfied. Remembering that state $i = 1$ represents the full-up dispatchable configuration equation 5.4.4 becomes:

$$\lambda_{TWA,bal} = \lambda_{HMC} + \lambda_{UR} + \sum_{i=1}^{n} t_i\lambda_{i,L}. \qquad 5.4.6$$

Upon examination it can be seen that this is equivalent to adding the failure rates due to mechanical/hydromechanical and unrevealed faults to the sum of the fractions of time dispatching in each of the i dispatchable configurations multiplied by the instantaneous LOTC rate from that dispatchable configuration. This is equivalent to the original definition given at the beginning of Section 5.3. A conclusion that may be drawn from this is that the balanced time coefficients given in [18] lead to a formula for the LOTC rate that will more accurately represent the original definition of the TWA LOTC rate than that given in [17]. Therefore we may reasonably assume that the balanced time coefficients are a better approximation of the time spent in each dispatchable system configuration than the original coefficients used in [17].

5.5 Reduced Fault State Markov Approaches

As its name suggests, this modelling approach is based on a Markov analysis of the FADEC system. The difference from a conventional Markov approach lies in the reduced number of system states. In this approach only lower order fault states, commonly only first order fault states, are included in the model. When only first order fault states are included in the model this is known as a single fault state Markov model [17], [18]. These first order fault states would normally be only STD and LTD category fault states. DND fault states may be included but as a DND condition would normally be caused by a combination of system faults, these would, by definition, not be included in a single fault state Markov model. In [18] it is stated that single fault state models are acceptable when the repair rates for the single fault states are approximately 10 times (or more) greater than the maximum failure rate into or out of the single fault states. It is also stated that, for the models to achieve a good degree of accuracy (where the computed LOTC rate is within 5% of the true value), the repair rate for STD faults must

be at least 10 times greater than the maximum failure rate into or out of any STD state and a similar situation must hold true for the LTD repair and failure rates. This is said to justify the validity of a single fault state Markov model, since when repairs are so frequent the likelihood of multiple fault states leading to a LOTC event is minimal. The probability of a LOTC event in these circumstances is very low.

For some systems the reduced state Markov model must include dual order fault states in addition to the single fault states. This could be the case if the repair rate and failure rates of any of the single fault states did not meet the criteria outlined above.

A general feature of both single fault state and dual fault state Markov models that is unconventional when one considers the usual Markov modelling technique is that a transition rate is included on the system model in addition to the repair, failure and LOTC rates. This transition rate is an artificial simulated repair, added in order to allow the steady state solution for the LOTC rate to be calculated [18]. It leads directly from the final LOTC state back to the full-up system state. This transition rate means that the model no longer contains any absorbing states. This simulated repair rate will be discussed in more detail later in Section 5.5.2.

What now follows is a description of a general single fault state Markov model, as described in [18].

5.5.1 Single Fault State Markov Approach

Consider again the general system described in Section 5.2, and assume that all of the STD and LTD dispatchable system states, numbered 2,...,n, are single fault states. Since mechanical/hydromechanical faults and unrevealed faults may cause LOTC from any of the dispatchable system configurations a transition to the LOTC state from each of the states representing these dispatchable configurations is added to the model (as defined in Sections 5.3.1 and 5.3.2). This results in a model shown in Figure 5.5.1.

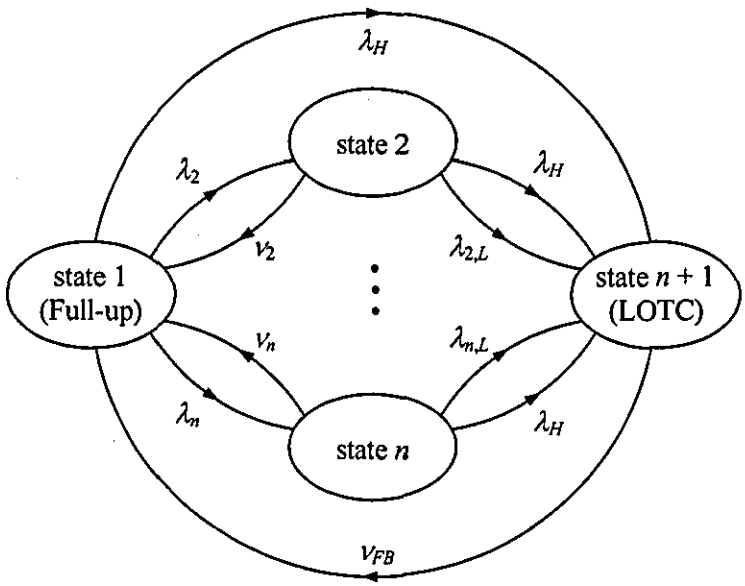


Figure 5.5.1. A single fault state Markov model.

λ_H represents the sum of λ_{HMC} and λ_{UR}, the failure rates due to the mechanical/hydromechanical and unrevealed failures respectively. λ_i is the failure rate into state i and $\lambda_{i,L}$ is the failure rate out of that state into the LOTC state. The repair rates, ν_i, are defined as the reciprocal of the dispatch intervals for each fault state. Thus, if T_i represents the dispatch interval for state i the repair rate, ν_i, is given by:

$$\nu_i = \frac{1}{T_i}. \qquad 5.5.1$$

The artificial, simulated repair rate that is added to the model is given by the feedback transition, ν_{FB}, from the LOTC state (state n + 1) to the full-up state (state 1).

5.5.2 Calculating the LOTC Rate

The Markov model shown in Figure 5.5.1 yields the transition rate matrix **A**, given by:

$$\mathbf{A} = \begin{bmatrix} -\left(\lambda_H + \sum_{i=2}^{n} \lambda_i\right) & \lambda_2 & \cdots & \lambda_n & \lambda_H \\ \nu_2 & -(\nu_2 + \lambda_{2,L} + \lambda_H) & 0 & 0 & \lambda_{2,L} + \lambda_H \\ \vdots & 0 & \ddots & 0 & \vdots \\ \nu_n & 0 & 0 & -(\nu_n + \lambda_{n,L} + \lambda_H) & \lambda_{n,L} + \lambda_H \\ \nu_{FB} & 0 & \cdots & 0 & -\nu_{FB} \end{bmatrix}. \qquad 5.5.2$$

All off-diagonal terms of **A** that are not members of the first or last row or column are equal to zero. This transition rate matrix leads to a system of n + 1 differential equations of the form:

$$\dot{\mathbf{Q}} = \mathbf{QA}, \qquad 5.5.3$$

where

$$\mathbf{Q} = [Q_1, Q_2, \ldots, Q_{n+1}], \qquad 5.5.4$$

with Q_i being the probability of the system being in state i. When finding the steady state solution of this Markov model the derivatives of these probabilities is set to zero (at steady state the rates of change of the probabilities is zero). This results in the following system of *dependent* linear simultaneous equations:

$$\mathbf{QA} = \mathbf{0}. \qquad 5.5.5$$

In order to obtain an *independent* system of equations one of the equations is arbitrarily chosen to be replaced by the constraint equation:

$$\sum_{i=1}^{n+1} Q_i = 1. \qquad 5.5.6$$

If the equation that is replaced is chosen to be the first of the $n + 1$ equations then the system given in equation 5.5.5 becomes:

$$\mathbf{Q}\hat{\mathbf{A}} = \mathbf{B}, \qquad 5.5.7$$

where

$$\hat{\mathbf{A}} = \begin{bmatrix} 1 & \lambda_2 & \cdots & \lambda_n & \lambda_H \\ 1 & -(\nu_2 + \lambda_{2,L} + \lambda_H) & 0 & 0 & \lambda_{2,L} + \lambda_H \\ \vdots & 0 & \ddots & 0 & \vdots \\ 1 & 0 & 0 & -(\nu_n + \lambda_{n,L} + \lambda_H) & \lambda_{n,L} + \lambda_H \\ 1 & 0 & \cdots & 0 & -\nu_{FB} \end{bmatrix}, \qquad 5.5.8$$

and

$$\mathbf{B} = [1, 0, \ldots, 0], \qquad 5.5.9$$

The LOTC rate of the system is found by considering the average failure rate *into* the LOTC state, i.e. state $n + 1$, and thus the single fault state Markov LOTC rate, λ_{SFS}, is defined as:

$$\lambda_{SFS} = \frac{\text{Transition rate } \textit{into} \text{ state } n+1}{1 - \text{Probability of being in state } n+1}, \qquad 5.5.10$$

which will be given by:

$$\lambda_{SFS} = \frac{\lambda_H Q_1 + \sum_{i=2}^{n} (\lambda_{i,L} + \lambda_H) Q_i}{1 - Q_{n+1}}. \qquad 5.5.11$$

This LOTC rate requires the solution of equation 5.5.7. However, as is noted in [18], it is possible to obtain an expression for λ_{SFS} that is independent of the state probabilities by considering all but the final simultaneous equation of the system given by equation

5.5.7. From the constraint equation, equation 5.5.6, one can obtain an expression for the denominator of λ_{SFS} given by:

$$1-Q_{n+1}=\sum_{i=1}^{n}Q_i, \qquad 5.5.12$$

and from the simultaneous equations numbered 2 to n one can obtain expressions for Q_i which is as follows:

$$Q_i=\frac{\lambda_i}{\nu_i+\lambda_{i,L}+\lambda_H}Q_1, \quad i=2,\ldots,n, \qquad 5.5.13$$

Substituting equations 5.5.12 and 5.5.13 into equation 5.5.11 gives the following expression for the single fault state Markov LOTC rate:

$$\lambda_{SFS}=\frac{\lambda_H Q_1+\left\{\sum_{i=2}^{n}\frac{\lambda_i(\lambda_{i,L}+\lambda_H)}{\nu_i+\lambda_{i,L}+\lambda_H}\right\}Q_1}{\sum_{i=1}^{n}Q_i}, \qquad 5.5.14$$

which, after substitution of equation 5.5.13 into the denominator and subsequent simplification yields:

$$\lambda_{SFS}=\frac{\lambda_H+\sum_{i=2}^{n}\left(\frac{\lambda_i(\lambda_{i,L}+\lambda_H)}{\nu_i+\lambda_{i,L}+\lambda_H}\right)}{1+\sum_{i=2}^{n}\left(\frac{\lambda_i}{\nu_i+\lambda_{i,L}+\lambda_H}\right)}. \qquad 5.5.15$$

This expression for the LOTC rate is now noted to be independent of the probabilities of residing in each of the system states but also, more importantly, the feedback rate, ν_{FB}. This would imply that the steady state LOTC rate of the system is independent of the value of the artificial feedback rate that is added to the model. As is stated in [18], this is despite the fact that the probabilities of lying in the various system states *will* depend on ν_{FB}. If this is not immediately obvious consider solving the system of simultaneous equations given in equation 5.5.7. The presence of ν_{FB} in the final column of equation 5.5.8 means that the solution will also depend on ν_{FB}. An explanation for why the LOTC rate of the system does not depend on the value of the feedback rate, ν_{FB}, is given in [18]. It is stated that the average LOTC rate "of a fleet of engine control systems is not affected by the length of time a control system is absent from the operational fleet following its failure." The feedback rate could be thought of as being representative of the fact that a LOTC event can occur for an engine and the engine would then be repaired or replaced at the end of the flight in which the LOTC event

occurred. This would be true for multi-engined aircraft where the feedback rate would represent the repair or replacement process.

5.5.3 First Order Approximation to the LOTC Rate

When calculating the LOTC rate of a system a common approach appears to be to set the length of the STD interval and then plot a graph of the predicted LOTC rate against a varying LTD interval [1], [17], [18]. This will involve fixing the STD interval, T_i ($i = 2,\ldots,m$), and hence the repair rate, v_i ($i = 2,\ldots,m$), from the STD dispatchable fault states and then calculating the LOTC rate for different values of the LTD interval, T_i ($i = m + 1,\ldots,n$). It is stated in [18] that this could cause difficulty when using a spreadsheet to compute the LOTC rate as the summations in the numerator and denominator of equation 5.5.15 need to be recomputed for each value of the LTD interval for which the LOTC rate must be calculated. For this reason approximations are obtained for the summations involved in the numerator and denominator of equation 5.5.15 that will then negate the need to continually recalculate these summations over the required range of LTD interval values.

The first step in the approximation process is to substitute equation 5.5.1 into equation 5.5.15 to give the single fault state Markov LOTC rate, λ_{SFS}, as a function of the dispatch intervals, T_i, as follows:

$$\lambda_{SFS} = \frac{\lambda_H + \sum_{i=2}^{n}\left(\frac{T_i\lambda_i(\lambda_{i,L} + \lambda_H)}{1 + T_i(\lambda_{i,L} + \lambda_H)}\right)}{1 + \sum_{i=2}^{n}\left(\frac{T_i\lambda_i}{1 + T_i(\lambda_{i,L} + \lambda_H)}\right)}. \qquad 5.5.16$$

In order to obtain an approximation to the LOTC rate, Maclaurin series expansions of the summations in its numerator and denominator are computed, truncated and substituted into equation 5.5.16. Since λ_i, $\lambda_{i,L}$ and λ_H, will essentially be held constant as the dispatch intervals, T_i, are varied, the summations may be considered as functions of one variable, T_i, as the Maclaurin series are calculated. Below are the derivations of both Maclaurin series, for the summations in the numerator and the denominator of equation 5.5.16.

A Maclaurin series is a Taylor series expansion of a function about 0 and as such takes the form:

$$f(x)=\sum_{k=1}^{\infty}\frac{x^k}{k!}\frac{d^k f(0)}{dx^k}. \qquad 5.5.17$$

Let $f_1(T_i)$ be the argument of the summation in the numerator of λ_{SFS}. Thus:

$$\begin{aligned} f_1(T_i)&=\frac{c\lambda_i T_i}{1+cT_i},\\ f_1'(T_i)&=\frac{c\lambda_i}{(1+cT_i)^2},\\ f_1''(T_i)&=\frac{-2c^2\lambda_i(1+cT_i)}{(1+cT_i)^4}, \end{aligned} \qquad 5.5.18$$

where c is the constant given by:

$$c=\lambda_{i,L}+\lambda_H. \qquad 5.5.19$$

Substituting equations 5.5.18 into 5.5.17 and truncating after terms of $O(T^2)$ yields the expression:

$$f_1(T_i)\approx c\lambda_i T_i - c^2\lambda_i T_i^2. \qquad 5.5.20$$

Similarly, if $f_2(T_i)$ is the argument of the summation in the denominator of λ_{SFS}, the following equations are obtained:

$$\begin{aligned} f_2(T_i)&=\frac{\lambda_i T_i}{1+cT_i},\\ f_2'(T_i)&=\frac{\lambda_i}{(1+cT_i)^2},\\ f_2''(T_i)&=\frac{-2c\lambda_i(1+cT_i)}{(1+cT_i)^4}, \end{aligned} \qquad 5.5.21$$

where c is as defined in equation 5.5.19. Substituting equations 5.5.21 into 5.5.17 and again disregarding terms of $O(T^3)$ and above gives:

$$f_2(T_i)\approx \lambda_i T_i - c\lambda_i T_i^2. \qquad 5.5.22$$

Returning now to the definition of the single fault state Markov LOTC rate, λ_{SFS}, the expressions for the arguments of the numerator and denominator, given by equations 5.5.20 and 5.5.22, may be substituted into equation 5.5.16 to obtain the first order approximation to the single fault state Markov equation, which will be called $\lambda_{SFS,1}$ here. This is thus given by:

$$\lambda_{SFS,1}=\frac{\lambda_H+\sum_{i=2}^{n}\left(\lambda_i(\lambda_{i,L}+\lambda_H)T_i\right)-\sum_{i=2}^{n}\left(\lambda_i(\lambda_{i,L}+\lambda_H)^2T_i^2\right)}{1+\sum_{i=2}^{n}(\lambda_i T_i)-\sum_{i=2}^{n}\left(\lambda_i(\lambda_{i,L}+\lambda_H)T_i^2\right)}. \qquad 5.5.23$$

This may not look any simpler at first glance than equation 5.5.16. However, by grouping all STD and LTD fault states one may obtain an expression for $\lambda_{SFS,1}$ wherein the dispatch intervals T_{STD} and T_{LTD}, defined in equation 5.4.3, are included in the expression as follows:

$$\lambda_{SFS,1} = \frac{\lambda_H + a_1 T_{STD} + a_2 T_{LTD} - b_1 T_{STD}^2 - b_2 T_{LTD}^2}{1 + c_1 T_{STD} + c_2 T_{LTD} - a_1 T_{STD}^2 - a_2 T_{LTD}^2}, \qquad 5.5.24$$

where:

$$\begin{aligned} a_1 &= \sum_{i=2}^{m} \lambda_i (\lambda_{i,L} + \lambda_H), \\ a_2 &= \sum_{i=m+1}^{n} \lambda_i (\lambda_{i,L} + \lambda_H), \\ b_1 &= \sum_{i=2}^{m} \lambda_i (\lambda_{i,L} + \lambda_H)^2, \\ b_2 &= \sum_{i=m+1}^{n} \lambda_i (\lambda_{i,L} + \lambda_H)^2, \\ c_1 &= \sum_{i=2}^{m} \lambda_i, \\ c_2 &= \sum_{i=m+1}^{n} \lambda_i. \end{aligned} \qquad 5.5.25$$

All of these summations are independent of the dispatch intervals, T_{STD} and T_{LTD}, and this means that they need only be calculated once, allowing the LOTC for different dispatch intervals to be easily obtained.

5.5.4 Simplified Approximations When Dispatch Intervals are Short

Reasonable approximations to the LOTC rate can be obtained in the case when dispatch intervals are short [18] (a value of less than 1000 hours in a MEL maintenance model is quoted, which equates to repair rates greater than 0.001 repairs per hour). Considering equation 5.5.15, it is noted that the terms representing the failure rates from the ith dispatchable configuration added to the failure rates due to unrevealed and mechanical/ hydromechanical faults are usually less than 50×10^{-6} failures per hour. This then leads to the approximation that the repair rates are roughly 10 times greater than the aforementioned failure rates and equation 5.5.15 is simplified to give:

$$\lambda_{SFS,short} = \frac{\lambda_H + \sum_{i=2}^{n} \left(\frac{\lambda_i (\lambda_{i,L} + \lambda_H)}{\nu_i} \right)}{1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\nu_i} \right)}. \qquad 5.5.26$$

Substituting equation 5.5.1 into equation 5.5.26 yields the following approximation for the single fault state LOTC rate:

$$\lambda_{SFS,short} = \frac{\lambda_H + \sum_{i=2}^{n}\left(T_i\lambda_i\left(\lambda_{i,L} + \lambda_H\right)\right)}{1 + \sum_{i=2}^{n}\left(T_i\lambda_i\right)}. \qquad 5.5.27$$

In fact, this expression can be obtained by truncating the Maclaurin series expansions of the numerator and denominator of equation 5.5.16 after terms of O(T) rather than O(T^2).

5.5.5 Dual Fault State Markov Approach

For the single fault state Markov model to be valid it is required that the repair rates, ν_i, of individual STD faults in the system must be 10 times or more greater than the failure rates into or out of the STD single fault states. A similar situation must hold for the LTD single fault states. If these criteria are not met, a dual fault state model is used in preference to a single fault state model. Another scenario that may require the use of a dual fault state model is if the presence of dual fault combinations brought about a reduction in the dispatch interval from say a LTD interval to a STD interval. A general dual fault state model is depicted in Figure 5.5.2. The model shown omits a large number of the states within it for clarity. Comparing this with Figure 5.5.1, the Markov model for the single fault state model, it can be seen that all of the single fault states are still present, and numbered from 2 up to n. The full-up system state and LOTC state are, as before, labelled 1 and $n + 1$ respectively. The dual fault states are represented by the states that are labelled with two indices. Thus, for example, state 2, 3 in the diagram is the state where both of the faults 2 and 3 are present.

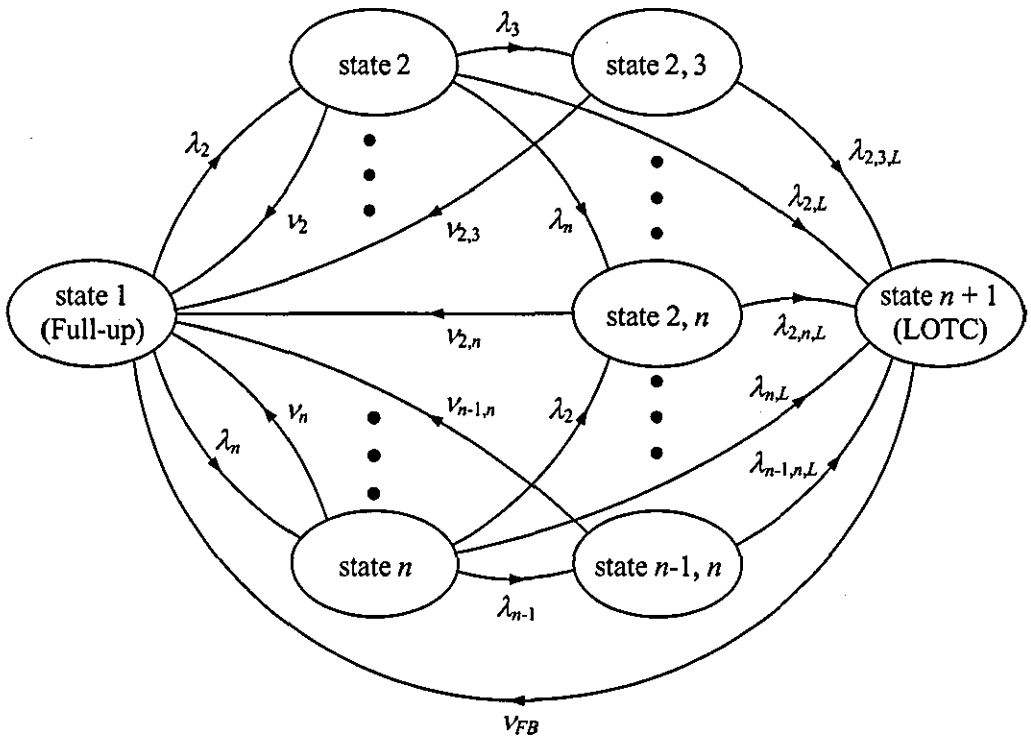


Figure 5.5.2. A general dual fault state Markov model.

Figure 5.5.2 may be taken to infer that all combinations of dual faults should be included in the model. This is not intended to be the case. In reality, a reduced number of dual fault states may be included. It should also be noted that Figure 5.5.2 shows a maintenance approach wherein all faults are cleared from the system at maintenance times. Thus all repair rates, ν_i and $\nu_{i,j}$ lead to the full-up system state, state 1. Also note that the failure rates due to mechanical/hydromechanical faults have not been included on the diagram. This is purely in the interests of clarity, the diagram becomes more cluttered when these are included. However, these failure rates could easily be added to the system and they would lead from each of the dispatchable system states directly to the LOTC state, as was the case in the single fault state Markov model in Figure 5.5.1. Taking the general model depicted in Figure 5.5.2, it is possible to obtain a transition rate matrix for this system, which is given in Figure 5.5.3. The diagonal terms of the matrix contain a term D_i or $D_{i,j}$, which take the following values:

$$D_k = \begin{cases} \sum_{i=2}^{n} \lambda_i & k = 1, \\ \nu_k + \sum_{\substack{i=2 \\ i \neq k}}^{n} \lambda_i + \lambda_{k,L} & k = 2, \ldots, n, \\ \nu_{x,y} + \lambda_{x,y,L}, & k = (2,3), \ldots, (n-1, n) \end{cases} \qquad 5.5.28$$

x and y represent the single faults that occur together to cause a dual fault. The LOTC rate of the system obtained from the dual fault model is calculated as it was for the single fault state model, by considering the failure rate into state $n + 1$, which is defined as:

$$\lambda_{DFS} = \frac{\text{Transition rate } \textit{into} \text{ state } n+1}{1 - \text{Probability of being in state } n+1}, \qquad 5.5.29$$

which will be given by:

$$\lambda_{DFS} = \frac{\sum_{i=1}^{n} \lambda_{i,L} Q_i + \sum_{i=2}^{n} \sum_{j=i+1}^{n} \lambda_{i,j,L} Q_{i,j}}{1 - Q_{n+1}}$$

$$= \frac{\lambda_{1,L} + \sum_{i=2}^{n} \lambda_{i,L} \frac{Q_i}{Q_1} + \sum_{i=2}^{n} \sum_{j=i+1}^{n} \lambda_{i,j,L} \frac{Q_{i,j}}{Q_1}}{1 + \sum_{i=2}^{n} \frac{Q_i}{Q_1}}, \qquad 5.5.30$$

where $Q_{i,j}$ is the probability of being in the dual fault state involving single faults i and j. These quantities can be obtained from the transition rate matrix, from the columns representing the dual fault states, and are given by the general equation:

$$\lambda_i Q_j + \lambda_j Q_i - D_{i,j} Q_{i,j} = 0, \qquad 5.5.31$$

which is rearranged to give:

$$Q_{i,j} = \frac{\lambda_i Q_j + \lambda_j Q_i}{D_{i,j}}. \qquad 5.5.32$$

Equation 5.5.32 can then be substituted into equation 5.5.30 to give a general dual fault state model LOTC rate. Since the dual fault state Markov model being considered is such a general one, it is difficult to continue with any meaningful analysis. The number of states in the model and subsequent analysis depends on the system being modelled.

$$
\begin{array}{c|c}
 & \begin{array}{cccccccccccccc}
\text{Full-Up} & \text{Single Fault States} & & & & & \text{Dual Fault States} & & & & & & & \text{LOTC} \\
1 & n-1 & & & & & n-2 & & & & & 2 & 1 & 1
\end{array} \\
\begin{array}{cc}
\text{Full-Up} & 1 \\
\text{Single Fault States} & n-1 \\
\text{Dual Fault States} & n-2 \\
 & 2 \\
 & 1 \\
\text{LOTC} & 1
\end{array}
&
\begin{bmatrix}
-D_1 & \lambda_2 & \lambda_3 & \cdots & \cdots & \lambda_n & 0 & \cdots & \cdots & 0 & \cdots & 0 & 0 & 0 & 0 \\
\nu_2 & -D_2 & 0 & \cdots & \cdots & 0 & \lambda_3 & \lambda_4 & \cdots & \lambda_n & \cdots & \vdots & \vdots & 0 & \lambda_{2,L} \\
\vdots & 0 & \ddots & 0 & & \vdots & \lambda_2 & 0 & \cdots & 0 & \cdots & 0 & 0 & \vdots & \vdots \\
\vdots & \vdots & 0 & \ddots & 0 & \vdots & 0 & \ddots & 0 & \vdots & \cdots & \lambda_{n-1} & \lambda_n & 0 & \vdots \\
\vdots & \vdots & & 0 & \ddots & 0 & \vdots & 0 & \lambda_2 & 0 & \cdots & \lambda_{n-2} & 0 & \lambda_n & \vdots \\
\nu_n & 0 & \cdots & \cdots & 0 & -D_n & 0 & \cdots & 0 & \lambda_2 & \cdots & 0 & \lambda_{n-2} & \lambda_{n-1} & \lambda_{n,L} \\
\nu_{2,3} & 0 & \cdots & \cdots & \cdots & 0 & -D_{2,3} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \lambda_{2,3,L} \\
\vdots & \vdots & & \cdots & & \vdots & 0 & \ddots & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & & \cdots & & \vdots & \vdots & 0 & \ddots & 0 & \vdots & \vdots & \vdots & \vdots & \vdots \\
\nu_{2,n} & 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 & -D_{2,n} & 0 & 0 & 0 & 0 & \lambda_{2,n,L} \\
\vdots & 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 & \ddots & 0 & \vdots & \vdots & \vdots \\
\nu_{n-2,n-1} & 0 & \cdots & \cdots & \cdots & 0 & \cdots & \cdots & \cdots & 0 & 0 & \ddots & 0 & \vdots & \lambda_{n-2,n-1,L} \\
\nu_{n-2,n} & 0 & \cdots & \cdots & \cdots & 0 & \cdots & \cdots & \cdots & 0 & \cdots & 0 & \ddots & 0 & \lambda_{n-2,n,L} \\
\nu_{n-1,n} & 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 & -D_{n-1,n} & \lambda_{n-1,n,L} \\
\nu_{FB} & 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 & 0 & -\nu_{FB}
\end{bmatrix}
\end{array}
$$

Numbers in this row give the number of state transitions represented in a particular section of a row of the matrix. Likewise, the numbers in the column opposite give the number of transitions represented in a particular section of a column.

Figure 5.5.3. The transition rate matrix for the general dual state Markov model shown in Figure 5.5.2.

5.6 Summary

During the course of implementing the modelling techniques described in this chapter a number of approximations and assumptions are made. These are summarised below for each method.

5.6.1 Approximations/Assumptions Involved in the TWA Approach

- The failure rate due to mechanical/hydromechanical faults, λ_{HMC}, is approximated by adding all of the failure rates of all individual mechanical/hydromechanical faults. This would appear to be a reasonable assumption.
- The failure rate for unrevealed faults, λ_{UR}, is approximated in a way that could prove to be prone to error (see equation 5.3.2). A constant, X, is used to represent the fact that the unrevealed fault rate of a channel generally lies between 0 and 5% of the total LOTC rate of that channel. This constant could potentially be manipulated in order to try to demonstrate a more favourable LOTC rate of the system. It is difficult to prove the suitability of the value chosen for X.
- The failure rate for full-up electronics, $\lambda_{FU,L}$, is approximated using a conservative upper bound. It is conservative since all channel faults when dispatching full-up are assumed to cause the loss of that channel. It is also conservative since during the calculation of the upper bound used the probability of both channels failing is approximated using the rare event approximation (see equation 5.3.19). Another point to note about this approximation is the fact that failure rates are approximated by dividing failure probabilities by the length of a flight in order to get the probability of failure per flight hour. This technique is also used when calculating the failure rates to LOTC from the ith dispatchable configuration, $\lambda_{i,L}$. This has the potential to lead to problems.
- The definitions of the average failure rates to LOTC from the STD and LTD dispatchable system configurations, $\lambda_{STD,L}$ and $\lambda_{LTD,L}$, are given in [17], [18] with little explanation as to why they take the values given. From Section 5.4, these definitions can be seen, when used in conjunction with the balanced fractional

coefficients, equation 5.3.15, to yield an expression for the TWA LOTC rate that contains a term representing the TWA of the failure rate to LOTC of the system from each of its dispatchable configurations. This would suggest that the definitions of the average failure rates to LOTC from the STD and LTD dispatchable system configurations are reasonable.

5.6.2 Approximations/Assumptions Involved in the Single Fault State Markov Approach

- Constructing a single fault state Markov model involves making a vast reduction in the number of states in the Markov model in comparison to a conventional Markov approach. This will lead to some reduction in the accuracy of any solutions obtained from the model. It is difficult to predict whether this reduction in states and the ensuing approximations taken will lead to a conservative solution or not. When so many states are removed from a Markov model it is possible that the Markov model might no longer represent the situation that is to be modelled.
- The feedback repair transition, ν_{FB}, that is added to the model in order to provide a path from the final fully-failed system state back to the full-up state with no faults present at first appears unconventional, particularly when one considers the fact that the value of this rate is said to be unimportant. However, this rate is added to the model in order for the fleet-wide average LOTC rate to be obtained and thus the feedback rate can be thought of as taking the place of the maintenance required in replacing or repairing a FADEC system after a LOTC event has occurred.
- The single fault state Markov model described considers the case where all faults will be cleared from the system at maintenance. The model could be adapted to cover other maintenance strategies. However, this requires the construction of a new model and the work involved in doing this may not be a trivial matter. Also note that Markov models inherently model the MEL maintenance approach, and as such require the manipulation of the dispatch interval to find the maximum allowable inspection interval if PIR is to be used.
- Approximations may be used in the single fault state Markov model, such as the one yielding the first order approximation to the single fault state Markov model (equation 5.5.24) or the approximation applied when dispatch intervals are short (equation 5.5.27). These approximations, when applied to a method that already

involves some substantial approximations, may bring further inaccuracies into the modelling process.

5.6.3 Approximations /Assumptions Involved in the Dual Fault State Markov Approach

- The number of fault states is again reduced for this dual fault state model in comparison to the number of states that would be present in a conventional Markov model. Because of the increase in the number of system states one may expect greater accuracy from the model in relation to the single fault state model. However, a loss in accuracy of the model would still be expected, and it is unclear whether any solutions obtained would be conservative or not.
- In order to use a *single* fault state Markov model a certain requirement must be met. The requirement is for the repair rates of individual fault states to be at least 10 times greater than the failure rates into or out of the single fault states. If this is not the case a dual fault state Markov model would be used. The value of 10 seems to be an arbitrary choice. It is unclear whether this requirement is a valid one.
- An artificial feedback loop is added to the dual fault state model, as was the case with the single fault state model. This feedback rate can be thought of as being applied under the same assumptions that held for the single fault state Markov model.
- As was the case with the single fault state Markov approach the dual fault state Markov model inherently models the MEL maintenance approach and requires the manipulation of the dispatch interval to find the maximum allowed inspection interval.

Chapter 6 : TLD Simulation Modelling Methodology

6.1 Introduction

TLD is essentially a very simple process in which maintenance is performed on a system within a set period of time after a fault has occurred. However, given the complexity of modern aircraft systems the application and modelling of TLD is not necessarily a simple task. As discussed in 0 the dispatch intervals that are applied to the different fault categories must be such that the system LOTC rate doesn't exceed the level set in the certification regulations. In addition, there is a further requirement that the computed LOTC rate of the system from individual dispatchable system configurations must not exceed specified levels also set in the certification regulations. These levels vary for each of the different categories of dispatch. When one also begins to take into account, for example, the presence of multiple faults within the system at any one time or the application of, say, MEL maintenance to STD faults and PIR maintenance to LTD faults, the modelling process can become more complex. Another item of potential importance is the ability to model different repair scenarios to show that they will, if applied, meet the certification requirements for given dispatch intervals.

Monte Carlo Simulation (MCS) lends itself very well to modelling the application of TLD to simple systems. Fault times can be added to a schedule, giving the chronological order in which the faults occur, with the system being checked after each fault to see if it has failed or if TLD criteria should be applied. MCS is sufficiently flexible to deal with the intricacies of TLD. For instance, a MCS can handle the occurrence of multiple faults within the system and the subsequent application of TLD and associated maintenance operations. The ordering of fault combinations can be accounted for, as can different repair strategies at maintenance. Perhaps the main advantage of a MCS code specifically designed to model TLD is the potential to model more than one system using the same code. This would be of particular use, for example, if a system design was changed at some point in time. If using an analytical technique to demonstrate certification compliance a totally new model must be

constructed for the system. If using MCS a single code could be used for any number of systems, the system structure being passed to the code, along with details about the dispatch intervals to be applied and the maintenance strategies to be used. Having in mind this idea of the potential functionality of a MCS, a number of characteristics that a MCS code for TLD should demonstrate can be outlined:

1. the ability to model any system to which TLD is applied,
2. handle different maintenance approaches and repair strategies,
3. deal effectively with the potential importance of the order of occurrence of faults and, more generally, fault combinations,
4. most importantly, demonstrate with some confidence whether or not a system will meet certification requirements.

Software was written in order to model TLD using MCS. The above four points were the main ones taken into account when constructing the MCS code. The code is written in C++ in such a way that it is as general as possible, allowing a potential user to easily apply it to their system of interest. The limitations of the other approaches, such as the requirement of basic event independence of fault tree analysis and the state space explosion encountered in Markov analysis, are overcome by using the MCS modelling approach developed here.

In this chapter important details of the TLD simulation modelling methodology and the computer code produced to perform the analysis are discussed. An overview of the TLD MCS code structure is given, more details of which can be found in Appendix 2. This is followed by the more important details of the TLD MCS code, beginning with a description of the schedules used within the code and continuing with a description of the method used to handle TLD criteria within the code. The method of propagating faults (and repairs) through the fault tree structure is described, along with how the different repair strategies at maintenance deadlines are handled and how opportunistic maintenance is performed. The method used to determine the times of deadlines added to the schedule for differing maintenance approaches is described. Finally, the output available from the code is described. All of these details of the code have been developed as part of the work for this thesis and provide the backbone of the Monte Carlo simulation methodology used to model TLD. This methodology overcomes the

limitations of the other possible techniques, such as the requirement of basic event independence in fault tree analysis, the state space explosion encountered when using Markov models or the number of approximations and assumptions used in the recommended models.

6.2 TLD MCS Code Outline

When modelling a system using the TLD MCS code there are a number of inputs that must be supplied. Figure 6.2.1 shows a schematic of the inputs supplied to the TLD MCS code and the outputs that can be obtained from it. The inputs that describe the system are the fault tree structure that describes the LOTC of the system and the component data for this fault tree. The component data are the failure distributions and parameters for the basic events of the fault tree. In addition the dispatch criteria for the system must also be provided. All of these inputs are provided to the system in text files, examples of which are given in Appendix 1 and Appendix 5. The fault tree file contains a list of the gates in the tree followed by their inputs. The basic event failure data file contains a list of the basic events, each of which has an identifier for its distribution and the parameter for that distribution associated with that basic event. The dispatch criteria for the system are defined in another text file. These may be described in terms of basic events or, as perhaps might be more likely in a real system, in terms of intermediate events from the system fault tree or, if required, as some combination of basic and intermediate events. If the TLD criteria are to be specified in terms of intermediate events then the appropriate intermediate events are also supplied to the code in a text file. The final text file that may be specified is one which comes into force as maintenance takes place. This file contains a list of any TLD faults that will be repaired and will always involve other faults being cleared from the system. For example, it might be the case that when a certain part of a system is repaired other parts of that system will always be repaired at the same time, if failed.

Given the system information described above the TLD code performs a Monte Carlo simulation procedure, producing the system LOTC rate for given values of dispatch intervals. The code can also produce the instantaneous failure rates to LOTC and can also be used to give importance measures for the basic events in the system fault tree and also the faults that are considered for the dispatch criteria.

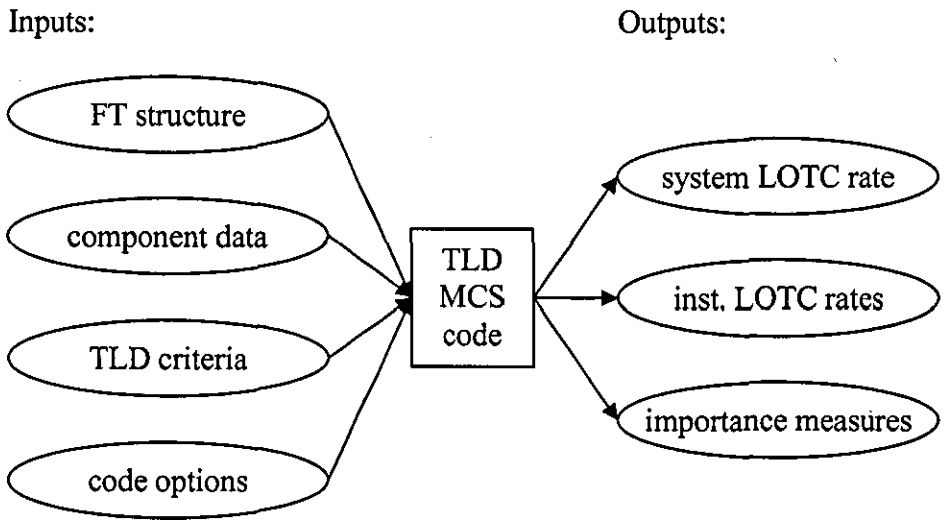


Figure 6.2.1. Inputs and outputs of the TLD MCS code.

6.3 TLD MCS Code Structure

The TLD MCS code has a modular format. Along with relatively simple modules, such as one that is used to generate fault times, see Section 4.4, or one used to construct names for the files used for data output, there are a number of modules that form the core of the TLD MCS code. These cover the main TLD operations applied to the system and the handling of the relevant system data during the course of the simulations. These major modules are listed (in no particular order), along with basic descriptions of the functions that they perform, in Table 6.3.1. Along with these modules the TLD MCS code requires an input file, to be specified by the user before running the code. This data file contains options that allow certain features of the code to be 'turned on' or 'turned off' according to the results and output required and also contains certain constants or variables that will be used by the code. An example of such a file is given in Appendix 1.

The modules and options specified in the input data file are described in Appendix 2. Points of particular relevance to the simulation of TLD are described in more detail than those considered to be not as important to modelling TLD.

Module/s	Function/s
main	Main driver program.
auto_set_tld_criteria	Called from *main* when dispatch criteria are to be set automatically within the code.
dispatch_configuration	Identifies the system dispatchable configurations, for which instantaneous LOTC rates must be calculated.
vary_user_options	Group of modules used to ready information supplied by the user for use in a simulation of a system, e.g. steps through a range of STD intervals.
initialisation	Contains a group of modules used to initialise data at different points in the simulation process.
control_sims	Sets data according to user options (using *vary_user_options*) and calls *simulations* module.
simulations	Implements a group of simulations to find the system LOTC rate for a certain set of user options (supplied by *control_sims*).
schedule	Handles data regarding to fault ordering (for both faults that have yet to occur and faults that have already occurred).
failure_update	Upon fault occurrence, propagates faults through the system fault tree.
component_repair	Updates the relevant data as a fault is cleared from the system.
repair_update	Propagates repairs through the system fault tree.
LOTC_i	Amends data used to calculate the instantaneous LOTC rates from the dispatchable system configurations.
TLD_check	Checks to see whether any TLD criteria must be applied after a fault occurrence.
TLD_maintenance	Decides the correct faults to be repaired at a TLD maintenance deadline.
LOTC_convergence *IMP_convergence*	Check for convergence of the system LOTC rate, instantaneous LOTC rates from the dispatchable system configurations and importance measures.

Table 6.3.1. Major modules of the TLD MCS code and outlines of their functions.

6.4 Event Scheduling

Schedules play a very important role in the TLD MCS code. It is important that the times of faults and maintenance deadlines are used by the code in a strict chronological order. In this way the simulations can move from one event occurrence (fault or

maintenance deadline) to the next, the system status being checked at the time of each event. As these event times are generated from the appropriate distributions (for fault times) or calculated (for maintenance deadlines) they are added to a schedule. From this schedule it must, at various times throughout the implementation of the TLD MCS code, be possible to identify the first event chronologically or to find the position of certain events within the schedule.

There is a second use for a schedule within the TLD MCS code. A schedule with the same structure described above is used to keep track of the faults that are present within the system and the order in which they occurred. Thus, two schedule structures are used, the first records the order of the events that are *still to occur*, the second records the order of faults that *have already occurred* and are still present in the system. Both of these schedules use identical structures. The schedule that records the order of occurrence of faults that are present in the system is used when implementing some of the repair strategies to enable maintenance. The different repair strategies at maintenance are discussed in Sections 6.7 and 6.8. The remainder of this section describes the schedule as it is applied in the case where a record is kept of the events that are still to occur. Exactly the same principles are used in the case where a record is kept of the events that have already occurred.

The schedule used in the TLD MCS code is constructed around two main array structures. One is an integer array that holds reference numbers of events that will occur at some point in the system's future, and the other an array that holds the time that these events are due to occur. For an illustration of how such arrays are represented consider Table 6.4.1. Entries with a dash signify empty elements of arrays. The left hand column represents the index of the array. There are a number of events that may occur, these being stored in the central column, and the times at which these events will occur, given in the right hand column. Upon examination of this particular schedule it can be seen that the event with identifier 6000 will occur first at a simulation time of 965.0, to be followed by event 2 at 1053.2, event 1 at 1210.4 and event 7003 at 1425.0. Note that these events are not in chronological order. Clearly, if the schedule could only hold five entries such as the one in Table 6.4.1 it would be no problem to search the table to determine the event order. However, if the schedule was an array of hundreds or thousands of events the task of finding the first event in the schedule time after time

Array Index	Event Identifier	Event Time
0	2	1053.2
1	-	-
2	7003	1425.0
3	1	1210.4
4	6000	965.0
5	-	-

Table 6.4.1. A representation of schedule arrays.

would become particularly draining on system resources. For this reason the schedule structure is extended to contain other information for each event, which refers to the position within the array of the next and previous events chronologically. This information is stored in two extra integer arrays. Thus, the schedule represented in Table 6.4.1 will take the form given in Table 6.4.2. This information now allows the addition of or removal of events and times from any position within the schedule and allows a record to be kept of the relative positions of all events in the schedule array.

Array Index	Event Identifier	Event Time	Position of previous event	Position of next event
0	2	1053.2	4	3
1	-	-	-	-
2	7003	1425.0	3	-
3	1	1210.4	0	2
4	6000	965.0	-	0
5	-	-	-	-

Table 6.4.2. A schedule representation with reference to previous and next events.

6.4.1 Removing the First Event From the Schedule

In order to illustrate the use of a schedule within the code consider initially the task of removing the first event from a schedule, a procedure that is carried out many times during the implementation of simulations. Considering the example just discussed the first event is known to be stored at position 4. Thus the event and its associated time are removed from position 4 in the appropriate arrays. In this way the occurring event would be 6000 and would occur at a time of 965.0. The simulation time would be advanced to 965.0 and the relevant procedures to be implemented for event 6000 would be carried out if a TLD MCS was being undertaken. As the event and time were

removed from the schedule the arrays storing the position of the next and previous events must be updated, along with the variable storing the position of the first event chronologically in the schedule. Since 0 is the position of the event that occurred after event 6000 this would now be the first event. At this point all elements of arrays at index 4 can be cleared. The new first event, 2, at position 0 now has no event previous to it (since it is the first event), hence the element of the previous element array at position 0 will be cleared. Thus the schedule arrays will now take the form given in Table 6.4.3. This procedure can be implemented repeatedly until the schedule is emptied. Note that the schedule being used to store events that are to occur should never be emptied, since there will always be faults or maintenance deadlines that will occur at some time in the future. However, the array used to store the order of faults that have occurred may be emptied completely, since there will be numerous times during the implementation of the code that there will be no faults present in the system.

Array Index	Event Identifier	Event Time	Position of previous event	Position of next event
0	2	1053.2	-	3
1	-	-	-	-
2	7003	1425.0	3	-
3	1	1210.4	0	2
4	-	-	-	-
5	-	-	-	-

Table 6.4.3. A representation of the schedule after the removal of event 6000.

6.4.2 Adding an Event to the Schedule

Adding an event to the schedule is a slightly more complex procedure than removing the first event from the schedule. The event and the time of its occurrence are added to a free element of the relevant arrays. In the TLD MCS code the first element that is checked for its availability is the last element that had an event removed from it. In order to do this a variable is used to store the array index of the previous event that was removed from the schedule. If the array elements at this position are not free for some reason the arrays are searched from that point, element by element, until a free position is found. If the end of the array is encountered while it is being searched the search restarts at the beginning of the array. Once free positions are found the event identifier and its time of occurrence are stored at that position in the relevant arrays. The position

of the event in the schedule arrays is also stored in the relevant array. The next step is to update the arrays containing the positions in the schedule of the next and previous events. This involves comparing the time of the event being added to the schedule to the times of the events that are already in the schedule. The search of the event times for comparison with the time of the added event begins with the time of the last event chronologically. If the added event occurs later than the final event it will become the new last event chronologically. If not the time of the previous event to the last event is checked and if the added event occurs between these two events the position is known. This process continues until the position of the added event is known and then the arrays are updated.

In order to illustrate the process of adding an event to the schedule consider that event 0 is to be added to the schedule represented in Table 6.4.3 and that the time of its occurrence is 1373.7. The first task is to locate a free element. Considering that event 6000 was just removed from position 4 the first element checked for its availability has index 4. It is discovered to be empty and the event and its time are added to position 4 of the relevant arrays. Next the location of the event in the chronological ordering scheme must be found. As the time of the event is not greater than that of the last event chronologically (event 7003) the time of the previous event to that last event (event 1) is compared to the time of the added event. Event 0 is found to occur between these two events. Thus the data from the schedule is as given in Table 6.4.4. The first event chronologically will still be at position 0, the final event will still be at position 2 of the array. Note that the position of the previous event to event 7003 has now been updated to 4 and the position of the next event to 3 has been updated to 4. At the same time the positions of the events occurring before and after the added event 0 are 3 and 2 respectively.

Array Index	Event Identifier	Event Time	Position of previous event	Position of next event
0	2	1053.2	-	3
1	-	-	-	-
2	7003	1425.0	4	-
3	1	1210.4	0	4
4	0	1373.7	3	2
5	-	-	-	-

Table 6.4.4. A representation of the schedule after the addition of event 0.

6.4.3 Removing an Event From Any Position In the Schedule

This procedure is carried out in much the same way as that which is used to remove the first event chronologically from the schedule. This is a simple process so long as the positions of previous and next events are updated accordingly.

Suppose that event 1 is to be removed from the schedule, despite the fact that it is not the first event chronologically. This scenario can occur when faults are repaired. In order to do this the position of the event in the array is obtained from the relevant array that is used to store the positions of all the events in the schedule array. After identifying that it is in position 3 of the schedule arrays the entry for the event identifier and its time are cleared from the array and the arrays storing the positions of the next and previous events are updated. The schedule will now look like the representation in Table 6.4.5. Note that the position of the next event to the event that was previous to event 1, i.e. event 2 at position 0, is now updated to 4. Also, the previous event of event 4 is now changed to event 2 which is entered in the schedule at position 0.

Array Index	Event Identifier	Event Time	Position of previous event	Position of next event
0	2	1053.2	-	4
1	-	-	-	-
2	7003	1425.0	4	-
3	-	-	-	-
4	0	1373.7	0	2
5	-	-	-	-

Table 6.4.5. Representation of the schedule after the removal of event 1.

6.4.4 Event Times Added to the Schedule

There are two types of events that may be added to the schedules in the TLD MCS code. These are faults and maintenance deadlines. The schedule that records the events and times that are yet to occur within the system will, at various points, have faults and maintenance deadlines added to it. When this happens the time of the faults must be generated using the appropriate failure distributions, see Section 4.4, and the time of the maintenance deadlines must be calculated according to the maintenance procedures being implemented, see Section 6.8. The second type of schedule used within the code, that which records the faults and times at which they occurred, uses the times removed from the first schedule type. This means there is no generation or calculation of event times needed for the second type of schedule, only for the first.

6.5 Handling System Dispatch Criteria

When simulating the application of TLD to a system one must efficiently handle the dispatch criteria that will be applied to that system. These dispatch criteria will involve either an individual fault or some, perhaps ordered, combination of faults. To illustrate this consider an example where a fault A, occurring in isolation, initiates a LTD interval. The further occurrence of fault B after fault A may cause the allowable dispatch interval to be reduced to the STD category. For the moment ignore the occurrence of fault B alone or the occurrence of fault B followed by fault A. In this case the dispatch criteria would be defined as shown in Table 6.5.1.

Dispatch Criteria	Dispatch Category
A	LTD
AB	STD

Table 6.5.1. Dispatch criteria for two faults, A and B.

In order to include a representation of these dispatch criteria within the TLD MCS code one must first identify the important fault within the group of faults that causes the criteria to be applied. Consider how the TLD MCS code works, by removing faults and other events from a schedule in a chronological order. Consider also how these dispatch criteria are defined, in terms of an ordered group of faults. Following this thought process, one may reach the conclusion that the final fault of the group of faults causing

the implementation of certain dispatch criteria is the most important. In this way, the final fault of the group may be considered as a kind of initiating event for the group of faults that brings about the implementation of certain dispatch criteria. The preceding events could be thought of as being analogous to enabling events. Thus, it is reasonable to use these dispatch criteria within the code by checking, as a fault occurs, for other faults that preceded it to see if any dispatch criteria have been met. In this way the dispatch criteria for the two faults, A and B, given in Table 6.5.1 would be stored in the TLD MCS code under a reference to the final fault of the group of faults. The dispatch criteria shown in the table are stored referring to fault A (for the fault involving A alone) and fault B (for the dual fault condition). Therefore the dispatch criteria, as stored in the code, take the form given in Table 6.5.2. Note that the dash in the table signifies that when A is the final fault there are no preceding faults required for the initiation of the LTD interval. Fault B must be preceded by fault A in order for the STD interval to be initiated.

Final Fault (used as reference)	Preceding Faults	Dispatch Category
A	-	LTD
B	A	STD

Table 6.5.2. A representation of dispatch criteria as used in the TLD MCS code

In order to further illustrate this type of representation of the dispatch criteria consider a second example for dispatch criteria, shown in Table 6.5.3. In this case there are three faults covered by these dispatch criteria, these being A, B and C. The occurrence of fault A or fault B in isolation initiates a LTD category. Further failures give rise to the application of STD and DND dispatch categories.

Dispatch Criteria	Dispatch Category
A	LTD
AB	STD
ABC	DND
B	LTD
BA	STD
BC	STD

Table 6.5.3. Dispatch criteria for three faults, A, B and C.

Now consider the representation of these criteria that must be used in the TLD MCS code. Since the final event of the group that triggers the initiation of a dispatch interval is the one by which the group will be referenced there will be two criteria referenced by fault A, two by fault B and two by fault C. The representations used in the code will therefore be those given in Table 6.5.4.

Final Fault (used as reference)	Preceding Faults	Dispatch Category
A	-	LTD
	B	STD
B	A	STD
	-	LTD
C	AB	DND
	B	STD

Table 6.5.4. A representation of dispatch criteria as used in the TLD MCS code.

The dispatch criteria supplied to the TLD MCS code must cover all combinations of faults that can occur within a system. This does not mean that a dispatch category must be applied to every possible combination of faults. There will be certain faults and combinations for which dispatch is forbidden (DND faults) and any higher order fault combinations that contain the lower order DND fault will automatically be DND themselves. As such the higher order faults just described need not be included in the dispatch criteria since, in all situations, the lower order DND fault will cause the implementation of the relevant DND maintenance deadline. In this way there comes a point when dealing with dispatch criteria, where all fault combinations of a certain order and above will be DND. As the dispatch criteria are input by the TLD MCS code this number is stored for later use when determining applicable dispatch categories and setting maintenance deadlines.

6.6 Propagating Faults/Repairs Through the Fault Tree Structure

The occurrence and subsequent repair of faults within the TLD MCS code is fundamental to its success in modelling TLD. As a fault occurs the *failure_update* module is called to update the status of the relevant gates and events within the system fault tree by propagating the original fault up through the fault tree. In a similar way, as

a fault is cleared from the system at maintenance deadlines the status of the relevant gates and events in the system are updated by propagating the repair up through the fault tree structure. This task is performed in the code using the *repair_update* module.

Propagating faults or repairs up through the structure of a fault tree is a relatively simple task. The TLD MCS code first identifies the parent gates of the fault that has occurred or is being repaired. It then checks whether the status of the parent gate will change (working to failed or failed to working) due to the change in status of its input. Whether the status of the parent gate changes is dependent on the type of the gate and sometimes the status of its other inputs. If the status of the parent gate changes its own parent gates are then checked, to see if their status changes and this process continues until the status of no more parent gates will change (if, following a fault occurrence, one of the parent gates is the top event and it fails the system will clearly fail). Table 6.6.1 shows the requirements for a change in status of the parent gate following a fault occurrence (if the gate is to change status from working to failed) and the requirements for a change in status of the parent gate following a fault repair (if the gate is to change status from failed to working). Note that, because the failure of a NOT gate's input causes the 'repair' of the NOT gate, the NOT gate identity is passed to the *repair_update* module from the *failure_update* module in order to propagate the logic up the fault tree. Similarly, when the NOT gate's input is repaired, the NOT gate fails and thus *failure_update* is called from *repair_update* in order to propagate the appropriate logic.

Gate type	After fault occurrence	After fault repair
OR	1 input failed causes gate failure.	All inputs working cause gate to work.
AND	All inputs failed cause gate failure.	1 input working causes gate to work.
k/*n* voting	*k* inputs failed cause gate failure.	$n - k + 1$ inputs working cause gate to work.
priority AND	All inputs failed in order cause gate failure.	1 input working causes gate to work.
NOT	Input failure causes gate repair.	Input repair causes gate failure.
null	Input failure causes gate failure.	Input repair causes gate repair.

Table 6.6.1. The requirements for a change in status of the parent gate when propagating faults/repairs through the fault tree (used in *failure_update*, *repair_update*).

6.7 Implementing Maintenance Within the TLD MCS Code

If, during the process of a simulation, a maintenance deadline is encountered in the schedule, repairs must take place to rectify certain faults within the system before further dispatch is allowed. The nature of the repairs to be carried out at maintenance deadlines during the course of a simulation determines which deadlines are added to the schedule after a fault occurrence. The addition of maintenance deadlines to the schedule is discussed in Section 6.8.

If, upon reaching maintenance deadlines, there would always be only one fault present within the system, this would simply be cleared to allow further dispatch. However, since it is possible for more than one fault to exist within the system at maintenance deadlines it is necessary to have a repair strategy that will decide what faults must be repaired as a maintenance deadline is encountered. In an ideal world, one would probably require that all faults could be cleared from the system at the maintenance deadline, returning the system to a fully-up state and allowing subsequent indefinite dispatch of the aircraft. However, in reality, it may be the case that, despite the best planning of maintenance crews, replacement parts may not be available in time, or sufficient time is not available to implement all the repairs before the aircraft is due to enter service again. Provided that the faults that cannot be cleared from the system in time are not the ones that directly caused this maintenance deadline (and hence may remain in the system for some further period of time) the aircraft could be dispatched again and the final repairs effected at some later date. For this reason a number of repair strategies at maintenance deadlines were included in the TLD MCS code.

As mentioned earlier, in practice perhaps the most likely maintenance strategy would involve clearing all faults from the system at the first opportunity, the first maintenance deadline that arises. However, as was also briefly discussed, this may not happen for some reason, for instance an unavailability of replacement parts or lack of maintenance time. There will clearly be varying amounts of repairs that can be implemented at maintenance deadlines. Four different repair strategies at maintenance deadlines may be implemented in the TLD MCS code. Each of these strategies was devised to represent varying amounts of work as the maintenance procedure is carried out. This ranged from implementing the minimum amount of maintenance possible at a deadline that will still

allow further dispatch of the aircraft, to doing the maximum amount of work possible (basically clearing all faults from the system at maintenance deadlines). Which of the four repair strategies to use at maintenance deadlines is a choice specified in the input data. Each of these repair strategies is described in the following sections, with examples of the concepts involved, where appropriate. The repair strategies themselves are listed below:

1. Repair the last fault of the group that caused the current maintenance deadline,
2. Repair all faults in the group causing the current maintenance deadline,
3. Repair all faults in the same category as the group causing the current deadline,
4. Repair all faults present in the system.

Also possible within the TLD MCS code is opportunistic maintenance, wherein repairs that would be carried out at future maintenance deadlines are carried out at the present maintenance deadline if they would occur within a specified length of time. This is described in Section 6.7.5.

It is also possible within the TLD MCS code to associate certain fault tree events (basic or intermediate) with faults that appear in the dispatch criteria. This allows repairs to be carried out to relevant parts of the system as particular dispatchable faults are repaired. This is described in Section 6.7.6.

6.7.1 Repair Strategy 1 – Repair the Last Fault of the Group Causing the Current Maintenance Deadline

This repair strategy involves carrying out the least possible amount of work at a maintenance deadline in order to allow further dispatch of the system. For this reason it is probably the worst possible maintenance strategy that could be applied in practice. When maintenance deadlines are encountered during a simulation the final fault in the group that caused the deadline to be added to the schedule is repaired. In order to illustrate this repair process consider the occurrence of three faults, A, B and C, the dispatch criteria for which are shown in Table 6.7.1.

Dispatch Criteria	Dispatch Category
A	LTD
AB	STD
BC	DND

Table 6.7.1. The dispatch criteria for 3 faults, A, B and C.

Considering the dispatch criteria given in Table 6.7.1, if the faults occur in the order A followed by B followed by C the situation could resemble that depicted in Figure 6.7.1. Fault A initiates a LTD interval ending at the maintenance deadline at t_3. Fault B occurs and, in combination with fault A, initiates a STD interval ending at the maintenance deadline at t_2. Fault C, as it occurs, acts in combination with B and initiates a DND maintenance deadline at t_1. Note that it is assumed that the maintenance deadlines are assumed to have been adjusted to occur between flights and that the DND deadline at t_1 is set at the end of the flight in which fault C occurs.

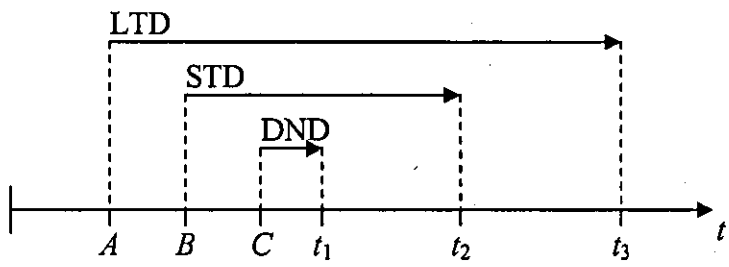


Figure 6.7.1. The occurrence of A, B and C and the setting of maintenance deadlines.

Now, when employing this repair strategy, repairs will be implemented at the deadlines t_1, t_2 and t_3 in the following way. Upon reaching time t_1 the group of faults that caused the deadline is known to be BC. The last fault of the group is identified as fault C, which is repaired, and the system may be dispatched again until time t_2. Faults A and B are still present within the system. At time t_2 the group of faults causing the deadline is identified as AB and the last of these faults to occur, fault B, is repaired. This allows dispatch until time t_3 with fault A still present within the system. At this point the group of faults causing the deadline consists of A alone. This is repaired and the system may now be dispatched indefinitely, until further faults occur.

The situation described above for the three faults, A, B and C, was relatively simple to work through because of the MEL maintenance approach that was being employed for

all of the faults. A situation that could arise and complicate the process when using this repair strategy happens when the PIR maintenance approach is used. In this case more than one group of faults may bring about the initiation of a maintenance deadline. In this case the final fault of all the groups causing the deadline would be repaired in the TLD MCS code. In order to illustrate this consider a situation in which four faults, D, E, F and G, have the dispatch criteria shown in Table 6.7.2.

Dispatch Criteria	Dispatch Category
DE	LTD
FG	LTD

Table 6.7.2. The dispatch criteria for 4 faults, D, E, F and G.

Consider the case where these faults occur in the way depicted in Figure 6.7.2 and LTD category faults are maintained using the PIR maintenance approach. If D, E, F and G all occur between consecutive inspections I_1 and I_2 then they will be discovered at I_2 and assumed to have occurred at t_m, giving a maintenance deadline at t_1. As t_1 is reached, faults E and G will be cleared from the system, since this will involve repairing the last fault of each group associated with the deadline and will allow further dispatch of the system. Faults D and F will remain in the system, to be repaired at some other time that will be related to some other dispatch criteria that are applied to the system in addition to the criteria given in Table 6.7.2.

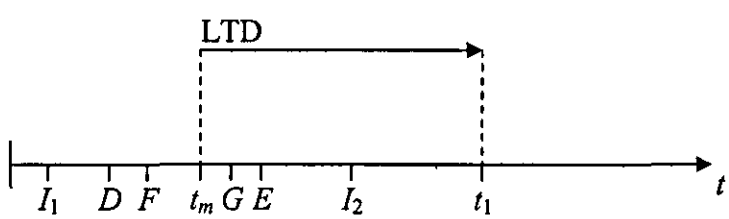


Figure 6.7.2. The occurrence of D, E, F and G and the associated maintenance deadline.

It is highly likely that this repair strategy would not be used in practice during the application of TLD to a real system. It is difficult to imagine a repair strategy that is worse than this one. However, it is included in the TLD MCS code in order to be able to analyse the system for a worst-case scenario. In the TLD MCS code, this will give the most pessimistic results for the system LOTC rate and instantaneous LOTC rates from the dispatchable system configurations.

6.7.2 Repair Strategy 2 – Repair All Faults in the Group Causing the Current Maintenance Deadline

This repair strategy is, in reality, perhaps more likely to represent the minimum amount of work that would be employed at a maintenance deadline. As opposed to the previous strategy, in which the final fault of the group that initiated the maintenance deadline is repaired, the entire group of faults is cleared from the system. In order to illustrate this, consider the two examples given in the previous section, those of the three faults, A, B and C and the four faults, D, E, F and G.

Consider first the case of the faults A, B and C, with dispatch criteria as shown in Table 6.7.1 and that occur in the order shown in Figure 6.7.1. As the first deadline is reached, at t_1, faults B and C are repaired, since this is the group of faults that caused the deadline. At this point, if this scenario occurred within the TLD MCS code, future maintenance deadlines would be checked to see if they should still be scheduled. Both deadlines must remain in the schedule at this point in order for all of the faults causing them to be repaired. At t_2, the second deadline, the group of faults causing the deadline is AB. In this case fault B has already been repaired at the DND maintenance deadline, t_1. Fault A is repaired now meaning the group of faults causing the deadline has been repaired and the system could then be dispatched until t_3. At this maintenance deadline fault A would be repaired. However, at this point (i.e. at t_2 after the maintenance has taken place) the future maintenance deadlines would be checked in the TLD MCS code and it would be seen that the fault causing the deadline, fault A, is no longer present and therefore the deadline would be removed from the schedule. See Section 6.8.8 for further information on removing future deadlines that are no longer necessary.

In the second case that was discussed for repair strategy 1, with faults D, E, F and G, the dispatch criteria were given in Table 6.7.2 and the faults were assumed to have occurred as shown in Figure 6.7.2. If the maintenance strategy involves repairing all faults in the group that caused the initiation of the dispatch deadline all four faults will be repaired at t_1, E being repaired along with D and F along with G.

6.7.3 Repair Strategy 3 – Repair All Faults in the Same Category as the Group Causing the Current Maintenance Deadline

In this repair strategy, as repairs are carried out to the group of faults that caused the maintenance deadline to be initiated, other faults that are of the same TLD category are also cleared from the system. In this way, for example, as a maintenance operation initiated by a STD category fault was carried out, all STD faults would be cleared from the system. Any faults from other TLD categories would be left in the system until the appropriate maintenance deadlines were reached.

In order to illustrate this maintenance strategy consider a case where faults A, B and C occur. The dispatch criteria involve A and B initiating a LTD interval and C initiating a STD interval. If both STD and LTD category faults are addressed using MEL maintenance then the situation depicted in Figure 6.7.3 could occur. As the first maintenance deadline was reached at t_1 the fault that initiated it, A, would be repaired. At this point all other LTD faults would be cleared from the system, in this case only fault B. Since fault C is a STD fault it would remain in the system at this time. As t_2 is reached fault C would be cleared from the system. At t_3 no further repairs need to be implemented, since B was repaired earlier. In the TLD MCS code the deadline at t_3 would have been removed from the schedule as soon as it became apparent that it would be redundant. This process is explained further in Section 6.8.8.

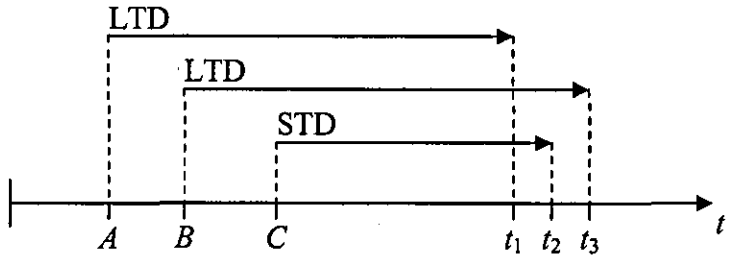


Figure 6.7.3. The occurrence of 3 faults, A, B, and C, and the associated maintenance deadlines.

Repair strategy 3 differs from repair strategy 2, which, when employed in the TLD MCS code, involves all deadlines related to the fault that just occurred being added to the schedule. In repair strategy 3, if a STD maintenance deadline is already in the schedule and a fault occurs that initiates another STD maintenance deadline the second

deadline will not be added to the schedule, since it is known that one already exists within the schedule and all STD repairs will be conducted at the first schedule.

When PIR maintenance is used to address a category of faults during this repair strategy the TLD MCS code repairs faults of that category that have yet to be 'discovered' at a periodic inspection, if any have occurred. This is in contrast to repair strategy 2, where only faults causing the deadline will be repaired. To illustrate this point, consider again the example shown in Figure 6.7.2, the dispatch criteria for which are shown in Table 6.7.2. When using repair strategy 2, if another LTD fault occurred in the time after inspection I_2 and before t_1 it would be repaired after the next periodic inspection for LTD faults. However, when using repair strategy 3, if another LTD fault occurred in this manner it would be repaired at t_1, along with D, E, F and G. When the PIR maintenance approach is employed and repair strategy 2 and 3 are used, this is the only difference between the two strategies in the TLD MCS code.

6.7.4 Repair Strategy 4 – Repair All Faults Present in the System

This is the most self-explanatory of all of the repair strategies implemented in the TLD MCS code. It is perhaps the strategy that is most likely to be used during the application of TLD to a real system. As a maintenance deadline is reached all faults present in the system at that time are repaired. The system will be clear of faults and dispatched indefinitely until the next fault occurs.

6.7.5 Opportunistic Maintenance

There is a further maintenance option that may be specified in the input data to be carried out whilst simulating the application of TLD to a system – opportunistic maintenance. This is a reasonably simple concept wherein, upon the completion of scheduled repairs at a maintenance deadline, future scheduled maintenance times are checked and if these fall within a certain time of the current maintenance deadline then the maintenance scheduled to take place at the future maintenance deadlines will be completed also. Figure 6.7.4 shows a timeline on which are represented 3 scheduled maintenance deadlines, M_1, M_2 and M_3. Consider that maintenance is taking place at M_1. If opportunistic maintenance is required then an opportunistic maintenance window

is specified. This is illustrated on the diagram. If future scheduled maintenance deadlines fall within this window then the maintenance work required at them will be completed. Thus, in this example, because M_2 lies within the opportunistic maintenance window after M_1 the repairs scheduled at M_2 will also be implemented at M_1. However, because M_3 lies outside the opportunistic maintenance window, the repairs that are to be carried out at M_3 will remain to be completed at M_3. As repairs at M_2 are carried out at M_1 the scheduled maintenance at M_2 is removed from the schedule.

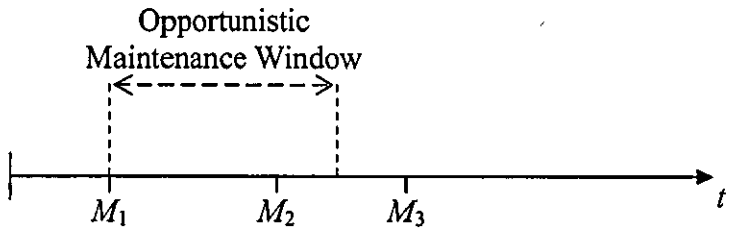


Figure 6.7.4. A demonstration of opportunistic maintenance at a maintenance deadline

Note that in the case of repair strategy 4 above, wherein all faults are cleared from the system at maintenance deadlines, opportunistic maintenance will not be implemented. This is because all faults are cleared from the system at the current maintenance deadline meaning that no future maintenance deadlines need to be scheduled.

If opportunistic maintenance is to be carried out in the TLD MCS code, a time (in hours) for the opportunistic maintenance window is specified. Certain limits for this value are applied within the TLD MCS code and the opportunistic maintenance window supplied is checked against these limits. The lower limit is the length of a single flight and the upper limit is arbitrarily chosen as the lowest value in the range of LTD interval values specified.

6.7.6 Repairing Associated Faults in the Schedule

One of the input data files supplied to the TLD MCS code is used to list faults that are 'associated' with faults which are included in the dispatch criteria for the system. To illustrate how this would be used consider a system where faults A and B occur and each causes a STD interval to be initiated. As the appropriate maintenance deadlines for

A and B are reached, these faults must be cleared from the system. Now consider that, in this particular system, a fault C can also occur but that this fault does not appear explicitly in the dispatch criteria for the system so that when it occurs a dispatch interval is not initiated and maintenance specific to it is not scheduled. Consider also that, as fault A is cleared from the system, the maintenance crew also either check for the existence of fault C as a matter of routine, or that in repairing fault A, fault C is also repaired by the maintenance crew as part of the standard procedure of maintenance. In order for this situation to be modelled within the TLD MCS code, the user includes fault A in the text file that is used to hold the associated faults. Following A will then be listed all of the faults that will then be repaired along with it at maintenance, i.e. fault C.

6.7.7 Maintenance When the System is Undispatchable

The input data determines a fault order above which the system will become unfit for dispatch. Once the number of TLD faults exceeds this number a DND category fault is assumed to have occurred and when the associated maintenance deadline arises in the simulation code, at the end of the current flight, all faults are cleared from the system. The assumption is made here that if the system degrades to such an extent that the number of TLD faults specified by the user is exceeded then the system will be returned to an 'as-new' state in a similar way to clearing all faults from the system in repair strategy 4. In the TLD MCS code this procedure is carried out by initialising the system in a similar manner to that used when initialising the system at the beginning of a simulation or after a system failure has occurred during a simulation.

6.8 Adding Maintenance Deadlines to the Schedule in the TLD MCS Code

During the course of a MCS there are many cases where a fault occurs in the system and that fault does not cause the system to fail. Every time this happens the TLD criteria relating to that fault should be checked after the fault has occurred to ascertain whether or not a maintenance deadline must be added to the schedule that is used to log the events that are still to occur. If a deadline must be added to the schedule the time of the deadline will depend on the maintenance approaches being used (MEL or PIR) for that category of deadline. The repair strategies that will be employed at maintenance deadlines will also affect how many deadlines relating to the particular fault that has

occurred should be added to the schedule. For this reason, the addition of maintenance deadlines to the schedule will be discussed for each of the maintenance strategies. Before this another important aspect of the TLD MCS code is discussed. Maintenance operations cannot occur during a period of time when the aircraft is flying – it must be on the ground. For this reason the flight time specified (see Section A2.2.2) must be taken into consideration when adding maintenance deadlines to the schedule.

6.8.1 DND Maintenance Deadlines

When DND maintenance deadlines need to be added to the schedule of events the flight time of the aircraft is taken into consideration. The deadline is added to the schedule with a time set to be at the end of the flight in which the initiating fault occurs.

6.8.2 MEL Maintenance Deadlines

In order to view how MEL maintenance deadlines are added to the schedule within the TLD MCS code consider a fault A that occurs within a system during a simulation. If the system dispatch criteria detail that fault A must be addressed using a MEL maintenance approach a scenario that arises could be similar to that depicted in Figure 6.8.1. As the fault A occurs the appropriate dispatch interval is initiated and this dispatch interval ends at t_d, the time of the maintenance deadline. However, the deadline cannot be assumed to occur at this exact time. Also shown on the diagram are the times t_{fs} and t_{fe}, which respectively represent the times of the start and end of the flight in which t_d occurs. The time of the maintenance deadline, and hence the time that must be added to the schedule, must be shifted to one of these times. In the TLD MCS code the input data determines whether the maintenance deadline is to be shifted to the start, t_{fs}, or the end of the flight, t_{fe}. Shifting the deadline to the end of the flight is pessimistic in view of calculating LOTC rates since the exposure to faults is increased slightly in relation to the dispatch interval. However, in [2] it is stated that the time of fault occurrence may be shifted to midnight on the day the fault occurred. This could be considered analogous to shifting t_d to t_{fe}.

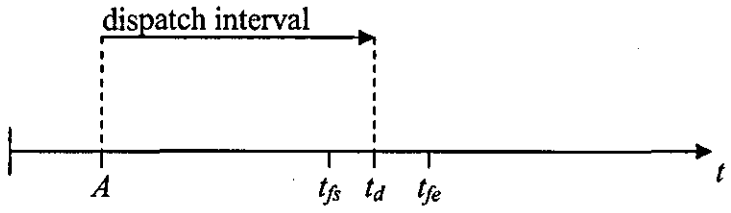


Figure 6.8.1. The occurrence of a fault A, addressed using MEL maintenance.

6.8.3 PIR Maintenance Deadlines

Consider now a fault B that occurs within a system during a simulation. In a similar manner to the situation that occurs in the general case of adding a maintenance deadline to the schedule when MEL maintenance is used, one must again consider the times at which flights begin and end before the deadline can be added. When PIR maintenance is used one could also consider the times of the periodic inspections for faults and whether these will fall between flights or mid-flight. This approach is not taken in the TLD MCS code. In order to visualise the approach used consider Figure 6.8.2. This shows a fault B that occurs and is to be addressed using a PIR maintenance approach. In order to be able to add a maintenance deadline at the correct time after fault B has occurred the first step is to find the time of the periodic inspection that is previous to it, I_1. Since only maintenance deadlines are added to the schedule in the code and periodic inspections are not added, the times of the inspections are not adjusted to occur between flights. The assumption is made that the adjustment of the time of the dispatch interval to occur at the start or end of the flight in which it falls means that the adjustment of the time of the inspection is not required.

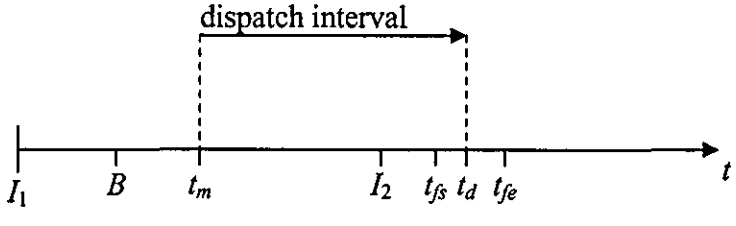


Figure 6.8.2. The occurrence of fault B, addressed using PIR maintenance.

Once the time of I_1 has been calculated, the midpoint, t_m, of the consecutive inspections, I_1 and I_2, is calculated by adding half of the length of the inspection interval. The dispatch interval is then added from this midpoint giving the time of the maintenance deadline, t_d. This maintenance deadline may or may not occur between flights because there is no guarantee that the midpoint will occur between flights or that the dispatch interval is an integer number of flights. For this reason the maintenance deadline is shifted to either the start, t_{fs}, or the end of the flight, t_{fe}, in which it occurs, if indeed it does occur mid-flight. Note that if the maintenance deadline is initially calculated to occur between flights (i.e. t_d was equal to either t_{fs} or t_{fe}), the shifting process used will leave t_d at its calculated position, it will not be moved forwards or backwards one flight. This is also the case with the MEL maintenance approach.

6.8.4 Repair Strategy 1

This repair strategy involves repairing, at a maintenance deadline, only the last fault of the group of faults that caused the deadline to occur. For this reason, after a fault occurs only the earliest-occurring maintenance deadline initiated by the fault will be added to the schedule. As this maintenance deadline arises the final fault in the group, i.e. the one that initiated the maintenance deadline can be cleared from the system. This is demonstrated below using a simple example.

Consider a case where three faults A, B and C occur within a system during a simulation. The dispatch criteria for these faults are as given in Table 6.8.1.

Dispatch Criteria	Dispatch Category
AC	STD
BC	LTD

Table 6.8.1. Dispatch criteria for 3 faults, A, B and C.

Assume that the faults occur in the order A followed by B followed by C. If both STD and LTD faults are to be addressed using MEL maintenance the situation will appear something like that depicted in Figure 6.8.3.

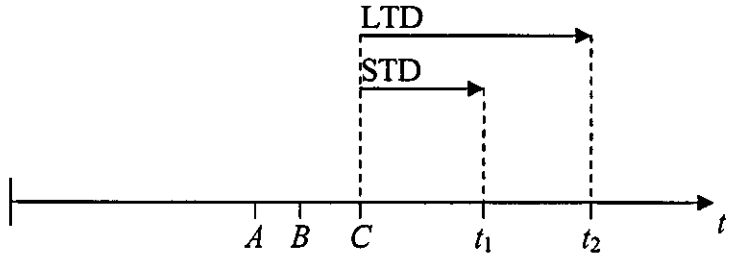


Figure 6.8.3. The occurrence of 3 faults, A, B and C.

In this case, as fault C occurs the two dispatch criteria that were given for the three faults are satisfied. This means that, due to the fact that repair strategy 1 is being applied, and that MEL maintenance is being used to address both STD and LTD faults, the STD maintenance deadline at time t_1 must be added to the schedule. The LTD fault need not be added. As time t_1 is reached in the TLD MCS code fault C will be cleared from the system, leaving A and B to be cleared as some other dispatch criteria, not noted here, will lead to maintenance deadlines at which they will be repaired.

Next, consider again the case described above with the three faults A, B and C occurring in exactly the same way. One may imagine that, whatever the maintenance approaches (MEL or PIR) being applied to STD and LTD faults, one would always add the STD deadline to the schedule since it is bound to arise before the LTD deadline. However, this is not always the case. This will be illustrated with a counterexample. Consider Figure 6.8.4, where the situation is depicted in which the faults A, B and C occur at exactly the same times as before. The difference from the previous example is the fact that LTD faults will now be addressed using PIR maintenance instead of MEL maintenance. Thus STD faults are addressed using MEL, LTD using PIR.

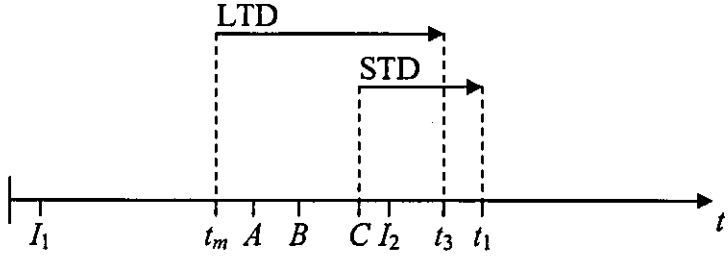


Figure 6.8.4. The occurrence of 3 faults A, B and C.

Again, as fault C occurs an STD or an LTD maintenance deadline could be added to the schedule of events to occur. In the situation depicted fault C occurs relatively close to

inspection I_2, at which the fault combination BC would be discovered. Because of this and the inspection interval, when the midpoint t_m is calculated and the LTD deadline applied (exactly the same length as in the previous example) it ends at t_3 which is actually earlier than t_1. Thus in this case the LTD deadline would be added to the schedule and not the STD deadline as before. Therefore when adding the maintenance deadlines for repair strategy 1 in the TLD MCS code the time at which the deadlines would occur is checked before any are added. Clearly, if a DND maintenance deadline is initiated by a fault group this will be added to the schedule regardless of the time of occurrence of the other categories. This is because it would be added at the end of the current flight and neither of the STD or LTD deadlines could possibly occur before this.

One final point must be made before moving on to the next repair strategy. The way that PIR faults are dealt with in the TLD MCS code is to list the faults that must be repaired at maintenance deadlines. This can be illustrated with an example. Consider two faults will occur within a system, fault D and fault E. Both of them initiate LTD maintenance deadlines and LTD faults will be addressed using PIR maintenance. The situation is depicted in Figure 6.8.5.

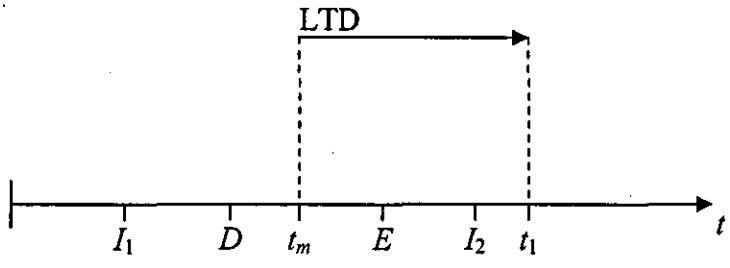


Figure 6.8.5. The occurrence of 2 faults, D and E.

As fault D occurs its maintenance deadline is calculated to occur at t_1 and is added to the schedule. Similarly, when E occurs its fault is calculated to occur at t_1. In this case the maintenance deadline need not be added to the schedule since it is already there. What happens instead is that E is added to the list of faults that will be repaired at t_1. This is relatively straightforward. However, there is a situation that can occur where E would not be repaired at the same time as D. Such a situation is depicted in Figure 6.8.6.

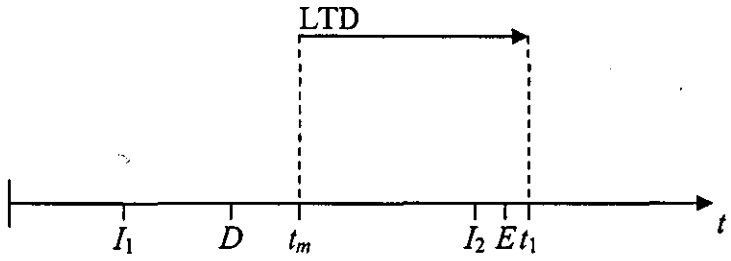


Figure 6.8.6. The occurrence of 2 faults, D and E.

In this case fault E occurs after inspection I_2, the one at which D is discovered. It is possible that fault E might be repaired along with D at t_1, but it may be left in the system and treated as though discovered at the next inspection for LTD faults, to be repaired at the maintenance deadline that occurs after that inspection. When repair strategy 1 is applied in the code the second option is chosen, where fault E would be repaired at the later of the maintenance deadlines. For this reason fault E would be added to a second list, separate from the one that contains fault D. Note that if faults D and E as depicted above actually represented two groups of faults that caused the relevant maintenance deadlines to be added to the schedule, only the final fault of the group would be repaired at the deadlines.

6.8.5 Repair Strategy 2

In repair strategy 2 all faults in the group causing the current maintenance deadline are repaired. For this reason all maintenance deadlines related to a fault are added to the schedule of events to occur as the fault occurs. This differs from repair strategy 1, where only the earliest of the deadlines would be added. Also recorded is the group or groups of faults that caused the maintenance deadline, rather than just the final fault in the group, as was the case in strategy 1. This means that, as the maintenance deadline is encountered, the correct faults can be cleared from the system.

To illustrate repair strategy 2 consider again the example given in Figure 6.8.3, where faults A, B and C cause a STD and a LTD maintenance deadline to be added to the schedule. Both categories of fault will be maintained using MEL maintenance. As fault C occurs both maintenance deadlines, t_1 and t_2, are added to the schedule. Then, as t_1 is reached faults A and C are cleared from the system. At time t_2 faults B and C would be cleared from the system. However, since C has already been repaired, only fault B

would be repaired at this point. Note that, in this situation, if both maintenance deadlines were not added to the schedule, only faults A and C would be repaired. This is the reason for adding all maintenance deadlines in repair strategy 2.

In the case of PIR maintenance being applied for repair strategy 2, all maintenance deadlines will be added to the schedule as faults occur. The same consideration is taken of the periodic inspection at which faults will be discovered as was for repair strategy 1, which was illustrated by the examples shown in Figure 6.8.5 and Figure 6.8.6. Again, if a fault occurred between an inspection and that inspection's associated maintenance deadline it would not be repaired at that maintenance deadline, but instead at the next one for faults of that category.

Note that this repair strategy is one of the strategies for which future scheduled maintenance deadlines must be checked after repairs have taken place at the current deadline. This process is described in Section 6.8.8.

6.8.6 Repair Strategy 3

This repair strategy involves repairing all faults in the same category as the fault/group of faults that caused the current maintenance deadline. It is very similar to repair strategy 2 but there are a few small differences. In this case as a fault occurs all of the deadlines of different maintenance approaches that are related to it are added to the schedule in much the same way as was the case for strategy 2. However, for strategy 3 as further faults occur the groups of faults that meet dispatch criteria are added to the lists of faults to be repaired at maintenance deadlines that already exist in the schedule, if these deadlines are related to the correct category. In this way all faults in the same category that are present in the system at a maintenance deadline will be repaired at the first maintenance deadline that arises for faults of that category. In order to illustrate this consider Figure 6.8.7.

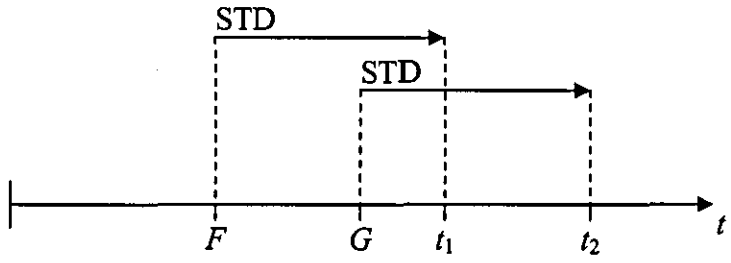


Figure 6.8.7. The occurrence of 2 faults, F and G.

Two faults, F and G, occur in that order and the dispatch criteria for the system say that both faults will initiate STD intervals ending at t_1 and t_2 respectively. When this situation arises in the code and repair strategy 3 is being employed the scenario will unfold as follows. Firstly fault F will occur. Assuming that the system was in a full-up state before this (i.e. no faults present) the STD interval is calculated and the STD maintenance deadline ending at t_1 is added to the schedule. When fault G occurs it is identified as a STD category fault and is added to the list of faults to be repaired at t_1. The STD maintenance deadline is still added to the schedule at t_2 despite the fact that it will not be required if STD faults are to be repaired at t_1. This may seem illogical at first but if one considers that the first deadline may be removed if fault F was repaired at some other time before t_1 (perhaps if F acted in combination with another fault to produce a DND maintenance deadline) then the maintenance deadline at t_1 would be removed and G would be repaired at t_2. If both faults were simply repaired at t_1 the maintenance deadline at t_2 would be checked at t_1 to see if it is still a relevant maintenance deadline. Since fault G was no longer in the system the deadline at t_2 would be removed.

The above situation represented MEL maintenance. For PIR maintenance a similar situation takes place. To see what happens here we can consider again Figure 6.8.5 and Figure 6.8.6. Both of these show a situation where two faults, D and E, occur and are addressed using the PIR maintenance approach. In the case of the first of these examples, Figure 6.8.5, as D occurs the maintenance deadline at t_1 is added to the schedule, and as E occurs it is added to the list of faults to be repaired at t_1. This would remain the same for repair strategy 3. However, in the second example, Figure 6.8.6, as fault E occurs it is seen to appear after the periodic inspection that will discover fault D and is therefore added to a list of faults that will be repaired after the next periodic

inspection. In the case of repair strategy 3 fault E is added to the list of faults to be repaired at t_1. In this way, both faults D and E will be repaired at t_1.

As for repair strategy 2, when repair strategy 3 is applied, future scheduled maintenance deadlines must be checked after repairs have taken place at the current maintenance deadline in order to verify that they are still relevant when one considers the faults that are present in the system after repairs.

Repair strategy 3 is perhaps the most realistic of the three repair strategies so far. It is a strategy that could conceivably be attempted to be used on a real system. The first two strategies may be more likely to be applied occasionally, if spare parts are unavailable or maintenance is taking place at an inconvenient time or location. One might also expect that, because faults are likely to be rare in the systems being modelled, that results obtained when modelling repair strategy 3 might not differ greatly from those obtained using repair strategy 2.

6.8.7 Repair Strategy 4

Repair strategy 4 involves repairing all faults present in the system at maintenance deadlines. This is likely to be the repair strategy that is applied for the majority of the time in a real application of TLD. It is also the simplest method to apply in the TLD MCS code. As a fault occurs and certain dispatch criteria are met, the earliest occurring deadline is added to the schedule, as was the case with repair strategy 1. Recall that, if PIR maintenance is used, the earliest occurring deadline may not necessarily be the STD deadline. If a DND dispatch criterion is met for that fault then it will always be the earliest occurring maintenance deadline.

As further faults occur the same process takes place, where the earliest deadline is added to the schedule in the TLD MCS code. The faults are just added to a list of all faults present in the system so that they may be repaired at the first deadline to occur. As the first deadline is met all of the faults in the system are repaired. Then all other deadlines are removed from the schedule.

6.8.8 Handling Maintenance Deadlines

Situations arise within the TLD MCS code where future maintenance deadlines must be removed from the schedule after maintenance has taken place at the current deadline. In order to illustrate this in a simple manner, consider Figure 6.8.8, which depicts the occurrence of two faults, A and B. Consider that the dispatch criteria stipulate that A is categorised as a LTD fault and that the dual fault AB is categorised as a STD fault. Thus maintenance deadlines ending at t_2 and t_1 are scheduled to occur as shown. If one considers that repair strategy 2 is being applied, in which all faults causing the maintenance deadline must be repaired, then, as time t_2 is reached faults A and B must be cleared from the system before further dispatch is allowed. The maintenance deadline at t_1 will now be redundant when it occurs because fault A, which is due to be repaired as the deadline occurs, has already been repaired at t_2. Therefore the maintenance deadline at t_1 may be removed from the schedule after repairs have been implemented at t_1.

This process of removing maintenance deadlines would most likely occur in reality when applying TLD to a real system, since if there is no longer a need to carry out repairs at a maintenance operation at some point in the future then this maintenance operation is likely to be cancelled.

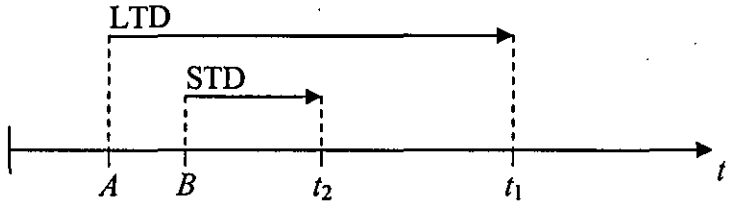


Figure 6.8.8. The occurrence of two faults, A and B.

The process of checking future maintenance deadlines is only required to be implemented for repair strategies 2 and 3. Repair strategy 1, where only the last of the group of faults causing the maintenance deadline to be scheduled is repaired, does not require future maintenance deadlines to be checked because when the deadlines are scheduled only one deadline is added to the schedule for every fault that occurs. When implementing repair strategy 4 all faults are cleared from the system at the maintenance deadlines.

Note also that even if it is not possible to remove a maintenance deadline from the schedule, it may be the case that certain repairs that were due to be carried out at the deadline no longer need to be performed. To illustrate this consider the example shown in Figure 6.8.9, where repair strategy 3 is being implemented at maintenance. Faults A and B, when occurring alone, are LTD faults. Therefore, when A occurs a LTD maintenance deadline is scheduled at t_1. When B occurs it is also an LTD fault and will therefore be repaired along with A at t_1, since all faults of the same category are to be repaired together. However, at this time a maintenance deadline is also added at t_3. The dispatch criteria for the system also specify that the dual fault BC is a STD category fault combination. Therefore, as C occurs a STD maintenance deadline is scheduled at t_2. As the maintenance deadline at t_2 is encountered faults B and C, the combination causing the deadline to be scheduled, are cleared from the system and repairs are complete at this point. Now future maintenance deadlines must be checked to see if they need to be removed from the schedule. Two separate faults were due to be repaired at the deadline at t_1, these being fault A and fault B, so these faults are checked to see if they still need to be addressed at t_1. Clearly, since fault B has been repaired it no longer needs to be repaired so this fault is now removed from the list to be repaired at t_1. However, since fault A is still present in the system the maintenance deadline cannot be removed because A must be repaired at t_1. Next, the maintenance deadline at t_3 is checked. This was caused by fault B, which has now been repaired and this means that the maintenance deadline at t_3 can be removed from the schedule. This means that, after the repair of B and C at t_2, the scheduled maintenance at t_3 will no longer be needed and the maintenance deadline at t_1 will only involve the repair of fault A.

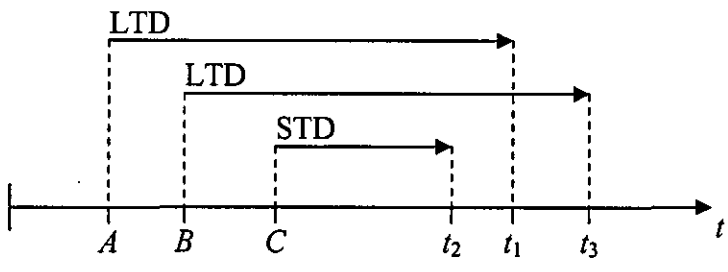


Figure 6.8.9. The occurrence of 3 faults to be repaired using repair strategy 3.

6.9 TLD MCS Code Output

MCS is a tool that has great potential if one wishes to model the effects of different maintenance approaches, maintenance strategies and fault combinations within a system. Another benefit of MCS when modelling TLD is the large amount of information that is potentially available both during the course of simulations and after they have completed. In implementing the TLD MCS code the main aim, as with any other form of TLD analysis, is to find the LOTC rate of a system and demonstrate that it meets the levels of safety required from it. The system LOTC rate is therefore output from the MCS code. Also, because of the way that TLD is applied to a system, one must demonstrate that the dispatch categories and associated intervals that are applied are suitable for the dispatchable faults within the system. For this reason the instantaneous LOTC rates from the dispatchable system configurations may also be output by the TLD MCS code. The methods used to obtain both the system LOTC rate and the instantaneous LOTC rates from the dispatchable configurations are outlined in the following sections. Also available are two separate types of importance measures, one of which gives a measure of the importance of basic events from the system fault tree and the other of which measures the importance of the dispatchable faults and fault combinations.

Also described in the following sections are the ways in which the output from the TLD MCS code is formatted. This depends mainly on the options specified in the input data, for instance whether dispatch intervals or the operating life of the system are to be varied or whether the ordering of faults is considered important when calculating the instantaneous LOTC rates. Also possible is an option which results in a detailed list of events that occur within the modelled system during the course of a number of individual simulations. This will also be briefly described below.

6.9.1 Results Obtained From the TLD MCS Code

The results that are obtained from the TLD MCS code are LOTC rates, essentially failure rates, and importance measures, essentially probabilities. They will initially vary due to the random nature of Monte Carlo simulations, before settling down and converging to a specific value. The LOTC rates obtained are the system LOTC rate and

the instantaneous LOTC rates from dispatchable system configurations. How these and the importance measures are calculated is explained below. Also explained is how updated dispatch criteria are output from the code when the dispatch criteria are to be automatically set for the system. Note that when the code is used to automatically set dispatch criteria for a system the system LOTC rate, instantaneous LOTC rates from the dispatchable system configurations and the dispatch criteria used at each step of the iterative process are output.

All results are output to the folder specified in the options data file. The TLD MCS code constructs filenames according to options specified input data file and adds a numerical identifier to the filename. A similar numerical identifier is added to the name of the file to which new dispatch criteria are output. These numerical identifiers allow the progress of simulations to be tracked, for instance when the dispatch criteria are automatically set by the code (see Chapter 9) and a number of sets of results are produced at each iteration.

6.9.1.1 The System LOTC Rate

This quantity is calculated by dividing the total number of failures in a period of simulation time by the length of that simulation time. Therefore, during the course of a group of simulations the code counts the number of system failures that occur and also the total operating life of the system throughout those simulations.

In order to illustrate the process of calculating the system LOTC rate consider Table 6.9.1. This shows the results of five consecutive simulations carried out on a system whose maximum operating life is 10000 hours. Thus, if the system does not fail during the course of one simulation the length of that simulation will be 10000 hours. Note how, after each simulation is complete, the total number of system failures up to that point is known, as is the total simulation time. One is added to the total number of system failures if the simulation just implemented failed. The total simulation time is obtained by adding the times taken for all of the simulations completed so far. The total system LOTC rate is calculated by dividing the total number of system failures by the total simulation time.

Simulation Number	System Failed?	Length of Simulation	Total Number of Failures	Total Simulation Time	Total LOTC rate
1	No	10000.0	0	10000.0	0.0
2	Once	10000.0	1	20000.0	$5.00*10^{-5}$
3	No	10000.0	1	30000.0	$3.33*10^{-5}$
4	No	10000.0	1	40000.0	$2.50*10^{-5}$
5	Once	10000.0	2	50000.0	$4.00*10^{-5}$

Table 6.9.1. Calculating the system LOTC rate.

6.9.1.2 The Instantaneous LOTC Rates From Dispatchable System Configurations

It is important to know the instantaneous LOTC rates from dispatchable system configurations when applying TLD to a system. These instantaneous LOTC rates must lie within certain ranges for dispatchable faults that are covered by certain TLD categories [2]. The calculation of these failure rates to LOTC from different system fault states is rather more complex than the calculation of the system LOTC rate but is essentially carried out in the same manner.

In the case of the instantaneous LOTC rates from dispatchable system configurations the TLD MCS code keeps track of the total length of time for which the system is dispatched from each of the dispatchable configurations. The number of system failures that occur when dispatching from those system configurations must also be recorded in order to be able to calculate the instantaneous LOTC rates.

The first step in calculating the total time spent dispatching from a certain dispatchable system configuration is to define what is actually meant by dispatching from a certain configuration. The definition used in the TLD MCS code is as follows. If a fault or fault combination is present within the system as the aircraft is dispatched then the aircraft is being dispatched from that dispatch configuration. To illustrate this point consider Figure 6.9.1, which shows a number of consecutive flights (flight 1 ends at f_1, flight 2 at f_2, etc.) on a timeline and the occurrence within the system of three faults, A, B and C. The initial dispatch of the system, as indicated by FU, is fully-up and the system is returned to a fully-up state at f_4, when a maintenance operation clears faults A, B and C from the system.

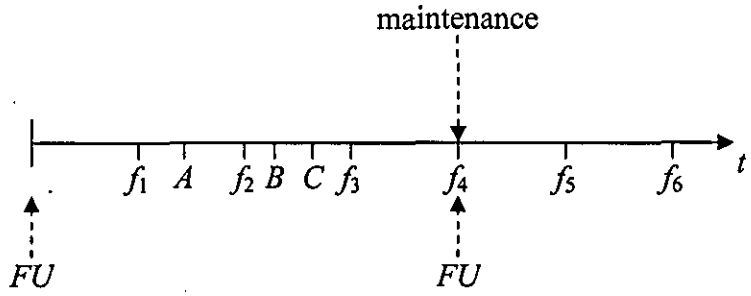


Figure 6.9.1. An example to illustrate dispatchable system configurations.

If one considers the configuration as the system is dispatched before each flight for this system the methods used in the TLD MCS code can be demonstrated. Initially the system is dispatched fully-up. At time f_1 the system is still in a fully-up state and so is dispatched for the next flight from the fully-up state. As fault A occurs the system switches from the fully-up state to the state where A has occurred. However, despite this, the system was still dispatched in the fully-up state at the start of the flight. For this reason the dispatch configuration remains as fully-up until the flight ends (or, of course, the system could fail due to further faults). Thus, as time f_2 is reached, the system has been dispatched for a time equal to two full flights in the fully-up state.

At time f_2 the system is dispatched from the configuration where fault A is present. Faults B and C occur in the next flight. The system does not fail in this flight and upon reaching f_3 the time spent dispatching from configuration A is updated to the length of one flight. At f_3 there are now three faults present in the system, A, B and C, which occurred in that order. As the system is dispatched at f_3 there are a number of fault combinations that are present. These are A, B, C, AB, AC, BC and ABC. The system is dispatched from these configurations at f_3 until the maintenance operations at time f_4. Thus the times spent dispatching from these five configurations have been increased by the length of one flight at f_4. The system is then returned to the fully-up state and the same process starts again. Thus, the system state at the start of a flight governs the dispatchable configuration for the rest of that flight. Similarly, when a system fails the dispatch configuration or dispatch configurations at the start of the flight are the ones for which the number of system failures will be increased.

The problem confronted when writing the TLD MCS code was how to find the times spent dispatching from the various dispatchable system configurations in an efficient manner. It is clearly not computationally efficient to check the system configuration at the start of every flight during a simulation. This would slow the code unnecessarily. Four points in the simulations were identified at which the times spent in the particular dispatchable configurations and the number of failures from those configurations could be updated:

1. *Before* maintenance takes place at a deadline,
2. *After* maintenance has taken place at a deadline,
3. When the system maximum lifetime (as specified in the input data file) has been reached in an individual simulation,
4. When the system fails during an individual simulation.

Consider the example depicted in Figure 6.9.2. It shows a simple scenario for a short simulation of a system. The system begins in a fully-up state, a number of faults occur, maintenance is implemented, further faults occur and the simulation ends. If the instantaneous LOTC rates from the dispatchable system configurations are to be calculated then the second schedule used in the TLD MCS code (the one that stores the faults that have occurred) keeps track of the faults that have occurred and their ordering. Given the four points of the simulations at which times spent in each of the dispatchable system configurations is calculated, the first point at which these times will be updated is at time f_4, when maintenance takes place. At this point there are three faults, A, B and C, present in the system.

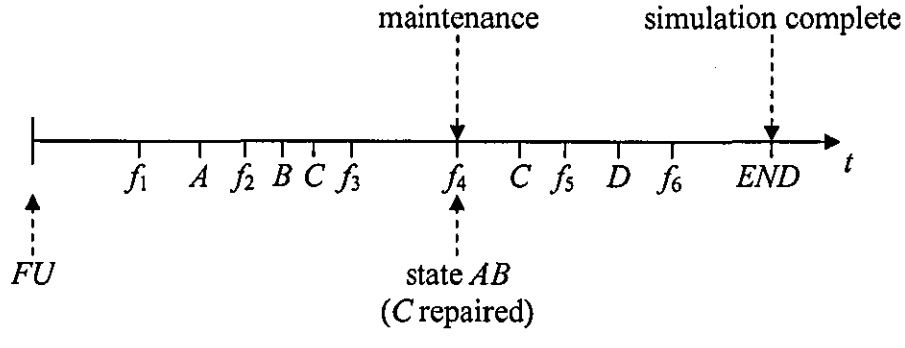


Figure 6.9.2. A representation of a system simulation.

One needs to know whether the system was fully-up at any point in the time since the start of the period since the last maintenance event or start of the simulation. A flag is used within the TLD MCS code which identifies whether this was the case. In this case the system was fully-up at the start so the first task is to update the time spent in this state. If t_{FU} represents the time spent in the fully-up system state then it will be updated at this time as follows:

$$t_{FU} = t_{FU} + (f_2 - t_{FU:START}), \qquad 6.9.1$$

where $t_{FU:START}$ is the time at which the fully-up dispatch configuration first started. The time of f_2 is calculated in the TLD MCS code by shifting the time that fault A occurred to the end of the flight. Note that, in the general case, the first point shown on the timeline, when the fully-up dispatchable configuration begins, need not be the start of the simulation – it could be the last maintenance event when the system was returned to a fully-up state.

After the fully-up time has been updated the time spent in the other dispatchable system configurations must also be updated. This is done using the schedule that records the order of occurrence of faults and the times at which they occurred. Starting with the first fault in this schedule, the faults are stepped through, building the appropriate system states and the times are updated accordingly. Thus, considering the ordered faults in the example under consideration, we get:

$$\begin{aligned} t_A &= t_A + (f_4 - f_2), \\ t_B &= t_B + (f_4 - f_3), \\ t_{AB} &= t_{AB} + (f_4 - f_3), \\ t_C &= t_C + (f_4 - f_3), \\ t_{BC} &= t_{BC} + (f_4 - f_3), \\ t_{AC} &= t_{AC} + (f_4 - f_3), \\ t_{ABC} &= t_{ABC} + (f_4 - f_3). \end{aligned} \qquad 6.9.2$$

Note that, for each of these dispatchable configuration times, the time f_4 appears in all of the equations. This is because f_4 represents the current simulation time. The other times that appear in the brackets, are f_2 and f_3, which represent the ends of the flights in which fault A and faults B and C occur, respectively. These times are calculated using the time of occurrence of the faults as stored in the schedule of faults that have occurred. If the ordering of the faults was deemed unimportant then, for example, we would have

$$t_{AB} = t_{BA}. \qquad 6.9.3$$

After the maintenance has taken place, fault C has been cleared from the system, leaving faults A and B present. Now the system configuration must be checked. If the configuration is now fully-up (which it is not) the flag indicating a fully-up configuration is set to the appropriate value and the time of the start of the fully-up dispatch period ($t_{FU:START}$) is set to the current time (f_4 in this case). This would give a situation similar to that at the start of the example shown in Figure 6.9.2. However, the system is not in the fully-up configuration, since faults A and B are present. This, together with the fact that the dispatch times relating to faults A and B have been updated up to f_4, the current time, the times of the fault occurrences stored in the schedule are changed to the current time. This prevents the time before the current maintenance deadline being added again to the total time spent in the relevant dispatchable configurations.

The next events in the example are the occurrences of faults C and D, then at some time after this the simulation ends. It is not specified here whether this is because of the system maximum lifetime being reached or the system failing. Both cases will now be discussed. Considering firstly the case of the system maximum lifetime being reached, the procedure carried out is identical to that which is carried out at maintenance. Thus:

$$\begin{aligned} t_A &= t_A + (t_{END} - f_4), \\ t_B &= t_B + (t_{END} - f_4), \\ t_{AB} &= t_{AB} + (t_{END} - f_4), \\ t_C &= t_C + (t_{END} - f_5), \\ &etc\ldots \end{aligned} \qquad 6.9.4$$

All the times spent in the dispatchable system configurations are updated accordingly, using the current time, t_{END}, and the time of the end of the flight in which the last fault in the configuration occurred, f_5 or f_6.

In the case where the simulation ends as the system fails a similar procedure is again carried out. There are two small differences from the procedure used before maintenance and as the system maximum lifetime is reached. Firstly, any faults that occurred in the final flight during which the system failed (including the fault that caused the system failure) are not included in any of the dispatchable configurations whose times and number of failures must be updated. This is because the system has not been *dispatched* with these faults present within it at this time. Note that, in the case of

the procedure carried out before maintenance or as the system maximum lifetime is reached, if we try to calculate the time spent dispatching for a fault that occurs in the flight immediately prior to the maintenance or maximum lifetime then the dispatch time will not be updated. This is because the time of the end of the flight in which the fault occurs is equal to the current time (time of the maintenance or maximum lifetime) and thus the quantity in brackets given in equations 6.9.2 and 6.9.4 is equal to zero, meaning no time will be added. This characteristic cannot be utilised when the system fails, since if the system fails in mid-flight the time of the end of the flight in which a fault occurred could be after the system has failed, i.e. after the current time. This would make the quantity in brackets negative meaning that time would be subtracted from the total time spent in the dispatchable system configurations. This is the reason for testing for faults in the last flight to occur only in the case of system failure. The test is not carried out before maintenance or as the maximum system lifetime is reached since it will most likely be rare, meaning a test would perhaps unnecessarily slow the code. Also, the fact that the quantity in brackets above would be zero means that the test is unnecessary anyway.

The second difference of the procedure carried out at system failure from the procedure carried out before maintenance or as the maximum system lifetime is reached is that, because the system has failed the number of failures from the dispatchable configurations must also be updated. This is performed in a similar manner to how the times are updated, considering the different order configurations, and updating the total system failures after dispatching from each of them accordingly.

6.9.1.3 Importance Measures

Two importance measures can be calculated during the process of system simulation. The first is a measure of basic event importance, essentially the Fussell-Vesely measure of importance, and the second is an importance measure that is similar to the Fussell-Vesely measure of minimal cut set importance, but which provides importance measures for the dispatchable system faults and fault combinations. Some of the data needed to calculate the importance measures is found when calculating the instantaneous failure rates to LOTC.

The basic event importance measure given by the TLD MCS code is the probability that the basic event has contributed to the cause of system failure given that the system has failed. It is obtained by dividing the total number of times that the basic event is present when the system fails by the total number of system failures. Similarly, the dispatchable fault importance measure provided by the TLD MCS code gives the probability of existence of a dispatchable fault given that the system has failed. Therefore, it is calculated by dividing the total number of times that the fault or fault combination is present at system failure by the total number of times that the system fails. When calculating the system LOTC rate and the instantaneous failure rates to LOTC from the various dispatchable system configurations for the system the total number of system failures and the number of times that the dispatchable faults and fault combinations are present when the system fails are already recorded. This means that the only data required in order to calculate the importance measures for the system are the number of times that each basic event is present when the system fails. This is carried out in the MCS code by checking the status of all basic events as the system fails and incrementing a counter specific to them if the basic event is in a failed state. As was the case for the instantaneous failure rates to LOTC, the importance measures are checked for convergence before being output.

6.9.1.4 Updated Dispatch Criteria

When the TLD MCS code is used to automatically set dispatch criteria for a system the dispatch criteria are set for, in turn, first order faults, second order faults, third order faults and so on in an iterative process, see Chapter 9. As this iterative process is implemented in the TLD MCS code a set of dispatch criteria is produced before each iteration, which is then tested against the LOTC rates from each of the dispatchable configurations. Each of these sets of dispatch criteria is output before the simulations are carried out, allowing the user to examine the intermediate steps involved when automatically setting the dispatch criteria.

Note that the updated dispatch criteria are output to file in different ways depending on whether the ordering of faults is important, as specified in the options input data file. Thus, if the ordering of faults is unimportant, and dispatch criteria are to be output for two dispatchable faults, A and B, dispatch criteria will be output for combinations, A, B

and AB. If fault ordering was important the combination BA would be output along with the three combinations just listed.

6.9.2 Results' Output Format

Due to the flexibility of Monte Carlo simulation techniques, it is possible to output data in a number of different ways. However, presented here are brief descriptions of the ways chosen to output results from the TLD MCS code. There are three main outputs available after system simulations. These are the system LOTC rate, the instantaneous LOTC rates from the dispatchable system configurations and the importance measures. These will be output to file together with information about the user options that brought about the results (such as the convergence options and repair strategy options).

If it is selected not to vary the operating life of the system the system LOTC rate is output to file in tabular form for varying values of the STD and LTD intervals. According to the options specified, separate files and tables of results will be produced for different maintenance approaches, repair strategies and inspection intervals. The tables of results are produced for values of the LTD interval that are greater than or equal to the STD interval. This means that the values of these intervals in the input data file must satisfy this inequality.

If it is specified that the instantaneous LOTC rates from the dispatchable system configurations is also to be output the following happens. A file for the instantaneous LOTC rates is constructed every time one is constructed for the system LOTC rate. Then, for the same values of the dispatch intervals used for the system LOTC rates, the instantaneous LOTC rates are listed for each of the dispatchable system configurations. A similar process also takes place if importance measures are to be calculated.

If the operating life of the system is varied the system LOTC rate is output to file alongside the operating life of the system that was applied to obtain that LOTC rate.

One more output format is possible. This shows a chronological list of events that occur within n simulations of the TLD MCS code (n specified) and allows the behaviour of the system in the simulations to be checked in an accountable, visual fashion.

Examples of all types of input and output files are provided in Appendix 1, Appendix 3 and Appendix 4.

6.10 Summary

This chapter has presented a general outline of the methodology behind the MCS code that was written to model the application of TLD to systems and explained the inputs required and outputs available. The main modules and functions within the code are described in Appendix 2. The method used to propagate faults and repairs was described and maintenance techniques were also explained. Four different repair strategies at maintenance were defined, along with how these repair strategies are implemented within the code. Opportunistic maintenance was also introduced and the option to include associated repairs at maintenance was discussed. The techniques used within the code to calculate the LOTC rate of the system, the instantaneous LOTC rates from the various dispatchable system configurations and importance measures were described. The varying output that may be obtained from the code was also outlined.

Chapter 7 : Analysis of Transient Markov Models for TLD

7.1 Introduction

The Markov analysis of systems and methods to construct Markov models of systems were described in Chapter 3. One of the assumptions made in constructing such a Markov model is that the system is stationary. This means that it behaves in the same way regardless of the current time. This is fine if the rates in the model are constant and failure and repair times follow exponential distributions. This can be the case for failure rates, but it may be preferable to model the repairs as taking place at periodic inspections. This behaviour can be incorporated in a Markov model and the resulting model can be solved using numerical integration techniques to give transient solutions. Only failure transitions are included in the Markov model and hence in the transition rate matrix that is developed for it. Repairs are carried out at discrete points within the integration procedure. As time is advanced during the numerical integration state probabilities are shifted at the appropriate times according to the repairs that must be implemented at these times. Once repairs have been implemented in this manner the numerical integration procedure continues until the time of the next maintenance operation is reached, when repairs will be carried out once more by shifting state probabilities. The following sections detail how this method can be used to address the application of TLD and, in particular, how such a numerical integration procedure to solving a Markov model was coded in the TLD MARKOV code using C++.

7.2 TLD MARKOV

A program, TLD MARKOV, was developed that constructs a Markov model and finds a transient solution to that Markov model. Figure 7.2.1 shows a diagram of the inputs required for and output obtained from the TLD MARKOV code. The construction of the Markov model is automated within the code since it was considered that constructing a Markov model (manually or otherwise) and correctly inputting all transitions within it would be a procedure open to the possibility of errors that could prove difficult to track. For this reason, the code developed takes as one of its inputs a

text file containing a fault tree of the system to be analysed, in a similar fashion to the TLD MCS code. The states of the Markov model are constructed by considering all of the basic events as components that are either working or failed and building all possible combinations of these working and failed components. Thus, if there are N basic events in the fault tree, there will be 2^N states in the Markov model. The system fault tree is used to determine the system status for each of these Markov model states by setting the relevant basic events to true or false and propagating this logic up through the fault tree structure using a similar technique to that used in the TLD MCS code. In this way the failed system states (or LOTC states) can be identified within the Markov model for use when calculating the system LOTC rate.

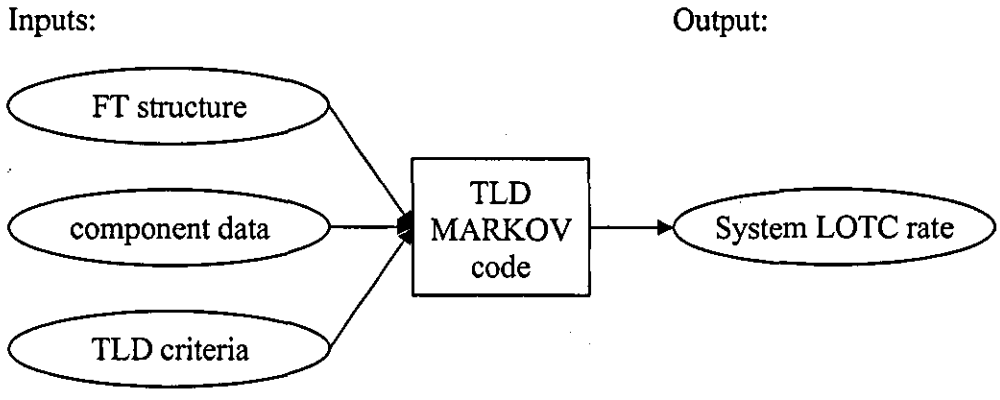


Figure 7.2.1. Inputs and output of the TLD MARKOV code.

Along with the system fault tree text file another text file containing the basic event failure rates must be provided. Each of these failure rates will then be used when constructing the transition rate matrix for the Markov model. There are a number of times that must also be provided to the code. These are: the length of a flight (used to calculate when DND maintenance operations should take place), the STD and LTD intervals that will decide when their respective TLD maintenance operations must take place, and also the total time for which the system should be modelled. This is the time over which the numerical integration procedure will be performed and a time step size, dt, must also be provided that will be used in this numerical integration.

Since the application of TLD is being modelled the dispatch criteria for this system must also be included in the model. These are also provided to the code in an input data

file where they are listed individually as groups of faults followed by a dispatch category. As these are input each of the *non-failed* system states is checked for the presence of the particular group of faults in the dispatch criterion. If that group of faults is present then the state is allocated the relevant dispatch category. If, when inputting the dispatch criteria, more than one dispatch category could be applied to a system state, the most restrictive dispatch category is applied. For example, if a system state could be allocated to either the DND or STD category then the DND category would be applied since it is the more restrictive of the two. Once all of the dispatch criteria have been input and the appropriate dispatch categories applied to each of the *non-failed* system states these dispatch categories can be used to decide what repairs will be carried out and when these repairs will take place.

Each system state now has a dispatch category that has been assigned to it and the system states for each of the categories (DND, STD and LTD) are added to lists. This enables the appropriate faults to be repaired in an efficient manner at the correct times during the implementation of the Markov model. Rather than working through the full list of system states at times when maintenance must be conducted, only the faults relevant to the maintenance will be considered as the maintenance takes place.

Once the dispatch criteria are known for the various system states it remains to define, for each state, in which other state the system will reside after repairs have taken place. For instance, if all faults are to be cleared from the system at maintenance then all repairable system states will be returned to the full-up system state as maintenance operations take place. This repair strategy is automated within the Markov code but if other strategies are used they must be included in the code manually to cover any intricacies within the strategy.

7.3 The Transition Rate Matrix

Failure transitions lead from each of the system states to at least one other system state. This is true except in the case of the final fully failed system state from which there are no failure transitions since all of the system components are already failed in this system state. The transition rate matrix is constructed for the Markov model by comparing the system states and allocating the appropriate failure rates to the matrix.

Then the diagonal elements are added to the matrix by adding the other terms in the rows in which they are contained. Thus the elements of the transition rate matrix, **A**, are given by:

$$a_{ij} = \begin{cases} 0, & \text{if } i \neq j \text{ and there is no transition from state } i \text{ to state } j, \\ \lambda_{i,j}, & \text{if } i \neq j \text{ and there is a failure transition from state } i \text{ to state } j, \\ -\sum_{k=1}^{n} a_{ik}, & \text{if } i = j, \end{cases} \qquad 7.3.1$$

where n is the number of states in the Markov model and $\lambda_{i,j}$ is the failure rate from state i to state j. This transition rate matrix is stored in the Markov code as a two-dimensional array.

7.4 Calculating the LOTC Rate

The LOTC rate is defined here as being the total (failure) transition rate into the states of the Markov model that represent LOTC from non-LOTC (i.e. non-failed) model states. Transitions from LOTC states to other LOTC states are not considered when calculating the system LOTC rate since the LOTC event has already occurred before this transition takes place. To illustrate this consider a number of states from a Markov model, shown in Figure 7.4.1. A single LOTC state from the model (state i) is shown, along with the y non-failed states which have a failure transition leading from them to the LOTC state i, labelled as (i, 1) through (i, j) to (i, y). Each of these y states has a failure transition characterised by rate $\lambda_{i,j}$ that leads to the LOTC state i.

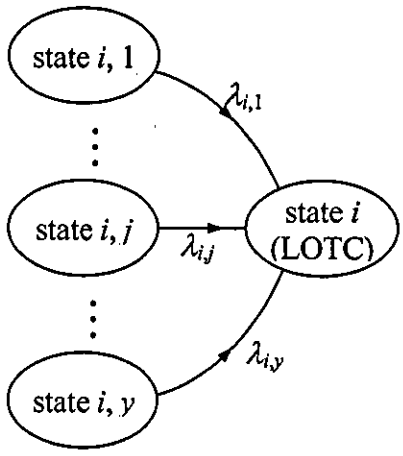


Figure 7.4.1. Failure transitions into a general LOTC state.

Calculating the LOTC rate into state i involves adding the products of the probabilities of being in each of the states (i, 1) to (i, y) multiplied by the failure rates from those states to the LOTC state, i.e.

$$\begin{aligned} LOTC_i &= \lambda_{i,1}Q_{i,1} + \ldots + \lambda_{i,j}Q_{i,j} + \ldots + \lambda_{i,y}Q_{i,y} \\ &= \sum_{j=1}^{y} \lambda_{i,j}Q_{i,j} \end{aligned} \qquad 7.4.1$$

If there are x LOTC states in the model then the system LOTC rate will be given by adding the LOTC rates. Thus, if LOTC state i has y_i associated system states that have a failure transition leading from them to LOTC state i then the system LOTC rate is given by:

$$LOTC_{SYS} = \sum_{i=1}^{x} \sum_{j=1}^{y_i} \lambda_{i,j}Q_{i,j} \qquad 7.4.2$$

Equation 7.4.2 is constructed within the Markov code using the transition rate matrix to find which model states involve a failure transition to LOTC states (i.e. failed system states). Knowing these states allows the failure rates, $\lambda_{i,j}$, to be stored in an array type structure together with the associated state reference which in turn allows the state probabilities, $Q_{i,j}$, to be multiplied by the correct failure rates when calculating the system LOTC rate.

7.4.1 The Average System LOTC Rate

The average system LOTC rate can be calculated by taking an average of $LOTC_{SYS}$ over the length of the numerical integration procedure, which will cover the system operational lifetime. In the Markov code this is conducted at two separate times, these being at the start of flights and at the end of flights. These give slightly different values and to illustrate this consider two consecutive flights. As the first flight ends the system LOTC rate can be calculated. Then, before the next flight begins the LOTC rate can again be calculated. However, these values will not be the same since, in the intervening period repairs will have been implemented (i.e. probabilities will have been shifted).

7.5 Numerical Integration

The transition rate matrix, **A**, forms a system of ordinary differential equations:

$$\dot{\mathbf{Q}}(t) = \mathbf{Q}(t)\mathbf{A}, \qquad 7.5.1$$

where

$$\mathbf{Q}(t) = [Q_1(t), Q_2(t), \ldots, Q_n(t)]. \qquad 7.5.2$$

$Q_i(t)$ is the probability of the system being in state i at time t and there are n system states in the model. Therefore, in order to find a solution to the Markov model, the system of ordinary differential equations given in equation 7.5.1, together with the initial conditions, must be solved. A common method of solution is to use a numerical integration technique. The code employs a 4th order Runge Kutta algorithm based on pseudo code given in [19]. Initial conditions are used to start the numerical integration procedure. These are:

$$Q_i = \begin{cases} 1, & \text{when } i \text{ is the full-up state,} \\ 0, & \text{for all states other than the full-up state,} \end{cases} \qquad 7.5.3$$

i.e. the system initially resides in the full-up state.

The Runge Kutta algorithm gives the values of the system state probabilities, Q_i, after a time step of specified length dt. These state probabilities can be used at the appropriate times in the code to calculate the system LOTC rate by substituting the relevant Q_i's into equation 7.4.2. The 4th order Runge Kutta algorithm gives:

$$Q_i(t+dt) = Q_i(t) + \frac{1}{6}(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i})dt, \qquad 7.5.4$$

where

$$\begin{aligned} k_{1i} &= \sum_{j=1}^{n} a_{ji} Q_j(t), \\ k_{2i} &= \sum_{j=1}^{n} a_{ji} \left[Q_j(t) + \tfrac{1}{2} k_{1i} dt\right], \\ k_{3i} &= \sum_{j=1}^{n} a_{ji} \left[Q_j(t) + \tfrac{1}{2} k_{2i} dt\right], \\ k_{4i} &= \sum_{j=1}^{n} a_{ji} \left[Q_j(t) + k_{3i} dt\right] \end{aligned} \qquad 7.5.5$$

However, in the code written to perform these operations, it would be inefficient to evaluate each of the summations in equations 7.5.5 since the majority of the a_{ji}'s are zero. Thus, the non-zero terms of these a_{ji}'s, which correspond to the columns of the

transition rate matrix **A**, are stored in array-type structures along with the value of j – the row in the matrix from which the transition rate is taken.

Therefore, if column i has n_i non-zero terms, and the position (j, above) of the mth of these within the column of the transition rate matrix **A** is represented by say b_m, then the equations 7.5.5 can be written as:

$$
\begin{aligned}
k_{1i} &= \sum_{m=1}^{n_i} a_{b_m i} Q_{b_m}(t), \\
k_{2i} &= \sum_{m=1}^{n_i} a_{b_m i} \left[Q_{b_m}(t) + \tfrac{1}{2} k_{1i} dt \right], \\
k_{3i} &= \sum_{m=1}^{n_i} a_{b_m i} \left[Q_{b_m}(t) + \tfrac{1}{2} k_{2i} dt \right], \\
k_{4i} &= \sum_{m=1}^{n_i} a_{b_m i} \left[Q_{b_m}(t) + k_{3i} dt \right]
\end{aligned}
\qquad 7.5.6
$$

Calculating these values during the integration procedure within the Markov code will now be far more efficient. This is especially true as the number of states in the Markov model increases.

7.6 TLD maintenance operations

Repairs are carried out within the model at times in the integration that correspond to the times at which TLD maintenance should take place. DND repairs take place after the system of ordinary differential equations have been integrated for the length of a flight. STD and LTD repairs then take place at the appropriate intervals after an integer number of flights. The repairs in this type of Markov model represent the real process of repair that would take place in practice wherein faults are repaired between flights and not at a constant rate from the moment that the failure occurs. The process of repair involves identifying, for each repairable state, which of the other model states represents the system after the appropriate repairs have been carried out. In order to conduct the repairs the probability of the state to be repaired is set to zero after it has been added to the probability of the state in which the system will reside after repairs have been implemented. In effect the state probabilities are shifted from the states to be repaired to the states resulting from the repairs. In the Markov code the repairs are carried out in the order DND followed by STD followed by LTD. Repairs are carried

out in this order so that state probabilities are redistributed from the higher order fault states down through the lower order fault states, i.e. from states with higher numbers of faults to states with lower numbers of faults. This is true because it is generally the case that, given a certain fault combination, if more faults occur within the system the TLD category applied will become more restrictive. For example, one could imagine a LTD category being caused by the occurrence of a single fault. A further fault may cause the LTD category to be reduced to STD. This STD category could then be reduced to DND if a third fault occurred. In this process it could be the case that a repair strategy is implemented wherein the repair of 3 fault state would take the system from that DND state back to the 2 fault STD state and the repair of this state would take the system back to the LTD state. Repair from this LTD state would then lead to the 0 fault (full-up state).

If one reached a stage in the Markov code where, for instance, all fault categories were to be repaired, then state probabilities would be redistributed for this group of fault states in the following manner. Firstly, the DND repairs would be implemented and the probability of being in the state with 2 faults present would be increased by the value of the probability of being in the state with 3 faults present. The probability of being in the state with the 3 faults present would then be set to zero, signifying that repairs had been completed. This process is represented below:

$$\begin{aligned} Q_{0,DND} &= Q_{0,orig}, \\ Q_{1,DND} &= Q_{1,orig}, \\ Q_{2,DND} &= Q_{2,orig} + Q_{3,orig}, \\ Q_{3,DND} &= 0. \end{aligned} \qquad 7.6.1$$

$Q_{i,orig}$ represents the original probability of being in the state with i faults present and $Q_{i,DND}$ represents the probability of being in state i after the DND repairs have been carried out. Next, the STD repairs would be carried out, by increasing the value of the single fault state probability by the value of the 2 fault state probability then clearing the 2 fault state probability. This is demonstrated below:

$$\begin{aligned} Q_{0,STD} &= Q_{0,DND}, \\ Q_{1,STD} &= Q_{1,DND} + Q_{2,DND}, \\ Q_{2,STD} &= 0, \\ Q_{3,STD} &= Q_{3,DND}. \end{aligned} \qquad 7.6.2$$

Finally, the LTD repairs would be implemented by shifting the probability of the single fault state to the state where no faults are present in the system. The computations involved are outlined below:

$$\begin{aligned} Q_{0,LTD} &= Q_{0,STD} + Q_{1,STD}, \\ Q_{1,LTD} &= 0, \\ Q_{2,LTD} &= Q_{2,STD}, \\ Q_{3,LTD} &= Q_{3,STD}. \end{aligned} \qquad 7.6.3$$

Now, we may represent the probabilities of being in each of the states after the LTD repairs have been conducted by substituting equations 7.6.1 and 7.6.2 into equations 7.6.3. This gives:

$$\begin{aligned} Q_{0,new} &= Q_{0,orig} + Q_{1,orig} + Q_{2,orig} + Q_{3,orig}, \\ Q_{1,new} &= 0, \\ Q_{2,new} &= 0, \\ Q_{3,new} &= 0, \end{aligned} \qquad 7.6.4$$

where $Q_{i,new}$ represents the probability of being in the state with i faults present after the DND, STD and LTD maintenance operations have been implemented. Thus, as is desired, after all three of the TLD maintenance operations have been implemented the probability of being in the system states with 1, 2 and 3 faults present is zero. The probability of being in the state where no faults are present has been increased by the sum of the original probabilities of being in the other states.

If, instead of carrying out maintenance in the order described above (i.e. DND, then STD, then LTD), one was to implement it in the order LTD, then STD, then DND the first step in the repair process would be conducted as follows:

$$\begin{aligned} Q_{0,LTD} &= Q_{0,orig} + Q_{1,orig}, \\ Q_{1,LTD} &= 0, \\ Q_{2,LTD} &= Q_{2,orig}, \\ Q_{3,LTD} &= Q_{3,orig}. \end{aligned} \qquad 7.6.5$$

Here, it can be seen that the probability of being in the state with one fault present has been shifted to the state with no faults present. Next STD faults would be carried out as shown below:

$$
\begin{aligned}
Q_{0,STD} &= Q_{0,LTD}, \\
Q_{1,STD} &= Q_{1,LTD} + Q_{2,LTD}, \\
Q_{2,STD} &= 0, \\
Q_{3,STD} &= Q_{3,LTD}.
\end{aligned}
\qquad 7.6.6
$$

The probability of being in the state with 2 faults present has been shifted to the state with 1 fault present. Finally, DND maintenance will be carried out:

$$
\begin{aligned}
Q_{0,DND} &= Q_{0,STD}, \\
Q_{1,DND} &= Q_{1,STD}, \\
Q_{2,DND} &= Q_{2,STD} + Q_{3,STD}, \\
Q_{3,DND} &= 0.
\end{aligned}
\qquad 7.6.7
$$

Substituting equations 7.6.5 and 7.6.6 into equation 7.6.7 gives the probabilities of being in each of the fault states after repairs have been implemented. These are shown below:

$$
\begin{aligned}
Q_{0,new} &= Q_{0,orig} + Q_{1,orig}, \\
Q_{1,new} &= Q_{2,orig}, \\
Q_{2,new} &= Q_{3,orig}, \\
Q_{3,new} &= 0.
\end{aligned}
\qquad 7.6.8
$$

Now the probabilities of being in the system fault states after the maintenance has been carried out in the two different orders can be compared. Equations 7.6.4 show the probabilities after maintenance has been carried out in the order DND followed by STD followed by LTD. Equations 7.6.8 show the probabilities after maintenance has been carried out in the reverse order. It can be seen from equations 7.6.8 that the maintenance has not been carried out as would be desired, since, if we assume that the original probabilities of lying in each of the states, $Q_{i,orig}$, was non-zero then the probability of being in the states with 1 and 2 faults present is non-zero. This would correspond to the situation where the repairs of the STD and LTD had not been completed.

Conversely, equations 7.6.4 show that carrying out repairs in the order DND, STD, LTD results in the state probabilities being as is needed after all of the repairs have been implemented. This demonstrates why the maintenance operations are carried out in this specified order in the Markov code when more than one of the TLD categories of fault must be cleared at the same time. It should also be noted that other repair strategies at maintenance could be covered within the Markov code. For instance, if all faults were

to be cleared from the system at maintenance then, irrespective of the ordering of the TLD maintenance operations, state probabilities as shown in equations 7.6.4 would result. However, as shown above, the ordering of maintenance operations prescribed yields desirable results whatever the repair strategy at maintenance.

7.7 Maintenance after LOTC events

When a LOTC event occurs due to the failure of a FADEC system the most likely course of action taken will be to replace or fully repair the FADEC before further dispatch of the aircraft. In this way one can think of all probability from the LOTC states in the Markov model being shifted to the full-up state at the end of every flight. Thus, in the Markov code, at the end of every flight the probability of being in the LOTC states is added to the probability of being in the full-up state and the probability of being in the LOTC states is set to zero. If there are x LOTC states in the Markov model the process carried out is as follows:

$$\begin{aligned} Q_{full-up,new} &= Q_{full-up,orig} + \sum_{\substack{i=1 \\ i \text{ is a} \\ \text{LOTC state}}}^{x} Q_i, \\ Q_i &= 0, \qquad i = 1,\ldots,x. \end{aligned} \qquad 7.7.1$$

7.8 Maintenance Approaches

Before using the Markov code whose methods have just been described it is necessary to think about the way that the model used impacts on the TLD analysis performed. Consider the way that maintenance will be implemented in the Markov code. At certain times within the code all probabilities from relevant dispatchable system states will be shifted to the full-up state. For example, in the case of DND faults all probabilities of DND category fault states will be shifted to the full-up state at the appropriate discrete time points during the numerical integration. These time points will be the multiples of the length of a flight. In this way, throughout the numerical integration, DND faults will effectively be repaired after each flight, giving a reasonable representation of DND maintenance.

Considering STD category faults in carrying out maintenance within the model one must specify the time points when probabilities are shifted and maintenance takes place. This notion leads to the requirement that, when finding transient solutions to Markov models, the repair intervals specified for use within the models must in fact relate to the inspection intervals applied for PIR maintenance. Thus a maximum of twice the dispatch interval applied to faults can be applied as the period of time between repairs in the transient Markov model. This is true for both STD and LTD category faults. Note that the transient Markov model is limited in that it can only be thought of as representing PIR maintenance with differing dispatch intervals and the inspection interval set to twice the dispatch interval.

7.9 Summary

A method of calculating a transient solution to the application of TLD to a system has been described. The method takes a fault tree description of a system and uses it to construct a complete (i.e. non-reduced) Markov model to which failure transitions are added according to basic event failure rates. Maintenance within the Markov model is conducted by considering the dispatch criteria that must be applied to each of the system states (these are also supplied to the code) and shifting state probabilities at the appropriate times within the numerical integration procedure carried out within the code. Consideration is taken of the order in which the shifting of the probabilities must be implemented during the modelling of this repair. The similarity of this modelling approach to the PIR maintenance approach with the inspection interval set at twice the dispatch interval is noted.

It is also noted that the way that maintenance is included within the model will not represent exactly the maintenance approach that would be applied in reality when TLD is applied. A point not noted previously, but worthy of note, is the fact that the repair strategy at maintenance described throughout this chapter is the one wherein all faults will be cleared from the system at maintenance. It might be possible to include other maintenance approaches in transient Markov models but the implementation of this will be far from straightforward.

Chapter 8 : Modelling TLD for Simple Systems

8.1 Introduction

In the previous chapters a number of modelling techniques have been described for analysing the application of TLD to systems. In this chapter the FAA recommended methods described in Chapter 5, the transient Markov solution described in Chapter 7 and the Monte Carlo simulation technique described in Chapter 6 are used to study a simple system. This system was given as an example in [1] and results on this were published in [20]. Other systems that were extensions of this were considered in work published in [21] and [22].

8.2 Two Component System Example

Consider the system depicted in Figure 8.2.1, which consists of two components, labelled as unit 1 and unit 2, configured in parallel.

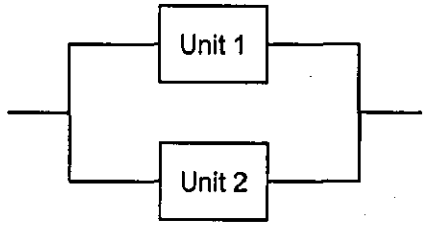


Figure 8.2.1. Two component system.

This example is given in [1] in order to demonstrate the application of the TWA and single fault state Markov approaches to modelling TLD. Each unit has associated with it a failure rate, 2.0×10^{-4} failures per hour for unit 1 and 1.0×10^{-4} failures per hour for unit 2. The flight time for the system is assumed to be 5 hours and the operational lifetime of the system is assumed to be 130000 hours. This would correspond to a total lifetime of about 20 years if the system was used for an average of 15-20 hours per day throughout its life cycle. The STD interval is to be set at 100 hours and the LOTC rate is to be calculated over a range of values for the LTD interval, rising from 100 hours to 500 hours.

Due to the simplicity of the system there are only two dispatch criteria that will be applied to it. Ignoring, for the sake of this simple example, the allowable failure rates to LOTC from the dispatchable system states for STD and LTD category faults, the dispatch criteria are defined as follows. When unit 1 fails a LTD interval is initiated and when unit 2 fails a STD interval is initiated. The presence of one failed unit in the system means that the failure of the remaining unit will cause system failure. Thus the failure rate to LOTC from each of the failed dispatchable configurations (i.e. with one unit failed) is simply the failure rate of the remaining working unit. Recalling the general system model described in Section 5.2, the full-up system state with no faults will be represented as state $i = 1$, the STD dispatchable system state (i.e. that with unit 2 failed) will be state $i = 2$ and the LTD dispatchable system state (with unit 1 failed) will be state $i = 3$. Thus, for the STD and LTD dispatchable system states, the associated failure rates into the states (λ_i) and failure rates from the states to LOTC ($\lambda_{i,L}$) take the values as shown in Table 8.2.1.

State, i	λ_i	$\lambda_{i,L}$	T_i
2	1.0×10^{-4}	2.0×10^{-4}	T_{STD}
3	2.0×10^{-4}	1.0×10^{-4}	T_{LTD}

Table 8.2.1. Failure rates into the failed dispatchable system states and from them to LOTC and dispatch intervals.

With this information one may now apply the TWA and reduced-state Markov modelling techniques. The following sections outline these processes. Since the two component system considered here is so simple the failure rate due to mechanical/hydromechanical faults, λ_{HMC} (see Section 5.3.1), and the failure rate due to unrevealed faults, λ_{UR} (see Section 5.3.2), are not used in the applications of the TWA and reduced state Markov models. Thus set

$$\lambda_{HMC} = \lambda_{UR} = 0. \qquad 8.2.1$$

8.2.1 TWA Model

Before commencing with the process of modelling the system using TWA it should be noted that two different models exist according to the fractional coefficients that are used within the model. Section 5.4 describes why the balanced fractional coefficients

could be considered as the best of the representations of the fractional coefficients and these balanced fractional coefficients are the ones that are used in this TWA analysis.

The values of λ_i displayed in Table 8.2.1 may be substituted into equations 5.3.4 to give:

$$\lambda_{STD} = 1.0 \times 10^{-4}, \quad \lambda_{LTD} = 2.0 \times 10^{-4}. \qquad 8.2.2$$

Similarly, substituting the values of λ_i and $\lambda_{i,L}$ from into equations 5.3.6 gives the following values for the average failure rates to LOTC:

$$\lambda_{STD,L} = 2.0 \times 10^{-4}, \quad \lambda_{LTD,L} = 1.0 \times 10^{-4}. \qquad 8.2.3$$

It now remains to estimate the failure rate for full-up electronics, $\lambda_{FU,L}$, using equation 5.3.18, which gives:

$$\lambda_{FU,L} = 1.125 \times 10^{-7}. \qquad 8.2.4$$

Given the above values one may obtain the following expression for the TWA LOTC rate using the balanced fractional coefficients, obtained using equation 5.4.6:

$$\lambda_{TWA,bal} = \frac{1.125 \times 10^{-7} + 2.0 \times 10^{-8} \times T_{STD} + 2.0 \times 10^{-8} \times T_{LTD}}{1 + 1.0 \times 10^{-4} T_{STD} + 2.0 \times 10^{-4} T_{LTD}}, \qquad 8.2.5$$

This expression for the TWA LOTC rate of the system can be used to compute LOTC rates by setting the STD interval, T_{STD}, and then varying the value of the LTD interval, T_{LTD}. The calculated LOTC rate obtained for varying T_{LTD} when T_{STD} is set at 100 hours is shown in the second column of Table 8.2.2.

T_{LTD}	*TWA – balanced fractional coeffs*	*Single fault state (SFS) Markov*	*First order approx. to SFS Markov*	*Simplified SFS approx. for short TLD intervals*
100	3.99	3.83	3.83	3.88
150	4.94	4.73	4.73	4.83
200	5.88	5.61	5.60	5.77
250	6.81	6.46	6.45	6.70
300	7.73	7.29	7.27	7.62
350	8.64	8.10	8.07	8.53
400	9.54	8.88	8.84	9.43
450	10.43	9.65	9.59	10.33
500	11.32	10.39	10.31	11.21

Table 8.2.2. LOTC rates (no. of failures per 10^6 flt. hrs.) obtained for the two unit system for varying T_{LTD} (hrs.) with T_{STD} = 100 hrs. Results shown are for FAA recommended methods.

8.2.2 Single Fault State Markov Model

The single fault state Markov model for the two component system makes use of the same values for λ_i and $\lambda_{i,L}$ that were used in the TWA approaches just demonstrated, given in equations 8.2.2 and 8.2.3. Using equation 5.5.15 and recalling that $\lambda_{HMC} = \lambda_{UR} = 0$ the single fault state Markov LOTC rate is thus given by:

$$\lambda_{SFS} = \frac{\left(\dfrac{\lambda_2\lambda_{2,L}}{\nu_2+\lambda_{2,L}} + \dfrac{\lambda_3\lambda_{3,L}}{\nu_3+\lambda_{3,L}}\right)}{\left(1+\dfrac{\lambda_2}{\nu_2+\lambda_{2,L}} + \dfrac{\lambda_3}{\nu_3+\lambda_{3,L}}\right)}. \qquad 8.2.6$$

Substituting into this the values of λ_i and $\lambda_{i,L}$ and the repair interval from equation 5.5.1 gives:

$$\lambda_{SFS} = \frac{\left(\dfrac{2.0\times10^{-8}\times T_{STD}}{1+2.0\times10^{-4}\times T_{STD}} + \dfrac{2.0\times10^{-8}\times T_{LTD}}{1+1.0\times10^{-4}\times T_{LTD}}\right)}{\left(1+\dfrac{1.0\times10^{-4}\times T_{STD}}{1+2.0\times10^{-4}\times T_{STD}} + \dfrac{2.0\times10^{-4}\times T_{LTD}}{1+1.0\times10^{-4}\times T_{LTD}}\right)}, \qquad 8.2.7$$

which may be rearranged to give:

$$\lambda_{SFS} = \frac{2.0\times10^{-8}(T_{STD}+T_{LTD})+6.0\times10^{-12}T_{STD}T_{LTD}}{1+3.0\times10^{-4}(T_{STD}+T_{LTD})+7.0\times10^{-8}T_{STD}T_{LTD}}, \qquad 8.2.8$$

Similarly to when the TWA LOTC rate was calculated, T_{STD} is now fixed at 100 hours and the LOTC is calculated for varying values of T_{LTD}. The results are shown in the third column of Table 8.2.2.

8.2.2.1 1st Order Approximation to the Single Fault State Markov LOTC Rate

The first order approximation to the single fault state Markov model LOTC rate of the two component system may also be computed (see Section 5.5.3). Substituting the appropriate transition rates into equation 5.5.24 gives the following expression:

$$\lambda_{SFS,1} = \frac{2.0\times10^{-8}\left(T_{STD}+T_{LTD}\right)-2.0\times10^{-12}\left(2T_{STD}^{2}+T_{LTD}^{2}\right)}{1.0+10^{-4}\left(T_{STD}+2T_{LTD}\right)-2.0\times10^{-8}\left(T_{STD}^{2}+T_{LTD}^{2}\right)}. \qquad 8.2.9$$

The results of the model are given in the fourth column of Table 8.2.2.

8.2.2.2 Simplified Approximation for Short Dispatch Intervals

The simplified approximation for short dispatch intervals described in section 5.5.4 and given in equation 5.5.27 is shown below for the two component system. After the substitution of the transition rates into and out of the system states and the dispatch intervals for each of the states in the model the following expression is obtained:

$$\lambda_{SFS,short} = \frac{2.0\times10^{-8}\left(T_{STD}+T_{LTD}\right)}{1+1.0\times10^{-4}\left(T_{STD}+2T_{LTD}\right)}. \qquad 8.2.10$$

This is equivalent to the first order approximation to the single fault state Markov LOTC rate, $\lambda_{SFS,1}$, without the squared terms in T_i. The results are given in the fifth column Table 8.2.2.

8.2.3 Transient Markov Model Solutions

The conventional Markov model of the two component system takes the form shown in Figure 8.2.2. Note that the transition rates, λ_i and ν_i, shown on the model represent the failure and repair of unit i in this model, and not the failure and repair rates into and out of state i, as was the case with the single fault state Markov model discussed previously.

In the case where transient solutions to the model are found and the maintenance is incorporated in the model at discrete time points during the numerical integration the Markov model will not contain any repair transitions. The state probabilities will instead be shifted at the appropriate times during the numerical integration from state 2 to state 1 and from state 3 to state 1. In this case the repair interval will be set at 200 hours for the STD fault and similarly the repair interval for the LTD fault will be set at twice the desired dispatch intervals for the LTD fault. The model is as shown in Figure 8.2.2, with the repair rate for the STD and LTD faults included as the reciprocals of T_{STD} and T_{LTD} respectively.

In both cases the state 4 probability will be shifted to state 1 at the end of each flight by the numerical integration procedure. In order to compute the solutions to the problem the fault tree for the two component system is passed to the Markov code described in Chapter 7, along with files containing the failure rates of the individual units of the system and the dispatch criteria that will be applied to the system. A step length of 0.1 is used in the Runge-Kutta integration routine.

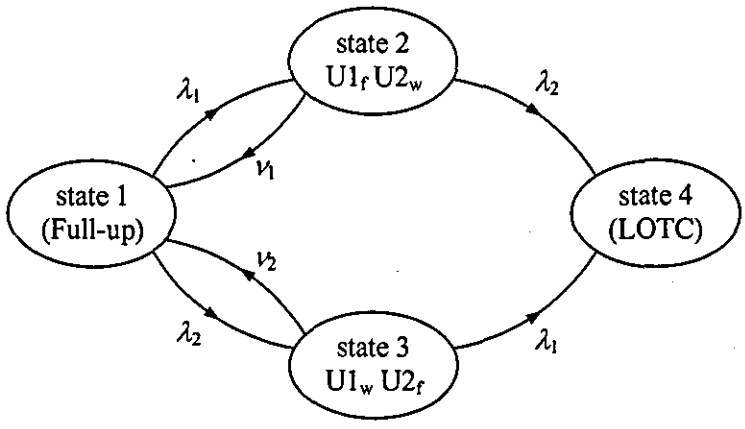


Figure 8.2.2. Conventional Markov Model of the Two Component System.

8.2.4 Monte Carlo Simulation

The TLD MCS code was also used to simulate this two component system. Due to the flexible nature of the code it was possible to simulate the use of different maintenance approaches (MEL or PIR) applied to each category of fault (i.e. each unit). It is also possible to model the four different repair strategies at maintenance deadlines described

in Section 6.7. However, in the case of this simple example system the differences between the repair strategies do not make any difference. To illustrate this, consider repair strategy 1, where only the fault causing the maintenance deadline is repaired and strategy 4, where all faults are cleared from the system at maintenance. In fact, these represent the same strategy for this system, since there can only ever be a maximum of one fault present in the system at dispatch. In fact, all of the repair strategies are equivalent for this system. The fault tree structure was passed to the code, along with the failure rates of the units and the dispatch category that should be applied to each of them. The results are shown in Table 8.2.3. In order to obtain the results an accuracy of 3 s.f. was specified for the results and 1 000 000 simulations were carried out before checking for convergence of the results. The results were subsequently checked every 50 000 simulations. Three consecutive values for the LOTC rate were required to match to 3 s.f. before convergence was assumed.

	Transient Markov Model			MCS
T_{LTD}	repairs as rates	discrete repair - start	discrete repair - end	STD - MEL LTD - MEL
100	3.82	3.79	3.98	3.96
150	4.73	4.70	4.89	4.85
200	5.60	5.62	5.80	5.75
250	6.45	6.49	6.68	6.62
300	7.28	7.37	7.56	7.47
350	8.08	8.21	8.40	8.30
400	8.86	9.06	9.24	9.10
450	9.62	9.87	10.05	9.88
500	10.40	10.70	10.87	10.70

Table 8.2.3. LOTC rates (no. of failures per 10^6 flt. hrs.) obtained for the two unit system for varying T_{LTD} (hrs.) with T_{STD} = 100 hrs. Results shown are for the transient Markov model and the MCS.

8.2.5 Comparison of the Modelling Approaches

The LOTC rate of the two component system has been evaluated using several different modelling approaches. An expression was obtained for the LOTC rate using the TWA method using the balanced time coefficients. Of the reduced-state Markov models, the only one suitable for this two component system is the single fault state model. An expression has been derived for the LOTC rate of the system using this model, along

with further expressions obtained by truncating series expansions for the original single fault state Markov model LOTC rate. The final analytical model used was a conventional Markov model, for which an expression for the asymptotic failure rate was calculated. All of these expressions for the LOTC rate contain terms for the STD and LTD intervals, T_{STD} and T_{LTD}. For this simple system the LOTC rate was calculated for a value of 100 hours for T_{STD} and for values of T_{LTD} ranging from 100 to 500 hours in increments of 50 hours. These values of dispatch intervals were also used in the TLD MCS code and, as was mentioned previously, the LOTC rate was obtained for the different possible maintenance approaches.

Table 8.2.2 and Table 8.2.3 show the results obtained using the methods outlined above for the 2 unit example system. Table 8.2.2 shows the results obtained using the TWA technique (column 2) and the single fault state Markov model (column 3). The two recommended approximations for the single fault state Markov model (columns 4 and 5) were also used. Table 8.2.3 shows the LOTC rate calculated for the system using the transient Markov model (columns 2, 3 and 4) and also the MCS (column 5). Results were obtained using the transient Markov model in 3 ways, all of which have been described previously. The first way was to include repairs in the Markov model as transitions with associated rates that appear in the transition rate matrix for the model and carrying out a numerical integration. The LOTC rate is calculated as time progresses within the integration and averaged over the period of the integration. This average converges and gives the results shown. The second and third ways both involve including repairs in the model at certain discrete time points within the integration procedure by shifting the appropriate state probabilities. The average LOTC rate is again obtained by averaging the LOTC rate over the course of the numerical integration and the results are shown after this average has converged. The difference between the two sets of results lies in when the LOTC rates are calculated – either at the start of a flight or at the end. The LOTC rate is higher when calculated at the end of a flight since it is before maintenance has taken place. At the start of a flight the maintenance will have been carried out and this means the LOTC rate, when calculated, will no longer include transitions from the repaired states whose probabilities will be zero because of the repairs.

First note that a desired behaviour is obtained when using all of the models in that the LOTC rate is seen to increase as T_{LTD} increases for the fixed value of T_{STD}. This is as would be expected, since the probability of a LOTC event occurring should be higher if dispatching with faults present in the system for longer periods of time. Looking at the values of LOTC rate obtained using the different methods it can be seen that they are all quite similar, although some differences can be noted. One might reasonably expect similar results from such a simple example, due to the fact that the small size of the model doesn't allow any approximations within the models to have a great effect on the results.

In order to be able to better compare the models it is better to consider the graphical representations of the results tables. These are shown in Figure 8.2.3 and Figure 8.2.4. In each figure there are two representations of the same data. The first shows a line graph of the data, which demonstrates the comparative LOTC rates obtained using the different methods. However, in both figures, some of the line graphs are so close as to be virtually indistinguishable and for this reason a bar chart displaying the same data is also included. This shows the values of the LOTC rate obtained using the different methods for each value of T_{LTD} considered. The bar charts more clearly show the underlying trends in the differences of the LOTC rates calculated using the different analysis techniques.

Consider first Figure 8.2.3, which shows the results obtained for the TWA approach with balanced coefficients (TWA, bal), the single fault state Markov model (SFS), the first order approximation to the single fault state Markov model (SFS,1) and the approximation to the single fault state Markov model used when dispatch intervals are short (SFS,short).

- The TWA approach gives, across the range of T_{LTD} covered, higher LOTC rates than the single fault state Markov approaches, although the single fault state Markov approximation used when dispatch intervals are short gives similar results to the TWA approach, but marginally lower.
- The single fault state Markov model and its first order approximation are close in their calculated LOTC rates with the approximation giving results slightly below the single fault state Markov results.

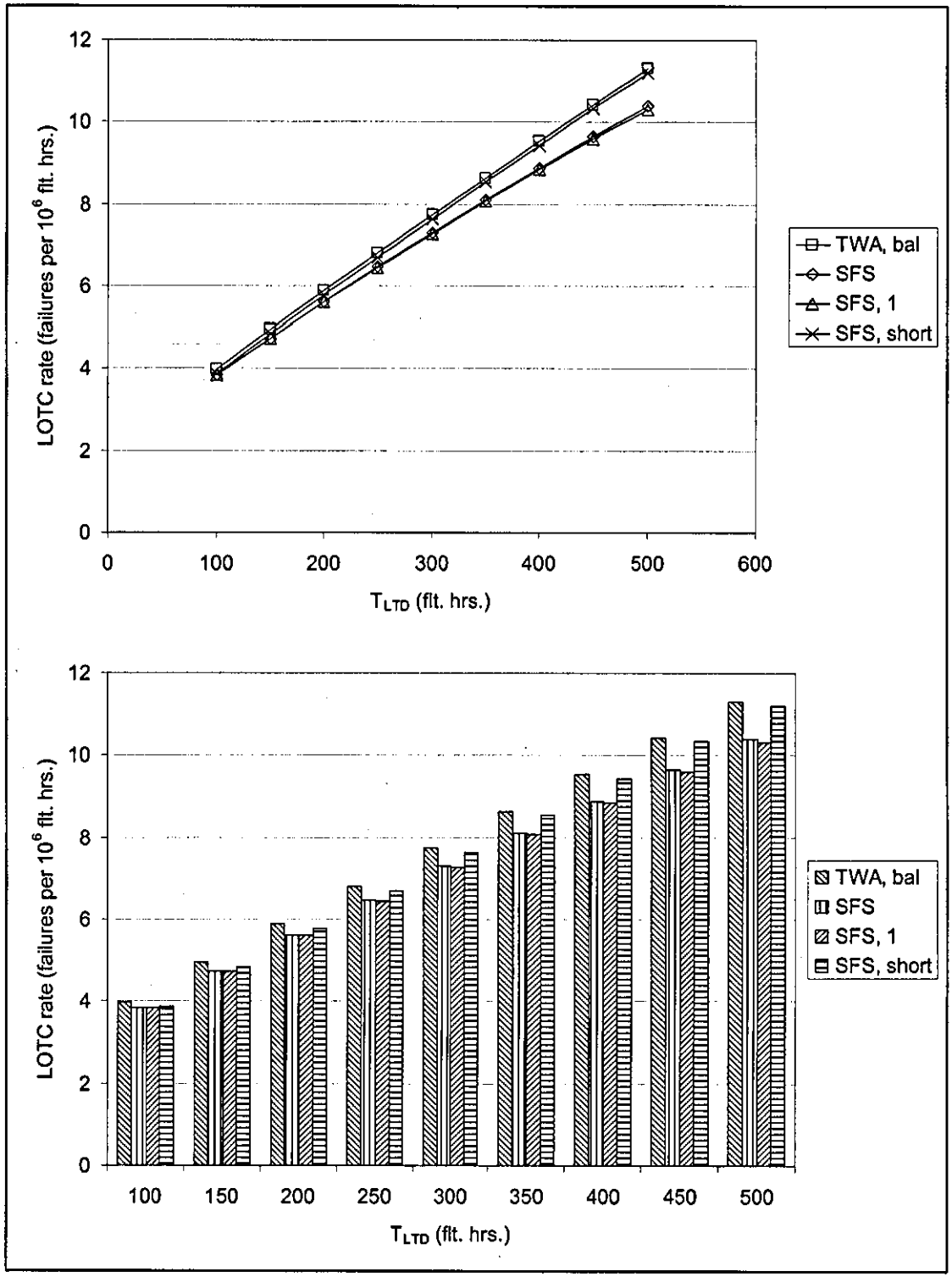


Figure 8.2.3. LOTC rates calculated for the 2 unit system using the TWA approach and 3 versions of the single fault state Markov approach for a STD interval of 100 flight hrs.

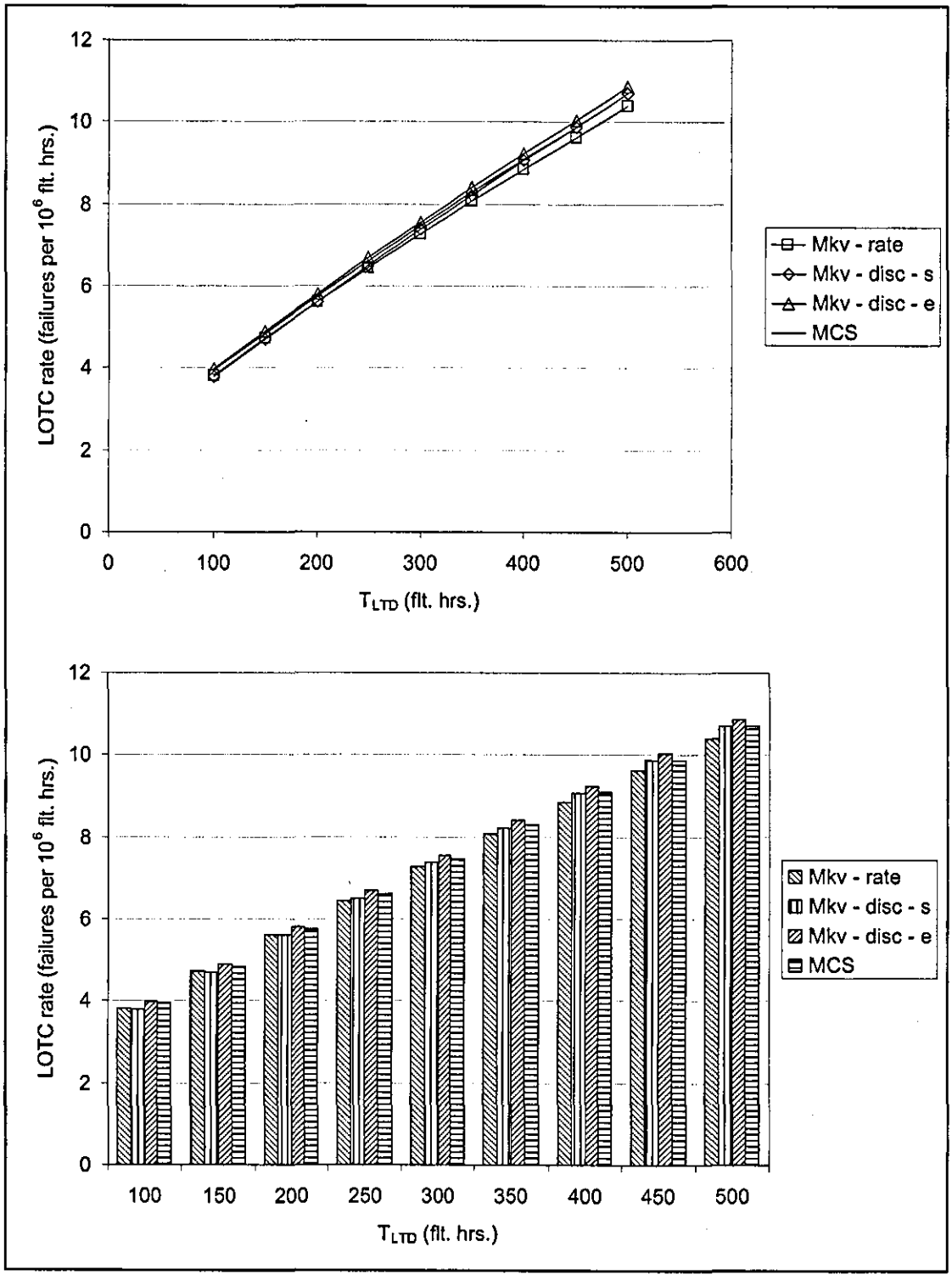


Figure 8.2.4. LOTC rates calculated for the 2 unit system using the transient solutions of the Markov model and the MCS approach for a STD interval of 100 flight hrs.

Moving on to Figure 8.2.4, this shows the MCS results (MCS) obtained when both STD and LTD faults are maintained using MEL maintenance, and the three different sets of results obtained when transient solutions are found to Markov models. Repairs are included in the transition rate matrix (Mkv – rate), or implemented at discrete points in time within the integration procedure (Mkv – disc). When repairs are implemented at discrete points in time the average LOTC rate is calculated using values obtained at the start (– s) or end of flights (– e).

- The transient Markov model in which LOTC rates are calculated at the start of flights gives generally similar results to the MCS model.
- The transient Markov model in which repairs are included in the Markov model as rates gives a LOTC rate that falls further and further below the MCS rate and transient model with repairs included discretely as T_{LTD} increases. This can be explained by the fact that the repairs are included more realistically in the latter three models. When the T_{LTD} interval becomes larger the repairs are effectively 'spread out' over a longer period of time rather than acting instantaneously.
- As expected, the solutions obtained using the discrete repair times in the transient Markov model differ, with the average LOTC rate obtained at the start of flights lower than the average LOTC rate obtained at the end of flights. This is explained by the repairs that take place between the end of one flight and the start of the next.

Figure 8.2.5 shows a comparison of the results obtained using the MCS approach and the TWA and single fault state Markov approaches. The LOTC rates obtained using the MCS is sandwiched by the other two methods, the TWA approach giving a higher rate and the single fault state Markov approach giving a lower rate. It is reasonable to assume that the MCS approach gives the more accurate results since it models the system in the most accurate manner, with dispatch and maintenance taking place exactly as intended. Bearing this in mind, the fact that the TWA approach overestimates the LOTC rates in comparison to MCS is good in that it is at least pessimistic. However, the fact that the single fault state Markov model underestimates the LOTC rate in comparison to the MCS approach is not good since this, in effect, gives an optimistic measure of the LOTC rate. For this simple system this is not a problem. However, if this kind of underestimation was to be displayed for a real system then the

consequences could be undesirable because the dispatch intervals set may be too lenient.

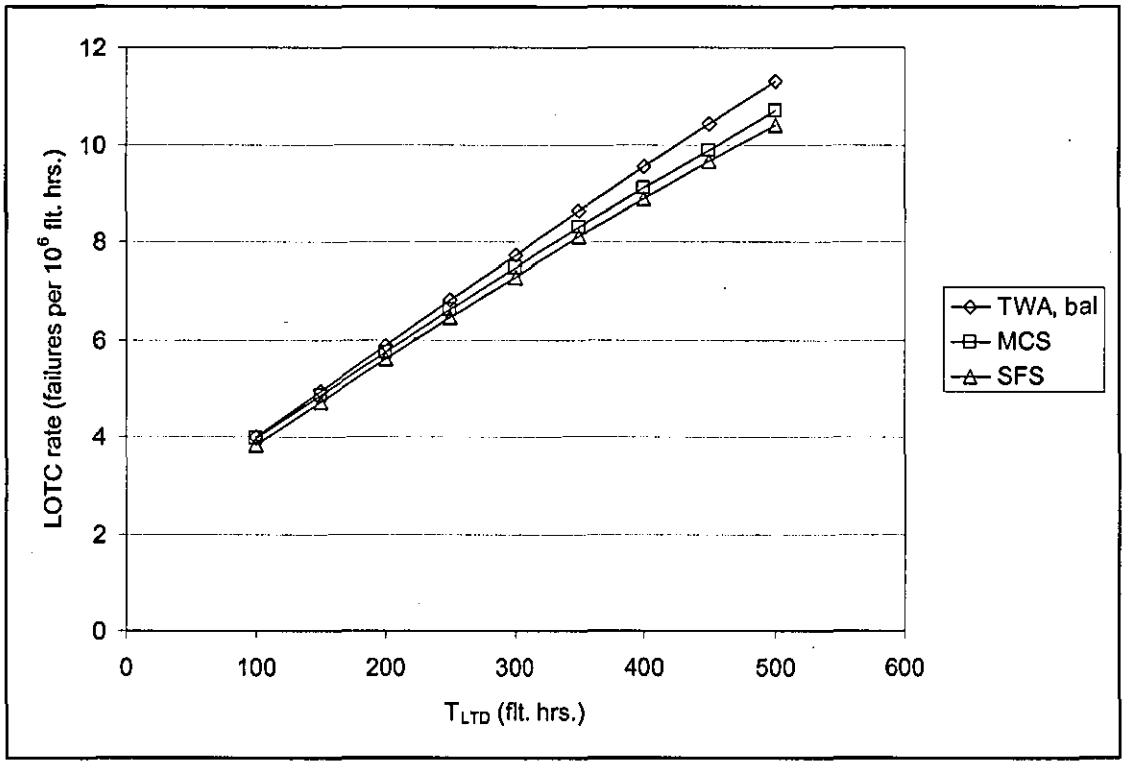


Figure 8.2.5. A comparison of the LOTC rates obtained for the 2 unit system using the TWA, single fault state Markov and MCS approaches for T_{STD} = 100 flt. hrs.

The dispatch intervals allowed for use with the 2 unit system will differ according to the model used. Recall that the limit for the average LOTC rate of the system is 10 failures per 10^6 flt. hrs. Given that, for each of the models, T_{STD} was set to be 100 flt. hrs. it remains to find the largest allowable value of T_{LTD} that can be applied whilst remaining below the 10 failures per 10^6 flt. hr. limit. This could be done using the graphs shown in Figure 8.2.3 and Figure 8.2.4 and tracing across from the 10 failures per 10^6 flt. hrs. mark on the y-axis to the relevant curve and down to the corresponding value of T_{LTD} on the x-axis. Table 8.2.4 shows the allowed value of T_{LTD} for each of the modelling approaches. The values in the table were actually calculated by linear interpolation of the results given in Table 8.2.2 and Table 8.2.3 and then rounded to the nearest 5 flt. hrs. It can be seen that the values vary from 425 hours for the TWA method to 480 hours for the first order approximation of the single fault state Markov model. This difference equates to 11 flights of 5 hours. Taking the results of the MCS, it can be seen that the LTD interval can be set at 460 hours. If one takes this as the benchmark (which

is considered reasonable given the techniques used in the MCS code) then three of the methods overestimate the LTD interval, these being the transient Markov solution with repair rates included in the transition rate matrix, the single fault state Markov model and the first order approximation to the single fault state Markov model.

	Transient Markov Model			MCS
	repairs as rates	discrete repair - start	discrete repair - end	STD - MEL LTD - MEL
T_{LTD}	475	460	450	460

	$\lambda_{TWA,bal}$	λ_{SFS}	$\lambda_{SFS,1}$	$\lambda_{SFS,short}$
T_{LTD}	425	475	480	430

Table 8.2.4. The values of T_{LTD} for which the LOTC rate is at the 10 failures per 10^6 flt. hrs. allowable limit (values given to the nearest 5 flt. hrs.)

8.3 Summary

A simple example system has been presented and used to demonstrate the application of the different techniques to modelling TLD. The results obtained using each of the methods were presented and compared. This example allowed the techniques to be introduced. However, when using this simple example it is not possible to utilise many of the options available in the TLD MCS code and also the system is not of sufficient complexity to properly analyse the differences between the models. For this reason a larger example system is discussed in the next chapter.

Chapter 9 : Automatically Setting Dispatch Criteria

9.1 Introduction

As part of this research project a method has been developed that allows dispatch criteria to be automatically set within a MCS. This method is included in the TLD MCS code and is described in the following section. The module of the TLD MCS code that performs this process is described in Appendix 2, Section A2.3.13. After the method is described here an example system is discussed and the dispatch criteria are set for that system and the different models available are applied. The system considered in this chapter is a progression from the system considered in the previous chapter, containing a greater number of components and applying TLD to different elements of the system.

9.2 The Methodology

When dispatch criteria are to be automatically set by the TLD MCS code a set of dispatch criteria can be specified in the data input file. An iterative process is then started using this set of dispatch criteria as the initial criteria which will be iteratively tested and amended. If no dispatch criteria are specified then the code will initially assume the most lenient scenario, where all dispatchable faults are assumed to be in the LTD category.

The iterative process of setting dispatch criteria begins by testing single faults for the initial set of dispatch criteria. The instantaneous failure rates to LOTC are found for these faults. Of course, when carrying out this procedure dispatch intervals must be specified for each of the STD and LTD categories of faults. Since ranges of these values within which the system LOTC rate will be calculated for the final set of allocated dispatch criteria must be specified in the input data file, the largest values of these intervals are used when setting the dispatch criteria. To illustrate this, consider that it has been specified that the system LOTC rate will ultimately be calculated for values of STD interval between 100 and 500 hours inclusive and values of LTD interval between 1000 and 5000 hours inclusive. When the instantaneous failure rates to LOTC are to be

calculated, the largest possible values for these intervals will be chosen, that is 500 hours for STD faults and 5000 hours for LTD faults. This ensures that ultimately, when the system LOTC rate is found, the dispatch criteria that have been applied to the system will be valid for the full range of values of dispatch intervals specified. This is the case since the instantaneous failure rates to LOTC for the various system configurations will have been calculated for the largest possible times of exposure to the faults. Once the instantaneous failure rates to LOTC are found, by calling the *control_sims* module, for the single faults they are checked for agreement with the dispatch categories that were applied to those single faults. At this stage any faults categorised as LTD faults whose failure rate lies above 75 or 100 failures per 10^6 flight hours will be amended to STD or DND faults respectively. STD faults whose failure rate is less than 75 failures per 10^6 flight hours will be amended to the LTD category (but only if no restrictions on the TLD category in the initial dispatch criteria have been supplied). STD faults whose failure rate is greater than 100 failures per 10^6 flight hours will be amended to the DND category. Finally, any faults that were initially DND faults will remain so. This adds an element of conservatism to the practice of setting dispatch criteria automatically. In the same way, throughout the entire process of automatically setting dispatch criteria, any DND faults will remain in that category once assigned. This is considered a reasonable practice because of its conservative nature.

Once TLD criteria have been set and adjusted for single faults the process continues for dual faults in the same way. The TLD criteria just calculated for the single faults are applied to the system and instantaneous failure rates to LOTC are calculated for all dispatchable faults up to dual faults. These are then checked for agreement with the dispatch criteria applied and new dispatch criteria are set accordingly. This process continues until TLD criteria have been set for faults up to the order specified. Once this order has been reached, if the TLD criteria had to be reassigned then the instantaneous LOTC rates will be checked against the TLD criteria again in order to verify that whatever dispatch criteria have just been amended have not affected the dispatch criteria applied to other faults. In this way the dispatch criteria are, in a sense, optimised. This process of re-checking specific orders of faults does not need to be carried out for each order of fault up to that specified by the user as the maximum since this procedure will automatically take place. This is because all dispatch criteria are tested each time the fault order is increased. To illustrate this concept consider a case

where 3 single faults, *A*, *B* and *C* are initially assigned to the LTD category, their instantaneous failure rates to LOTC are checked and faults *A* and *B* are seen to need placing in the STD category, whereas fault *C* should be in the DND category. If these criteria are applied it could be possible that the restriction placed on fault *C* could lead to fault *A* no longer needing to be in the STD category and actually being LTD, while fault *B* must remain in the STD category. The single faults alone could be tested to find out this information but if the dual faults are also tested (i.e. the instantaneous failure rates to LOTC for dual faults are also tested) then the single fault dispatch criteria will be updated appropriately along with the dual faults' criteria. The only time that this process wouldn't occur is once the order of faults specified has been reached. Hence the requirement for the instantaneous failure rates to LOTC to be checked for the highest fault order until the TLD criteria do not need to be reassigned.

During the process of setting dispatch criteria a number of output files are produced, which give the TLD criteria applied at each iteration of the automated process, the instantaneous failure rates to LOTC from the dispatchable system configurations for each of these sets of dispatch criteria and also the system LOTC rate obtained for these. As previously mentioned, these instantaneous LOTC rates are calculated for the upper values of the STD and LTD intervals specified as input data. The final values output by the TLD MCS code are the system LOTC rates that vary over the ranges of STD and LTD intervals specified. Since the TLD criteria allocated to the system satisfy the requirements related to the instantaneous failure rates to LOTC for the maximum specified values of the STD and LTD intervals, it is certain that the same requirements will be met for the lower STD and LTD intervals in the ranges, since the maximum possible exposure to the various fault combinations will be lower. If the code was used to automatically set the TLD criteria and then calculate the average system LOTC rate using the code for values of STD and LTD interval higher than those specified when setting the criteria for the system then checks should be carried out to ensure that the instantaneous failure rates to LOTC from the various dispatchable system configurations meet the certification requirements for the category applied to those dispatchable system configurations.

It should also be noted here why the TLD criteria are assigned in an iterative manner (assigning criteria for single faults, then dual faults, etc.) when the code is used to

automatically set the dispatch criteria. This method was chosen because of the increase in time involved in using the MCS code to calculate instantaneous failure rates to LOTC for higher order fault combinations, which, as a general rule, occur less frequently than lower order fault combinations. Thus, if dispatch criteria for lower order faults are set to the correct categories before higher order faults, there will be, in general, less chance that the higher order faults will also need to be changed.

9.3 Eight Component System Example

Consider the system depicted in Figure 9.3.1, which shows a reliability block diagram of a system containing 8 components. The system comprises of two channels, *X* and *Y*, each of which performs two functions, F_1 and F_2. Function F_1 on channel *X*, F_1X, is performed by the components *A* and *B* in parallel, F_2X is performed by *C* and *D* in series. The equivalent functions on channel *Y*, F_1Y and F_2Y, are performed respectively by *A* and *E* in a parallel configuration and *F* and *G* in a series configuration. There is also a link between the channels that, in the event of a function failure on one channel, allows the corresponding function on the other channel to be used in its place. This link is represented on the diagram by component *H*. The component failure rates for this system are given in Table 9.3.1 and the system fault tree is shown in Figure 9.3.2.

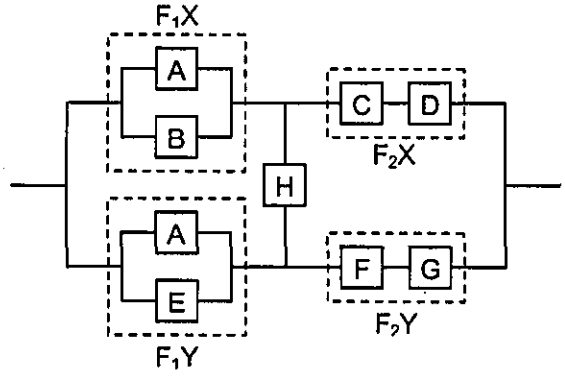


Figure 9.3.1. Eight component system.

Component/s	Failure rate (failures per hour)
A	7.5×10^{-5}
B, E	4.0×10^{-5}
C, F	3.0×10^{-5}
D, G, H	2.0×10^{-5}

Table 9.3.1. Component failure rates for the system.

The loss of the functions F_1X, F_2X, F_1Y and F_2Y and the loss of the link between the channels, H, are the system faults that will be included in the following TLD analyses. This represents the fact that the faults included in a TLD analysis of a real system will not necessarily correspond to the basic events of a fault tree, which will most likely correspond to component level failures, as is the case here. The flight length is assumed to be 10 hours and the system operational lifetime is assumed to be 130 000 hours. This system will, when modelled, require the inclusion of combinations of faults of order two and above in the dispatch criteria. These combinations will play a role in the LOTC rate of the system. As such, the TWA model is inappropriate for use in modelling this system because of the (dispatchable) combinations of faults that can occur and the fact that failure rates into these fault combination states would need to be calculated for use in the model. This brings in an extra level of approximation over and above that used in the dual fault state Markov model. This model is the most appropriate of the SAE techniques because it will at least account for transition into the dual fault states by including transitions from the single faults states to the dual fault states. This is in contrast to the TWA approach, which instead requires that a failure rate into the dual fault states must be calculated. For this reason the dual fault state Markov model is the only SAE model covered for this example. This is considered acceptable here, since, even in the case of the 2 unit example covered previously, the results obtained from the TWA model were the least accurate of the SAE models.

Rather than deciding on a set of dispatch criteria before modelling the system, the dispatch criteria are to be set using methods appropriate to the technique used to perform the analysis. Thus, for the MCS the TLD MCS code will be used to automatically set the dispatch criteria. For the dual fault state Markov approach the dispatch criteria will be set by first approximating the failure rates to LOTC from each of the system states using the technique of dividing a failure probability by a flight time described in section Section 5.3.6, equation 5.3.17, then applying the appropriate dispatch category. When finding the system LOTC rate using the transient solution of the conventional Markov model, the dispatch criteria from the other methods (i.e. MCS model or dual fault state Markov model) will be applied in turn. The STD interval, T_{STD}, is set at 500 hours and the LOTC rate must be modelled for values of LTD interval, T_{LTD}, ranging from 500 to 2500 hours in 500 hour intervals.

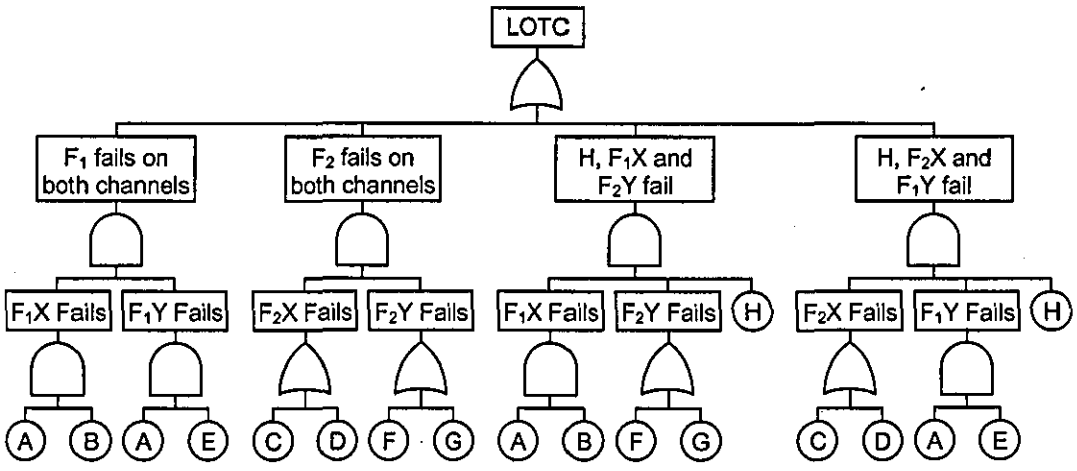


Figure 9.3.2. Fault tree of the eight component system.

9.3.1 The Dual Fault State Markov Model

Before the dual fault state Markov model can be constructed for the eight component system the instantaneous failure rates to LOTC for each of the fault combinations must be calculated. This then determines which of the fault combinations are dispatchable and should be included in the dual fault state Markov model. Once this process has been carried out the transition rate matrix for the model can be constructed and then used to calculate a solution. These steps are described below.

9.3.1.1 Determination of the Instantaneous LOTC Rates

Calculating the instantaneous failure rates to LOTC is the first step that must be carried out before a dual fault state Markov model can be constructed. The technique used to do this is that described in section 5.3.6, equation 5.3.17, where the failure probability of LOTC from each dispatchable configuration is divided by the length of a flight. To illustrate the process that will be used here consider, for example, the fault F_1X, which occurs when components A and B are failed. When these component failures are present in the fault tree the Boolean representation of the system top event LOTC is given by:

$$LOTC_{F1X} = E + C.F + C.G + D.F + D.G + F.H + G.H \qquad 9.3.1$$

where $LOTC_{F1X}$ represents the LOTC of the system given that F_1X is failed and '+' and '.' represent the Boolean OR and Boolean AND operators respectively. The rare event

upper bound approximation for the system unavailability given that F_1X is failed is given by:

$$Q_{SYS|F1X} = q_E + q_C q_F + q_C q_G + q_D q_F + q_D q_G + q_F q_H + q_G q_H, \qquad 9.3.2$$

where q_i represents the probability of failure of component i and is calculated as follows:

$$q_i = 1 - \exp(-\lambda_i t_{flt}), \qquad 9.3.3$$

where t_{flt} represents the length of a flight in hours. The use of the upper bound approximation brings a degree of conservatism to the calculations. In order to calculate the instantaneous failure rate to LOTC for F_1X equation 9.3.2 is divided by t_{flt}. Substituting the component failure rates from Table 9.3.1 into equation 9.3.2 gives a value of 4.0027×10^{-5} failures per flight hour for the instantaneous failure rate to LOTC from fault state F_1X. The instantaneous failure rate to LOTC for fault state F_1Y is identical due to the symmetry of the system.

Consider now fault state F_2X, which occurs when either component C or component D is failed. A degree of conservatism can be introduced if, when approximating the failure probability of LOTC given that F_2X is failed, both components are considered to be failed. Thus, by considering both components to be failed and carrying out the same procedure as that demonstrated above for F_1X, the rare event upper bound approximation for the system unavailability (itself an upper bound here) given that F_2X is failed is given by:

$$Q_{SYS|F2X} = q_F + q_G + q_A q_E q_H + q_A q_B q_E, \qquad 9.3.4$$

which yields a value of 4.9994×10^{-5} failures per flight hour for the instantaneous failure rate to LOTC from fault state F_2X. The instantaneous failure rate to LOTC for fault state F_2Y is identical due to the symmetry of the system.

The single fault state where H is failed has a fault tree whose Boolean representation is given by:

$$LOTC_H = C.F + C.G + D.F + D.G + A.B.E + A.B.F + A.B.G + A.C.E + A.D.E \qquad 9.3.5$$

which leads to the following rare event upper bound approximation for the system unavailability given that H has failed:

$$Q_{SYS|H} = q_C q_F + q_C q_G + q_D q_F + q_D q_G + q_A q_B q_E + q_A q_B q_F + \\ + q_A q_B q_G + q_A q_C q_E + q_A q_D q_E. \qquad 9.3.6$$

This gives a value of 2.5035×10^{-7} failures per flight hour for the instantaneous failure rate to LOTC from fault state H.

For the dual fault states the same technique is applied, with a conservative estimate applied (as above) when either of F_2X or F_2Y are present. The upper bound approximations for the system unavailability given that the various faults and fault combinations have occurred are then divided by the flight time (10 hours) to give the approximations for the instantaneous failure rates to LOTC for fault combination i, $\lambda_{i,L}$, given in Table 9.3.2. Also shown in this table are the TLD categories that must then be applied to the faults and fault combinations given the value of $\lambda_{i,L}$. Recall that if the instantaneous failure rate to LOTC is greater than 100 failures per 10^6 flight hours (i.e. 10^{-4} failures per hour) then the dispatch category must be DND, if the rate lies between 75 and 100 failures per 10^6 flight hours (i.e. between 7.5×10^{-5} failures per hour and 10^{-4} failures per hour) then the dispatch category must be STD and, finally, if the rate lies below 75 failures per 10^6 flight hours then the dispatch category must be LTD.

Fault (i)	$\lambda_{i,L}$	TLD Category
H	2.5035×10^{-7}	LTD
F_1X	4.0027×10^{-5}	LTD
F_1Y	4.0027×10^{-5}	LTD
F_2X	4.9994×10^{-5}	LTD
F_2Y	4.9994×10^{-5}	LTD
$H.F_1X$	8.9986×10^{-5}	STD
$H.F_1Y$	8.9986×10^{-5}	STD
$H.F_2X$	5.0023×10^{-5}	LTD
$H.F_2Y$	5.0023×10^{-5}	LTD
$F_1X.F_2X$	8.9986×10^{-5}	STD
$F_1X.F_2Y$	1.0998×10^{-4}	DND
$F_1Y.F_2X$	1.0998×10^{-4}	DND
$F_1Y.F_2Y$	8.9986×10^{-5}	STD

Table 9.3.2. The eight component system faults, their associated upper bound instantaneous failure rates to LOTC (calculated for the dual fault state Markov model) and the associated TLD categories.

9.3.1.2 The Dual Fault State Markov Model

Given the instantaneous failure rates to LOTC and the associated dispatch categories for each of the fault combinations a dual fault state Markov model can be constructed. There are 11 dispatchable system fault states and the model is depicted in Figure 9.3.3. Note that repair transitions are excluded from the model for clarity and that they will lead from each of the dispatchable fault states (numbered 1 to 11 inclusive) back to the full-up state. The feedback transition, with rate μ_{12} is shown on the model. Failure transitions lead from each of the dispatchable system states (including the full-up state) to the LOTC state 12.

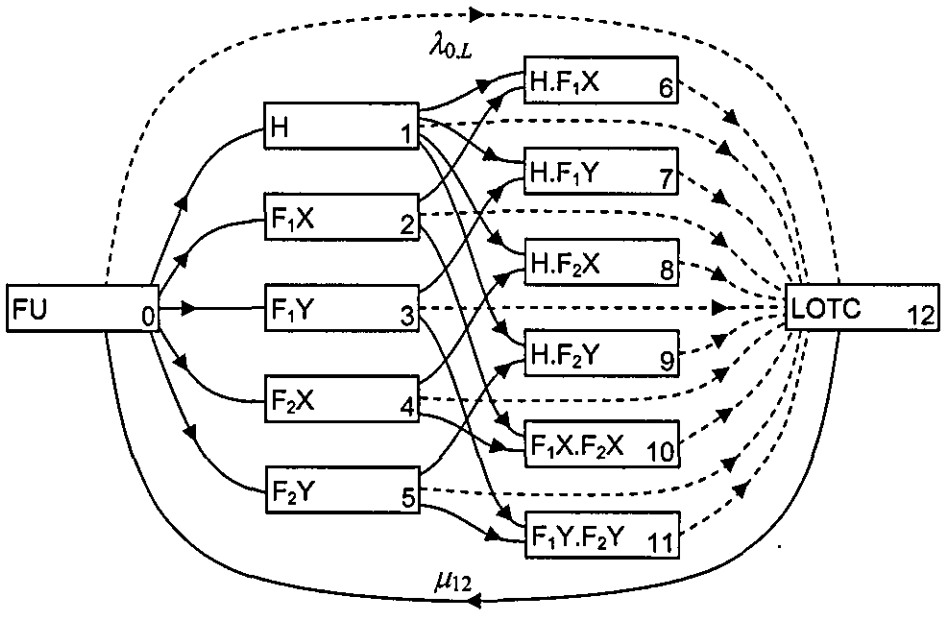


Figure 9.3.3. Dual fault state Markov model showing only failure transitions.

The dual fault state Markov model gives rise to a system of ordinary differential equations of the form given in equation 3.3.7. **A** is the transition rate matrix for the dual fault state Markov model, given by:

$$\begin{bmatrix} -\Sigma_0 & \lambda_H & \lambda_{F1X} & \lambda_{F1Y} & \lambda_{F2X} & \lambda_{F2Y} & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{0,L} \\ \mu_1 & -\Sigma_1 & 0 & 0 & 0 & 0 & \lambda_{F1X} & \lambda_{F1Y} & \lambda_{F2X} & \lambda_{F2Y} & 0 & 0 & \lambda_{1,L} \\ \mu_2 & 0 & -\Sigma_2 & 0 & 0 & 0 & \lambda_H & 0 & 0 & 0 & \lambda_{F2X} & 0 & \lambda_{2,L} \\ \mu_3 & 0 & 0 & -\Sigma_3 & 0 & 0 & 0 & \lambda_H & 0 & 0 & 0 & \lambda_{F2Y} & \lambda_{3,L} \\ \mu_4 & 0 & 0 & 0 & -\Sigma_4 & 0 & 0 & 0 & \lambda_H & 0 & \lambda_{F1X} & 0 & \lambda_{4,L} \\ \mu_5 & 0 & 0 & 0 & 0 & -\Sigma_5 & 0 & 0 & 0 & \lambda_H & 0 & \lambda_{F1Y} & \lambda_{5,L} \\ \mu_6 & 0 & 0 & 0 & 0 & 0 & -\Sigma_6 & 0 & 0 & 0 & 0 & 0 & \lambda_{6,L} \\ \mu_7 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_7 & 0 & 0 & 0 & 0 & \lambda_{7,L} \\ \mu_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_8 & 0 & 0 & 0 & \lambda_{8,L} \\ \mu_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_9 & 0 & 0 & \lambda_{9,L} \\ \mu_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_{10} & 0 & \lambda_{10,L} \\ \mu_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_{11} & \lambda_{11,L} \\ \mu_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_{12} \end{bmatrix} \qquad 9.3.7$$

where λ_H, λ_{F1X}, λ_{F1Y}, λ_{F2X}, λ_{F2Y} represent the failure rates of the faults considered for TLD. μ_i represents the repair rate for state i and $\lambda_{i,L}$ represents the instantaneous failure rate from state i to the LOTC state. Σ_i represents the sum of all elements except the ith in the ith row of the matrix. The failure rates for the faults H, F_1X, F_1Y, F_2X and F_2Y are calculated here using a similar method to that used to calculate the instantaneous failure rates to LOTC, i.e. the probability of the relevant fault occurring (approximated using the rare event upper bound) is divided by the length of a flight (i.e. 10 hours). For example, consider fault F_1X. This fault occurs if both of components A and B fail. Thus, for a fault tree top event F_1X the Boolean representation would be:

$$F_1X = A.B, \qquad 9.3.8$$

giving a failure probability of:

$$Q_{F1X} = q_A q_B. \qquad 9.3.9$$

This is then divided by the length of a flight to give an approximation for the failure rate into fault state F_1X. Considering now fault F_2X, the Boolean representation of the a fault tree with this top event would be:

$$F_2X = C + D, \qquad 9.3.10$$

giving a failure probability of:

$$Q_{F2X} = q_C + q_D. \qquad 9.3.11$$

This is then divided by the length of a flight to give an approximation for the failure rate into fault state F_2X. The failure rates for faults F_1Y and F_2Y are found in the same way. These failure rates are shown in Table 9.3.3

Fault (i)	λ_i
H	2.0000×10^{-5}
F_1X	2.9983×10^{-8}
F_1Y	2.9983×10^{-8}
F_2X	4.9994×10^{-5}
F_2Y	4.9994×10^{-5}

Table 9.3.3. Failure rates (failures per flight hour) for each of the faults considered for TLD.

The final failure rate to be calculated for the model is the full-up failure rate to LOTC, $\lambda_{0,L}$, which is calculated in a similar way to previous rates in the model by finding a rare event approximation of the failure probability of LOTC from the full-up state and dividing this by the length of a flight. Taking the rare event approximation to the system unavailability and dividing by 10 hours gives a value of 2.5×10^{-8} failures per flight hour.

All transition rates that are needed for solution of the dual fault state Markov model are now calculated and equation 5.5.29 can be used to give:

$$\lambda_{DFS} = \frac{\lambda_{0,L} + \sum_{i=1}^{11} \frac{Q_i}{Q_0} \lambda_{i,L}}{1 + \sum_{i=1}^{11} \frac{Q_i}{Q_0}}, \qquad 9.3.12$$

which is the dual fault state Markov LOTC rate for the eight component system. The equations represented by columns 2 to 12 (i.e. where $i = 1,\ldots,11$) of the transition rate matrix given in equation 9.3.7 are used to give algebraic expressions for Q_i/Q_0 which are then substituted into equation 9.3.12 to give an expression for the dual fault state LOTC rate of the system. This is then used to calculate the dual fault state LOTC rate of the system for the desired STD and LTD intervals by approximating the repair rates by the reciprocal of the relevant dispatch interval, see equation 5.5.1. Note that the final result for the dual fault state LOTC rate will not depend on the value of the feedback rate μ_{12}. The results obtained using this solution are given in column 4 of Table 9.3.4.

T_{LTD}	Transient Markov Model: repairs as rates	Transient Markov Model: discrete repair - start	Transient Markov Model: discrete repair - end	dual fault state Markov model
500	5.62	5.64	5.71	2.35
1000	7.95	8.18	8.24	4.37
1500	9.91	10.36	10.41	6.15
2000	11.60	12.27	12.33	7.71
2500	13.00	14.07	14.12	9.11

Table 9.3.4. Average system LOTC rates (failures per 10^6 flight hours) obtained for T_{STD} = 500 hours using the transient Markov model and the dual fault state Markov model.

9.3.2 The Transient Markov Model

The construction of the conventional Markov model for this system first involves identifying all of the possible system states. Since there are 8 components in the system the Markov model produced in the Markov code will contain $2^8 = 256$ states. As discussed previously, and carried out for the two unit system, two solution methods are possible when implementing the transient Markov solution. These involve including repairs as rates within the Markov model or modelling the repairs at discrete points in time within the numerical integration procedure. The fault tree and repair rates for individual components are passed to the Markov code described in Chapter 7, along with a file containing the TLD criteria for the system in terms of the basic events. This was done for the dispatch criteria calculated for the system for use in the dual fault state Markov model just described and also for the dispatch criteria calculated using the TLD MCS code. A step length of 0.1 is used in the Runge-Kutta integration routine.

In the case where repairs are included as rates within the transition rate matrix for the Markov model the STD interval, T_{STD}, is set at 500 hours and the LTD interval, T_{LTD}, set from 500 to 2500 hours in 500 hour intervals. Recall that, in the case where repairs are included at discrete points in the integration procedure, the points at which repair must take place are set at twice the equivalent value for the other case, since the repair intervals are analogous to the inspection intervals that are used in PIR maintenance. The results obtained using the transient Markov model code are given in columns 2 to 3 of Table 9.3.4.

9.3.3 Monte Carlo Simulation

When using the TLD MCS code to model the eight component system the option was taken to set the dispatch criteria automatically using the code, before the system LOTC rate was calculated over the range of STD and LTD intervals required. This was implemented for each of the repair strategies that are possible at maintenance within the TLD MCS code with MEL maintenance applied to both STD and LTD category faults. The results obtained are shown in Table 9.3.5. In all cases modelled an accuracy of 3 s.f. was specified for the results and 1 000 000 simulations were carried out before checking for convergence of the results. The results were subsequently checked every 50 000 simulations. Three consecutive values for the LOTC rate were required to match to 3 s.f. before convergence was assumed.

	MCS Repair Strategy			
T_{LTD}	1	2	3	4
500	11.10	11.10	11.00	5.68
1000	13.50	13.40	13.30	8.17
1500	15.50	15.40	15.40	10.40
2000	17.30	17.20	17.10	12.40
2500	18.90	18.80	18.70	14.10

Table 9.3.5. Average system LOTC rates (failures per 10^6 flt. hrs.) obtained to 3 s.f. for T_{STD} = 500 hours using the TLD MCS code with MEL maintenance applied to both STD and LTD faults.

9.3.4 Comparison of Results Obtained from Alternative Models

The first comparison to be made between the different methods is the dispatch criteria that are obtained for use with the system. Here only the dual fault state Markov model and the MCS models can be compared since the transient Markov model solutions are found using these dispatch criteria. The dispatch criteria for the dual fault state Markov model were calculated earlier. The dispatch criteria were set automatically by the TLD MCS code for the 4 different repair strategies possible at maintenance. Table 9.3.6 shows the instantaneous LOTC rates obtained for each of the repair strategies using the TLD MCS code (rep1...rep 4) and also the approximations obtained for use in the dual fault state Markov model. The rates are shown as the number of failures per 10^6 flight

hours. Also indicated on the table are the dispatch criteria associated with each failure rate.

Fault	Rep 1	rep 2	rep 3	rep 4	DFS
H	20.5	21.3	22.4	20.2	2.5
F_1X	48.0	48.5	49.1	49.6	40.0
F_1Y	48.0	48.5	49.1	49.6	40.0
F_2X	60.9	60.8	60.8	56.2	50.0
F_2Y	60.9	60.8	60.8	56.2	50.0
$H.F_1X$	90.0	89.9	90.0	89.9	90.0
$H.F_1Y$	90.0	90.1	89.8	90.4	90.0
$H.F_2X$	79.5	80.1	80.1	72.7	50.0
$H.F_2Y$	79.4	80.0	80.3	72.5	50.0
$F_1X.F_2X$	89.9	89.9	90.1	90.1	90.0
$F_1X.F_2Y$	110.0	110.0	110.0	110.0	110.0
$F_1Y.F_2X$	110.0	110.0	110.0	110.0	110.0
$F_1Y.F_2Y$	90.0	90.0	89.9	90.2	90.0
$H.F_1X.F_2X$	91.9	90.9	91.5	90.0	-
$H.F_1Y.F_2Y$	89.3	91.4	90.4	89.4	-

Key:	LTD	STD	DND

Table 9.3.6. Instantaneous failure rates to LOTC (in failures per 10^6 flight hours) for the eight component system obtained using the MCS code (repair strategies 1 – 4) and approximated for the dual fault state (DFS) Markov model.

It can be seen from the pattern of shading on the table that the dispatch criteria applied for the MCS where repair strategies 1, 2 and 3 are implemented are identical. The dispatch criteria for the MCS where repair strategy 4 is implemented and the dual fault state Markov model are also identical, differing from repair strategies 1, 2 and 3 only for the $H.F_2X$ and $H.F_2Y$ fault combinations (note that there are no rates calculated for the triple fault dispatchable criteria for the dual fault state Markov since, by definition, these states are not considered). However, there are discernable differences in the instantaneous failure rates to LOTC obtained for the different models. The values used in the dual fault state Markov model for all single faults and the $H.F_2X$ and $H.F_2Y$ fault combinations are markedly different from those obtained using the TLD MCS code. This is particularly true for fault H, which has a rate 10 times smaller in the dual fault state Markov model than that obtained using the TLD MCS code.

Note that the LOTC rates calculated using the TLD MCS code show less symmetry as the number of faults increases. For instance, due to the symmetry of the system one

would expect F_1X and F_1Y to have identical instantaneous failure rates to LOTC. This is the case. Indeed, in both cases where results should match for single fault states, this is so. When one looks at dual fault states, a slight variance is noted between states whose rates should be identical, although it is no more than a fraction of a percent. Looking at the triple fault states, the difference is greater, but is still less than one percent. This behaviour is easily explained. As the order of faults increases the occurrence of the faults and fault combinations become rarer. Thus, within the TLD MCS code, convergence is assumed when the instantaneous LOTC rates for the similar combinations have different values. The potential for differences is greater for higher order fault combinations. This is no problem here and could easily be overcome by changing the convergence criteria within the TLD MCS code or changing the number of simulations carried out or a combination of both of these solutions. The results are left in this format here in order to demonstrate the point that some variation is possible in the MCS results according to the convergence criteria chosen. Also, in this example the rates have converged sufficiently to allow the TLD criteria to be set. It is unlikely that the rates would need to be calculated to a greater accuracy than this.

The dispatch criteria found above were used in the system model for the appropriate solution technique and also when finding the transient Markov solution. Since the dispatch criteria are identical for the MCS when using repair strategy 4 and for the dual fault state Markov approach, these were the dispatch criteria used in finding the transient Markov solutions. The dispatch criteria found for the other repair strategies in the MCS were not used because the transient Markov code is written such that it models the situation where all faults are cleared from the system (i.e. repair strategy 4).

Table 9.3.4 shows the results obtained using the conventional transient Markov model and the dual fault state Markov model for a value of T_{STD} of 500 hours. The system LOTC rate obtained using the transient Markov model is again averaged at the start of flights and at the end of flights.

The average system LOTC rates obtained using the TLD MCS code for repair strategies 1 – 4 when MEL maintenance is applied to both STD and LTD category faults are shown in Table 9.3.5. As would be expected, as the repair strategy changes from 1 to 4 (and maintenance is carried out in a more stringent way) the LOTC rate obtained drops.

The most notable drop occurs from repair strategy 3 to repair strategy 4. This should be expected, since strategy 4 involves repairing all faults present in the system at maintenance. Consider the difference between repair strategy 1 (repair only the last of the faults causing the maintenance deadline) and repair strategy 2 (repair only the group of faults causing the maintenance deadline). This difference will probably be small for all systems, since one would expect single faults to dominate the LOTC rate. However, for a system of this size, with only 5 TLD faults, the difference could be expected to be very small indeed, which it proves to be. Next consider the difference between repair strategy 2 and repair strategy 3 (repair all faults of the same category as the one initiating the maintenance deadline). Again, this difference between the two strategies could be expected to be small because the extra time that faults remain in the system in strategy 2 will be relatively small in comparison to the system lifetime. This should again be accentuated by the fact that the system is relatively small. This appears to be borne out by the results.

The LOTC rates obtained are more easily examined using the diagrams in Figure 9.3.4 and Figure 9.3.5. Look first at Figure 9.3.4. The results obtained using the transient solution of the conventional Markov model display the behaviour exhibited by the same technique when used to model the 2 unit system, i.e.:

- When repair is included at discrete points within the numerical integration procedure the average LOTC rate calculated at the end of the flights is higher than the average LOTC rate calculated at the start of the flights,
- When repair is included as a transition rate in the model the average LOTC rate is closer to the results obtained using the discrete repair model for smaller values of T_{LTD}. This can be explained by the fact that the repairs are included more realistically when included at discrete points in time in the transient Markov model. When the T_{LTD} interval becomes larger the repairs (in the model where repairs are included as transition rates) are effectively 'spread out' over a longer period of time rather than acting instantaneously.

The LOTC rates obtained using the dual fault state (DFS) Markov model have the following attributes:

- The LOTC rates are generally far lower than for the transient Markov models,

- The gradient obtained over the range of T_{LTD} is generally similar to that obtained when using the transient models.

Figure 9.3.5 illustrates the points made earlier that the system LOTC rate falls as more repairs are carried out at maintenance (i.e. as we move from repair strategy 1 to repair strategy 4) and that there is a marked difference between repair strategy 4 and the other maintenance strategies. The interesting point to note here is that all of these strategies comply with the regulations but there are considerable differences between the different maintenance approaches.

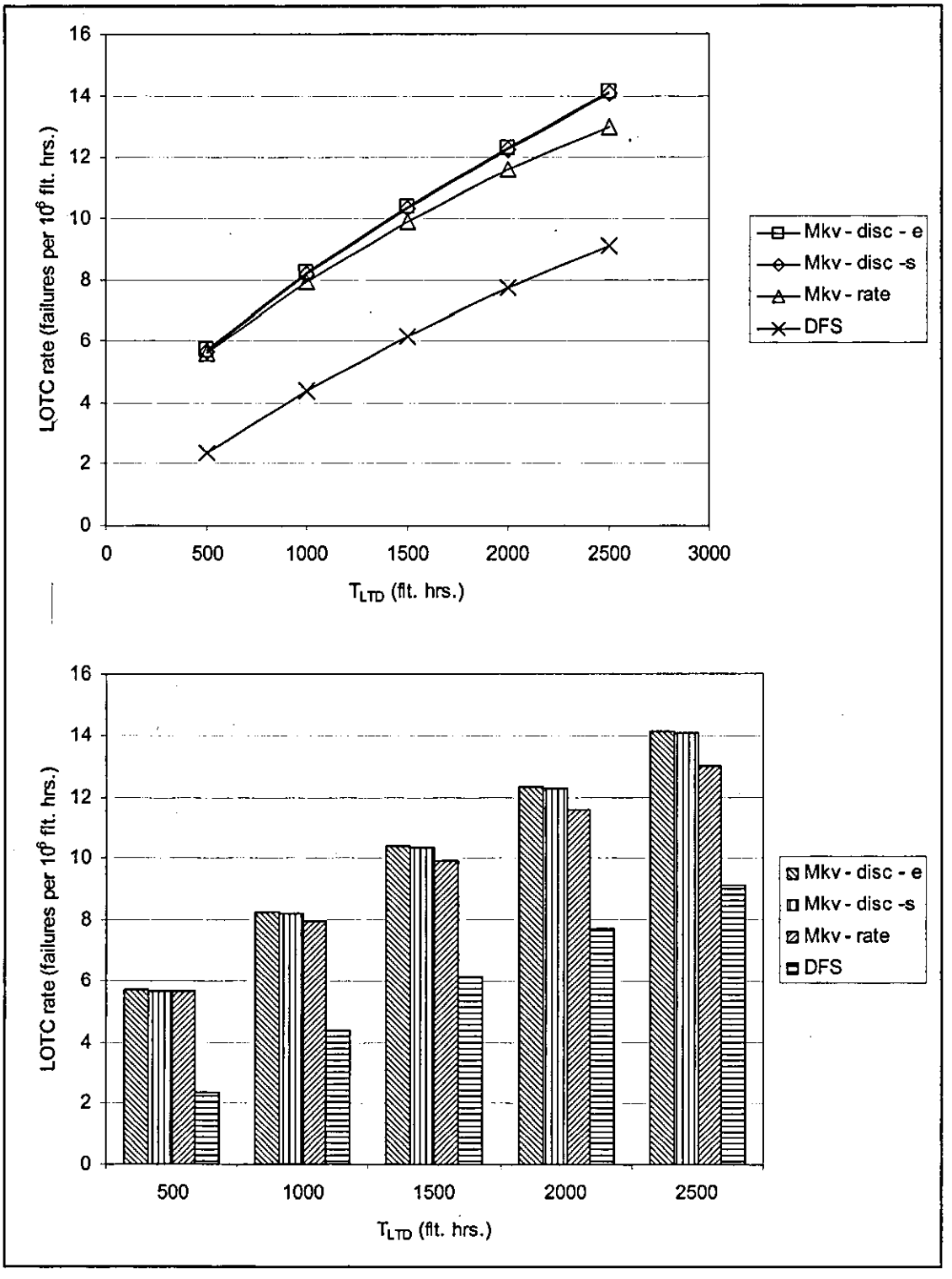


Figure 9.3.4. LOTC rates calculated for the eight component system using the transient solutions of the Markov model and the dual fault state Markov approach for T_{STD} = 500 hrs.

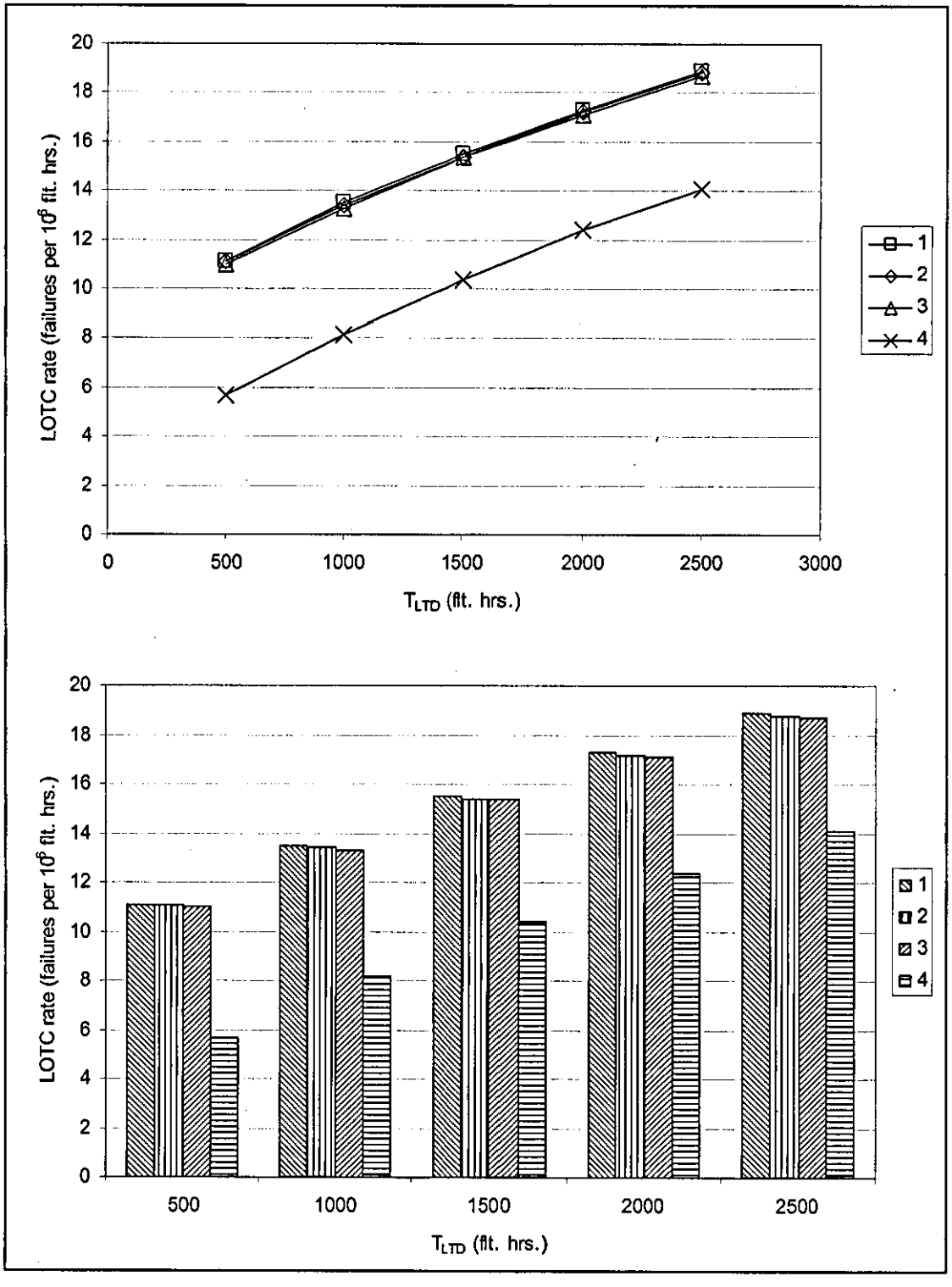


Figure 9.3.5. LOTC rates calculated for the eight component system using the four repair strategies available in the TLD MCS code with T_{STD} = 500 hrs. and STD and LTD category faults maintained using MEL maintenance.

The next step is to compare the MCS results with the other approaches, shown in Figure 9.3.6. Due to the similarity between some of the results a restricted dataset is considered. The MCS results for repair strategies 1 and 4 are compared with the transient Markov results obtained when repair is included at discrete points in the integration and the LOTC rate is averaged at the end of flights. The final results included in the figure are for the dual fault state Markov model.

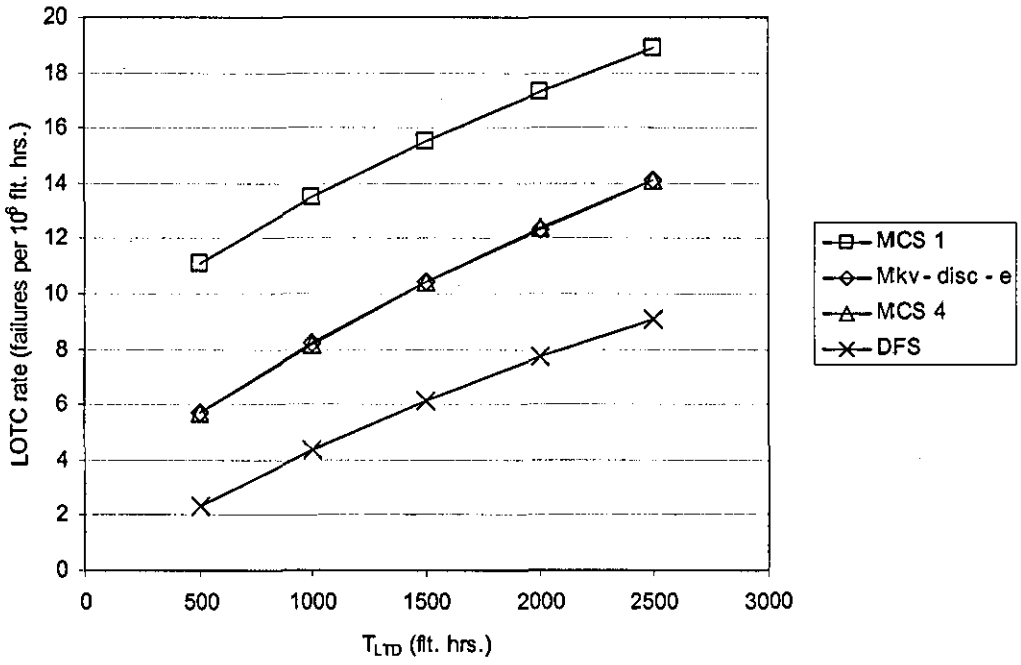


Figure 9.3.6. A comparison of the LOTC rates obtained for the eight component system using the TLD MCS code, the dual fault state Markov model and the transient solution to the conventional Markov model.

As was the case with the 2 unit system, it is assumed that the results obtained using the TLD MCS code are the most accurate because of the way that the dispatch and maintenance methods are most faithfully reproduced within the model. With this in mind the transient solution to the conventional Markov model compares favourably with the MCS results using repair strategy 4 (i.e. where all faults are cleared at maintenance). The dual fault state Markov model underestimates wildly in comparison to both of the other techniques. To illustrate the potential consequences if this was a real system imagine that the dual fault state Markov model was used to certify the system. This would allow a LTD interval of over 2500 hours for a STD interval of 500 hours and the LOTC rate would be under 10 failures per 10^6 flight hours. However, if one

looks at the MCS results where repair strategy 4 is used at maintenance one can see that the likely allowable value of LTD interval is below 1500 hours, a difference of 1000 hours or 100 flights. Then again, if one considers the worst-case scenario at maintenance, repair strategy 1, the allowable LTD interval would be below 500 hours, over 2000 hours lower. Clearly this would not be applied in reality since the LTD interval would be shorter than the STD interval. Therefore the STD interval would most likely be shortened as well as the LTD interval.

9.3.5 Further Dual Fault State Markov Model

It is clear that the dual fault state Markov model is not modelling the system in the correct manner for this eight component system. A possible reason for this is that the failure rates included in the dual fault state Markov model are incorrect. Recall that, in Table 9.3.6, a difference was noted between some of the instantaneous failure rates to LOTC obtained using the TLD MCS code and those approximated for use in the dual fault state Markov model. Note that the instantaneous failure rates to LOTC were lower for the dual fault state Markov model than obtained in the MCS. Thus, it could be useful to use the instantaneous failure rates to LOTC from the TLD MCS code in the dual fault state Markov model to see if the poor results obtained for the LOTC rate is caused by the approximations used to calculate the instantaneous failure rates to LOTC (despite the fact that these were approximated in a prescribed, conservative manner). Recall that the instantaneous failure rates to LOTC obtained using the TLD MCS code showed a small lack of symmetry for the dual and triple fault states. Before using the rates for the dual fault states in the dual fault state Markov model the appropriate rates were averaged to recapture the symmetry for the system. For example, the LOTC rates obtained for the dual faults $F_1X.F_2X$ and $F_1Y.F_2Y$ were averaged and the average value assigned to each of the combinations in the dual fault state Markov model. Also included in the model is the full-up failure rate calculated using the TLD MCS code – 7.34×10^{-6} failures per hour. Note that it is likely that the instantaneous failure rates to LOTC for the different faults and fault combinations will be slightly lower when the LTD interval is reduced. The instantaneous failure rates to LOTC substituted into the dual fault state Markov model from the TLD MCS code are calculated for a LTD interval of 2500 hours. The rates are likely to be lower for a LTD interval of 500 hours, at the bottom end of the LTD range for which average system LOTC rates are to be

calculated. This is not considered a problem here, however, since the use of the higher values will introduce a degree of conservatism into the model.

This procedure of substituting the instantaneous failure rates to LOTC from the TLD MCS code into the dual fault state Markov model was implemented and the results are shown (labelled as DFS – new) in Table 9.3.7, along with the results obtained using the TLD MCS code using repair strategies 1 and 4 and the original dual fault state Markov approach (labelled as DFS – orig). The same results are displayed in Figure 9.3.7. It can be seen that the dual fault state Markov model with the MCS rates included within it now produces system LOTC rates that lie closer to those produced by the MCS code with repair strategy 4. However, the LOTC rates obtained are still generally lower than for the MCS. This suggests that, given the right failure rates for inclusion in the model, a reasonable approximation to the LOTC rate can be obtained. However, a question remains as to how these failure rates should be accurately calculated. Also, the accuracy of the dual fault state Markov model must be questioned when it provides an underestimate of the results obtained using the TLD MCS code. This suggests that, structurally, there is a problem with the dual fault state Markov model. A final point should be made that echoes a point made previously – that the system could be maintained in practice using the techniques outlined for repair strategies 1-3, which all give similar results to repair strategy 1 for this system (strategy 1 is plotted). If the system was to be maintained in this way at any point the dual fault state Markov model provides no way of calculating the appropriate allowed TLD intervals.

T_{LTD}	MCS – 1	MCS – 4	DFS – orig	DFS – new
500	11.10	5.68	2.35	5.45
1000	13.50	8.17	4.37	7.75
1500	15.50	10.40	6.15	9.76
2000	17.30	12.40	7.71	11.55
2500	18.90	14.10	9.11	13.13

Table 9.3.7. Average system LOTC rates (failures per 10^6 flt. hrs.) obtained to 3 s.f. for T_{STD} = 500 hours using the TLD MCS code with MEL maintenance applied to both STD and LTD faults compared with the LOTC rates obtained by substituting the instantaneous failure rates to LOTC from the TLD MCS code into the dual fault state Markov model.

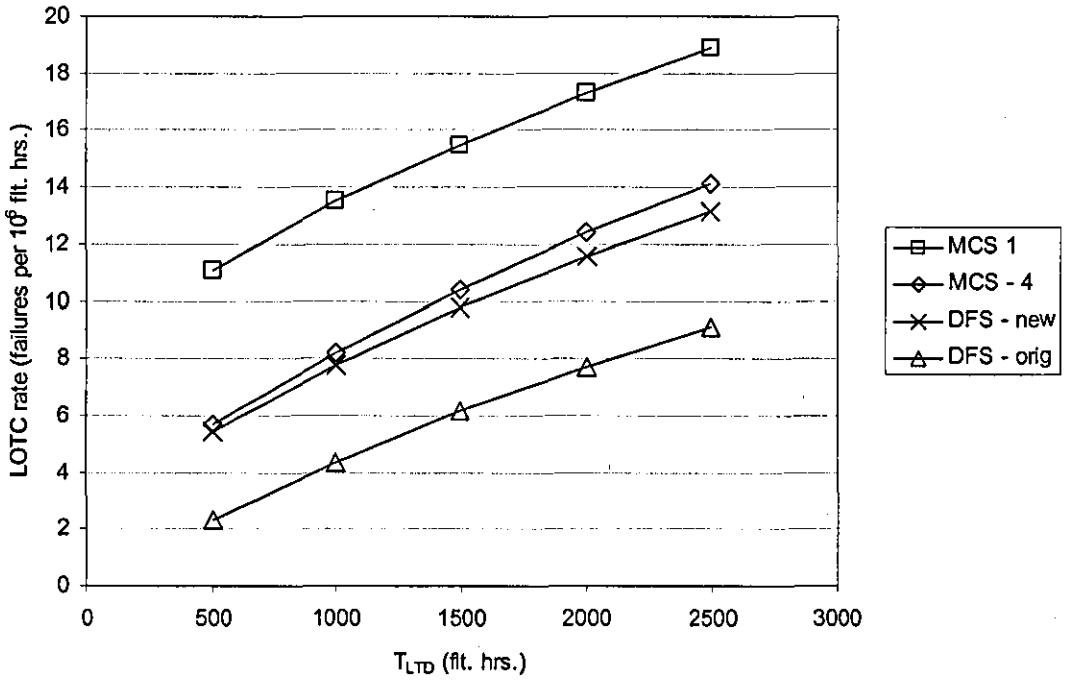


Figure 9.3.7 . A graph of the results shown in Table 9.3.7.

9.3.6 Conclusions

A number of conclusions may be drawn about the modelling of the eight component system using the different modelling techniques. The first point to make is that the TLD MCS code appears to produce the more accurate results for this system, which is due to the fact that it models the system more realistically. Dispatch is covered in the desired way and the appropriate repairs are carried out at maintenance that is carried out at the correct times. The results obtained using the conventional Markov model when transient solutions are found appear to back this up. The conventional Markov model with repairs included at discrete points in time in particular produces results that correspond well to those of the TLD MCS code. However, the conventional Markov model has a number of drawbacks, even for this small system. In no particular order, these are:

- The conventional Markov model requires dispatch criteria to be set for the system in some way. Here the techniques used to set the criteria for the TLD MCS code and the dual fault state Markov model were used.

- The conventional Markov model will increase in size exponentially. Even for this small system, the model contains 256 fault states, not a problem here, but this would soon become a problem for larger systems even if conventional methods of reducing the size of the Markov model were used. Ultimately, this is why conventional Markov models cannot be used to model TLD.

The dual fault state Markov model seems to not compare favourably with the other two methods used here. The possible reasons for this are listed below:

- The methods used to calculate the transition rates used in the model, while supposed to be conservative, cannot guarantee upper bounds, at least for this system. However, since the system considered here is still relatively small and simple, this could suggest a problem in executing reduced state Markov methods for larger systems.
- Given the instantaneous failure rates to LOTC calculated using the TLD MCS code, these were substituted into the dual fault state Markov model. This improved the results obtained using the dual fault state Markov model. However, the average system LOTC rates were still underestimated somewhat. This problem could be exacerbated when larger systems are considered. The fact remains that the dual fault state Markov model, if used in isolation, simply produces results that would be difficult to verify. The inaccuracies of the model, even for such a small system with correct rates included in it, suggests that the fundamental structure of the model might not be appropriate.
- The final point about the dual fault state Markov model, and the other commonly used reduced state Markov models (the SAE models, not conventional reduced Markov models), is that they are constructed in such a way that they model a particular maintenance method (equivalent to repair strategy 4 in the TLD MCS code). However, as has been demonstrated for this simple system, the results obtained for different repair strategies at maintenance can vary. This could cause problems when analysing the application of TLD to systems for certification.

9.3.7 Further Results Obtained Using the MCS Code

As discussed in Chapter 6, the TLD MCS code allows great flexibility when modelling a system. The following sections give brief examples of some of the different information that can be gleaned from the code and some of the other situations that can be modelled. This is intended to give a flavour of the options available when using the TLD MCS code. The options considered are the inclusion of PIR maintenance, opportunistic maintenance, calculating importance measures and varying the system operational lifetime. Sample output obtained from a single simulation is also shown. The length of a flight was 10 hours and the operational lifetime of the system was 130000 hours.

9.3.7.1 PIR Maintenance

The eight component system was modelled using the TLD MCS code for the case where MEL maintenance is used for STD faults and PIR maintenance is used for LTD faults. The ratio of inspection interval to dispatch interval was set at 2, the maximum allowed. A system LOTC rate was obtained for T_{STD} = 500 hours and T_{LTD} = 500...2500 hours in 500 hour intervals. The TLD criteria used were those calculated for the system when using repair strategy 4 shown in Table 9.3.6. Table 9.3.8 shows the results obtained. As would be expected, the LOTC rate generally increases as the ratio of inspection interval to dispatch interval is also increased. As the ratio increases the average exposure of the system to TLD faults is also increased. This increase in average exposure leads to the observed increase in LOTC rate.

T_{LTD}	PIR ratio of inspection to dispatch interval		
	1.0	1.5	2.0
500	5.61	5.66	5.67
1000	8.05	8.06	8.20
1500	10.10	10.20	10.40
2000	11.90	12.00	12.30
2500	13.50	13.60	14.10

Table 9.3.8. LOTC rates (failures per 10^6 flt. hrs.) obtained using the TLD MCS code for the eight component system with LTD faults obtained using PIR maintenance.

9.3.7.2 Opportunistic Maintenance

The eight component system was modelled using the TLD MCS code for a situation where opportunistic maintenance is included in the maintenance strategy and repair strategy 2 is used. The opportunistic maintenance window was set to be 250 hours. As a reminder, this means that as maintenance takes place, if further maintenance is planned for within the next 250 hours then this maintenance will also be carried out at the current maintenance. A system LOTC rate was obtained for $T_{STD} = 500$ hours and $T_{LTD} = 500...2500$ hours in 500 hour intervals. The TLD criteria used were those calculated for the system when using repair strategy 2 shown in Table 9.3.6. Table 9.3.9 shows the results obtained. As would be expected, when opportunistic maintenance is used the system LOTC rate can be seen to fall. This is an important result in practice, since it is likely that opportunistic maintenance will be implemented in real situations. Opportunistic maintenance will give the opportunity for faults to be repaired in advance of their scheduled maintenance time, thus giving a potential benefit in terms of reducing potential delays brought about by maintenance operations. It will also allow the number of maintenance operations to be reduced by utilising the flexibility of TLD. As such it is important that opportunistic maintenance can be modelled in this way for systems. As seen here, using opportunistic maintenance reduces the LOTC rates observed for the eight component system. These reduced LOTC rates would allow longer dispatch intervals to be legally set for the system, fully utilising the benefits of TLD.

T_{LTD}	with	w/o
500	11.00	11.10
1000	13.30	13.40
1500	15.10	15.40
2000	16.60	17.20
2500	17.90	18.80

Table 9.3.9. LOTC rates (failures per 10^6 flt. hrs.) obtained using the TLD MCS code for the eight component system with and without (w/o) opportunistic maintenance with a window of 250 hours.

9.3.7.3 Varying the System Operational Lifetime

All the previous MCS models of the eight component system have considered an operational lifetime of 130 000 hours for the system. However, it should be clear that the operational lifetime of the system could have an effect on its computed LOTC rate. In order to demonstrate this a system LOTC rate was calculated for repair strategy 4, with $T_{STD} = 500$ hours and $T_{LTD} = 1500$ hours. The length of a flight remained at 10 hours but the system operational lifetime was varied between 1 000 and 130 000 hours. Between 1 000 and 10 000 hours the LOTC rate was calculated at 1 000 hour intervals and between 10 000 and 130 000 hours the LOTC rate was calculated at 5 000 hour intervals. The TLD criteria used were those calculated for the system when using repair strategy 4 shown in Table 9.3.6. Table 9.3.10 shows a restricted set of the results. Figure 9.3.8 shows the full set of results obtained. The system LOTC rate can be seen to rise sharply initially as the operational lifetime increases. After this it tends towards a value of just over 10 failures per 10^6 flight hours. What this example demonstrates is that the operational lifetime of a system does indeed have an impact on the average system LOTC rate. This is yet another facet of the application of TLD that cannot be modelled using the recommended methods of reduced state Markov and TWA approaches.

Operational Lifetime	LOTC Rate
5000	6.58
10000	7.96
15000	8.60
20000	9.01
25000	9.33
30000	9.51
35000	9.67
40000	9.71
45000	9.77
50000	9.84
55000	9.88
60000	9.92
65000	9.96
70000	9.99
75000	10.00

Table 9.3.10. LOTC rates (failures per 10^6 flt. hrs.) obtained using the TLD MCS code for the eight component system for a varying system operational lifetime.

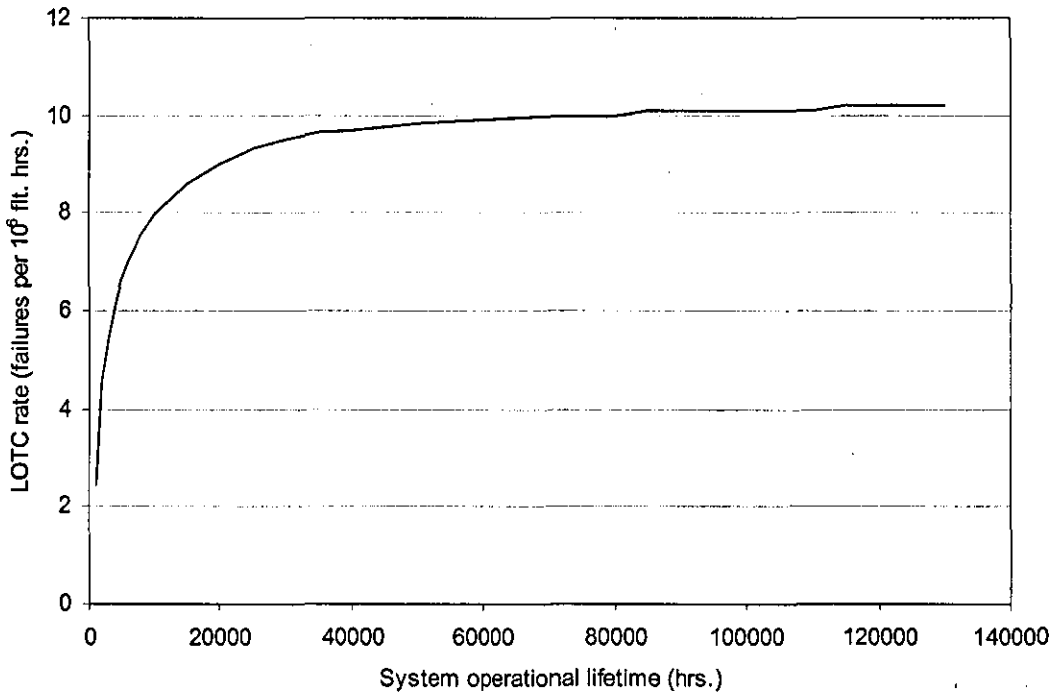


Figure 9.3.8. A graph showing the average system LOTC rate for the eight component system for varying system operational lifetimes.

9.3.7.4 Importance Measures

The eight component system was modelled using the TLD MCS code for repair strategy 4 with a flight length of 10 hours and an operational lifetime of 130 000 hours, as before. Importance measures were obtained for the components and TLD faults of the system for T_{STD} = 500 hours and T_{LTD} = 1500 hours. The TLD criteria used were those calculated for the system when using repair strategy 4 shown in Table 9.3.6.

Table 9.3.11 shows the results obtained for the system components. Figure 9.3.9 shows a bar chart that depicts the same results. These importance measures rank the components as one might expect for this system, with component A being ranked as the most important and component H being ranked as the least important. These importance measures give an indication of which components attention should be focussed upon in order to improve the system performance (in terms of the average system LOTC rate).

Fault	Importance Measure
A	5.55×10^{-1}
B	4.99×10^{-1}
C	3.69×10^{-1}
D	2.46×10^{-1}
E	4.99×10^{-1}
F	3.69×10^{-1}
G	2.46×10^{-1}
H	3.55×10^{-2}

Table 9.3.11. Importance measures (to 3 s.f.) for the components of the eight component system obtained using the TLD MCS code.

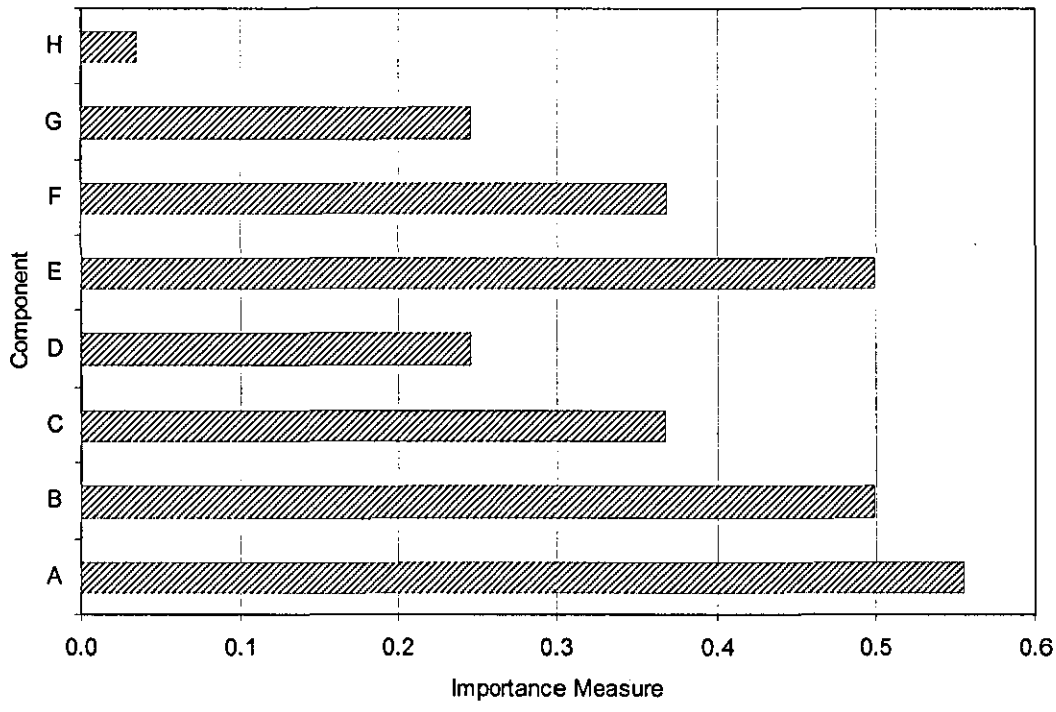


Figure 9.3.9. Component importance measures for the eight component system.

Table 9.3.12 shows the importance measures obtained for the single and dual dispatchable TLD faults. The corresponding bar charts are shown in Figure 9.3.10.

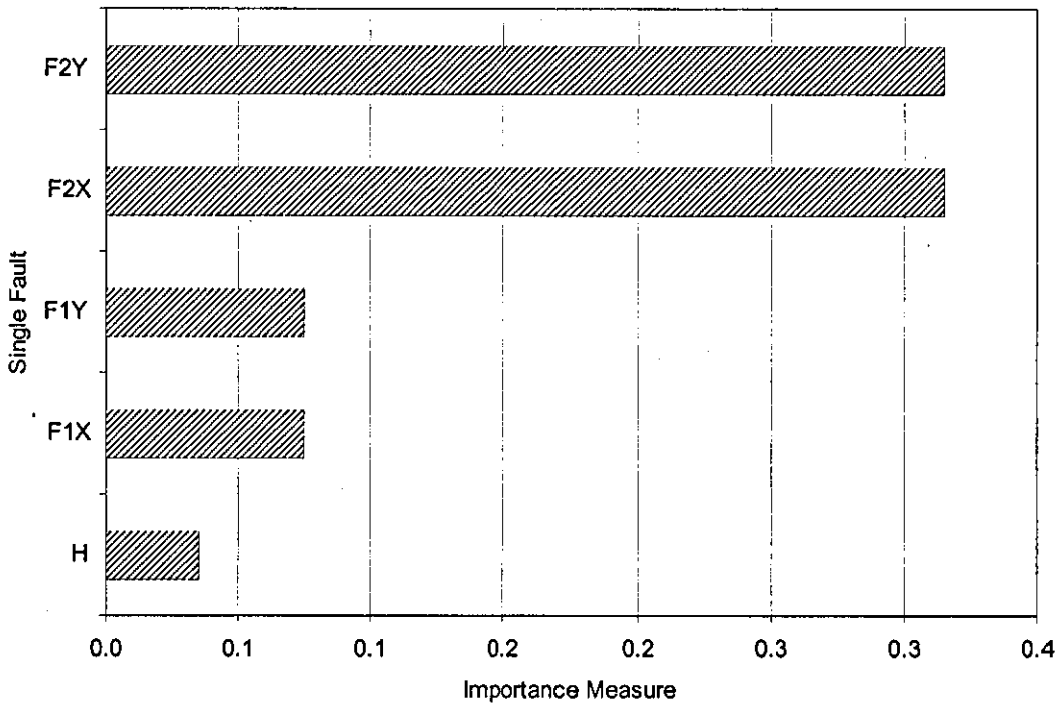
The single dispatchable faults are generally more important than the dispatchable dual faults in this model. This could be expected. Of the single faults it can be seen that F_2X and F_2Y are ranked as by far the most important faults. This could also be expected because of the fact that they both consist of components in series, although one could equally expect that faults F_1X and F_1Y would be ranked as more important because of

the fact that they contain the most important component, A. Considering the dual faults, $H.F_2X$ and $H.F_2Y$ are ranked as the most important.

Importance measures such as these will be useful in the event of a real analysis because they demonstrate where resources are best expended in order to improve the system performance.

Fault	Importance Measure
H	3.55×10-2
F_1X	7.48×10-2
F_1Y	7.48×10-2
F_2X	3.15×10-1
F_2Y	3.15×10-1
$H.F_1X$	2.69×10-3
$H.F_1Y$	2.69×10-3
$H.F_2X$	1.17×10-2
$H.F_2Y$	1.17×10-2
$F_1X.F_2X$	6.67×10-3
$F_1Y.F_2Y$	6.67×10-3

Table 9.3.12. Importance measures (to 3 s.f.) for the dispatchable single and dual faults of the eight component system obtained using the TLD MCS code.



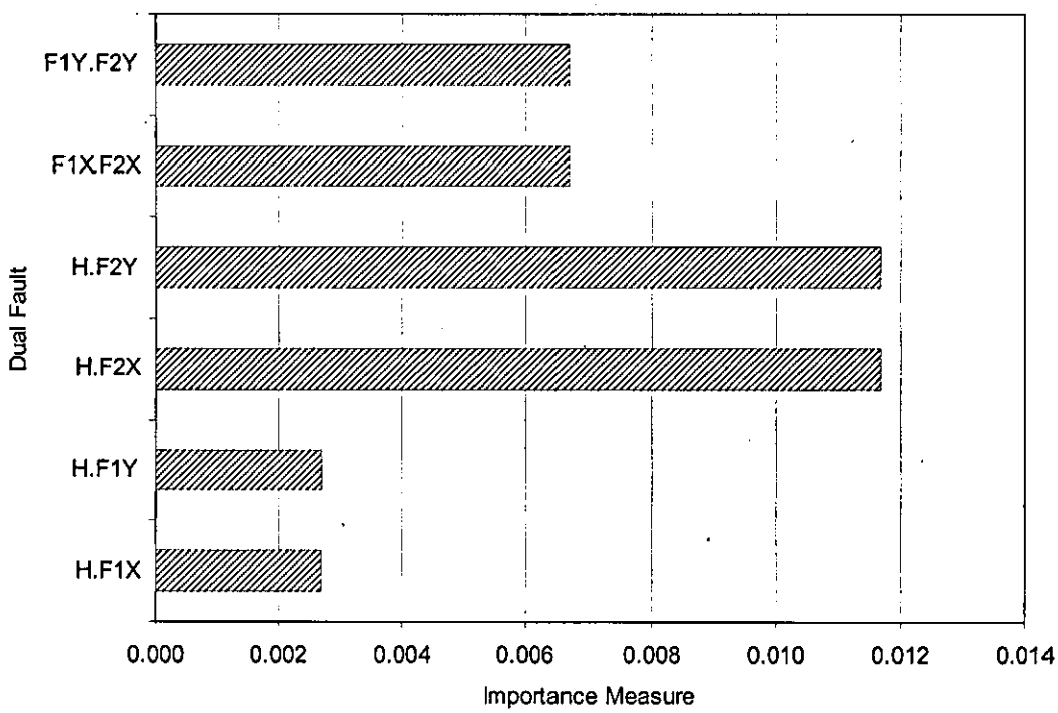


Figure 9.3.10. Single and dual fault importance measures for the eight component system.

9.3.7.5 Optional Simulation Output

When modelling TLD using the TLD MCS code it is possible to output a detailed account of the failures and maintenance operations that occur within simulations and their times of occurrence. Failure propagation is also detailed, along with the TLD faults present in a system. Figure 9.3.11 shows a sample of such an output file, for repair strategy 4 with both STD and LTD faults covered by MEL maintenance and having TLD intervals of 500 and 1000 hours respectively. In the file a number of failures occur and also a number of maintenance operations relating to these failures.

For example, initially component C fails (shown as basic event C – E:C) at time 3083.40. The failure is propagated up through the fault tree and fault F_2X is seen to occur (shown as G:F2X_FAILS) and the appropriate LTD maintenance deadline (code 70000) is added to the schedule 1000 hours after the end of the current flight at time 4090.00. After all options for propagation of the fault up the fault tree are checked the TLD faults present in the system are output before the simulation continues.

The next event in the schedule is the maintenance event. Because repair strategy 4 is being used all of the faults are cleared from the system. This means that F_2X and C are no longer failed and a new failure time is generated for component C, which will fail again at time 46788.16. The simulation then continues.

This kind of file is particularly useful when checking if TLD operations and maintenance strategies are being correctly implemented within the TLD MCS code. Files such as this were used extensively when writing the TLD MCS code in order to ensure that repair strategies were being implemented in the desired way.

```
-----------------------------------------------------------------------------
        3083.40    | * E:C fails
                   |   checking new status of parent G:F2_X_FAILS, OR gate
                   | * G:F2_X_FAILS fails
                   | * Adding 70000 to the schedule, time:   4090.00
                   |   checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |   checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
        4090.00    | *** Maintenance ***
                   | G:F2_X_FAILS now works
                   | E:C                 next failure:        46788.16
-----------------------------------------------------------------------------
        5627.39    | * E:E fails
                   |   checking new status of parent G:F1_Y_FAILS, AND gate
                   | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
        6112.40    | * E:B fails
                   |  E:E is also failed
                   |   checking new status of parent G:F1_X_FAILS, AND gate
                   | * G:F1_X_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
        8706.50    | * E:H fails
                   |  E:B is also failed
                   |  E:E is also failed
                   | * Adding 70000 to the schedule, time:   9710.00
                   |   checking new status of parent G:GATE_3, AND gate
                   | * G:GATE_3 works
                   |   checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: E:H
-----------------------------------------------------------------------------
        9710.00    | *** Maintenance ***
-----------------------------------------------------------------------------
       12405.77    | * E:A fails
                   |   checking new status of parent G:F1_X_FAILS, AND gate
                   | * G:F1_X_FAILS works
                   |   checking new status of parent G:F1_Y_FAILS, AND gate
                   | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       13183.33    | * E:G fails
                   |  E:A is also failed
                   |   checking new status of parent G:F2_Y_FAILS, OR gate
                   | * G:F2_Y_FAILS fails
                   | * Adding 70000 to the schedule, time:  14190.00
                   |   checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |   checking new status of parent G:GATE_3, AND gate
                   | * G:GATE_3 works
TLD faults present: G:F2_Y_FAILS
-----------------------------------------------------------------------------
       14190.00    | *** Maintenance ***
                   | G:F2_Y_FAILS now works
                   | E:G
-----------------------------------------------------------------------------
       21305.73    | * E:E fails
                   |   checking new status of parent G:F1_Y_FAILS, AND gate
                   | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       23079.31    | * E:A fails
.cont............
```

Figure 9.3.11 Sample output from the first part of a single simulation of the eight component system. See Appendix 4 for the complete output.

9.4 Summary

In this chapter, and the previous one, a number of different models have been used to obtain results for two simple systems, one containing just two components, the other containing eight components. It has been argued that MCS provided the most accurate results for these systems and there is no reason to expect that this would not be the case for larger systems. Certain flaws in the currently recommended approaches were identified:

- The TWA approach yielded inaccurate results even for the two component system and cannot be used for systems where dual faults are to be considered.
- The method of approximating failure rates for the SAE methods by dividing a failure probability by a flight time was seen to be potentially inaccurate.
- The results obtained using the reduced state Markov models were shown to be heavily dependent upon the rates used within them – a problem when the failure rates themselves could be inaccurate.
- The structure of the reduced state Markov models was called into question since the results obtained when more accurate failure rates were substituted into the dual fault state Markov model were seen to be optimistic in comparison to the MCS results.
- The currently recommended models would require some (perhaps impossible, certainly complex) changes in order to be able to cover some of the options available to the user of the MCS code, such as different repair strategies and opportunistic maintenance.

The flexibility of MCS as a means of modelling the application of TLD to systems was demonstrated by implementing some of the options open to the user of the TLD MCS code. This included demonstrating MEL and PIR maintenance, opportunistic maintenance, varying the operational lifetime of the system, viewing individual simulation output and calculating importance measures for system components and TLD faults. The TLD MCS code was seen to provide a powerful tool for the analysis of TLD.

Chapter 10 : Large Scale System TLD Modelling

10.1 Introduction

In the previous chapters the examples covered looked at modelling TLD for simple systems. These examples served to demonstrate the flexibility of the TLD MCS code and its ability to model the application of TLD to those systems. Comparisons with a conventional Markov model solved using numerical integration procedures and the models recommended in SAE ARP 5107 [17], [18] showed greatest agreement between the results obtained using the TLD MCS code and the conventional Markov model. This suggests a greater degree of accuracy in these two models and, it is argued, particularly in the MCS model, since this is the model which can most closely represent the application of maintenance at the correct times within the operational lifetime of the system. The conventional Markov results are close to the MCS results since the models include all system states and because the systems are small any inaccuracies in the results caused by approximations of repair rates or repair times within the code are also small. The SAE models appear to produce the least accurate results for the examples considered, in part due to the approximations required in calculating failure rates for use in the models and in part due to the actual structures of the models.

In the second example that was considered in the previous chapter the system that was modelled bore some features that might be expected of real aircraft systems. The faults that were included in the dispatch criteria did not correspond to basic events in the fault tree representation of the LOTC system failure mode. Instead, the faults included in the dispatch criteria were represented by intermediate events in the fault tree. A structural dependency was also introduced between TLD faults by including a single component that performed a certain task in two separate functions. In this chapter a larger scale system is considered. The fault tree text file for the system is shown in Appendix 5. Also given in the appendix are the other text files provided to the TLD MCS code for this system, including that containing the failure rates relating to the basic events of the fault tree. Due to issues of commercial confidentiality the fault tree of a real aircraft system cannot be used. However, the fault tree contains features that are adapted from

that of a real aircraft system but no part of the fault tree used here is identical in any way to that of the real aircraft system. The basic event failure rates and other features are intended to be representative of the kinds of features that might occur when modelling a real aircraft system. As a result of the changes that are made in adapting the fault tree for use here the results produced are illustrative of those that would be expected or indeed obtained when modelling a real system. The results obtained in this chapter will demonstrate that the TLD MCS code is capable of producing results for a large scale system.

The system to be modelled has a number of structural dependencies that exist between the various functions that are performed by the system. These functions are intermediate events in the fault tree and must be included in the dispatch criteria. The dispatch criteria for the system are not known and are set automatically for the system using the TLD MCS code. These dispatch criteria are then used when the application of TLD to the system is analysed further.

10.2 The System to be Modelled

The text file for the fault tree of the system to be modelled is given in Appendix 5. This is the text file provided to the TLD MCS code and the fault tree has 142 gates and 81 basic events. The input file containing the basic event failure distributions is also given in the appendix. The system for which the fault tree is drawn contains two channels that each perform 4 functions. Various combinations of the failures of these functions can cause the top event, LOTC, to occur. Each channel of the system is powered by a primary power source and in the case of failure of this power source a secondary power supply can be called upon to power the channel. The fault tree also contains a number of basic events that correspond to HMU faults. In addition to this there are a number of dependencies which exist between functions and channels if certain basic events occur. If a certain function is known to have failed on a particular channel then that function is used from the other channel, if available, or a modelled function output can be used.

The four functions on each channel can each fail in two ways – either invalid or erroneous. If the function fails invalid then it is known to have failed and these are the faults that will be included in the dispatch criteria. If the function fails erroneous then,

for the purpose of this example, it is assumed that the failure is effectively unrevealed and, as such, no action is taken as a direct result of a function failing erroneously. For example, consider the function FN1 performed by channel A. The intermediate event representing it having failed invalid is G:FN1_INV_A and the intermediate event representing it having failed erroneous is G:FN1_ERR_A. As G:FN1_INV_A occurs the dispatch criteria will be checked to see what TLD restrictions should be applied. However, as G:FN1_ERR_A occurs the dispatch criteria will not be checked. When the faults are repaired all faults relating to the relevant function will be repaired. Thus, any faults relating to function 1 on channel A, including function 1 erroneous on channel A, would be repaired as the fault relating to function 1 failing invalid on channel A was repaired. The system will not be described here in any more detail than this. However, this is not considered to be detrimental to the analysis because it is not important for the task that is being conducted here, namely using the TLD MCS code to model a larger scale system containing features that could be encountered in a real system.

The faults included in the TLD criteria are shown in Table 10.2.1. It is assumed that if the power supply to either channel is lost that the dispatch criteria applied will be DND, i.e. the loss of power to either channel requires that repairs must be carried out to the system as soon as the current flight ends. This takes advantage of one of the options available to the user when using the TLD MCS code to automatically set the dispatch criteria, namely that an initial set of required dispatch criteria can be supplied. Thus the initial TLD criteria supplied to the TLD MCS code are as given in Table 10.2.2.

G:PWR:A:LOSS	G:PWR:B:LOSS
G:EEC:A:LOSS	G:EEC:B:LOSS
G:FN1_INV_A	G:FN1_INV_B
G:FN2_1_INV_A	G:FN2_1_INV_B
G:FN2_2_INV_A	G:FN2_2_INV_B
G:FN3_INV_A	G:FN3_INV_B
G:FN4_INV_A	G:FN4_INV_B

Table 10.2.1. The faults to be included in the dispatch criteria.

G:PWR:A:LOSS	DND
G:PWR:B:LOSS	DND

Table 10.2.2. The initial TLD criteria supplied to the TLD MCS code.

As mentioned earlier, there are certain faults within the fault tree that will be repaired along with the TLD faults at maintenance. The repair of these 'associated' faults was described in Section 6.7.6 and in Appendix 2, Section A2.2.11. Table 10.2.3 shows the dispatchable TLD faults for the system and their associated faults that would be repaired along with them at maintenance. Repairing any of these intermediate events in the fault tree at maintenance involves restoring all gates and basic events below the intermediate events, and the intermediate events themselves, to their working status. In addition to these 'associated repairs' at maintenance there are other situations where extra repairs will be carried out as certain faults are repaired. This is in the case of the occurrence of the EEC and power loss faults that are included in the dispatch criteria. If any of these four faults are repaired then all of the faults that are present in the same channel as the fault being repaired are also cleared from the system. For example, if the fault G:PWR:A:LOSS is repaired then so are all faults that are present in channel A.

TLD Fault	Associated Faults	TLD Fault	Associated Faults
G:FN1_INV_A	G:FN1_ERR_A E:FN1:CM_FAULT_1 E:FN1:A:FAULT G:COMMS_CORRUPT_A	G:FN1_INV_B	G:FN1_ERR_B E:FN1:CM_FAULT_1 E:FN1:B:FAULT G:COMMS_CORRUPT_B
G:FN2_1_INV_A	G:FN2_1_ERR_A G:FN2_2_INV_A G:FN2_2_ERR_A E:FN2:CM_FAULT_1 G:COMMS_CORRUPT_A	G:FN2_1_INV_B	G:FN2_1_ERR_B G:FN2_2_INV_B G:FN2_2_ERR_B E:FN2:CM_FAULT_1 G:COMMS_CORRUPT_B
G:FN2_2_INV_A	G:FN2_2_ERR_A G:FN2_1_INV_A G:FN2_1_ERR_A E:FN2:CM_FAULT_1 G:COMMS_CORRUPT_A	G:FN2_2_INV_B	G:FN2_2_ERR_B G:FN2_1_INV_B G:FN2_1_ERR_B E:FN2:CM_FAULT_1 G:COMMS_CORRUPT_B
G:FN3_INV_A	G:FN3_ERR_A E:FN3:CM_FAULT_1 G:COMMS_CORRUPT_A	G:FN3_INV_B	G:FN3_ERR_B E:FN3:CM_FAULT_1 G:COMMS_CORRUPT_B
G:FN4_INV_A	G:FN4_ERR_A E:FN4:CM_FAULT_1 G:COMMS_CORRUPT_A	G:FN4_INV_B	G:FN4_ERR_B E:FN4:CM_FAULT_1 G:COMMS_CORRUPT_B

Table 10.2.3. Selected TLD faults and the faults associated with them at maintenance.

When modelling the system using the TLD MCS code the operational lifetime of the system was assumed to be 130 000 hours and the average length of a flight was

assumed to be 10 hours. The dispatch criteria are initially to be set automatically for the system when MEL maintenance is used to address both STD and LTD faults. The STD interval is to be varied from 500 to 2500 hours in steps of 500 hours and the LTD interval is to be varied from 2500 to 5000 hours in steps of 500 hours. The first check for convergence is made after 2 500 000 simulations and subsequent checks are made every 100 000 simulations. Convergence is required to 3 significant figures and 3 consecutive values of LOTC rates are required to match for convergence to be assumed. Note that in some cases, for the results presented, the system LOTC rate was seen to rise above 10 failures per 10^6 flight hours and these results were recalculated to 4 significant figures in order to examine the results in greater detail. Before any convergence checks are made for the LOTC rate a LOTC event must have occurred from the relevant dispatchable configuration a minimum of 1 000 times. The dispatch criteria applied to the system is DND if more than 3 faults are present in the system at any time and the dispatch criteria are set for fault combinations up to order 3, i.e. for single, dual and triple dispatchable fault combinations.

The four repair strategies that are available in the TLD MCS code are applied to the system. Note that the limit for the instantaneous failure rates to LOTC above which the DND category will be applied to faults is lowered (from 100) to 34 failures per 10^6 flight hours for this example and the limit above which the STD category will be applied is lowered (from 75) to 17 failures per 10^6 flight hours. These changes were made because of the range of values taken by the instantaneous LOTC rates (when calculated) for the various dispatchable fault states of the system. Due to the complexity of the system it proved difficult to construct an example that produced instantaneous LOTC rates that led to the utilisation of all dispatch categories for the system. The fact that these limits have changed does not matter for this example, since the model is only being used to demonstrate that the TLD MCS code is capable of dealing with more complex systems with characteristics that might be encountered in real aircraft systems. The values of these limits do not affect how the TLD MCS code will be used.

10.3 The Average System LOTC Rate

The system was modelled according to the assumptions and initial conditions outlined above. For repair strategy 4, when all faults are cleared from the system at maintenance,

the average system LOTC rates obtained for the system when the dispatch criteria are automatically set using the TLD MCS code are shown in Table 10.3.1. The system LOTC rates behave in an expected manner in that the rate rises as either the STD or LTD interval is increased. The rates produced by the system do not rise to the level of 10 failures per 10^6 flight hours. Thus, if this system were to be put through the certification process then any of the modelled values of T_{STD} and T_{LTD} could be used.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	7.42	7.56	7.70	7.82	7.94	8.07
	1000	7.49	7.63	7.76	7.88	8.01	8.13
	1500	7.54	7.68	7.81	7.95	8.07	8.20
	2000	7.59	7.73	7.87	7.99	8.13	8.25
	2500	7.65	7.78	7.91	8.05	8.18	8.31

Table 10.3.1. The average system LOTC rates obtained when automatically setting dispatch criteria using the TLD MCS code for repair strategy 4 (no. of failures per 10^6 flt. hrs.).

Given in Table 10.3.2 are the average system LOTC rates obtained for the system when repair strategy 1, where the last fault of the group causing the maintenance deadline to be applied, is used at maintenance and the dispatch criteria are set automatically using the TLD MCS code. Note that the LOTC rates obtained using this strategy are above 10 failures per 10^6 flight hours and because of this the rates displayed were calculated to 4 significant figures. This allows the small increases in the LOTC rates observed as the dispatch intervals increase to be seen in detail, as was the case with the results for repair strategy 4. In effect all results are calculated to the same number of decimal places. We can see again that the system LOTC rates rise as the STD and LTD intervals increase. In general the system LOTC rate is about 3.5 failures per 10^6 flight hours higher for repair strategy 1 than for repair strategy 4.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	10.94	11.00	11.06	11.13	11.18	11.23
	1000	11.04	11.10	11.16	11.22	11.28	11.32
	1500	11.11	11.18	11.25	11.31	11.36	11.41
	2000	11.20	11.27	11.34	11.40	11.45	11.49
	2500	11.29	11.35	11.42	11.47	11.53	11.57

Table 10.3.2. The average system LOTC rates obtained when automatically setting dispatch criteria using the TLD MCS code for repair strategy 1 (no. of failures per 10^6 flt. hrs.).

Table 10.3.3 shows the results obtained for the system when repair strategy 2, where all of the fault combination causing the dispatch deadline are repaired at maintenance, is used in the TLD MCS code and the dispatch criteria are set automatically. Again, the results shown are obtained to 4 significant figures. Note that the general trend is for the modelled system LOTC rate to be slightly lower than that obtained using repair strategy 1 but that, in general, there is little to choose between the results obtained. In some cases the system LOTC rate obtained is identical to that obtained using repair strategy 1. In reality there could be a slight difference but this might only be seen if the LOTC rate is calculated to a greater accuracy. However, there is also one value of system LOTC rate, obtained for a STD interval of 2500 hours and a LTD interval of 2500 hours, which is greater, although the difference observed is only 0.01 failures per 10^6 flight hours. This result could be explained by the fact that the true value for the LOTC rates are close for the two repair strategies and the random nature of the MCS has led to results that are slightly above or below the true value for each strategy.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	10.93	10.98	11.03	11.09	11.16	11.19
	1000	11.03	11.09	11.14	11.19	11.25	11.29
	1500	11.11	11.19	11.24	11.30	11.34	11.37
	2000	11.20	11.26	11.33	11.38	11.43	11.48
	2500	11.30	11.35	11.42	11.47	11.52	11.55

Table 10.3.3. The average system LOTC rates obtained when automatically setting dispatch criteria using the TLD MCS code for repair strategy 2 (no. of failures per 10^6 flt. hrs.).

The LOTC rates obtained for the remaining repair strategy, strategy 3, where all faults from the TLD category of that causing the maintenance deadline are repaired at maintenance, are shown in Table 10.3.4 and are obtained after the dispatch criteria have been set automatically using the TLD MCS code. Again, as would be expected, the general trend is for the computed system LOTC rate to be slightly lower than that obtained using repair strategy 2, which was in turn slightly lower than that computed using repair strategy 1. However, there is clearly not a great deal of difference between these three sets of results.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	10.90	10.95	11.01	11.06	11.10	11.12
	1000	10.99	11.05	11.10	11.15	11.20	11.23
	1500	11.08	11.14	11.20	11.25	11.29	11.33
	2000	11.16	11.23	11.28	11.33	11.37	11.41
	2500	11.25	11.31	11.37	11.41	11.46	11.49

Table 10.3.4. The average system LOTC rates obtained when automatically setting dispatch criteria using the TLD MCS code for repair strategy 3 (no. of failures per 10^6 flt. hrs.).

Figure 10.3.1 shows a comparison of some of the results presented in Table 10.3.1, Table 10.3.2, Table 10.3.3 and Table 10.3.4, namely the ones obtained for a STD interval of 1000 hours. This serves to illustrate graphically the similarities in the results obtained using the repair strategies 1, 2 and 3 and the differences between those results and the results obtained using repair strategy 4. The differences observed when modelling the system for repair strategies 1, 2 and 3 are not very clear. Figure 10.3.2 shows the same results with the results for strategy 4 omitted and this shows more clearly the subtle differences in the results obtained when these three repair strategies at maintenance are simulated.

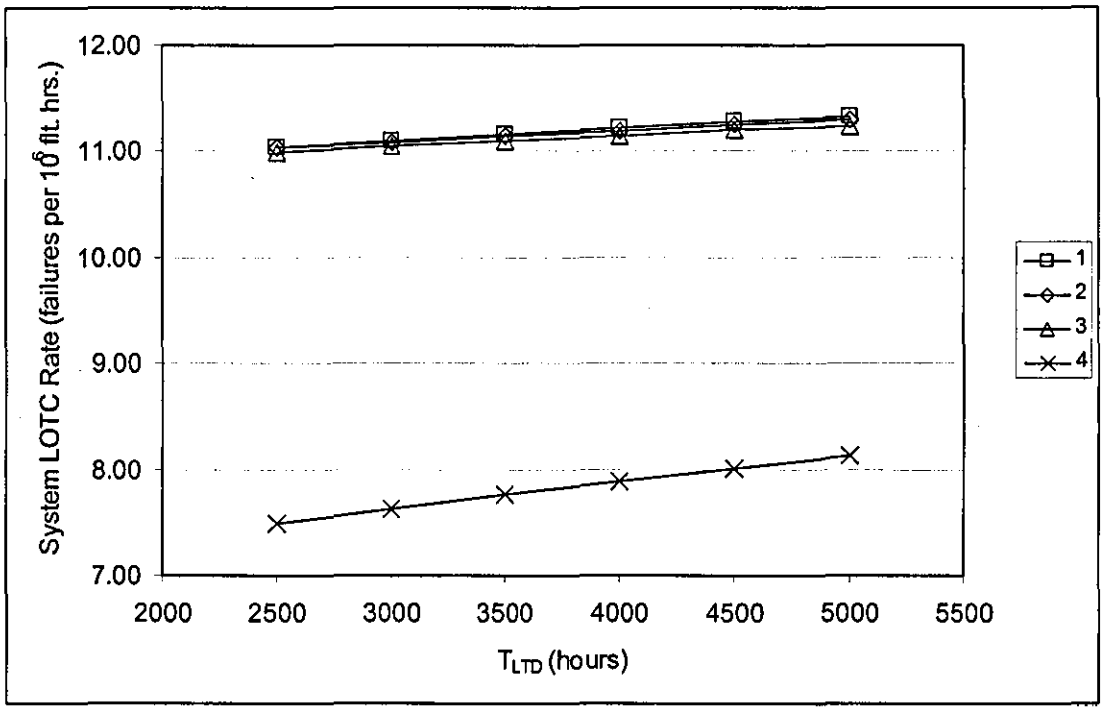


Figure 10.3.1. The results obtained for the average system LOTC rate for repair strategies 1 to 4. STD and LTD faults are maintained using MEL maintenance. T_{STD} = 1000 hrs.

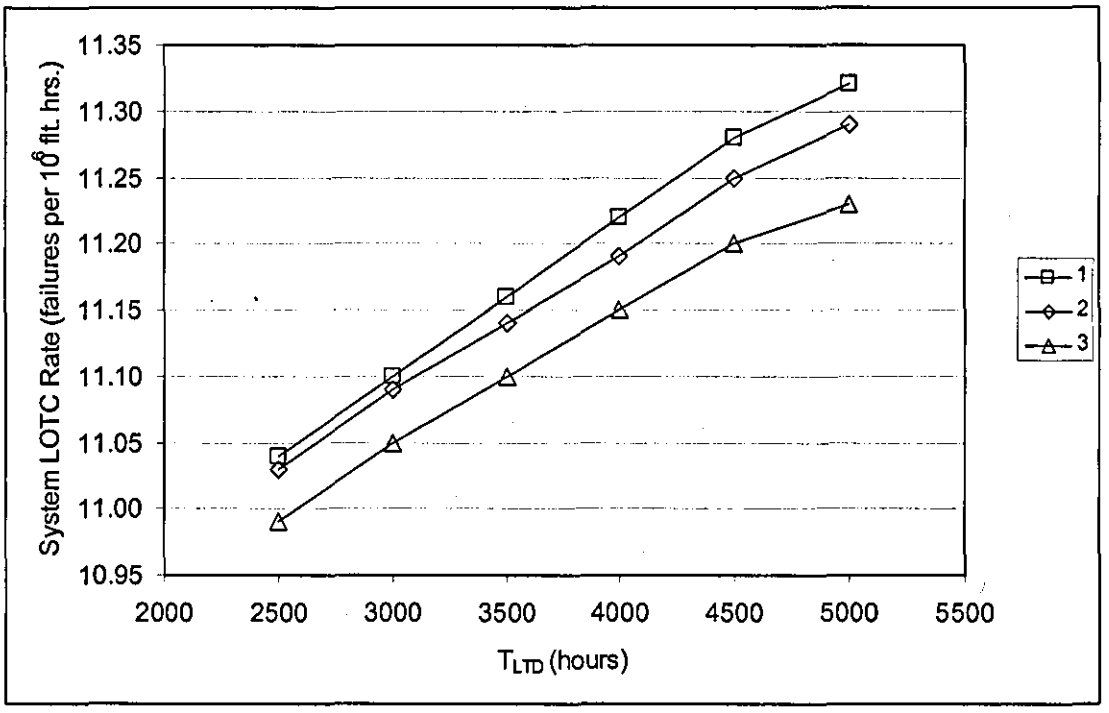


Figure 10.3.2. The results obtained for the average system LOTC rate for repair strategies 1 to 3. STD and LTD faults are maintained using MEL maintenance. T_{STD} = 1000 hrs.

A number of conclusions can be drawn from the results obtained for the system LOTC rates when using the different repair strategies. The first is that repair strategies 1, 2 and 3 give similar results. This means that, for this system, there would not be much difference observed in the LOTC rates obtained when using these three repair strategies. If one considers why this is the case there is a clear reason why this could be so. Since the occurrence of TLD faults are relatively rare, the difference between these three repair strategies at maintenance is relatively small because the majority of times when TLD faults are repaired at maintenance there will be only one TLD fault present in the system. Thus the different repair strategies, which amount to repairing fault combinations in different ways, do not have a great effect on the calculated LOTC rates, although differences do exist. If one now compares repair strategy 4, where all faults are cleared from the system at maintenance, to the other three repair strategies it can be seen that the average system LOTC rates obtained are lower, as would be expected when such a stringent maintenance policy is applied. In this case, the repair strategy does affect the system LOTC rate by a relatively large amount in comparison to the other repair strategies because there are certain faults in the system that are unrevealed. In all repair strategies except repair strategy 4 these unrevealed faults will remain in the system until the relevant TLD faults arise and cause the unrevealed faults to be repaired

along with the TLD faults. This could explain at least a part of the differences observed here, although in reality one might still reasonably expect that repair strategy 4 would still produce a lower average system LOTC rate. In reality the failure rates for the unrevealed faults might take values lower than those considered here and this could account for some of the differences. However, it could also be the case that these unrevealed faults could be repaired periodically. This will be investigated later in Section 10.5.

10.4 Instantaneous Failure Rates to LOTC

The instantaneous failure rates to LOTC calculated for all of the dispatchable faults will not be presented here because of the large number of dispatchable fault states. However, to provide a flavour of the results obtained, the instantaneous failure rates to LOTC from the dispatchable single fault states are presented. These are given in Table 10.4.1, along with the associated dispatch category for the fault. The instantaneous LOTC rates for the dual fault combinations are given in Appendix 6, since these will be used in the application of the dual fault state Markov model for this system later. Recall that the dispatch categories are assigned according to the 17 and 34 failures per 10^6 hours limits for this example. The instantaneous failure rates to LOTC shown are the final group of those which were calculated when the process of automatically setting the dispatch criteria using the TLD MCS code was implemented. This means that these are the instantaneous failure rates to LOTC obtained for STD and LTD intervals of 2500 and 5000 hours respectively. These rates will provide upper bounds for lower dispatch intervals. Indeed, the instantaneous LOTC rates could be calculated for lower dispatch intervals if required. In investigations the rates observed did not show a great deal of variation and so this avenue is not explored here. Also shown in this table are the computed values of the full-up failure rate to LOTC for the system. Note that, the way that the full-up failure rate is defined means that the failure rate shown is computed when there are no TLD faults present at dispatch. It can be seen that, as was the case for the average system LOTC rates obtained using the different methods, the instantaneous failure rates to LOTC obtained for repair strategies 1 to 3 are roughly similar and the corresponding instantaneous LOTC rates for strategy 4 are approximately 3 failures per 10^6 flight hours lower than for the other three strategies.

Fault	rep 1	rep 2	rep 3	rep 4
FULLUP	9.84	9.83	9.83	6.70
G:EEC:A:LOSS	22.90	22.90	22.90	17.30
G:EEC:B:LOSS	22.80	22.90	22.90	17.40
G:FN3_INV_A	14.60	14.80	14.90	12.10
G:FN3_INV_B	14.70	14.70	15.00	12.10
G:FN4_INV_A	12.80	12.90	13.10	9.64
G:FN4_INV_B	12.80	12.80	12.20	9.65
G:FN2_1_INV_A	12.50	12.60	12.80	9.32
G:FN2_1_INV_B	12.50	12.50	12.80	9.31
G:FN2_2_INV_A	12.70	12.70	12.90	9.38
G:FN2_2_INV_B	12.60	12.70	12.90	9.39
G:FN1_INV_A	15.30	15.10	15.20	12.40
G:FN1_INV_B	15.10	15.20	15.30	12.40

Key:	LTD	STD

Table 10.4.1. The instantaneous failure rates to LOTC for the dispatchable system faults and the associated TLD categories for repair strategies 1 to 4.

Note that the instantaneous failure rates to LOTC and hence the dispatch categories applied to the dual and triple faults also exhibit similar behaviour, with the LOTC rates obtained when repair strategies 1 to 3 are used being very similar. The rates obtained for strategy 4 are similarly slightly lower, as has been observed for the single fault states. Indeed this results in a small number of the dispatch criteria that are applied for the dual and triple fault combinations being more restrictive for repair strategies 1 to 3 than they are for the same combinations in repair strategy 4. This result could be reasonably expected due to the nature of the repair strategies at maintenance and the more complete nature of repairs when strategy 4 is implemented.

10.5 Varying the Operational Lifetime of the System

Table 10.5.1 shows the LOTC rates obtained when repair strategy 1 and repair strategy 4 are modelled for varying system operational lifetimes. The STD and LTD intervals are set at 500 and 2500 hours respectively. The dispatch criteria used in each case are the ones that were set for the relevant repair strategy using the TLD MCS code. Note how for each repair strategy, as one might expect, the computed LOTC rate is seen to rise as the operational lifetime of the system is increased. For each strategy the rate of change of the LOTC rate with respect to the operational lifetime of the system is seen to be steepest for the lower values of the operational lifetime modelled.

A feature of the results obtained is that for the lower values of the operational lifetime the system LOTC rate for the different repair strategies is seen to be most closely matched. As the operational lifetime increases there is little to choose between repair strategies 1 to 3 but repair strategy 4 diverges from these three sets of results. This is depicted in Figure 10.5.1, which shows the results obtained for repair strategy 1 and repair strategy 4. The explanation for this observed difference between the system LOTC rates obtained is the fact that a number of faults can occur within the system and essentially be unrevealed (the 'erroneous' faults mentioned earlier). For repair strategy 1, the greater the operational lifetime of the system, the greater the chance that a higher number of these faults will be present within the system and contribute to the failure of the system. In the case of repair strategy 4 any of these unrevealed faults that are present will be cleared at every maintenance deadline. This means that these faults effectively have fewer opportunities to contribute to system failure.

Operational Lifetime	1	2	3	4
5000	6.01	5.98	5.96	5.92
10000	6.83	6.81	6.78	6.44
15000	7.48	7.46	7.42	6.71
20000	8.01	8.00	8.00	6.91
25000	8.50	8.47	8.44	7.02
30000	8.85	8.84	8.81	7.11
35000	9.18	9.14	9.12	7.16
40000	9.45	9.42	9.38	7.21
45000	9.67	9.65	9.60	7.24
50000	9.84	9.82	9.79	7.27
55000	9.99	9.99	9.96	7.29
60000	10.13	10.12	10.08	7.31
65000	10.25	10.23	10.19	7.31
70000	10.34	10.32	10.30	7.34
75000	10.42	10.42	10.38	7.35
80000	10.50	10.49	10.47	7.36
85000	10.56	10.56	10.54	7.37
90000	10.63	10.61	10.59	7.37
95000	10.68	10.67	10.65	7.38
100000	10.72	10.72	10.69	7.40

Table 10.5.1. LOTC rates (failures per 10^6 flt. hrs.) obtained for each of the different repair strategies using the TLD MCS code for different system operational lifetimes.

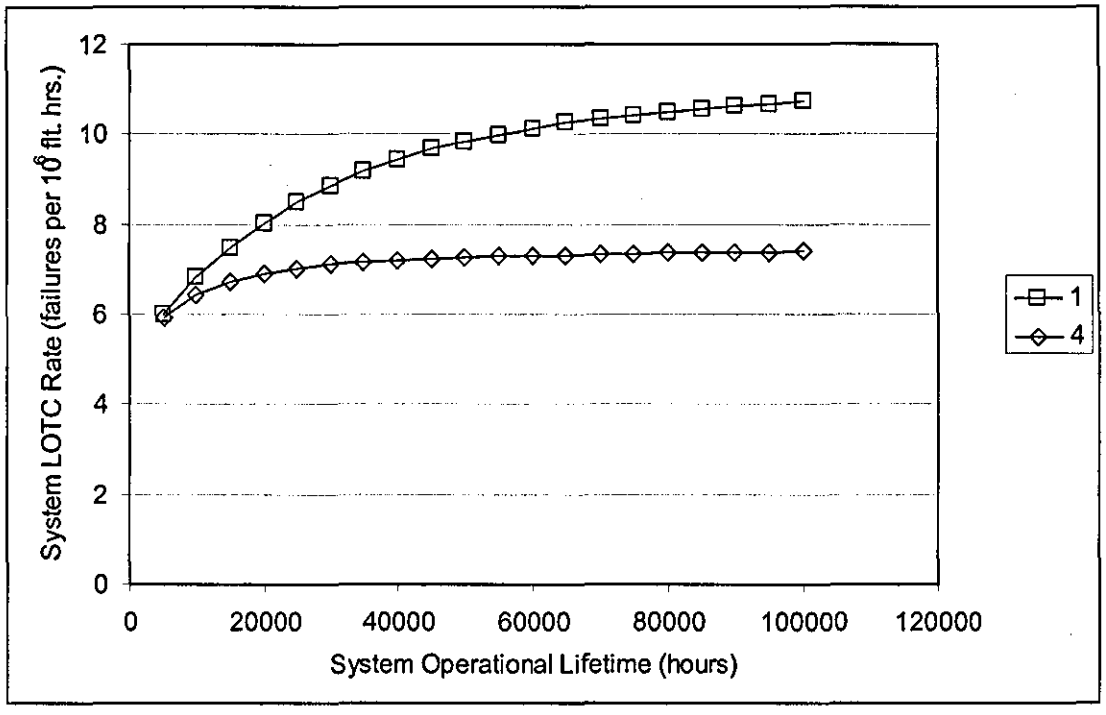


Figure 10.5.1. LOTC rates obtained for repair strategies 1 and 4 using the TLD MCS code for different system operational lifetimes.

The results presented above suggest that repair strategies 1, 2 and 3 could be improved for this system. It will clearly be of no benefit to allow the 'erroneous' faults to remain in the system for long periods of time. If faults such as these were observed in a real system it would likely be the case that they would lead to the system being serviced at certain points in its lifetime and returned to an 'as new' state, in a similar way to how repairs are conducted for repair strategy 4. This would in effect be a periodic major service of the system that would occur at intervals throughout its lifetime. This kind of periodic major service can be modelled here by reducing the operational lifetime of the system to the length of the interval between the services. This is because the system will effectively have a number of 'shorter lifetimes' that will occur throughout its entire operational lifetime. These shorter lifetimes will exhibit the same properties as the full operational lifetime in that they will involve the system initially containing no faults and being used for a certain length of time, with TLD being applied during that time.

Thus, for repair strategies 1, 2 and 3, the system could be modelled with a shorter operational lifetime, representing the fact that a major service will be conducted on a regular basis. This might represent more closely the type of repair strategy that would

be applied in reality and might give results that are more favourably comparable to those obtained for the system when repair strategy 4 is used at maintenance. This technique is investigated in the following section.

10.5.1 Simulating Major Maintenance Using Reduced Operational Lifetimes

When repair strategy 4 is used all faults are cleared from the system at maintenance deadlines. This is not the case when the other three repair strategies are used. To cover for the fact that unrevealed faults, which will not be repaired as TLD faults are addressed, could be present in the system it could undergo major maintenance periodically. At these times all faults will be cleared from the system and it will be restored to a fully working state. As discussed at the end of the previous section, this could be modelled by simulating a reduced system operational lifetime equivalent to the interval between the major maintenance operations. This process was carried out for repair strategy 1. Note that repair strategies 2 and 3 were not considered here because of the similarities of the results obtained so far with those obtained when repair strategy 1 is implemented. Repair strategy 4 is not considered since this is returned to a fully working state after every TLD maintenance operation anyway and if this strategy was applied it is unlikely that a regular major maintenance operation would be carried out. The system operational lifetime was set to 10 000 hours, representing the fact that in the lifetime of the system (which will still be 130 000 hours) it will undergo major maintenance once every 10 000 hours. As this maintenance is implemented the system will be returned to a fully working state. The results obtained for the average system LOTC rates obtained are shown in Table 10.5.2. Figure 10.5.2 shows a plot of the results obtained for a STD interval of 1000 hours compared with the results that were obtained for repair strategy 4.

Earlier we saw that the average system LOTC rates obtained for repair strategy 1 were higher than those obtained for repair strategy 4. However, as can be seen from Table 10.5.2 and as is seen for a STD interval of 1000 hours in Figure 10.5.2, the average system LOTC rate obtained when repair strategy 1 is implemented and the system undergoes a major maintenance operation every 10 000 hours is now lower. This shows that repair strategy 1 could be used to maintain the system, with the minimum amount of repairs possible being carried out, and that, as long as the system was cleared of all

faults regularly, the LOTC rate seen could be lower even than for repair strategy 4, where all faults are cleared from the system. The same would also be true for repair strategies 2 and 3. This would, in reality, perhaps be a more likely repair strategy to be applied than repair strategy 4 because it could be considered less likely to cause delays. This is because repair strategy 4 would always involve clearing all faults from the system at maintenance but this could bring with it a cost in terms of the time taken to repair all of the faults. If a strategy is adopted where just the final fault of the group causing the TLD deadline is repaired then delays could be less likely. If it is also known that the major repairs are to be carried out at regular intervals time could be allowed for these to take place that would ensure that delays are less likely to occur. Thus repair strategy 1 is not necessarily as bad a maintenance strategy as has been suggested thus far if it is married with a suitable periodic clearance of all faults in the system. This is a result of potentially great importance since this fact could also be used by the system manufacturer, if necessary, to gain certification for a system that might otherwise be considered to be uncertifiable by specifying a maintenance interval wherein all faults could be cleared from a system periodically.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	6.83	6.91	6.97	7.06	7.12	7.18
	1000	6.89	6.99	7.05	7.14	7.18	7.24
	1500	6.96	7.05	7.13	7.20	7.26	7.32
	2000	7.01	7.10	7.17	7.24	7.32	7.37
	2500	7.07	7.14	7.24	7.33	7.38	7.45

Table 10.5.2. Average system LOTC rates for an operational lifetime of 10 000 hours when repair strategy 1 is used at maintenance.

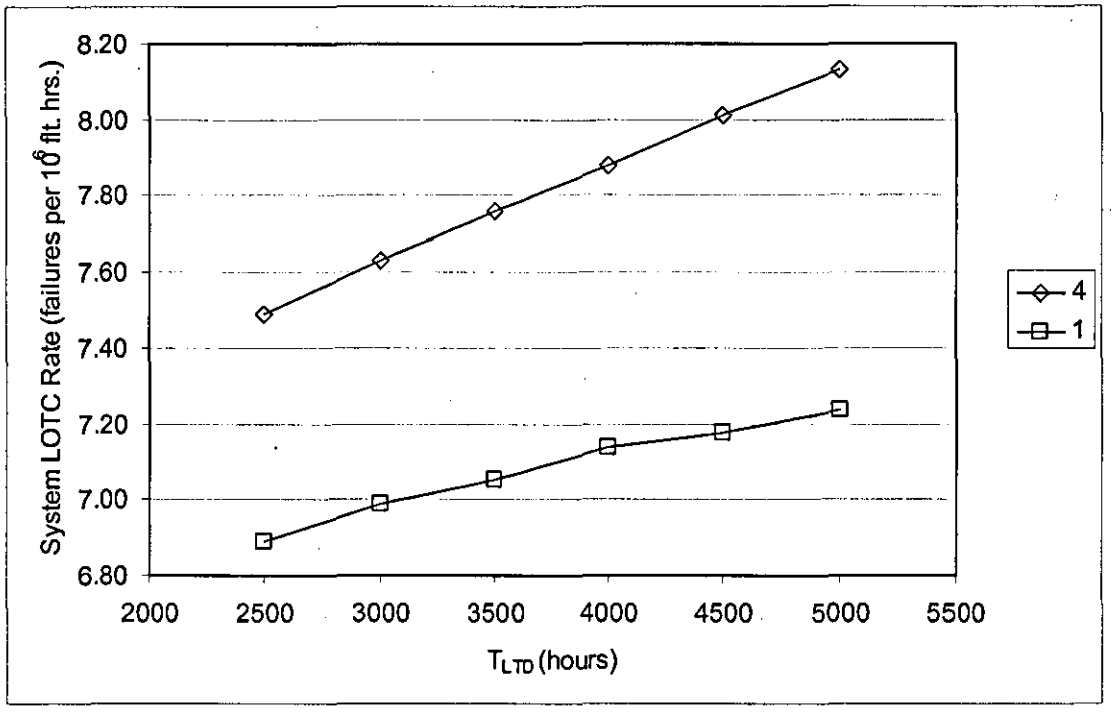


Figure 10.5.2. Plot of the average system LOTC rates for repair strategy 1 (with an operational lifetime of 10 000 hours) and repair strategy 4. T_{STD} = 1000 hours.

10.6 Importance Measures for the System

Consider again the average system LOTC rates obtained originally for a system operational lifetime of 130 000 hours when repair strategies 1 to 3 are applied. These results were presented in Section 10.3. Here it was deduced that the unrevealed system faults (caused by the erroneous failures) were causing an increase in the LOTC rates. Rather than introducing major maintenance overhauls into the service strategy for the system it might be preferable to resolve this issue by replacing certain components with more reliable alternatives. However, before implementing this task it would be important to know where efforts should be aimed in order to achieve the best possible effect. Importance measures can help when attempting to do this.

Shown in Table 10.6.1 and Table 10.6.2 are the importance measures calculated using the TLD MCS code for the single dispatchable TLD faults and the basic events of the system fault tree. The importance measures for the dispatchable TLD dual faults are given in Appendix 6. These importance measures were calculated for the system when repair strategy 1 was applied at maintenance. MEL maintenance was used to address both STD and LTD faults and the dispatch criteria applied to the system were those that

were obtained when dispatch criteria were automatically set using the TLD MCS code. The STD and LTD intervals were set at 2500 and 5000 hours respectively.

The importance measures presented show that, of the single TLD faults the faults G:EEC:A:LOSS and G:EEC:B:LOSS are the most important. There are a number of basic events whose importance measures are greater than 0.1 and these are perhaps where the first improvements should be aimed at the system. The importance measures give a good ranking of the importance of the basic events and TLD faults.

TLD fault	Importance Measure
G:EEC:A:LOSS	3.10%
G:EEC:B:LOSS	3.10%
G:FN3_INV_A	7.91%
G:FN3_INV_B	7.93%
G:FN4_INV_A	7.69%
G:FN4_INV_B	7.69%
G:FN2_1_INV_A	7.32%
G:FN2_1_INV_B	7.33%
G:FN2_2_INV_A	7.79%
G:FN2_2_INV_B	7.78%
G:FN1_INV_A	7.21%
G:FN1_INV_B	7.40%

Table 10.6.1. Importance measures calculated for the dispatchable TLD single faults using the TLD MCS code.

Basic event	Importance Measure	Basic event	Importance Measure
E:PWR1:A:FAULT_1	17.5%	E:FN4:B:FAULT_1	3.21%
E:PWR1:CM_FAULT_1	1.94%	E:FN4:B:FAULT_3	3.83%
E:PWR1:A:FAULT_2	4.23%	E:FN4_FN2:A:CM_FAULT_2	0.355%
E:PWR2:A:FAULT_1	17.7%	E:FN4:A:FAULT_2	4.02%
E:PWR2:CM_FAULT_1	1.94%	E:FN4:A:FAULT_4	3.95%
E:PWR2:A:FAULT_2	4.27%	E:FN4_FN2:B:CM_FAULT_2	0.346%
E:PWR1:B:FAULT_1	16.8%	E:FN4:B:FAULT_2	3.91%
E:PWR1:B:FAULT_2	4.24%	E:FN4:B:FAULT_4	3.93%
E:PWR2:B:FAULT_1	17.8%	E:FN4:CM_FAULT_1	3.00%
E:PWR2:B:FAULT_2	4.32%	E:FN2_1:CM_FAULT_1	2.54%
E:EEC:A:FAULT_3	2.47%	E:FN2_1:A:FAULT_1	2.37%
E:EEC:B:FAULT_3	2.62%	E:FN2_1:A:FAULT_3	3.06%
E:EEC:A:FAULT_2	2.14%	E:FN2_1:B:FAULT_1	2.54%
E:EEC:B:FAULT_2	2.11%	E:FN2_1:B:FAULT_3	3.29%
E:EEC:A:FAULT_1	2.26%	E:FN2_1:CM_FAULT_2	2.27%
E:EEC:B:FAULT_1	2.30%	E:FN2_1:A:FAULT_2	2.40%
E:FN1:A:FAULT	15.4%	E:FN2_1:A:FAULT_4	3.81%
E:FN1:B:FAULT	15.4%	E:FN2_1:B:FAULT_2	2.39%
E:HMU:FAULT_1	8.63%	E:FN2_1:B:FAULT_4	3.81%
E:HMU:FAULT_2	8.63%	E:FN2_2:CM_FAULT_1	2.53%
E:HMU:FAULT_3	8.63%	E:FN2_2:A:FAULT_1	2.37%
E:HMU:FAULT_4	8.63%	E:FN2_2:A:FAULT_3	4.10%
E:FN3:A:FAULT_3	8.50%	E:FN2_2:B:FAULT_1	2.56%
E:FN3_FN4:A:CM_FAULT_1	1.58%	E:FN2_2:B:FAULT_3	4.40%
E:FN3:A:FAULT_1	4.27%	E:FN2_2:CM_FAULT_2	2.35%
E:FN3_FN4:CM_FAULT_1	5.18%	E:FN2_2:A:FAULT_2	2.44%
E:FN3_FN4:B:CM_FAULT_1	1.60%	E:FN2_2:A:FAULT_4	4.37%
E:FN3:B:FAULT_1	4.30%	E:FN2_2:B:FAULT_2	2.41%
E:FN3:B:FAULT_3	8.39%	E:FN2_2:B:FAULT_4	4.36%
E:FN3_FN4:CM_FAULT_2	8.19%	E:FN2:CM_FAULT_1	2.22%
E:FN3_FN4:A:CM_FAULT_2	1.63%	E:FN1:CM_FAULT_1	3.60%
E:FN3:A:FAULT_2	6.47%	E:FN1:A:FAULT_1	5.52%
E:FN3:A:FAULT_4	6.46%	E:FN1:A:FAULT_3	6.64%
E:FN3_FN4:B:CM_FAULT_2	1.62%	E:FN1:B:FAULT_1	5.52%
E:FN3:B:FAULT_2	6.52%	E:FN1:B:FAULT_3	6.65%
E:FN3:B:FAULT_4	6.49%	E:FN1:A:FAULT_2	6.44%
E:FN3:CM_FAULT_1	3.56%	E:FN1:A:FAULT_4	6.45%
E:FN4_FN2:A:CM_FAULT_1	1.08%	E:FN1:B:FAULT_2	6.54%
E:FN4:A:FAULT_1	3.13%	E:FN1:B:FAULT_4	6.53%
E:FN4:A:FAULT_3	3.77%	E:COMMS_CORRUPTED	6.16%
E:FN4_FN2:B:CM_FAULT_1	1.14%		

Table 10.6.2. Importance measures calculated for the system fault tree basic events using the TLD MCS code.

10.7 Other Models of the System

So far in this chapter we have concentrated on the results obtained using the TLD MCS code. What follows is a brief discussion of two other methods – the reduced fault state Markov model and the conventional Markov model.

10.7.1 The Conventional Markov Model

The conventional Markov model gave results that best matched those obtained using the TLD MCS code for the eight component system considered in the last chapter. However, for this system a conventional Markov model is a non-starter, because there are 81 basic events in the system fault tree. This would lead to the generation of 2^{81} system states in the code written to build the conventional Markov model. This is considered far too large to be practical and, as such the conventional Markov model cannot be used to model this system.

10.7.2 The Dual Fault State Markov Model

Of the SAE techniques, the dual fault state Markov model is the most appropriate to model this system. Equation 5.5.30 gives the average system LOTC rate for a general dual fault state Markov model. This equation was used to calculate a solution for the system under consideration in this chapter. A number of failure rates must be calculated for use in the equation. The first of these is λ_i, the failure rate of TLD fault i. These are given in Table 10.7.1 and were estimated by calculating the probability of failure for fault i and dividing this by the length of a flight, the same method that was used for the previously modelled systems. The TLD faults G:EEC:A:LOSS and G:EEC:B:LOSS are gates in the fault tree that contain NOT logic. To calculate the probability of failure for these gates the coherent approximation was used, the negated event ignored.

TLD fault	λ_i
G:EEC:A:LOSS	2.35×10-5
G:EEC:B:LOSS	2.35×10-5
G:FN3_INV_A	1.88×10-5
G:FN3_INV_B	1.88×10-5
G:FN4_INV_A	1.98×10-5
G:FN4_INV_B	1.98×10-5
G:FN2_1_INV_A	1.90×10-5
G:FN2_1_INV_B	1.90×10-5
G:FN2_2_INV_A	2.00×10-5
G:FN2_2_INV_B	2.00×10-5
G:FN1_INV_A	1.60×10-5
G:FN1_INV_B	1.60×10-5

Table 10.7.1. The failure rates for each of the TLD faults used in the dual fault state Markov model for the system.

The other failure rates needed in the model were the instantaneous failure rates to LOTC for the dispatchable system faults. For these rates the values calculated using the TLD MCS code for repair strategy 4 were used. The values for the single fault states are shown in Table 10.4.1 and the values for the dual fault states are given in Appendix 6. Also, the system contains four basic events that represent hydromechanical/ mechanical faults (E:HMU:FAULT_1, E:HMU:FAULT_2, E:HMU:FAULT_3 and E:HMU:FAULT_4). These are included in the model as recommended in SAE ARP 5107 [18] by assuming that they can cause LOTC from any of the dispatchable fault states and so the instantaneous failure rate to LOTC for each of the dispatchable fault states is increased by adding the sum of the failure rates for these components (4.0×10^{-6}). The results obtained using this dual fault state Markov model are shown in Table 10.7.2.

		T_{LTD}					
		2500	3000	3500	4000	4500	5000
T_{STD}	500	9.26	9.52	9.76	9.96	10.15	10.32
	1000	9.55	9.80	10.03	10.22	10.40	10.56
	1500	9.82	10.06	10.28	10.47	10.64	10.80
	2000	10.07	10.31	10.52	10.70	10.87	11.02
	2500	10.31	10.55	10.75	10.92	11.08	11.23

Table 10.7.2. Average system LOTC rates obtained using the dual fault state Markov model (failures per 10^6 flight hours).

Figure 10.7.1 shows a comparison of the LOTC rates obtained when the system is modelled using the dual fault state Markov model with the rates obtained when the TLD MCS code is used and repair strategies 1 and 4 are implemented. This figure illustrates

how the dual fault state Markov results lie between the results obtained using the TLD MCS code. Despite the fact that the instantaneous failure rates to LOTC obtained using the MCS code are used in the dual fault state Markov model the average system LOTC rate is overestimated. This suggests that the structure of the dual fault state Markov model does not represent the system sufficiently well enough to give accurate results, echoing what was seen for the small systems in the previous chapter. If approximations such as the failure probability divided by the flight time were used to find the instantaneous failure rates to LOTC for the system then one could envisage a situation where even less faith should be held in the results obtained from the model.

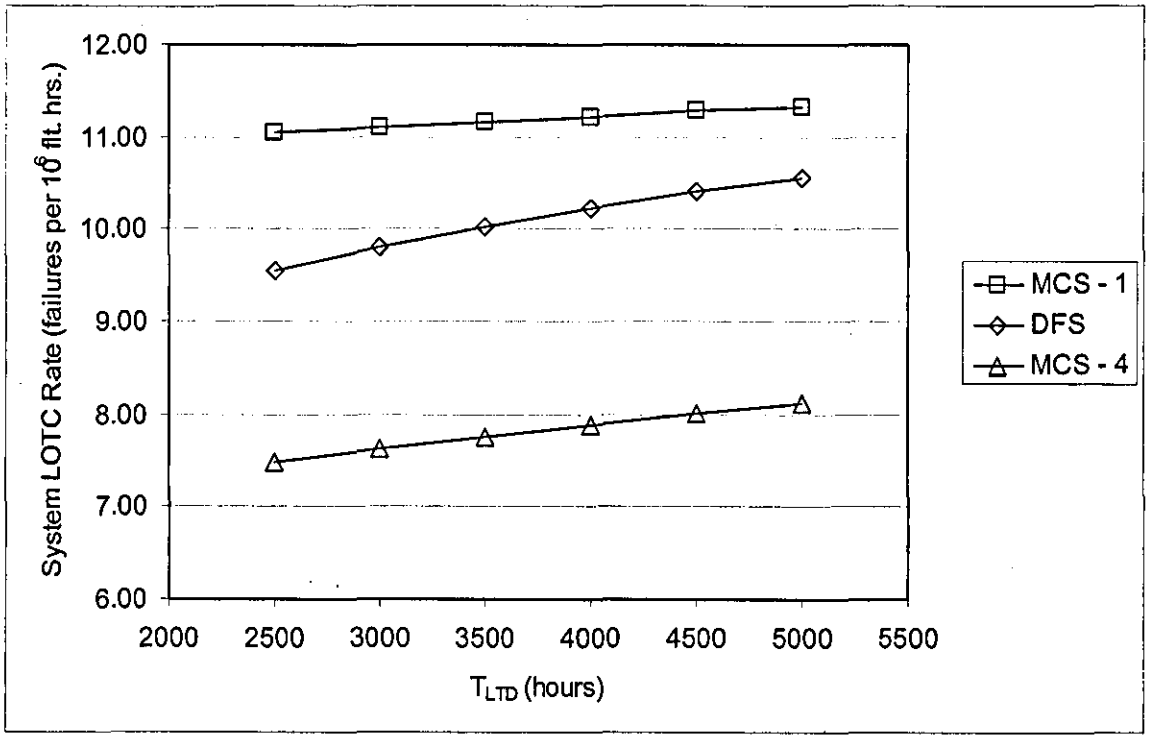


Figure 10.7.1. A comparison of the system LOTC rates obtained for T_{STD} = 1000 hours using the TLD MCS code with repair strategy 1 and 4 and the dual fault state Markov model.

10.8 Summary

The first point made here is in regards to the time taken to perform the simulations of the system considered in this chapter. In order to obtain results for a table of average system LOTC rates as given, for example, in Table 10.3.1, Table 10.3.2, Table 10.3.3 and Table 10.3.4 the simulations that took place generally took of the order of 5 minutes

on a 3.6GHz PC. When instantaneous LOTC rates were also calculated simulations took longer because of the greater amount of data that had to be recorded in order to perform the required calculations. For example, when calculating just the instantaneous failure rates to LOTC for the dual fault combinations the calculations took of the order of 10 minutes and for the triple fault combinations calculations took of the order of 40 minutes. This is not particularly excessive given the size of the system and indicates that, were a real system to be modelled, the time taken to perform the simulations makes the method a practical proposition.

The system considered in this chapter contained a number of features that were representative of those that might be seen on a real system to which TLD is to be applied. The system was modelled using the TLD MCS code and the code was used to automatically set dispatch criteria for the system for all four of the repair strategies that it is possible to include in the TLD MCS code. Repair strategy 4 was seen to offer clear benefits in terms of a lower average system LOTC rate than was obtained for the other three strategies. Importance measures were produced for the system that could be used to identify where effort should be applied in improving system components and hence the system LOTC rate, if required. An argument was also presented wherein repair strategies 1 to 3 could be used to repair faults in the system at maintenance and the LOTC rate could be set to a level that is effectively selected by the user. This would be done by performing a major system maintenance operation at intervals throughout the operational lifetime of the system. All faults would be cleared from the system at these major maintenance operations. The fact that the LOTC rate rises with the operational lifetime of the system could be used to help decide the length of the interval between these major maintenance operations.

A thorough review has been conducted of results obtained for the system using the TLD MCS code and, in particular, how these results can be interpreted. The results obtained using the TLD MCS code in this chapter demonstrate its flexibility and ability to deal with larger systems in reasonable lengths of time. Monte Carlo simulation has been demonstrated as a powerful tool to use for the analysis of TLD.

The other methods considered were a conventional Markov technique and a dual fault state Markov model. The conventional Markov technique could not be applied because

of the huge number of states in the system model and results were obtained for the dual fault state Markov model using the instantaneous failure rates to LOTC calculated using the TLD MCS code. The results did not compare favourably with those of the TLD MCS code. The differences in the results obtained suggested that less confidence could be held in results for which the instantaneous failure rates to LOTC for the dual fault state Markov model were approximated by a failure probability divided by a flight time.

The TLD MCS code offers a far greater choice of TLD scenarios than can be modelled using the dual fault state Markov model. The dual fault state Markov model is inflexible in comparison, modelling only one situation of maintenance and, even then, not modelling that scenario particularly accurately when compared to the TLD MCS code. There is no suggestion that the dual fault state Markov model could offer accurate results. The one saving grace for the dual fault state Markov model here was that it produced results for the system LOTC rate that were overestimates of the values obtained using the TLD MCS code when repair strategy 4 is applied. However, when one considers that the results for the other three repair strategies at maintenance give greater LOTC rates than the dual fault state Markov model then this does not seem such a benefit. Perhaps the main weakness of the dual fault state Markov model is the way in which it is used. Values are inserted into the model and a system LOTC rate is produced. There is no obvious way to see why the system LOTC rate produced takes the value that it does. In comparison the TLD MCS code is far more accountable. The different repair strategies can be evaluated and the system operational lifetime can also be seen to have an effect on the system LOTC rate obtained. This is something which is impossible to study using the dual fault state Markov model. From results obtained for the system modelled here and also those modelled in the previous chapter, the implication is that the TLD MCS code provides a better way to model the application of TLD to a system than the currently recommended methods.

Chapter 11 : TLD System Optimisation

11.1 Introduction

Monte Carlo simulation provides a powerful tool for use in the modelling of TLD. The TLD MCS code will calculate the LOTC rate of a system given a system fault tree and dispatch criteria. The code also allows the calculation of instantaneous failure rates to LOTC from the various dispatchable system configurations so that the dispatch categories applied to these configurations may be verified. In addition to this, a method for automatically setting dispatch criteria for a given system has been demonstrated, in which the dispatch criteria for a given system are assigned in an iterative process. However, it would also be useful at the design stage to be able to investigate how different system structures or designs would affect the application of TLD to systems. This would allow TLD to be integrated into the design process of a system, rather than being something that is checked for compliance with regulations after the system has been designed. Advantages could be gained if a system were optimised with regard to factors that are currently considered in system design, such as cost, weight and reliability and also include the application of TLD.

This chapter contains a methodology and hypothetical example of how the application of TLD could be included in the optimisation of a system at the design stage. A simple genetic algorithm technique is applied to optimise a system, the TLD MCS code being embedded within the algorithm to carry out calculations relating to the application of TLD to the system. Genetic algorithms are briefly introduced and then a simple example is studied.

11.2 Genetic Algorithms

Genetic algorithms provide a method of optimisation formed on a basis of natural selection techniques. In the field of system design a population of string structures is used to represent different system configurations and designs to which a fitness is associated. Initially, this population of strings is chosen at random. Then, in a process

analogous to natural reproduction, the fitter strings are mated to give a new population of strings that will each represent new system configurations, each of which will be evaluated for fitness before the mating procedure begins again. In every population, the fitter strings have a greater chance of reproduction (compare with the Darwinian idea of 'the survival of the fittest'). This process will hopefully lead to the production of fitter offspring to be included in future populations. Genetic algorithms are relatively simple and merely involve string manipulation processes. Genetic algorithms are described in more detail in [23] and a brief summary is given here. A simple genetic algorithm is composed of three simple operations:

1. Reproduction
2. Crossover
3. Mutation

The first of these, reproduction, is, as described above, in some way analogous to a way of allowing the fittest of a population of strings to progress to the next generation. The fitter a string, the more likely it is that the string will progress to the point of mating in order to produce the next generation of strings. Thus, depending on the fitness of the strings, a number of parents will be chosen to take part in mating. The other two operations, crossover and mutation, make up the mating process between the selected fitter strings from the parent population.

Crossover involves taking two parent strings and, at some point in them, selecting a crossing site and swapping all characters between that crossover site and the end of the string. In this way a string produced for the next generation may develop beneficial characteristics that will incorporate good points from each of the parents of that string. Figure 11.2.1 shows an example of the crossover operator as performed on two parent strings, shown in the left of the diagram. The first step of the crossover process is to choose a crossover site at random within the string. In the example depicted the crossover site is marked with a dashed line. The next step is to swap in the two parent strings the bits that occur after the crossing site. This gives two new strings for the next generation. These are shown on the right of the diagram.

Crossing site

1 0 1 1 0 0 1 ¦ 0 1 1	→	1 0 1 1 0 0 1 1 1 0
0 1 0 0 0 1 1 ¦ 1 1 0		0 1 0 0 0 1 1 0 1 1
Before Crossover		After Crossover

Figure 11.2.1. An example of the crossover operation. Parent strings are on the left.

The final operation, mutation, involves randomly changing individual bits within a population of strings. Mutation takes place at a relatively low rate so that useful genetic material (or string information) is generally not lost from one generation to the next. However, it takes place since the small change that does occur might have a beneficial effect on string fitness and, if this is the case, the characteristic would have a higher probability of being passed on to future generations. If the mutation had a bad effect on fitness the string would be less likely to be involved in producing the next population of strings. Figure 11.2.2 shows an example of mutation on a population of four strings. One mutation has taken place and the point of the mutation is chosen randomly within the population. When mutation takes place a 1 is changed to a 0 or vice versa. The number of mutations taking place in any generation is randomly chosen according to some low mutation rate. These processes of crossover and mutation mirror the natural processes involved in producing offspring, where each parent passes a certain amount of genetic material to offspring and a certain amount of mutation within the genetic material will also take place.

1 0 1 1 0 0 1 0 1 1	→	1 0 1 1 0 0 1 0 1 1
0 1 0 0 0 1 1 1 1 0		0 1 0 0 0 1 1 1 1 0
0 1 1 1 1 0 1 0 1 0		0 1 1 1 0 0 1 0 1 0
1 1 1 0 0 0 1 0 1 1		1 1 1 0 0 0 1 0 1 1
Before Mutation		After Mutation

Figure 11.2.2. An example of the mutation operation. The population on the right has undergone one mutation from the initial population on the left.

The genetic algorithms involved in this chapter and the C++ computer code used to implement these algorithms were based on the simple genetic algorithm (SGA) Pascal code given in [23]. The TLD MCS code was embedded within this since it was used in part to measure the string fitness.

11.3 An Example System

In order to illustrate an optimisation process involving TLD the genetic algorithm technique is applied to an example system similar in principle to that given in [22] and Chapter 9. Here a system to which TLD will be applied is constructed along with a number of performance constraints, such as the profit that can be made on such a system and the LOTC rate. An aircraft system is to be designed for sale to a customer, who will be charged for the system initially, generating revenue for the manufacturer, and then the manufacturer, who has designed and supplied the system, will maintain the system throughout its specified lifetime with no further cost incurred by the customer and all costs incurred by the manufacturer. This is similar to a 'power by the hour' deal that some aircraft engine manufacturers are beginning to use. The manufacturer wishes to supply a demonstrably good system to the customer, meeting requirements for TLD, minimising delays and minimising weight whilst also maximising profit.

11.3.1 The System Design

There are a number of options open to the system designers who must decide, for example, on the number of channels to be used in the system, the configuration of the functions within the system and the quality of components to be used. The completed system will consist of two or three channels that are configured as shown in Figure 11.3.1. Each channel of the system performs two functions, F_1 and F_2, which are performed in a series configuration as shown in Figure 11.3.2 for an individual channel N.

Figure 11.3.1. Possible system configurations – 2 channels (l) and 3 channels (r).

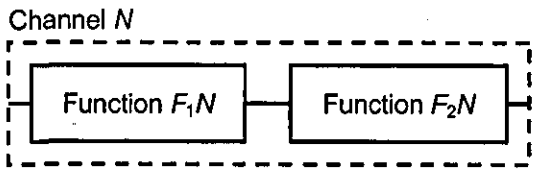


Figure 11.3.2. The configuration of a channel, N, of the system.

Function F_1 for each channel N is configured in such a way that two components, A_N and B_N, act in parallel, as shown in Figure 11.3.3. Function F_2 for each channel N can be configured in either one of two ways, with the proviso that whichever configuration is used for one channel is also used for the remaining channels in the system. In this way the function F_2 for each channel N can take one of the two forms also shown in Figure 11.3.3. The first of these involves components C_N and D_N in a series configuration and the second allows the system designer to add a further component, E_N, in parallel with these components.

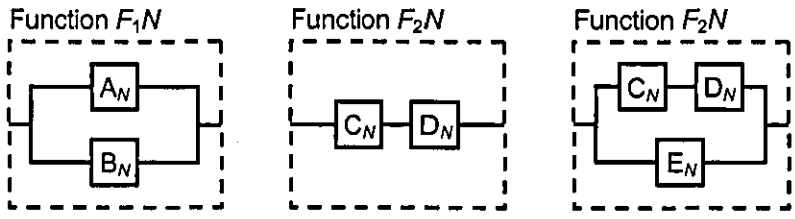


Figure 11.3.3. The sole configuration of function F_1N for each channel and the two possible configurations for function F_2N.

The system may also be constructed in such a way that there is sharing of information between channels. This means that if a function is lost on one (or more) channels that the corresponding function from another channel can be used to take its place. Figure 11.3.4 shows the layout of a two channel system constructed in this way.

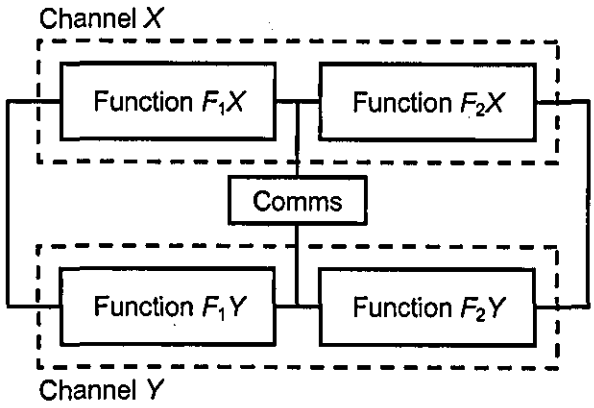


Figure 11.3.4. System layout for a 2 channel system with cross-channel communication (comms).

11.3.2 System Components

Other options available to the designer include using a single component A for function F_1 for each channel of the system, introducing a structural dependency between the channels, or using a separate, identical component A for each of the channels. There are two possible versions of each system component that may be used, each having advantages and disadvantages in terms of cost, weight and reliability. The designer can choose whether to include component E in the system or not for each channel, which will bring with it certain advantages or disadvantages. Table 11.3.1 shows a list of possible components that can be used in the system, along with their failure rates, weight, initial cost and also the maintenance cost and time.

Note that the components A_{common} and $A_{differs}$ represent the components that are available when A is used as a single component across all channels (A_{common}) or as a separate component for each channel ($A_{differs}$). The maintenance time shown for each component is the time taken to repair that component. The initial cost is the price that can be charged by the manufacturer for the specific component within a complete system and the maintenance cost is the cost to the manufacturer to repair the component when it fails within a system.

Component	Failure rate (failures per hour)	Weight	Initial Cost	Maint. Cost	Maint. Time
A_{common}	8.5×10^{-5}	80	200	25	2.0
	7.5×10^{-5}	40	300	50	
$A_{differs}$	8.5×10^{-5}	40	150	50	2.0
	7.5×10^{-5}	20	250	85	
B	4.0×10^{-5}	25	125	30	2.5
	3.6×10^{-5}	35	175	30	
C	3.0×10^{-5}	25	125	25	2.0
	2.7×10^{-5}	35	175	30	
D	2.0×10^{-5}	25	125	25	2.0
	1.8×10^{-5}	35	175	30	
E	3.5×10^{-5}	60	175	40	2.5
	2.5×10^{-5}	80	275	70	
Comms	2.5×10^{-5}	50	150	40	3.0
	2.0×10^{-5}	25	225	80	

Table 11.3.1. System components, their failure rates, weights, initial costs, costs incurred during maintenance and maintenance times.

The total initial revenue generated by the sale of a system is calculated by adding the initial costs of the individual components of the system and adding this to a fixed price that the customer will be charged. This fixed cost to the customer is set at 20 000 units. For components B, C, D, E and $A_{differs}$ calculating the additional cost to the customer means multiplying the number of channels by the initial cost of a single component, as shown in Table 11.3.1. When a single component A is used across all the channels in a system, i.e. A_{common} is used, the cost of just one component is added to the system, no matter how many channels the system has. When the cross-channel communications component, comms, is used in a system the cost added to the total is the initial cost for comms multiplied by the number of channels minus one.

The total weight of a system is calculated in a similar way. For components B, C, D, E, and $A_{differs}$ the weight added to the total for the system is obtained by multiplying the number of channels by the weight of a single component. When A_{common} is used the weight of just one component is added to the total for the system. When the cross-channel communications component, comms, is used the total cost is increased by the weight of just a single component, no matter how many channels the system has.

11.3.3 System Maintenance

The system designer may also choose between two repair strategies at maintenance, clearing all faults in the group that caused the maintenance deadline or clearing all faults from the system. These correspond to repair strategies 2 and 4 respectively in the TLD MCS code. There will be an associated cost for each maintenance event, calculated using the values given for the maintenance cost in Table 11.3.1. As a component is repaired the total maintenance cost increases by the maintenance cost of the component. This total cost is calculated over the course of the system lifetime and an average maintenance cost per system lifetime is calculated, which is checked for convergence as the simulations take place.

Also calculated at maintenance is the total time taken for the maintenance to take place. This is calculated by adding the maintenance times for the individual components being repaired. The designer requires that delays are kept to a minimum and assumes that there will be a delay if the total maintenance time for the system exceeds 4.5 hours. If this is the case then a delay is noted and the total number of delays is recorded. The average number of delays per system lifetime is calculated using this information. This value is checked for convergence as simulations take place.

11.4 Genetic Algorithm Strings

In order to incorporate the design process into a genetic algorithm a mapping must be developed between the various system design options and the strings to be used in the genetic algorithm. For this system 11 bit binary strings are used, each string representing a different system design and each bit of a string representing a particular option to the designer. Whether an individual bit is 0 or 1 will decide which options the designer includes in the system. Table 11.4.1 shows what each of the bits in the string represents in terms of the design options.

For example, consider the string `11100101000` (where the leftmost bit represents bit 0 and the rightmost bit represents bit 10). This system will have a different component A for each channel (1) each of which has a failure rate of 7.5×10^{-5} (1). Component B will have a failure rate of 3.6×10^{-5} (1), component C will have a failure rate of 3.0×10^{-5}

(0) and component D will have a failure rate of 2.0×10^{-5} (0). Component E will be included in the system (1) and will have a failure rate of 3.5×10^{-5} (0). Cross channel communications will also be included in the system (1) and will have a failure rate of 2.5×10^{-5} (0). The system will be maintained using repair strategy 2 (0) and will have 2 channels (0).

Bit	0	1
0	A common to all channels	Different A for each channel
1	$\lambda_A = 8.5\times10^{-5}$	$\lambda_A = 7.5\times10^{-5}$
2	$\lambda_B = 4.0\times10^{-5}$	$\lambda_B = 3.6\times10^{-5}$
3	$\lambda_C = 3.0\times10^{-5}$	$\lambda_C = 2.7\times10^{-5}$
4	$\lambda_D = 2.0\times10^{-5}$	$\lambda_D = 1.8\times10^{-5}$
5	E not included in system	E included in system
6	$\lambda_E = 3.5\times10^{-5}$	$\lambda_E = 2.5\times10^{-5}$
7	comms not included in system	comms included in system
8	$\lambda_{comms} = 2.5\times10^{-5}$	$\lambda_{comms} = 2.5\times10^{-5}$
9	Repair strategy 2	Repair strategy 4
10	System has 2 channels	System has 3 channels

Table 11.4.1. Interpretation of the bits in the system strings used in the genetic algorithm.

Now consider the same string, `11100101000` as an example for calculating the initial revenue and weight of the system. The system has two channels. Therefore the variable cost to the customer and the weight of the system will add up as follows:

- Since there is a different component A for each channel the cost for these will be 2×300 = 600 (2 being the number of channels and 300 being the cost of the version of component A with the lower failure rate). The weight will be 2×40 = 80.
- The version of component B with the lower failure rate is also used and the cost will be 2×175 = 350. The weight will be 2×35 = 70.
- The version of component C with the higher failure rate is used and the cost of this will be 2×125 = 250. The weight will be 2×25 = 50.
- The version of component D with the higher failure rate is used and the cost of this will be 2×125 = 250. The weight will be 2×25 = 50.
- The version of component E with the higher failure rate is used and the cost of this will be 2×175 = 350. The weight will be 2×60 = 120.

- The version of the cross-channel communications with the higher failure rate is used and the cost will be (2-1)×150 = 150. The weight will be (2-1)×50 = 50.

Thus the total weight of this system will be 80+70+50+50+120+50 = 420 and the total initial revenue will be the fixed revenue of 20 000 units added to the variable revenue calculated above which is 600+350+250+250+350+150 = 1 950 units. Therefore the total initial revenue is 21 950 units.

11.5 Evaluating String Fitness

For every system configuration there will be a number of advantages and disadvantages. These must be taken into account when calculating a fitness value for a string (and hence system configuration). These advantages and disadvantages will come in terms of the initial cost, maintenance cost and weight of the system, the number of delays due to the overrun of maintenance operations and, since TLD will be applied to the system, the system LOTC rate. It is assumed that the system can be configured in a number of ways as detailed in the previous sections and has an operational lifetime of 130 000 hours, flight length of 10 hours. Both STD and LTD faults are to be maintained using MEL maintenance and STD and LTD intervals are fixed at 500 and 2500 flight hours respectively. Instead of using the usual limits for deciding STD and LTD category faults, these are conservatively set at 80% of their original values, i.e. a new limit of 60 failures per 10^6 flight hours for LTD faults and a limit of 80 failures per 10^6 flight hours for STD faults. This was done in order that the TLD categories applied to the system were spread from DND through STD to LTD. It proved difficult to set up a simple example where the instantaneous LOTC rates observed for the example systems were such that they allowed the setting of the various dispatch categories for different faults.

String fitness is evaluated by considering the total profit that will be generated for the manufacturer. This is calculated by subtracting the maintenance costs incurred by the manufacturer from the total initial revenue gained from the sale of the system to the customer. However, since the weight of the system, the system LOTC rate and the number of delays experienced during the operational lifetime of the system due to maintenance activities are also important, these quantities must be taken into account when calculating the fitness of strings. Penalties are added to the profit whenever the

weight, LOTC rate and number of delays exceed certain amounts. These penalties increase in magnitude the more the value desired by the designer is exceeded. Thus the fitness is given in terms of a penalised profit function, given by:

$$P_{pen} = P - W_{pen} - L_{pen} - D_{pen}, \qquad 11.5.1$$

where:

P_{pen} is the penalised profit,

P is the unpenalised profit calculated by subtracting maintenance costs from the initial revenue,

W_{pen} is the penalty exerted due to excess weight,

L_{pen} is the penalty exerted due to excess system LOTC rate,

D_{pen} is the penalty exerted due to excess delays.

The penalty formulae for weight, LOTC rate and delays take forms similar to that given in [24] for a cost penalty function. The weight penalty for a system is given by:

$$W_{pen} = W_c \left(\frac{W - W_{\max}}{\frac{w_1}{100} W_{\max}} \right)^{w_2}, \qquad 11.5.2$$

where W is the weight calculated for the system, $W_{\max}$ is the desired value of the weight above which penalties are incurred by the system and w_1 and w_2 are constants. W_c is a constant given in terms of units of cost that is used to equate the excess weight of the system to a reduction in the profits gained. It reflects the fact that a customer using the system will incur extra costs due to the extra weight that must be carried throughout the operational lifetime of the system and that this will in turn reduce the amount that the customer is willing to pay for the system initially.

The delay penalty for a system is given by:

$$D_{pen} = D_c \left(\frac{D - D_{\max}}{\frac{d_1}{100} D_{\max}} \right)^{d_2}, \qquad 11.5.3$$

where D is the number of delays experienced during the operational lifetime of the system, $D_{\max}$ is the maximum value of the number of delays above which penalties will begin to be incurred by the system and d_1 and d_2 are constants. D_c is a constant given in terms of units of cost that is used to equate the excess delays incurred during the operational lifetime of the system to a reduction in the profits gained. It reflects the fact that a customer using the system will incur extra costs whenever delays imposed on

their passengers force them to provide monetary compensation to those passengers. As was the case when the weight of the system increased above a certain level, the amount that the customer is willing to pay the manufacturer for the system will fall as the expected number of delays rises.

The final penalty is for the system LOTC rate and this is constructed in the same manner as the penalties for excess weight and number of delays. The LOTC penalty for the system is given by:

$$L_{pen} = L_c \left(\frac{L - L_{\max}}{\frac{l_1}{100} L_{\max}} \right)^{l_2}, \qquad 11.5.4$$

where L is the calculated LOTC rate, $L_{\max}$ is the value of the LOTC rate above which penalties will begin to be incurred by the system and l_1 and l_2 are constants. L_c is a constant given in terms of units of cost that is used to equate the excess LOTC rate to a reduction in the profits gained. In this case the reduction in profits is mostly made to penalise the system for not complying with the certification requirement that the average system LOTC rate must not exceed 10 failures per 10^6 flight hours. Whilst this penalty function might not be ideal (a step function would be better, where as soon as the limit $L_{\max}$ is breached, the penalty steps from zero to a large value) it has been used here because of the difficulty encountered in constructing a simple example to be analysed using this GA methodology. It does, however, still punish higher LOTC rates very severely.

The constants used in each of the above equations for the penalties imposed on the systems are given in Table 11.5.1. The penalties are structured in such a way that small increases in the weight, number of delays or LOTC rate are tolerated more readily than larger increases in these variables. In order to explain the constants, consider the weight penalty function. The desired value of the weight above which penalties will be incurred is $W_{\max} = 450$ units. $w_1 = 20$ implies that a 20% increase in the weight above $W_{\max}$ will be tolerated before the penalty imposed rises above that which would be imposed if a linear penalty function was used. $w_2 = 1.5$ denotes that the penalty imposed will rise in line with a power of 1.5. $W_c = 1000$ means that a 20% increase in weight above $W_{\max}$ will lead to a penalty of 1000 units of cost being imposed on the system. The values of variables for delays and LOTC rate are explained in the same way. In

order to visualise the meaning of the constants used in relation to the penalties imposed consider Figure 11.5.1, Figure 11.5.2 and Figure 11.5.3, which show plots of the W_{pen}, D_{pen} and L_{pen} for a range of weights, delays and average system LOTC rates respectively. Note how the penalties begin to be imposed as the variable increases above the desired value and are initially relatively small but that, further increases are punished more severely, seen from the increasing gradient plotted curves. Note how, for the system LOTC rate, the penalty imposed is relatively huge in comparison to the penalties for the other two variables. This reflects the fact that any increase of the LOTC rate above a value of $L_{max} = 7.5$ is less-readily tolerated.

Constant	Value	Constant	Value	Constant	Value
W_{max}	450	D_{max}	5	L_{max}	7.5
w_1	20	D_1	25	l_1	10
w_2	1.5	D_2	1.25	l_2	2
W_c	1000	D_c	500	L_c	1000

Table 11.5.1. The values of constants used in the penalty functions.

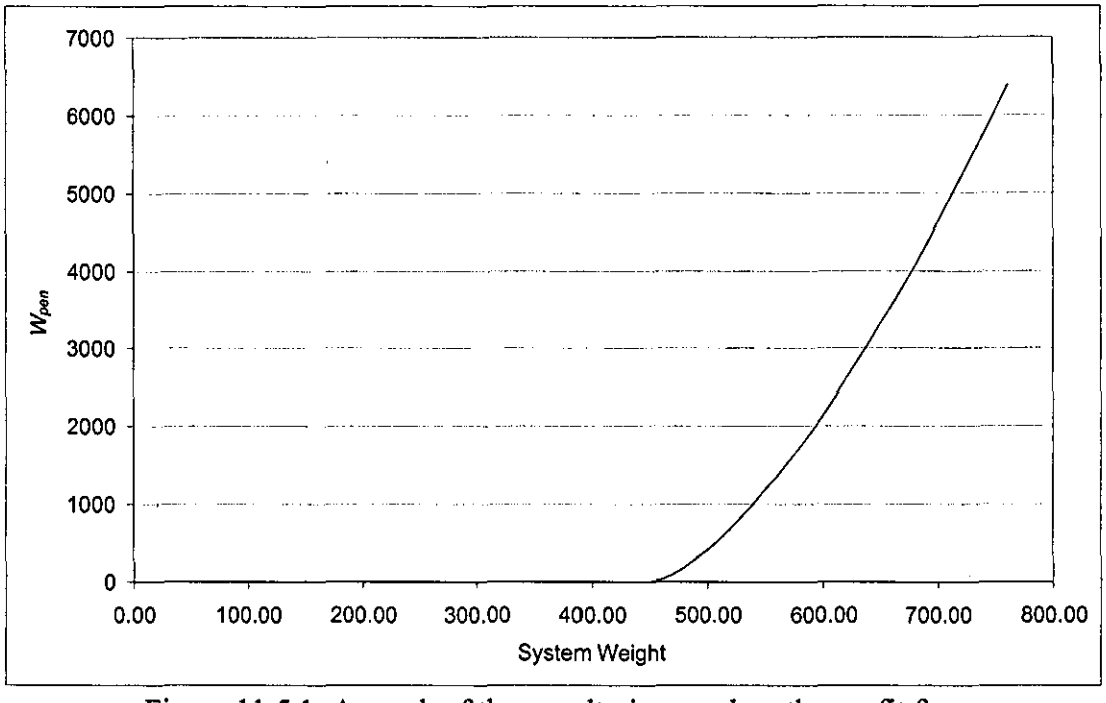


Figure 11.5.1. A graph of the penalty imposed on the profit for different system weights.

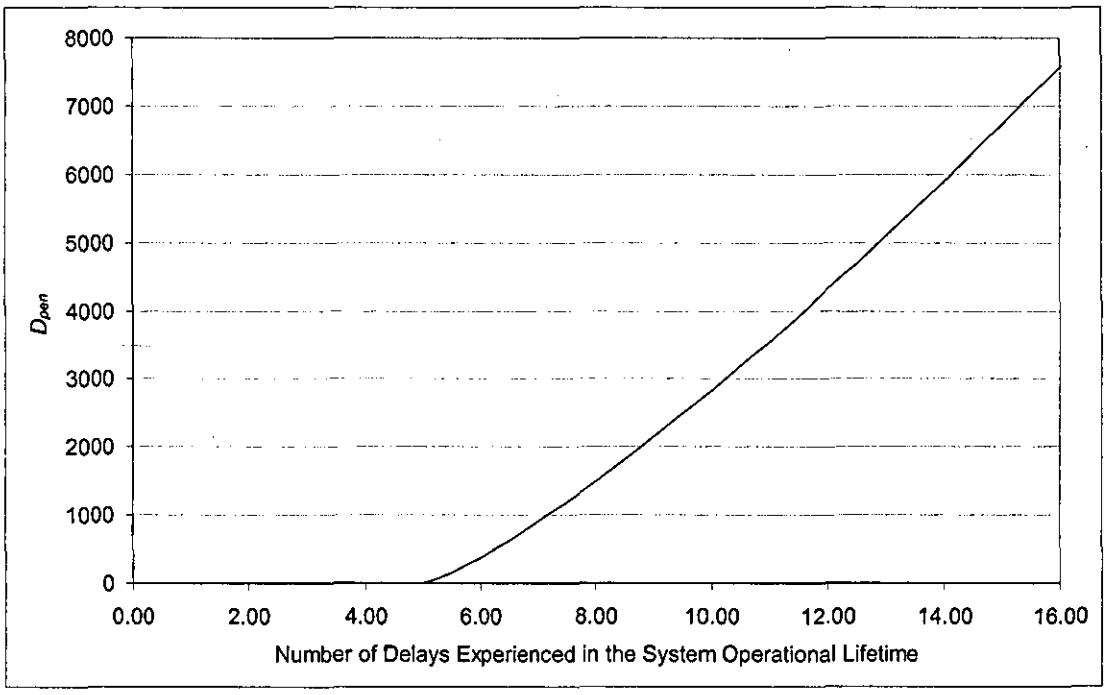


Figure 11.5.2. A graph of the penalty imposed on the profit for different expected numbers of delays experienced in the system lifetime.

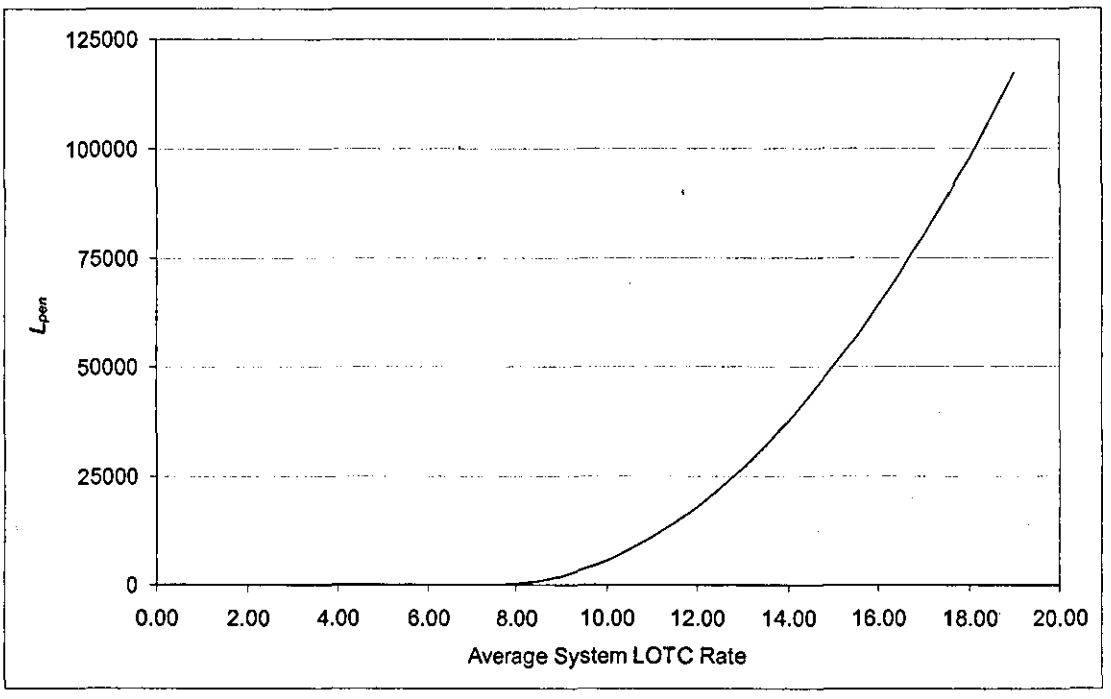


Figure 11.5.3. A graph of the penalty imposed on the profit for different values of the average system LOTC rate.

11.5.1 Calculating String Fitness

The string fitness for a system is calculated as the penalised system profit, as shown in equation 11.5.1. Calculating string fitness for each system first involves calculating the unpenalised profit and the total number of delays, the weight and the average system LOTC rate for that system. These latter three values are then used to calculate penalties to the system profit by substituting these values into equation 11.5.2, 11.5.3 or 11.5.4 as appropriate. The total profit gained by the manufacturer for the system will depend upon both the initial cost of the system to the customer and the maintenance costs incurred over the operational lifetime of the system by the manufacturer. The initial cost of the system, like the total weight of the system, is calculated knowing the system configuration and the weights and initial costs of the components as shown in Table 11.3.1. During the implementation of the genetic algorithm these two values (initial cost and weight) are calculated using the information contained in the strings relating to the various system configurations. The maintenance costs for the system are calculated using the TLD MCS code, which computes the average maintenance cost over a system lifetime. This then allows the calculation of the unpenalised profit. Similarly, the number of delays is computed using the TLD MCS code, which works out the length of maintenance operations using the values in Table 11.3.1 and calculates how many of these exceed 4.5 hours. This number is averaged over the simulations carried out to give a number of delays per system lifetime which can then be used to calculate a delay penalty in the genetic algorithm. The LOTC rate is also calculated using the TLD MCS code.

In order for the TLD MCS code to be called by the genetic algorithm code the relevant inputs for the system (fault tree, basic event failure rates, etc.) are provided to the TLD MCS code by the genetic algorithm code according to the string under consideration. The TLD MCS code is then used to automatically set dispatch criteria for the system for a STD interval of 500 hours and a LTD interval of 2500 hours and the system LOTC rate is obtained for these values of STD and LTD interval.

Whilst calculating penalties for the different variables, it is important to note that the penalties for system weight, number of delays and average system LOTC rate should only be calculated if W_{max}, D_{max} or L_{max} are exceeded.

11.6 Reproduction Probabilities

When the genetic algorithm is started a population of individual strings are constructed and the fitness of those strings is evaluated. The next step in the process is to select the strings such that the fitter strings have a greater chance of reproducing offspring to make up the next generation of individual strings. The probability of a string being selected for reproduction will depend on its fitness. The measure of fitness here is the penalised profit and fitter strings will have a higher penalised profit value. [23] describes a method of selection wherein strings are allocated a certain percentage of a hypothetical roulette wheel according to their fitness. Then, a number of random numbers are generated and these are used to identify the individuals which will be chosen from this 'weighted roulette wheel.'

To illustrate such a weighted roulette wheel consider a simple example where 4 individual strings are to be selected for breeding for the next generation and that the four individuals, i, have fitness values, f_i, as shown in Table 11.6.1.

i	f_i
1	50
2	1020
3	710
4	500

Table 11.6.1. Four individuals, i, and their fitness values, f_i.

Figure 11.6.1 shows the roulette wheel representation for these individuals, with the entire wheel representing the total fitness of the population, i.e. the sum of the f_i's (2280). Each individual is then allocated a segment of the roulette wheel representing its own proportion of the total fitness of the population. In order to select an individual form the population a random number between 0 and 1 is generated and, with the entire roulette wheel representing 1, a position on the wheel is chosen. This position will then relate to one of the individuals. Considering this example, individual 2 has the greatest probability of being selected, as would be desired since 2 is the fittest of the individuals, because it occupies the largest segment of the roulette wheel. Similarly, individual 1 has the lowest probability of being selected due to its small proportion of the roulette wheel.

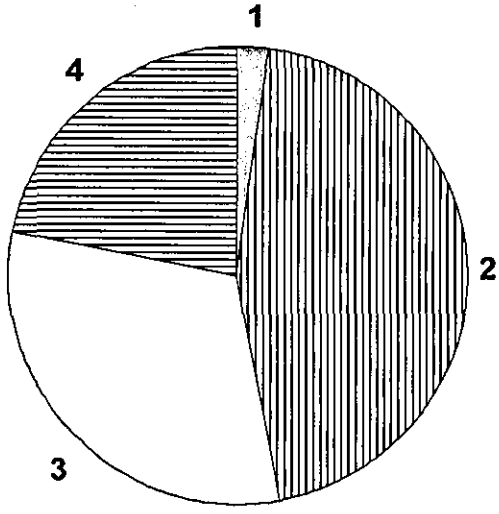


Figure 11.6.1. A representation of the 'roulette wheel' for the four individuals.

However, the method used to allocate a proportion of the roulette wheel for individual strings in this example does not rely solely on the penalised profit value for individual strings. Instead, a method similar to that used in [24] is applied wherein the fitness of a string means that it is assigned to a certain category. There are 9 categories to which a string can be assigned and each of these has a certain weight associated with it. This method is used because it copes with the diverse range of penalised profits possible. It also shows a high degree of sensitivity to fitter strings, due to the weighting applied to the fitter strings. The categories are given in Table 11.6.2, along with the associated range of penalised profit value and the weighting applied to the category.

Upper Limit	Category	Lower Limit	Weight
2000 ≥	1	no lower limit	1^2
6000 ≥	2	> 2000	2^2
10000 ≥	3	> 6000	3^2
12000 ≥	4	> 10000	4^2
14000 ≥	5	> 12000	5^2
16000 ≥	6	> 14000	6^2
17000 ≥	7	> 16000	7^2
18000 ≥	8	> 17000	8^2
no upper limit	9	> 18000	9^2

Table 11.6.2. The nine fitness categories used in the genetic algorithm, their associated upper and lower limits of penalised profit and weightings.

This allows all of the system strings in a population to be assigned to a category. The penalised profit for each string is multiplied by the weight of its associated category. These values are added for each category. These sums are then used to determine the proportion of the roulette wheel that will be assigned to each category. It then remains to allocate a certain proportion of this for each string in each category. Whilst doing this it is important that strings in the higher categories are allocated higher proportions than those in lower categories. This is ensured by working through the categories from the lowest to the highest in the following way. The average percentage allocated to each string in the previous non-empty category is calculated, then this percentage is equally distributed throughout the strings of the category currently under consideration. This leaves a certain excess that is then distributed between the strings in the current category according to the ratio of their fitness values. Note that if the string relates to a system that returns a loss rather than a profit there will be a negative value for the fitness value. To overcome this a fitness of 1000 is assigned throughout the process of these calculations for any string whose penalised profit is less than 1000. This has the effect of applying a higher percentage than would normally be applied for some systems. However, this is considered appropriate since it maintains an element of diversity within the gene pool.

11.7 Results

Results were obtained using the genetic algorithm code over 20 successive generations of individual system strings (plus one generation chosen at random to start off the genetic algorithm process). Each generation contained a population of 10 individual strings. As mentioned earlier, the options used in the TLD MCS code were for a flight length of 10 hours, a system operational lifetime of 130 000 hours and both STD and LTD faults were to be maintained using MEL maintenance. The STD and LTD dispatch intervals were set at 500 and 2500 hours respectively. The first check for convergence was performed after 500 000 simulations and subsequent checks were performed every 10 000 simulations. Convergence of LOTC rates was assumed after 3 consecutive values were seen to match to 3 s.f. The number of delays and maintenance cost per system lifetime were assumed to converge when matches between consecutive values were seen to 2 decimal places.

The results obtained from the genetic algorithm code are shown in Appendix 7. It can be seen from these results that the general trend is, as would be hoped, for the fitness of successive generations to improve. This can be confirmed by considering Figure 11.7.1, which shows the sum of the penalised profits for all individuals in the generation for all 21 generations considered. It can be seen that the overall fitness tends to rise, particularly after generation 1. Note that occasional falls in the sum total fitness for a generation will be due to the random effects of the genetic algorithm process, i.e. results of mutation or crossover as reproduction occurs.

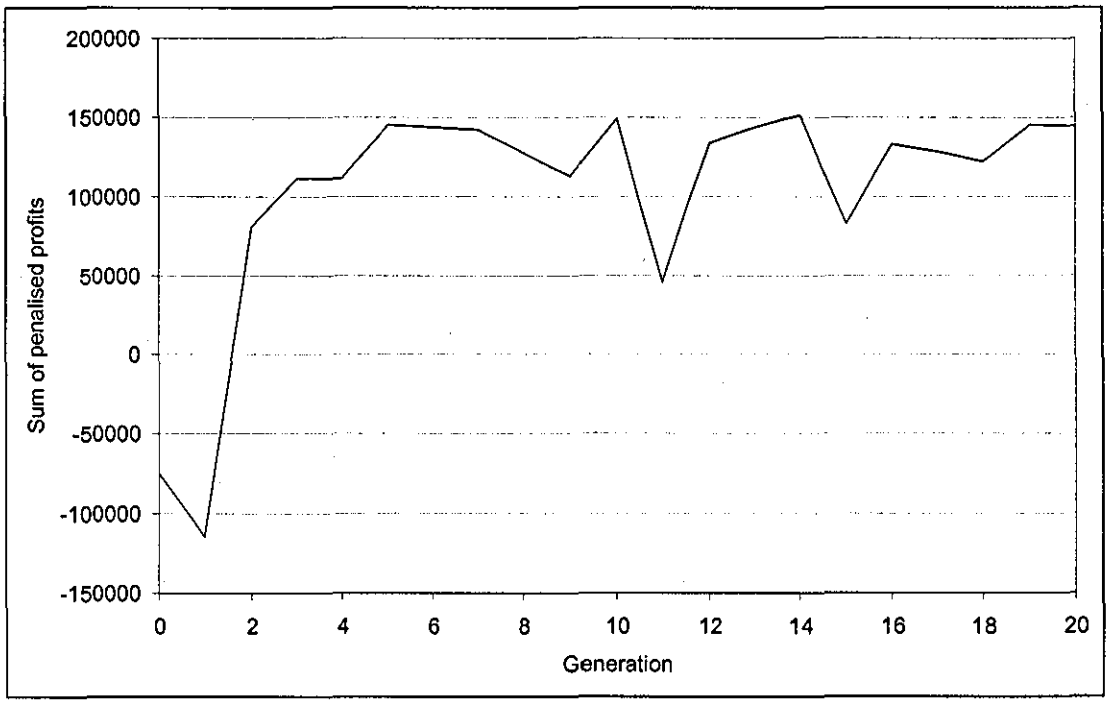


Figure 11.7.1. The sum of penalised profits for the generations of individual strings in the genetic algorithm code.

The overall fittest individual found was recorded in the genetic algorithm code after every generation. Figure 11.7.2 shows a graph of the value of the penalised profit obtained for the fittest individual after each generation. Note that this is not necessarily the fittest individual in each generation. By generation 3 the best penalised profit obtained had risen to 17482.47, which wasn't bettered until generation 9, when an individual was generated with fitness 17584.06. A fitter individual was not discovered throughout the remainder of the generations.

In total there were 5 changes of fittest individual identified throughout the course of completion of the genetic algorithm. These individuals were identified in generations 0,

1, 2, 3 and 9. The data gathered for these individuals during the genetic algorithm is shown in Table 11.7.1.

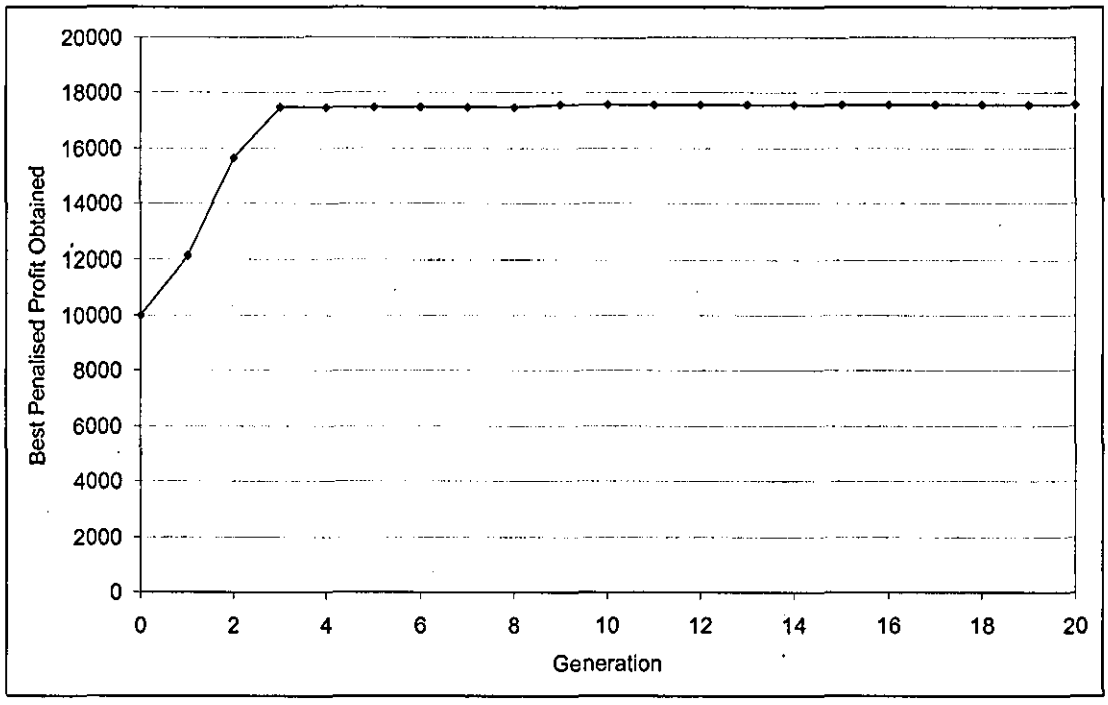


Figure 11.7.2. The penalised profit of the fittest individual discovered after each generation.

String	LOTC rate	Delays	Revenue	Maint. Costs	Weight	Penalised Profit
00001000011	8.98	6.99	22950	10111.27	335.0	9997.58
11101001011	1.50	13.20	24950	7575.34	395.0	12125.38
10110111101	0.32	10.90	26300	3341.17	670.0	15658.47
11110111101	0.28	10.60	26900	3788.29	610.0	17482.47
11100101111	0.32	11.20	26000	4028.98	520.0	17584.06

Table 11.7.1. The fittest individual strings identified throughout the genetic algorithm process and associated variables.

The fittest individual identified throughout the course of the entire genetic algorithm process is represented by string 11100101111. The system represented by this string can be decoded using the information shown in Table 11.4.1. It has 3 channels, with a separate component A used in function F_1 of each channel. Function F_2 of each channel is constructed in a parallel configuration, with component E included along with components C and D. The component representing communication between the channels, comms, is also included. Figure 11.7.3 shows the system layout. The failure rates for the individual components are as shown in Table 11.7.2. The final piece of

information contained in the string for this individual is for the repair strategy at maintenance. This is repair strategy 4, i.e. all faults are cleared from the system at maintenance. When the dispatch criteria were automatically set for this system (for all single, dual and triple fault combinations) they were all seen to be LTD. The LOTC rate of the system is 0.32 failures per 10^6 flight hours and there will be 11.2 delays on average in a system lifetime. The weight of the system is 520 units and the initial revenue generated from the system sale is 26000. The cost of maintenance throughout the system lifetime is 4028.98, giving a profit of 21071.02. However, the penalised profit was 17584.06.

Component	Failure rate (failures per hour)
$A_{differs}$	7.5×10^{-5}
B	3.6×10^{-5}
C	3.0×10^{-5}
D	2.0×10^{-5}
E	3.5×10^{-5}
Comms	2.5×10^{-5}

Table 11.7.2. The component failure rates for the optimal system.

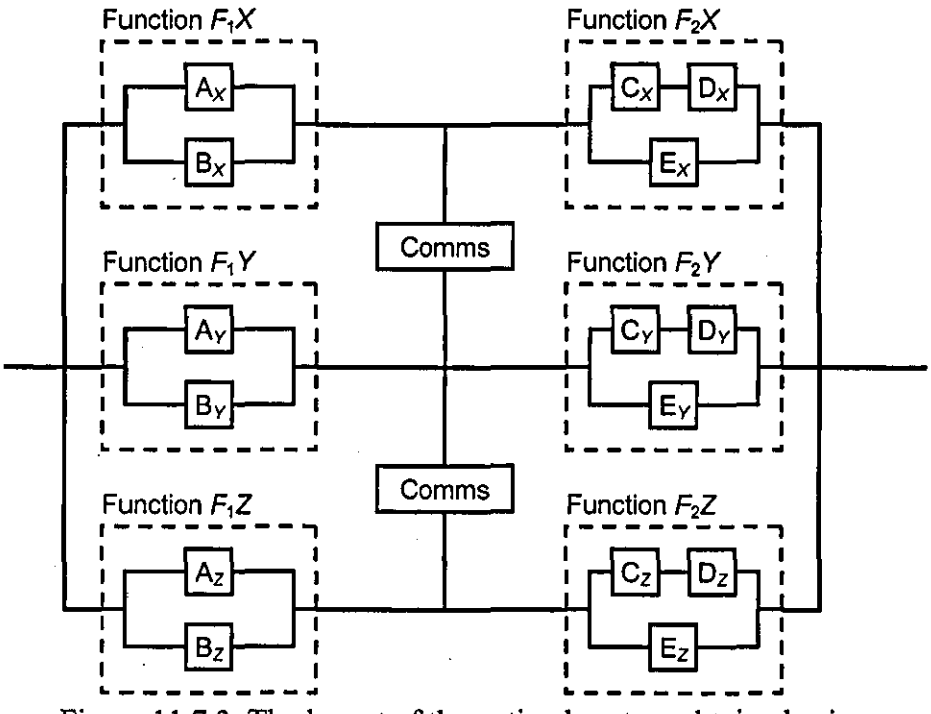


Figure 11.7.3. The layout of the optimal system obtained using the genetic algorithm code.

11.8 Summary

The TLD MCS code was embedded within further code written to perform a genetic algorithm optimisation procedure. A simple example was constructed which was then used to demonstrate a potential application of such a tool. The optimal configuration of the system design was achieved relative to certain constraints in order to maximise the profit obtained from the sale of such a system. The code written was used to find an optimal solution to the problem. Although the example is a simple one it nonetheless demonstrates the potential benefits that are available when using MCS to model TLD. The MCS can be used to obtain more information about the system and its operation during its operational lifetime. For instance, in the simple example considered here, the number of expected delays experienced by an airline operator in the system operational lifetime and the total cost to the system supplier of maintaining the system were calculated along with the system LOTC rate. This process was carried out in addition to the automatic setting of dispatch criteria for each system considered. The methods used to calculate such values were rudimentary for this simple system but, clearly, if such a process were to be implemented for a real system then the methods used could be applied as desired by constructing the algorithms for use in the code in the appropriate way.

The use of such a tool, which utilises MCS within a genetic algorithm environment, could conceivably be of great value during the design process of a real FADEC system. It could be used during the design phases to show that the FADEC system being considered met with the relevant certification requirements whilst also providing information that could assist in the choice of components used or the system architecture. The use of MCS to model the application of TLD would also allow the dispatch criteria for any systems to be set to automatically meet the certification requirements.

Chapter 12 : Conclusions and Future Work

12.1 Summary of Work

Time-Limited Dispatch (TLD) allows the dispatch of aircraft with known faults present in the engine control systems. In order to be able to apply TLD to a system the compliance of the system with certification regulations must be demonstrated. To show the certification requirements are being met and that the levels of system safety are sufficient a good mathematical model must be constructed. This must be able to deal with the mechanics of the TLD processes.

Fault tree analysis was discussed but this has a number of shortcomings when considered in relation to modelling TLD. The size of systems to which TLD is applied results in large fault trees that will require approximations to be carried out in order to perform any quantitative or qualitative systems analysis. The basic events of a fault tree must be statistically independent in order for analyses to be carried out. This is not the case for systems to which TLD is being applied. Aircraft operating procedures introduce dependencies and also maintenance and repair operations will introduce further dependencies. For these reasons fault tree analysis is unsuitable as a method of modelling TLD.

Markov analysis is capable of dealing with the dependencies that prevent the application of fault tree analysis to the application of TLD. However, Markov analysis requires that the system should be stationary, i.e. that the rates within the model are constant with time. This might be a reasonable assumption for systems to which TLD will be applied. However, it could be the case that certain system components have non-constant failure or repair rates. For example, a component could have a failure probability that follows a Weibull distribution. Perhaps the most important aspect of Markov models that makes them unsuitable for modelling TLD is their size. Markov models grow exponentially with the system size and, as such, problems of a state space explosion are possible on real, large aircraft systems.

Monte Carlo simulation was presented as a suitable technique for the modelling of TLD. It is simple, based on logical rules and capable of dealing with the problems that affect fault tree analysis and Markov analysis such as dependencies or large system sizes. Information can potentially be found when performing a Monte Carlo simulation that would be difficult or impossible to obtain using any other techniques. Monte Carlo simulations can use a fault tree representation of the system, which is a structured, logical method of representing a system. This is a good characteristic for the Monte Carlo simulation to use from fault tree analysis. Perhaps the main potential negative point of Monte Carlo simulation is the large amount of time needed for simulations to take place. However, the fast speed of modern digital computers and the potential for Monte Carlo simulation codes to be parallelised should take away the concerns associated with this problem.

Currently recommended techniques for modelling TLD were introduced. These involve a number of simplifying assumptions and require the calculation of the instantaneous failure rates to LOTC of the system. However, the method of calculating these rates is not a simple task, and would presumably require the use of fault tree analysis or Markov analysis. The recommended method involves using fault tree analysis to compute an approximation for the failure probability of the system and dividing this by an average flight time. This approximation will not necessarily be a good one. Each one of the recommended models involves a vast reduction in the number of system states and, as such, may not represent the system characteristics in an accurate way. The MEL maintenance technique is said to be inherently modelled by these methods.

A C++ code (TLD MCS) was developed that allows TLD analysis to be performed using Monte Carlo simulation. The TLD MCS code was capable of modelling many of the different maintenance options that are possible when implementing TLD, such as MEL maintenance, PIR maintenance and opportunistic maintenance. Four different repair strategies are possible at maintenance, each of which represents carrying out various amounts of repair work to the system at maintenance, from the minimum amount possible to the maximum amount possible. Many other options are available to the user of the TLD MCS code, including options such as convergence criteria, the system operational lifetime and flight length. The output produced can include files containing the events that occur within simulations, the average system LOTC rates, the

instantaneous system LOTC rates and the dispatch criteria that can be applied to the system in order to satisfy the requirements for the instantaneous system LOTC rates. Methods of calculating importance measures for the system fault tree basic events and the TLD faults were also developed.

Transient methods of solution for Markov models were introduced and a C++ code was developed that could be used to solve a Markov model in this way for a system. The numerical integration procedures within the code were implemented using a 4th order Runge-Kutta method. Different methods of including repairs within the Markov models were described, one of which involved the inclusion of repairs in the model as repair rates within the transition rate matrix for the system. The second method involved carrying out repairs at discrete points within the numerical integration procedure by shifting the values of the appropriate state probabilities. The importance of the ordering of the shifting of these probabilities was discussed in relation to carrying out the repairs in the required manner. Similarities were noted between the different methods of including repairs in the transient Markov models and the MEL and PIR maintenance approaches. However repairs are included, the maintenance approach will not exactly match the approach that is used in reality in the way that is possible when using Monte Carlo simulation.

Two simple systems were modelled using the different recommended methods just described, the transient Markov model code and the TLD MCS code. The Time-Weighted Averages (TWA) approach gave inaccurate results even for the small two component system whereas the results obtained using the dual fault state Markov model and the two other methods were in reasonable agreement. This backs up the claim that the reduced fault state Markov approach is the better of the recommended methods [1]. The method of approximating instantaneous failure rates to LOTC for the SAE methods by dividing a probability by a flight time was seen to be potentially inaccurate when the rates were compared to those obtained using the TLD MCS code. A problem associated with this for the dual fault state Markov models was that the models were shown to be heavily dependent upon rates used within them. This would be a potential source of problems if the failure rates in the models could themselves be inaccurate. The structure of the reduced fault state Markov models was called into question because of the fact that the results obtained using the model were seen to be optimistic in comparison to the

TLD MCS code when failure rates from the MCS were substituted into the reduced fault state Markov model. The MCS code's potential flexibility was demonstrated by modelling many different maintenance options for the simple system models. Comparisons could be made between the results obtained using the different maintenance techniques. The process of automatically setting dispatch criteria was demonstrated on a system and shown to produce good quality results. It was argued that the TLD MCS code produced the more accurate results due to the way that the TLD process was more accurately represented within it. The results obtained for the transient Markov model were in reasonable agreement with the TLD MCS code. However, the TLD MCS code is still considered to produce the more accurate results of the two techniques due to its accurate portrayal of TLD.

A large scale system was modelled, the fault tree of which contained features that were representative of a real aircraft system. This countered one of the earlier concerns about Monte Carlo simulation, namely that simulations could take a long time to perform to obtain results to the required levels of system convergence. Simulations for the system were seen to produce results converged to 3 significant figures in times of around 40 minutes. The TLD MCS code was used to set dispatch criteria for the system that would satisfy an adjusted set of requirements for the instantaneous failure rates to LOTC for the system. Importance measures were produced, which could be used to identify where attention should be focussed in order to improve the system performance. The option in the TLD MCS code allowing dependencies between basic event repairs to be modelled was used. This specified 'associated' repairs that ensured that certain faults would be cleared from the system whenever certain TLD faults were repaired. An argument was also presented wherein a shorter operational lifetime would be modelled that could be used to represent a major maintenance operation taking place at regular intervals throughout a system lifetime. Clearing all faults from the system at these major maintenance operations would allow the less rigorous repair strategies to be used while still allowing the system LOTC rates obtained to match those obtained using the more rigorous repair strategies. The conventional Markov model was too large to consider for the large scale system, the state space explosion problems coming to light here. Results were obtained for the dual fault state Markov model using instantaneous failure rates to LOTC calculated using the TLD MCS code. However, even with these rates included in the model, which are considered to be accurate, the dual fault state Markov model did

not produce similar results. If the decision had been taken to approximate the failure rates using the probability over flight time technique it is likely that the results obtained would be worse still. The dual fault state Markov model was seen to be inflexible in comparison to the TLD MCS code, which could model more situations applicable to TLD and was more accountable in the way the results were obtained.

Finally, a technique of system design optimisation was demonstrated for a simple example. The TLD MCS code was embedded within a genetic algorithm code and this was used to choose an optimal design for a system given a number of constraints. The TLD MCS code and the simulation technique in general demonstrated more flexibility and uses here by not only performing the automatic setting of dispatch criteria for various systems but also producing more information about the system, namely measures of the number of delays expected in a system lifetime and also the expected maintenance cost per system lifetime. This serves to demonstrate that the potential uses for a Monte Carlo simulation technique are potentially widely varied.

12.2 Conclusions

- Recommended methods of modelling fault tree analysis use certain assumptions and approximations that reduce their accuracy in modelling the TLD of aircraft systems:
    - The vastly reduced number of system states provides a poor representation of the system structure,
    - The technique of dividing a failure probability by a flight time to approximate the instantaneous failure rates to LOTC for dispatchable system faults is open to doubt as to its accuracy,
    - Even when instantaneous failure rates to LOTC that were considered accurate were used in the system the computed average system LOTC rate did not agree with those obtained using other techniques.
- The results obtained using the transient Markov model for the eight component system back up the suggestion that Monte Carlo simulation provides an accurate set of results for a system. The transient Markov model offers a full Markov model solution to the TLD problem, with repairs included in two different ways

conceptually similar to MEL and PIR maintenance. The agreement between the MCS and transient Markov model results reinforce the suggestion of inaccuracy in the recommended techniques.

- Monte Carlo simulation provides a flexible, accountable means of modelling the application of TLD to a system.
    - Great use can be made of the fact that dispatch criteria for the system can be iteratively set using a simulation technique, thus allowing the dispatch criteria for the system to be set according to the instantaneous failure rates to LOTC of the dispatchable fault states whilst assuring that targets for the system LOTC are met,
    - Many maintenance options can be modelled, which is potentially of great importance when TLD will be applied, since advantage should be taken of the option to use TLD to schedule maintenance operations in order that disruptions be minimised.
- As regards the time taken to perform a Monte Carlo simulation analysis of a system, the time taken to model a large scale system displaying properties similar to those that might be seen on a real aircraft was reasonably short. However, even if simulations were seen to take times of the order of days to run this would not be unacceptable when weighed against the length of time taken in designing and certifying an aircraft engine.
- When implementing TLD it could be prudent to perform a major maintenance operation regularly in order to clear all faults from the system at maintenance. This conclusion was reached after considering the results obtained from modelling various system operational lifetimes and observing that the lower the operational lifetime, the lower the system LOTC rate. This then led to the idea of using a lower operational lifetime to model a regular major maintenance operation.
- Basic event and TLD fault importance measures were produced which enable weaknesses within the system under consideration to be modelled.
- Monte Carlo simulation of TLD can be used within a genetic algorithm optimisation procedure in order to assist in the design of a system subject to certain constraints. Monte Carlo simulation can be used in such circumstances to obtain more information about the system than just the various LOTC rates

associated with the system. Examples were given for the calculation of expected delays and expected maintenance cost over the lifetime of the system.

12.3 Future Work

The TLD MCS code provides the basis for a tool that could be used for the system safety analysis of aircraft systems. Potential future work could look at the speed and efficiency of the code. In particular, the opportunity exists to parallelise the code, which could offer great benefits in terms of the speed of the simulation procedures. The same parallelisation procedure could also be applied to a genetic algorithm technique such as that presented in Chapter 11. Such increases in efficiency would be of most use when real, large aircraft systems were modelled.

Another option that might be worthy of investigation when considering TLD is a more flexible method of applying dispatch categories to faults. The potential exists for a sliding scale of dispatch categories, allowing different lengths of dispatch depending upon the LOTC rate of the system. For example, instead of just two dispatchable categories, STD and LTD, perhaps the system could be dispatched with a fault present for a time interval that was more directly related to that fault's instantaneous failure rate to LOTC. This could mean that the dispatchable range of LOTC rates for faults (up to 100 failures per 10^6 flight hours) could be split into a greater number of intervals, thus offering a number of grades of dispatch. Another alternative is that the dispatch interval for a fault could be in some way directly proportional to its instantaneous failure rate to LOTC.

The most obvious work for the future would be the modelling of a real aircraft system. This would provide the ultimate test and validation of the techniques used and could potentially provide further information as to what other information of interest could be extracted using Monte Carlo simulation.

References

[1] Larsen, H. and Horan, G., *Time-Limited Dispatch: An Interactive Training and Self-Study Course*, Keybridge Tecnologies, Inc., 2002, http://www.keybridgeti.com/videotraining/manual_download_index.php?file=24

[2] *FAA Memorandum: Policy for Time-Limited Dispatch (TLD) of Engines Fitted with Full Authority Digital Engine Control (FADEC) Systems*, Federal Aviation Administration, ANE-1993-33.28TLD-R1, June 29, 2001.

[3] Watson, H. A., *Launch Control Safety Study*, Bell Telephone Laboratories, 1961.

[4] Henley, E. J. and Kumamoto, H., *Reliability Engineering and Risk Assessment*, Prentice-Hall, 1981.

[5] Andrews, J.D.and Moss, T. R., *Reliability and Risk Assessment*, 2nd Edition, Professional Engineering Publishing, 2002.

[6] Haasl, D. F. et al, *Fault Tree Handbook*, US Nuclear Regulatory Commission, NUREG0492, 1981.

[7] Fussell, J. B., Henry, E. B., Marshall, N. H., *MOCUS: A Computer Program to Obtain Minimal Cut Sets From Fault Trees*, ANCR-1156, 1974.

[8] Schneeweiss, W. G., *Boolean Functions with Engineering Applications and Computer Programs*, Springer-Verlag, 1989.

[9] Foster, J. W., Phillips, D.T., Rogers, T.R., *Reliability, Availability and Maintainability*, M/A Press,1981.

[10] Frankel, E. G., *Systems Reliability and Risk Analysis*, 2nd Edition, Kluwer Academic Publishers, 1988.

[11] Billinton, R. and Allan, R. N., *Reliability Evaluation of Engineering Systems: Concepts and Techniques*, 2nd Edition, Plenum Press, 1992.

[12] Nagle, R. K. and Saff, E. B., *Fundamentals of Differential Equations*, 3rd Edition, Addison-Wesley, 1993.

[13] Villemeur, A., *Reliability, Availability, Maintainability and Safety Assessment*, Wiley, 1992.

[14] Gerald, C. F. and Wheatley, P. O., *Applied Numerical Analysis*, 7th Edition, Pearson Education, Inc., 2004.

[15] Boyd, M. A., *An Introduction to Markov Modelling: Concepts & Uses*, Annual Reliability and Maintainability Symposium, 1998.

[16] Davidson, J., *The Reliability of Mechanical Systems*, Mechanical Engineering Publications Limited for IMechE, 1988.

[17] *Guidelines for Time-Limited Dispatch Analysis for Electronic Engine Control Systems*, SAE International, SAE ARP5107, June 1997.

[18] *Guidelines for Time-Limited Dispatch Analysis for Electronic Engine Control Systems*, SAE International, Revision A, SAE ARP5107, January 2005.

[19] Chapra, S. C. and Canale, R. P., *Numerical Methods for Engineers: with Software and Programming Applications*, 4th Edition, McGraw-Hill, 2001.

[20] Prescott, D. R. and Andrews, J. D., *Aircraft Safety Modelling for Time Limited Dispatch*, Proceedings of the Annual Reliability and Maintainability Symposium, [CD-ROM], 2005.

[21] Prescott, D. R. and Andrews, J. D., *A Comparison of Modelling Approaches for the Time Limited Dispatch of Aircraft*, IMechE Part O Journal of Risk and Reliability, Vol. 220, No. 1, June 2006.

[22] Prescott, D. R. and Andrews, J. D., *The Safe Dispatch of Aircraft With Known Faults*, Condition Monitoring and Diagnostic Engineering Management: COMADEM 2006: Proceedings of the 19th International Congress,

[23] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[24] Pattison, R. L., and Andrews, J. D., *Genetic Algorithms in Optimal Safety System Design*, IMechE Part E Journal of Process Mechanical Engineering, Vol. 213, No. 3, 1999.

Appendix 1. User Options File For the TLD MCS Code

User Options File for TLD MCS Code

```
#################################################################
############################## times ##############################
#################################################################
AUTO_SET_TLD_CRITERIA           false                           #
#################################################################
#                                                               #
FLIGHT_LENGTH                   10                              #
SYSTEM_OPERATIONAL_LIFE         130000                          #
#                                                               #
VARY_OPERATIONAL_LIFE           false                           #
LOWER_OPERATIONAL_LIFE          1000                            #
UPPER_OPERATIONAL_LIFE          100000                          #
STEP_OPERATIONAL_LIFE           1000                            #
#                                                               #
#################################################################
##################### maintenance approaches ####################
#                                                               #
STD_MAINTENANCE_APPROACH        mel                             #
LTD_MAINTENANCE_APPROACH        mel                             #
#                                                               #
VARY_MAINTENANCE_APPROACH       false                           #
#                                                               #
#################################################################
####### ratio of inspection interval to dispatch interval #######
#                                                               #
PIR_LOWER                       1.5                             #
PIR_UPPER                       1.75                            #
PIR_STEP                        0.25                            #
#                                                               #
VARY_PIR_RATIO                  false                           #
#                                                               #
#################################################################
################ repair strategies at maintenance ###############
#                                                               #
OPPORTUNISTIC_MAINTENANCE       false                           #
#                                                               #
OPPORTUNISTIC_TIME              250                             #
#                                                               #
REPAIR_STRATEGY                 2                               #
#                                                               #
VARY_REPAIR_STRATEGY            false                           #
#                                                               #
#################################################################
######################## dispatch intervals #####################
#                                                               #
STD_DISP_LOWER                  500                             #
STD_DISP_UPPER                  1000                            #
STD_DISP_STEP                   500                             #
#                                                               #
LTD_DISP_LOWER                  1000                            #
LTD_DISP_UPPER                  2500                            #
LTD_DISP_STEP                   500                             #
#                                                               #
```

User Options File for TLD MCS Code

```
VARY_DISPATCH_INTERVALS          true                                  #
#                                                                      #
SHIFT_DEADLINE_TO_START_OR_END   end                                   #
#                                                                      #
#######################################################################
################# maximum number of simulations #################
#                                                                      #
NUMBER_OF_SIMULATIONS            100000000                             #
#                                                                      #
#######################################################################
###################### convergence options ######################
#                                                                      #
FIRST_CHECK                      1000000                               #
CHECK_EVERY                      50000                                 #
SIG_FIG                          3                                     #
CONSEC_VALS                      3                                     #
MINIMUM_FAILURES                 1000                                  #
#                                                                      #
#######################################################################
###################### generation of output ######################
#                                                                      #
PRINT                            false                                 #
PRINT_EVERY                      10                                    #
#                                                                      #
#######################################################################
################ instantaneous LOTC rate options ################
#                                                                      #
CHECK_INSTANTANEOUS_LOTC_RATE    false                                 #
UNORDERED                        true                                  #
#                                                                      #
#######################################################################
################### importance measure option ###################
#                                                                      #
CALCULATE_IMPORTANCE_MEASURES    false                                 #
#                                                                      #
#######################################################################
#################### random number generator ####################
#                                                                      #
SEED_AT_RANDOM                   true                                  #
#                                                                      #
#######################################################################
#### system non-dispatchable with more than n faults present ####
#                                                                      #
DND_above                        3                                     #
#                                                                      #
#######################################################################
# test the instantaneous LOTC rates for faults up to this order #
#                                                                      #
TEST_ORDER                       2                                     #
#                                                                      #
#######################################################################
# calculate the full-up instantaneous LOTC rate for the system  #
#                                                                      #
CALC_FU_RATE                     false                                 #
#                                                                      #
#######################################################################
# file names / folders
#
TREEFILE        C:\path\system_fault_tree.txt
F_R_FILE        C:\path\system_failure_rates.txt
```

User Options File for TLD MCS Code	
TLD_FILE	C:\path\system_initial_tld_criteria.txt
NEW_TLD_FILE	C:\path\system_updated_tld_criteria
FAULT_LIST	C:\path\system_fault_list.txt
RELATED_REPAIRS	C:\path\system_related_repairs.txt
OUTPUT_FOLDER	C:\path\results
#	

Appendix 2. TLD MCS Code Details

A2.1 Introduction

In Chapter 6 an outline of the methodology developed for the MCS modelling of TLD is presented. Appendix 2 looks at the options that are provided to the TLD MCS code in an input data file and the main modules of the TLD MCS code and the tasks that they perform.

A2.2 Simulation Options

The user of the TLD MCS code must specify a number of criteria prior to using the simulation code. These criteria are used to determine the way that the code is implemented lengths of dispatch and inspection intervals, and to how many significant figures the LOTC rates should be obtained. The text file also contains information that 'turns on' or 'turns off' certain features within the code. This information is provided to the code in the form of a text file the name of which is passed to the TLD MCS code by the user. An example of such a text file is given in Appendix 1. An overview of all of the options and of their effects on the ensuing simulations is given here, along with a brief description of other information required by the code.

A2.2.1 Specific Monte Carlo Simulation Options

The first of these is an option that determines whether or not the random number generator used should be seeded with a random number or not. If the random number generator is started from the same point at the beginning of different code implementations it allows the exact reproduction of a set of results. If the random number generator is to be seeded with a random value this random value is generated using the current time obtained from the computer.

The second of these options specifies the maximum number of simulations that are to be carried out when computing the appropriate LOTC rates (system LOTC rate and, in

some cases, instantaneous LOTC rates from dispatchable system configurations) and importance measures. This is used so that, should all of these values not have converged after this number of simulations, the simulations will stop.

A2.2.2 Flight Length Option

The length of a flight (in hours) is specified and this is used throughout all of the simulations. The flight length cannot be varied while simulations are in progress. Thus, throughout the course of the simulations the flights are of constant length. This is considered to be a reasonable assumption.

A2.2.3 Operating Life Options

There are a number of options available regarding the operating life of the system. The operating life is defined as the length of time that the system will be in service on an aircraft, or the length of time that the system will be in service before being returned to a guaranteed fully-up state, if it is possible to do so. The operating life is entered in flight hours. Each individual simulation will model the system throughout the operating life. There is also an option to obtain results for different lengths of operating life.

In the case where the operating life of the system is not varied one value is specified, which will be used in all of the simulations, as was the case with the flight length.

When the operating life of the system is to be varied a lower operating life, an upper operating life and a step length are specified. Simulations are implemented and a LOTC rate calculated for all operating lives specified. This option was included to investigate how the LOTC rate of the system varies with differing operating lives.

A2.2.4 Repair Strategy Options

The repair strategy to be employed at maintenance deadlines can be specified. Any one of four strategies may be specified or the results can be produced for all four of the strategies. The four strategies represent different levels of maintenance being applied to the system at maintenance deadlines, ranging from the very least amount of work that

may be carried out to allow further dispatch to the maximum amount of work that can be undertaken. The four different strategies are as follows:

1. Repair the last fault of the group that caused the current maintenance deadline,
2. Repair all faults in the group causing the current maintenance deadline,
3. Repair all faults in the same category as the group causing the current deadline,
4. Repair all faults present in the system.

There is one further option that may be specified, called opportunistic maintenance. If this is chosen to be implemented then the user must specify a time in hours that shall be called here the opportunistic maintenance window. If, at a maintenance deadline, any further maintenance deadlines are due to take place within this opportunistic maintenance window then the repairs that would take place at these deadlines will also be carried out. The four repair strategies listed above and opportunistic maintenance are described in more detail in Section 6.7.

A2.2.5 Maintenance Approach Options

These options specify the maintenance approaches to be used at STD and LTD deadlines or, alternatively, specify that simulations should be implemented for all possible combinations of the maintenance approaches.

Three options of maintenance approaches are considered. Both STD and LTD faults can be addressed using the MEL maintenance approach, both can be addressed using the PIR maintenance approach or, finally, STD faults can be addressed using the MEL approach and LTD faults addressed using the PIR approach. The final combination, where STD faults could be maintained using the PIR approach and LTD using the MEL approach was considered to be unrealistic. However, it is possible to specify *any* of the maintenance approaches for STD and LTD faults.

Note that the maintenance approaches cannot be varied when varying the operating life of the system being modelled. This restriction is made purely to allow manageable output of results.

A2.2.6 Dispatch Interval Options

This option determines for which value/s of dispatch interval results must be obtained. As was the case with the operating life of the system, the dispatch intervals may be set to a specific value and LOTC rates obtained from the TLD MCS code. Alternatively, as was also the case with the operating life, results may be obtained over a range of values for each of the STD and LTD intervals at a given step length. The dispatch intervals that are specified by the user are given in flight hours. An option also exists to specify that, should a maintenance deadline be due to fall mid-flight, it can be shifted to either the beginning or the end of the flight in which it is due to fall. This is explained further in Sections 6.8.2 and 6.8.3.

The dispatch intervals cannot be varied when varying the system operating life. This enables manageable output of results.

A2.2.7 Periodic Inspection Interval Options

If PIR maintenance is to be used during the simulations performed by the TLD MCS code the length of the inspection interval can be specified. Alternatively the user may specify a range of values of the inspection interval that are to be used to obtain results for the system. In this case, however, the inspection intervals are not given in terms of flight hours but by defining the ratio that the inspection interval will take to the dispatch interval. For example, the inspection intervals could be set at 1.5 times the size of the dispatch intervals used in the simulations. When a range of values are specified in the input file a step length is again supplied to increment between the lower and upper value.

When varying the operating life of the system the user may not specify that the inspection intervals are to vary. This is, again, to allow manageable output of results when the operating life is varied.

A2.2.8 Dispatchable Configuration Options

Options in this category specify whether the LOTC rates from individual system dispatchable configurations or importance measures are to be calculated. Basic event importance measures would be calculated in this case, along with importance measures for the system's dispatchable configurations. If LOTC rates and importance measures are to be calculated for the dispatchable configurations then it must be specified how the values will be calculated – for ordered combinations of faults or for unordered combinations of faults. If the ordering of faults is specified to be important the times spent dispatching from the dispatchable system configurations and the number of failures from those configurations is calculated for specific fault orders. If not the times and number of failures are calculated given that the faults are present in the system. This is perhaps easiest to explain given an example. Consider two faults, A and B, that may occur within a system and that instantaneous failure rates to LOTC must be calculated for this system. If the ordering of the faults is specified to be of importance LOTC rates will be calculated for the following dispatchable configurations; A, B, AB and BA. If the ordering of faults is not specified as important then the LOTC rates will be calculated for the following dispatchable system configurations; A, B and AB, where, in this case, AB describes the system configuration where A and B are present in the system in either order AB or BA.

This option was included in the TLD MCS code to optimise, when appropriate (i.e. when fault ordering is unimportant), the performance of the code when the instantaneous LOTC rates from the dispatchable system configurations and importance measures are to be calculated. This optimisation occurs because of the large reduction in the amount of bookkeeping that occurs during the implementation of simulations.

Also, when fault ordering is taken into account the time spent dispatching from certain system configurations and the number of failures from those configurations can require a large number of simulations to reach levels where convergence can be verified. This is particularly true for rare configurations and higher order fault combinations. When fault ordering is not taken into account the number of simulations required before convergence is achieved is lower.

A2.2.9 Convergence Options

When obtaining results for the LOTC rate of the system a value for this rate is calculated after every x simulations. If y consecutive values match to z significant figures the LOTC rate is deemed to have converged. The values of x, y and z may be specified along with after how many simulations these checks must begin. This allows for some settling of the LOTC rate before the values are to be calculated and compared. This is of particular use when calculating the instantaneous LOTC rates from the dispatchable system configurations in addition to the system LOTC rate, since the calculation and comparison of a large number of LOTC rates could take a significant processing effort. If the rates are unlikely to have converged there is little point in calculating them every x simulations. These values of x, y and z are also used when calculating importance measures.

A2.2.10 Output Options

This final option specified decides whether or not an output file should be produced showing the processes involved in applying the maintenance procedures specified by the user to a series of n individual simulations, where n may be specified by the user. This file can then be checked to see an example of how TLD will be applied to a system during its lifetime. It will show in chronological order, for each of the n simulations, times of fault occurrences and any subsequent TLD deadlines that are applied, what faults will be repaired at maintenance deadlines and whether the system fails before the maximum operating life is reached. This option was of particular use when debugging the code.

A2.2.11 Other Information Required

The only other information that must be supplied to the TLD MCS code relates to the files containing information about the system to be modelled. There are five that are required. Examples of such files are given in Appendix 1 and Appendix 5. The first file required contains a text representation of the system fault tree, in terms of a list of the gate names in the tree, along with their type and their inputs. The second file provided is also a text file, and lists all of the basic events in the fault tree, along with their failure

distributions and associated parameters. The TLD MCS code accepts the normal, exponential and Weibull failure time distributions, but could easily be adapted to accept other distributions if necessary. The third file required lists all of the basic or intermediate events from the system fault tree that will be considered as the faults to be included in a TLD analysis of the system. These are the faults that appear in the dispatch criteria. This is of particular importance when the code automatically sets dispatch criteria for the system but is also useful as it provides a means of verifying the faults that have been included in the dispatch criteria file. This dispatch criteria file is another text file that contains a complete list of the dispatch criteria that will be applied to the system. This must be given in terms of ordered combinations of the basic events from the fault tree along with an identifier (DND, STD or LTD) to define the TLD category that will be applied to the system when that combination is present. If the the dispatch criteria must be set automatically by the code then it is not necessary for a complete list of dispatch criteria to be supplied. In this situation any dispatch criteria for faults/fault combinations supplied in the text file will be held constant (or reduced if necessary) while dispatch criteria for other faults/fault combinations may be amended by the code. See Section A2.3.13 for more detail.

The final text file supplied to the code contains a list of faults (basic and intermediate events from the system fault tree) that must be repaired as particular faults within the dispatch criteria are repaired. Here, they will be called 'associated' faults. For instance, it may be the case that, when a certain fault from the dispatch criteria is cleared from the system, another fault from another part of the fault tree should also be repaired. This process is explained further in Section 6.7.6. The process itself will account for some of the dependencies that may be present within the system due to the maintenance procedures (the same dependencies that make the application of conventional Fault Tree analysis to the TLD problem unsuitable).

A number of faults above which the dispatch category applied to the system will be DND can be specified. This represents the assumption that, once the system has degraded to such an extent that there are a certain number of TLD faults present, further dispatch will be forbidden. If there are more faults than the number specified as the maximum allowed when maintenance takes place the system will be returned to an 'as-new' state.

In addition to the aforementioned text files, a folder to which files containing the results and output from the simulations will be sent and a filename to which the applied dispatch criteria will be sent when simulations are performed for a system are specified. This is of particular importance when the code is used to automatically set the dispatch criteria and these criteria are changed by the code. Before the dispatch criteria are output to file a numerical identifier is appended to the filename in order to relate it to the change of dispatch criteria. A similar process occurs for the other results output from the code, see Section 6.9.

A2.3 General Descriptions of TLD MCS Code Modules

A2.3.1 The 'main' Module

This is the main driver program used in the TLD MCS code. A general algorithm is shown in Figure A1. Prompts are issued for the name of the text file containing the options that will dictate how the simulations are carried out. Checks are made at this point on the validity of the data supplied. The system fault tree is then input, along with the failure time distributions for basic events. The system fault tree structure and failure data are stored in dynamically allocated arrays within the TLD MCS code. This allows memory resources to be used efficiently and also allows large structures to be defined (relating to the large number of basic events that could potentially be involved). An initialisation procedure is then carried out to the relevant arrays and structures at this starting point. What happens next depends on whether the dispatch criteria for the system are to be automatically set by the code or not. If they are to be set automatically then the *auto_set_tld_criteria* module is called to perform this task, from which *control_sims* is ultimately called to perform the necessary simulations. If not then the dispatch criteria supplied are input from file and stored in dynamically allocated arrays. Storing these dispatch criteria in an efficient manner is one of the crucial parts of the TLD MCS code. As a fault occurs within the system it must be possible to check the dispatch criteria in an efficient manner. The method used to store the dispatch criteria and the reasons for this particular choice of method are described in Section 6.5. If the instantaneous failure rates to LOTC are to be found for the system then *dispatch_configuration* is called in order to identify the dispatchable system

configurations and allocate memory for the structures used to store data for these configurations. At this stage *control_sims* is called to handle the necessary simulations.

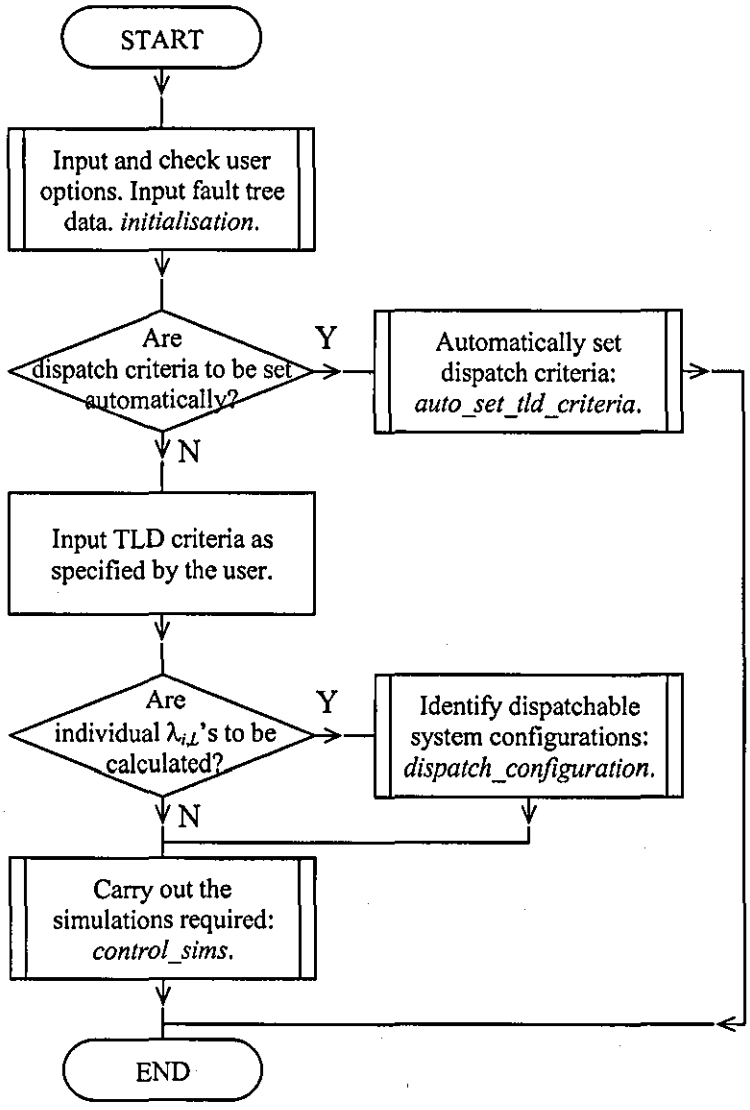


Figure A1. A general algorithm for the *main* module of the TLD MCS code.

A2.3.2 The 'control_sims' Module

This module of the code ensures that the simulations are carried out for the correct values or sets of values of the variables used within the code. The *vary_user_options* module is chosen to control the values of relevant variables. For instance, if STD intervals have been specified by the user to vary between 50 and 200 in steps of 50, this module will set the STD interval to 50, 100, 150 and 200 and ensure simulations are carried out using these values. The *simulations* module is called to carry out the

simulations with variables specified according to the user options by the *control_sims* module.

A2.3.3 The 'dispatch_configuration' Module

This module is used by the TLD MCS code only in the case when the user requires the LOTC rates, $\lambda_{i,L}$, from the dispatchable system configurations, i, to be calculated. This is one of the options provided in the option data file supplied to the code. The role of the *dispatch_configuration* module is to find which system configurations are actually dispatchable. The general algorithm is outlined in Figure A2.

The first step is to recognise that the full-up dispatchable configuration (i.e. that with no faults present) is dispatchable. The relevant structures are updated to hold the data for this dispatchable configuration. Next the module goes through a looping process, constructing system configurations (basically groups of faults) and testing to see whether these configurations are dispatchable. First of all the module checks for system failure with these faults present. If the system doesn't fail the TLD criteria being applied to the system are checked to see if this configuration is dispatchable. In the case that the group of faults does not cause system failure *and* is not a DND category fault group this particular fault configuration is dispatchable. In this case the relevant structures are updated to hold the data for this configuration. Whether or not the system configuration just tested is dispatchable, the module then generates another system configuration, if any exist, and tests its dispatchability. Note that the dispatchable system configurations can be ordered or unordered, as specified in the options data file (see Section A2.2.8).

Amongst the data that is stored in the structures made by this module for each dispatchable configuration is the total time spent in that dispatchable configuration and the total number of system failures when dispatching from that configuration. These are used to calculate the instantaneous LOTC rates from the dispatchable configurations. The remaining data structures made by this module for each dispatchable configuration are used to check for convergence of the LOTC rates calculated from each configuration. Similar data is stored for the dispatchable configurations for the

calculation of the importance measures that relate to the dispatchable faults. The calculation of importance measures is also explained in Section 6.9.1.3. The processes of calculating the LOTC rates and importance measures are explained in Section 6.9.1 and checking for convergence is described in Section A2.3.12.

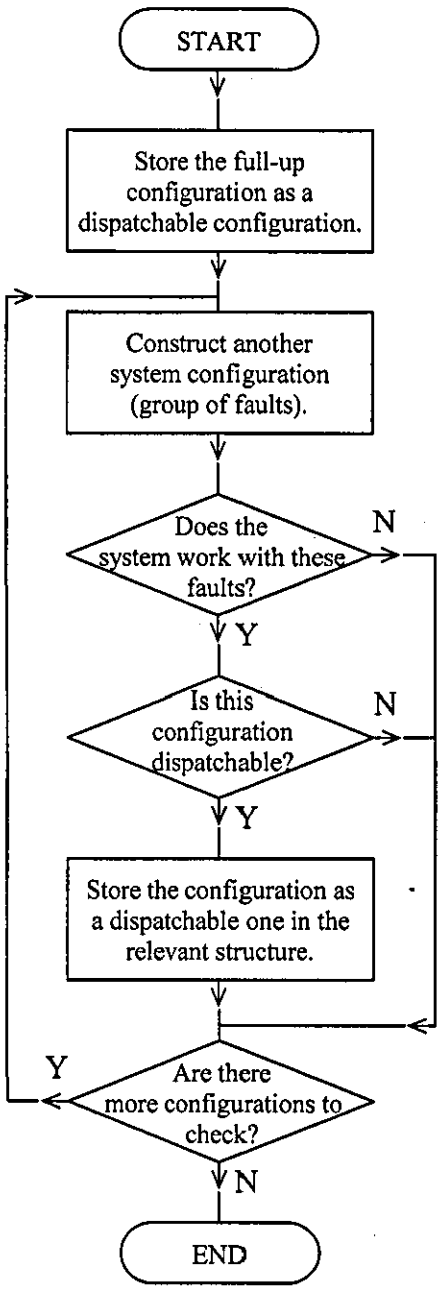


Figure A2. A general algorithm for the *dispatch_configuration* module of the TLD MCS code.

A2.3.4 The 'vary_user_options' Module

This module is used to vary parameters used by the TLD MCS code, if required. These variations, specified in the user options file, are discussed in Section A2.2 and include quantities such as the STD and LTD intervals, the periodic inspection intervals, the maintenance approaches and repair strategies to be applied and the operating life of the system. The *vary_user_options* module allocates values to the appropriate variables within the TLD MCS code to be passed to the *simulations* module, where a group of simulations is implemented using these variables. The *simulations* module is described in Section A2.3.6, where the term 'group of simulations' is explained.

A2.3.5 The 'initialisation' Modules

These modules are used to initialise data structures at various points in the TLD MCS code. An example of one of these initialisation modules is the process after a single simulation, when the system is returned to an 'as new' state. Another example of an initialisation process occurs after the system LOTC rate has been found for certain lengths of STD and LTD intervals. If the system LOTC rate is to be found for other STD and LTD intervals it (and the values used to calculate it) must be reset to zero before carrying out further simulations.

A2.3.6 The 'simulations' Module

This algorithm is used by the TLD MCS code to implement a group of simulations. The group of simulations will be carried out for certain values of STD and LTD intervals, and for specific values of other quantities that may be varied, such as the maintenance approaches, repair strategies or operating life of the system. These latter quantities are set to vary in the user options file. Thus the simulations module implements a number of simulations (referred to here as a group). This number is not predefined, simulations are implemented until the required data (LOTC rate/s and importance measures) has converged to a specified accuracy (described in Section A2.2). There is one exception to this. This exception occurs when the data does not converge within a specified number of simulations. If this number of simulations is exceeded the group of simulations is stopped. This provides a method of halting simulations in a controlled

manner if convergence is not achieved within a certain period. Note that the *simulations* algorithm shown in Figure A3 does not include this test, which is a trivial addition to the algorithm. The algorithm depicted also does not include the initialisations that occur before the group of simulations has begun.

The initial task in the *simulations* algorithm is to zero the time (in flight hours) elapsed and initialise variables for the individual simulation (i.e. one simulation of the lifetime of the system). During this initialisation of variables the initial time of each fault within the system will be produced (using the relevant failure distributions) and added to the schedule that is used to deal with the events that occur during the system lifetime. This schedule is described in more detail in Section 6.4.

The individual simulation begins by advancing the simulation time (i.e. time the system has been in service) to the time of the first fault, chronologically, from the schedule. If the fault time is less than the system lifetime the fault is propagated up through the system fault tree using the *failure_update* module. If the system doesn't fail the TLD criteria for the system are checked using the *TLD_check* module to see if any TLD maintenance deadlines should be added to the schedule. The simulation then continues.

The same procedure is carried out for any further faults. If a maintenance deadline is removed from the schedule the *TLD_maintenance* module is called to clear the appropriate faults from the system (according to the user options regarding maintenance). This *TLD_maintenance* module calls the *repair_update* module to propagate repairs through the fault tree as required. If opportunistic maintenance is to be implemented then any maintenance operations scheduled to occur within the opportunistic maintenance window of the current time will be carried out now also, see Section 6.7.5. Any remaining future scheduled maintenance deadlines are checked to see if they will still occur in view of the repairs that have just been carried out, as described in Section 6.8.8. When system failure occurs in a simulation all faults are cleared from the system at the end of the flight in which the failure occurs, before simulations continue in the same manner. This models the situation that would most likely occur in a real-life scenario, wherein the FADEC unit of an engine on a multi-engined aircraft would be replaced or repaired before the next time the aircraft was used.

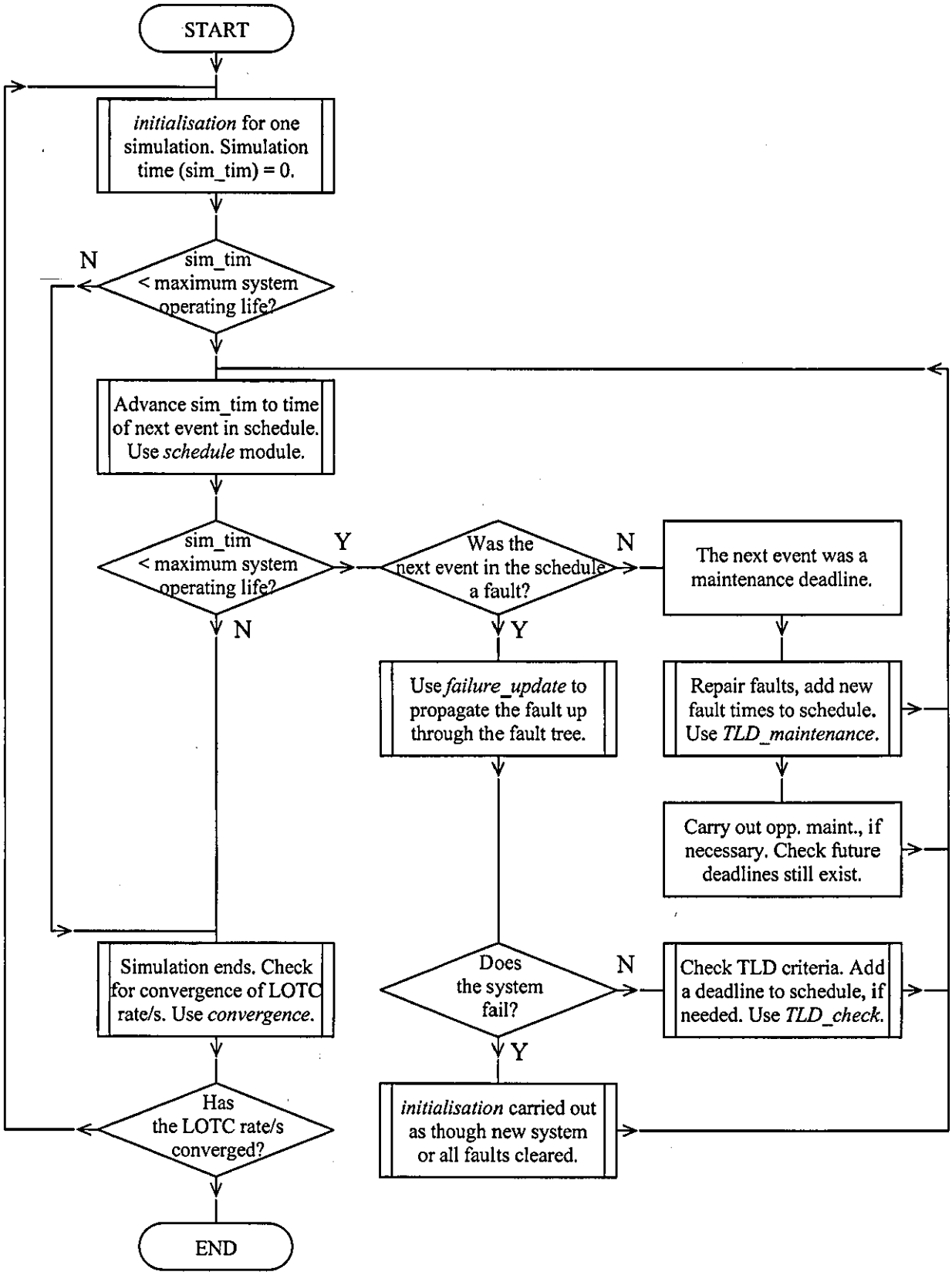


Figure A3. A general algorithm for the *simulations* module of the TLD MCS code.

The simulation continues with faults occurring, maintenance deadlines being initiated and faults being repaired until the operating life of the system is reached. At this point the *simulations* algorithm shown in Figure A3 is again simplified somewhat. The algorithm states that the *convergence* module is used to verify whether or not the required LOTC rates have converged (system LOTC rate and perhaps instantaneous LOTC rates from dispatchable system configurations). Also at this point the importance measures would be checked for convergence, if they are to be calculated. In reality what happens within the TLD MCS code, rather than checking for convergence after every simulation, is that a specified number of simulations (specified in the user options file) will be carried out before the first calculation of the LOTC rates and importance measures, after which tests of these values are performed every x simulations (where x is also specified in the user options file, see Section A2.2.9). The calculation of the LOTC rates and importance measures is described later in Section 6.9.1.

Simulation after simulation within this group is carried out in this manner. Once all of the required LOTC rates (system and perhaps dispatchable configurations) and importance measures have converged the group of simulations ends and the *simulations* module is exited, control returning to the *control_sims* module.

A2.3.7 The 'failure_update' Module

The *failure_update* module is called from the *simulations* module. Its role is to update the status of the relevant basic events and gates within the fault tree as faults occur during the system simulations. It propagates faults up through the fault tree structure by recursive calls to itself. This will be explained shortly. An algorithm for the *failure_update* module is shown in Figure A4. There is one important input to the module, which is shown in the algorithm diagram as **input**.

Upon the initial call to *failure_update* from *simulations* **input** will refer to the basic event within the system fault tree that corresponds to the fault that has just occurred. Firstly, **input**'s status is updated to failed. Then the failure time of **input** is logged – this will be of importance if a parent gate of **input** is a priority AND gate (to determine basic event ordering) or if the instantaneous LOTC rates from dispatchable system configurations and importance measures are to be calculated (See Section A2.3.11). The

failure_update function then loops through the parent gates of **input**, checking to see if any of them fail due to the failure of **input**. If the parent gate fails it is then passed to the *failure_update* module, where the gate's status is updated to failed and then it's own parent gates are checked to see if they have failed and the process repeats itself. This continues until all of the parent gates' statuses have been updated accordingly. If the parent gate that fails is that related to the fault tree top event the system fails and the *failure_update* module ends, passing control back through all the calls to *failure_update* until control once again returns to the *simulations* module.

The *failure_update* module has to be able check the five different types of gate allowed as inputs to the TLD MCS code. These are OR gates, AND gates, priority AND gates, *k/n* voting gates and NOT gates. The process of checking whether these gate types have failed is straightforward (see Section 2.2). OR gates fail if any one of their inputs fail, AND gates fail if all of their inputs fail, priority AND gates fail if all of their inputs fail in the required order, *k/n* voting gates fail if *k* of the *n* inputs to the gate fail and NOT gates fail if their single input is not failed. Note that, for clarity, the situation that arises if the input to a NOT gate fails is not shown in Figure A4. Since the failure of an input to a NOT gate means the NOT gate takes the opposite status, the NOT gate is effectively repaired if its input fails. For this reason, if the input to a NOT gate fails the *repair_update* module is called for the NOT gate in order to propagate the appropriate logic through the fault tree structure.

Note that null gates are also allowed in the code. These gates do not alter the flow of logic through the fault tree at all and might be included in the fault tree purely to assist in expressing the logic development in the fault tree. Null gates will have only one input and will exhibit identical failure logic to that input.

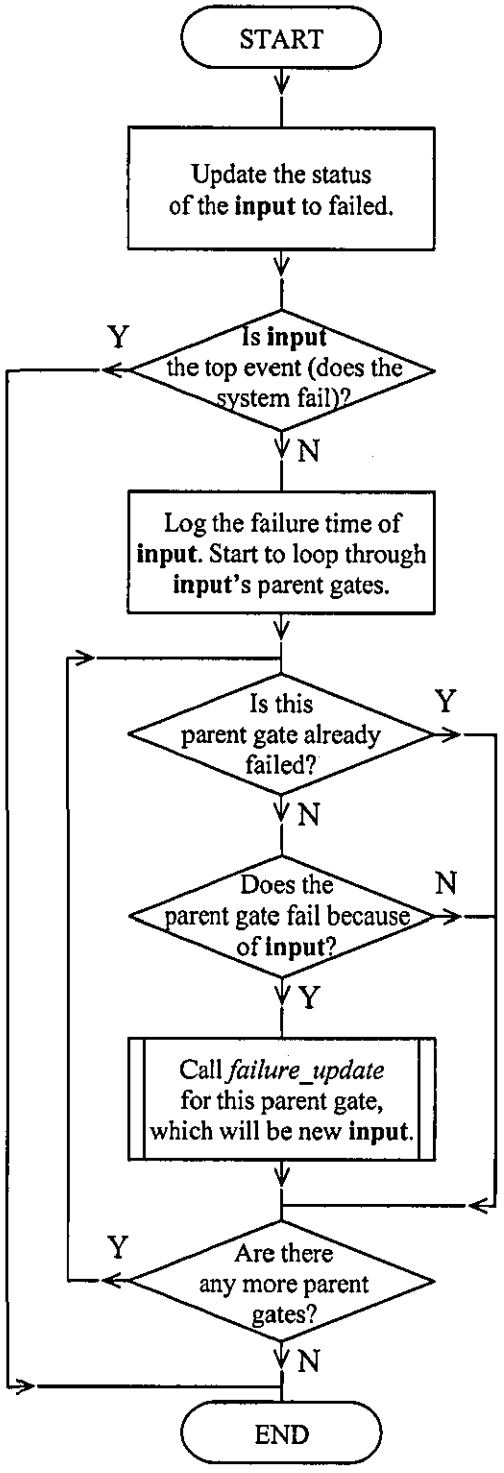


Figure A4. A general algorithm for the *failure_update* module of the TLD MCS code.

A2.3.8 The 'TLD_check' Module

This module is called from the *simulations* module of the TLD MCS code after a fault has occurred during a system simulation. It is used to check the faults present in the

system to see whether, given the order in which the faults occurred, a TLD maintenance deadline should be added to the schedule. The way in which the module works varies depending on the maintenance approach being applied to the system and the repair strategy that is being used at maintenance deadlines. However, the general function of the module is very similar no matter what the maintenance approach and repair strategy being used. This general function of the module is shown in the algorithm given in Figure A5.

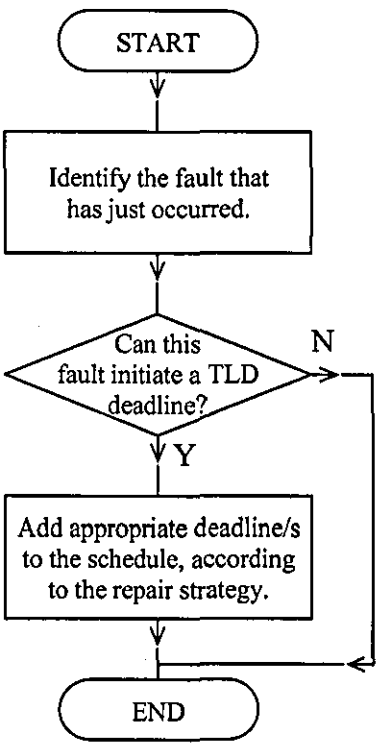


Figure A5. A general algorithm for the *TLD_check* module of the TLD MCS code.

After a fault has been occurred in the *simulations* module of the code it is passed to the *TLD_check* module. The TLD criteria are then checked for this fault (see Section 6.5 to see how this is done) to see if the system may be dispatched with this fault present. As these TLD criteria are checked the relevant maintenance deadlines are added to the schedule of events to occur. Which maintenance deadlines are added to the schedule and the times at which these deadlines will occur is decided by the maintenance approach and repair strategy being used to repair faults at the deadlines. The repair

strategies were described in Sections 6.7 and 6.8, along with how they affect the addition of maintenance deadlines to the schedule as faults occur. Also, depending on the maintenance approach and the repair strategy being employed at maintenance, the faults that must be repaired at the maintenance deadlines are also recorded. This results in maintenance operations (within the TLD MCS code at least) that are well organised, with all relevant faults being repaired as and when is necessary.

As an illustration of how the *TLD_check* module works, consider for now that all faults present will be cleared from the system at maintenance deadlines. If a fault occurs one need only add to the schedule the maintenance deadline associated with this fault that will occur soonest, since all faults will be cleared at this deadline.

A2.3.9 The 'TLD_maintenance' Module

This module of the TLD MCS code is called from the *simulations* module when a TLD maintenance deadline has occurred (i.e. has been removed from the schedule). It is used to implement the correct maintenance at TLD deadlines based on the maintenance approach that is being applied and the repair strategy that is being used at maintenance deadlines. As was the case with the *TLD_check* module, the details of how faults are repaired differs according to the maintenance approach and repair strategy. See Sections 6.7 and 6.8 for details.

The *TLD_maintenance* module uses the *component_repair* module to update the status of faults within the system and generate a new time at which the fault will occur. The *component_repair* module in turn calls the *repair_update* module to propagate fault repairs through the system fault tree as necessary, see Section A2.3.10.

A2.3.10 The 'repair_update' Module

This module is called from the *component_repair* module of the TLD MCS code (which is, in turn, called from the *TLD_maintenance* module). It is used, in a similar manner to the *failure_update* module, to propagate fault repairs through the fault tree structure using recursive calls to itself. The algorithm used in the *repair_update* module is shown in Figure A6.

The initial call to the *repair_update* module is made from the *component_repair* module after a fault has been repaired. The parent gates of the repaired fault are passed in turn to the *repair_update* module. The *repair_update* module then checks to see if the parent gate, shown in as **gate**, is failed. If **gate** is not failed there is no more work to be carried out and the module is exited. This is because the repair of a gate's input will have no effect on the status of the gate if the gate works anyway.

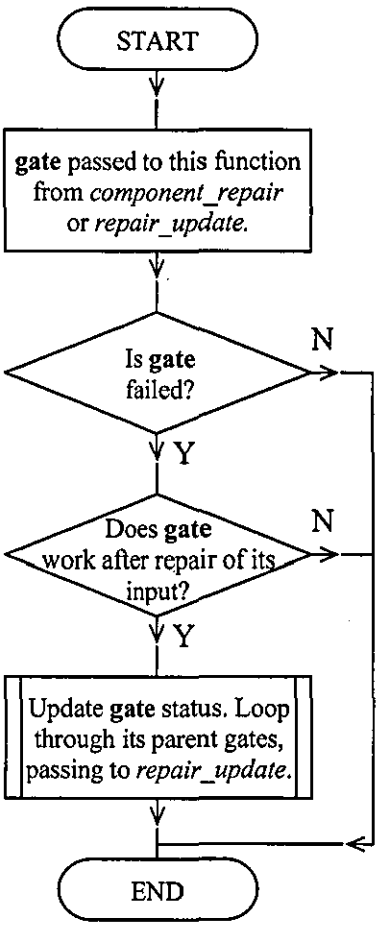


Figure A6. A general algorithm for the *repair_update* module of the TLD MCS code.

However, if **gate** is failed (i.e. *was* failed before the repair of its input) the status of **gate** is checked to see if the repair of its input has caused it to work again. If this is the case the *repair_update* module loops through the parent gates of **gate** and passes them to the *repair_update* module so that their status can be updated if necessary. The process just described continues until the statuses of all parent gates related to the initial fault that was repaired have been checked and updated where appropriate.

Note that, as was the case with the *failure_update* module, the algorithm shown for *repair_update* has been simplified by disregarding the case when the parent gate is a NOT gate. In this case, the repair of the input requires that the NOT gate will then fail, thus *failure_update* would be called to propagate this failure logic through the fault tree structure.

A2.3.11 The 'LOTC_i' Module

The *LOTC_i* module is used to update and amend the data that is used to calculate instantaneous LOTC rates from the dispatchable system configurations and is also used to calculate importance measures. It is called, when the user options file has indicated that LOTC rates and importance measures must be calculated, from four positions within the *simulations* module. These are:

1. *Before* maintenance takes place at a deadline,
2. *After* maintenance has taken place at a deadline,
3. When the system maximum lifetime (as specified by the user) has been reached in an individual simulation,
4. When the system fails during an individual simulation.

The reason for the calls at these times will be explained in more detail in Section 6.9.1.2, which details the methods used to obtain the LOTC rates from the system dispatchable configurations and the importance measures. In calculating these values one must calculate the time spent dispatching from each of the dispatchable system configurations and the number of system failures from these dispatchable system configurations. This is also explained in Section 6.9.1.2. The *LOTC_i* module updates these times and failure counts as required throughout the running of the TLD MCS code.

A2.3.12 The 'convergence' Modules

The *convergence* modules, LOTC_convergence and IMP_convergence, are used to check for convergence of the LOTC rate/s and importance measures that the user wishes to be output from the TLD MCS code. The modules are called from the

simulations module of the code after the completion of a certain number of individual simulations. There are a number of user options that decide exactly when the *convergence* modules are called, see Section A2.2.9. The first checks for convergence of the relevant LOTC rate/s and importance measures are conducted after a number of simulations specified in the user options file, then subsequent checks for convergence are conducted every time a further number of simulations (again specified in the options text file) are completed. Each of these checks involves a separate call to the *convergence* modules.

The algorithm used in the *LOTC_convergence* module is depicted in Figure A7 and the algorithm for the *IMP_convergence* module is similar to this, except for the fact that more than one set of importance measures is calculated, see Section 6.9.1.3. The following discussion is relevant to the *LOTC_convergence* module although the same basic procedure takes place for the importance measures in the *IMP_convergence* module. The *LOTC_convergence* module works in the same way for each LOTC rate to be calculated (e.g. system LOTC rate or instantaneous LOTC rate from a certain dispatchable system configuration). When called it works through all of the LOTC rates to be calculated. If the rate has not already converged, it calculates the current value of the LOTC rate and compares it to the previous value of the LOTC rate calculated. If the two values are equal (to the number of significant figures specified by the user) a counter is incremented for this LOTC rate. This counter is reset to zero if the values do not match. Once the counter has reached a total specified by the user the LOTC rate is deemed to have converged. If a specific LOTC rate converges it is not calculated again at the time of the next call to LOTC_*convergence* since there is no need to check for convergence again.

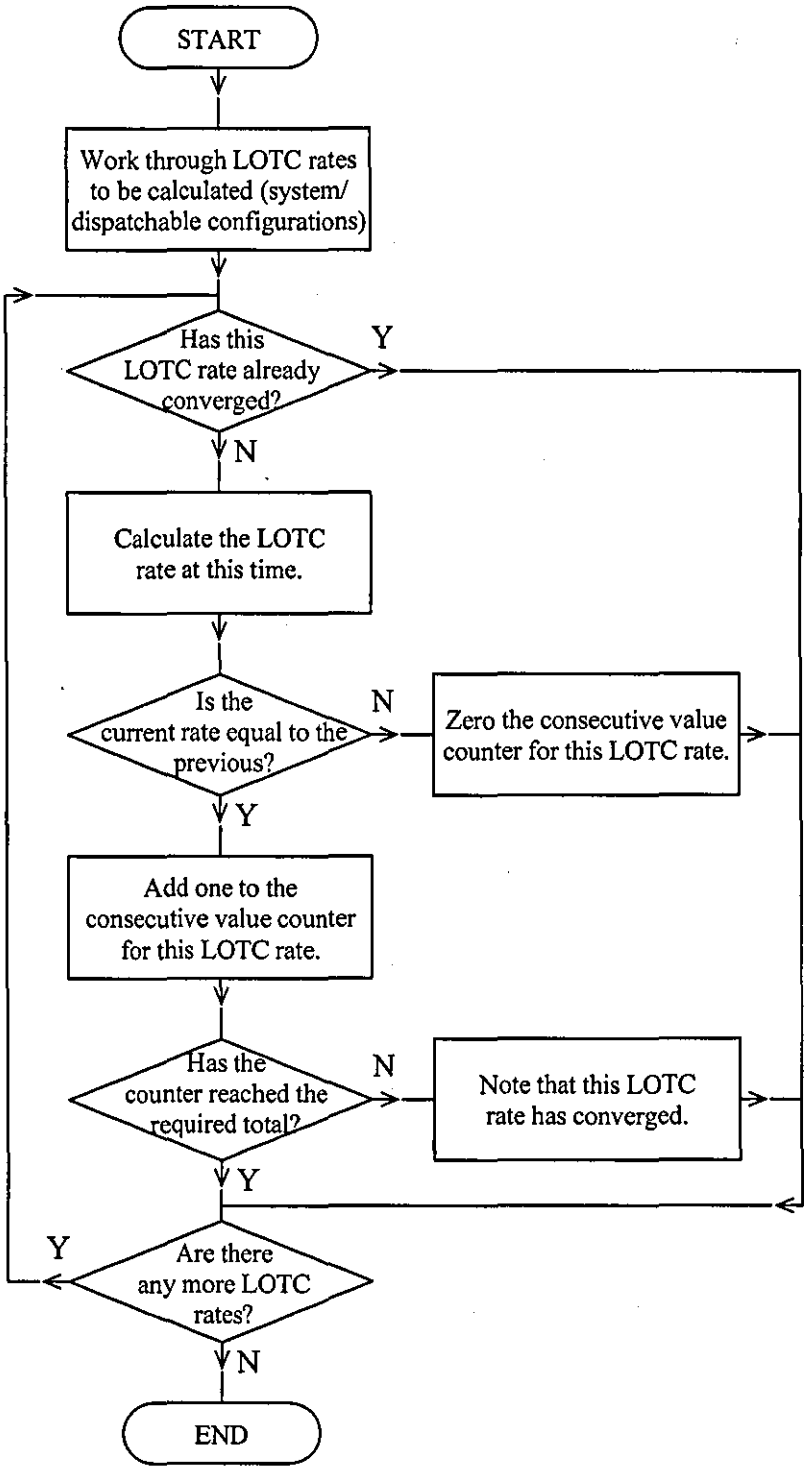


Figure A7. A general algorithm for the *LOTC_convergence* module of the TLD MCS code.

A2.3.13 The 'auto_set_tld_criteria' Module

This module is called by the code when the dispatch criteria for a system must be set automatically by the code. The call to this module comes from the *main* module. In

setting the dispatch criteria an iterative technique is used to build up a set of dispatch criteria for the system up to an order of faults specified in the input data. The instantaneous LOTC rates of the system from the various system configurations are checked against the allowed limits for the different categories of faults. If the instantaneous failure rate to LOTC for any of the faults doesn't agree with the TLD category to which that fault has been assigned then the category for that fault must be adjusted accordingly. A general algorithm for the *auto_set_tld_criteria* module is given in Figure A8.

When it has been selected that the code will automatically set dispatch criteria for a system it is still possible to specify certain dispatch criteria. For instance, there may be certain faults that could occur within the system that must be specified as DND, perhaps because of the nature of the faults or previous experience of this type of faults on previous applications. For this reason a list of TLD criteria can be specified. This list of criteria will then be used by the code to provide a maximum allowed dispatch category for each of the faults and fault combinations in the list. For example, consider a case where it is specified in the input file for the TLD criteria that a fault A should initially be assigned to the STD category. The code will assign that category to fault A as required. Then, as the TLD category for fault A is checked against its instantaneous failure rate to LOTC, the TLD category could be reduced to a DND category but would never be extended to a LTD category, even if the instantaneous failure rate to LOTC for fault A would allow this. This gives some degree of control over the dispatch criteria despite the fact that they are to be set automatically. The first step in this module is to allocate an initial set of dispatch criteria for the system. The faults are provided to the code in the text file that lists all of the faults to be included in the TLD analysis of the system. If some of the dispatch criteria are specified, as described above, they will be allocated to the relevant faults. Any faults for which no dispatch criteria are allocated (this could be all of the faults, if required) are assigned to the LTD category. The dispatch category can then be reassigned and further restricted as necessary as instantaneous failure rates to LOTC are calculated.

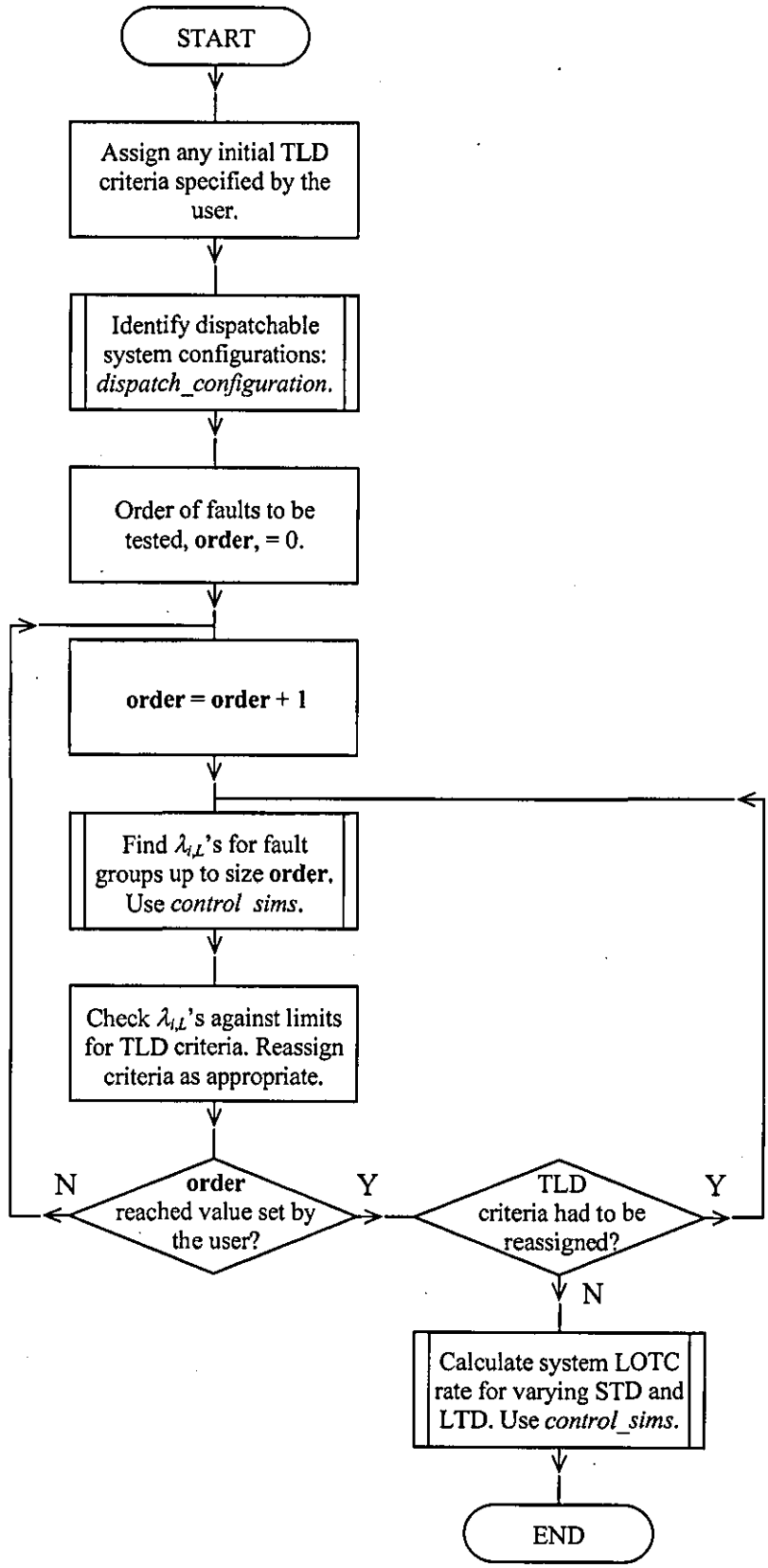


Figure A8. A general algorithm for the *auto_set_tld_criteria* module of the TLD MCS code.

After a set of initial dispatch criteria have been allocated for the system the *dispatch_configuration* module is called to identify the dispatchable system configurations. Once this action is completed the iterative process used to set the dispatch criteria for the system is started. As is the case when it is required to calculate instantaneous failure rates to LOTC when supplying a full set of dispatch criteria for the system, a maximum fault order is specified above which the instantaneous failure rates to LOTC of the system will not be calculated. It is possible to specify that all faults above a certain order will be categorised as DND faults.

It is possible, while TLD criteria are assigned using this iterative process, for the TLD criteria for certain faults or fault combinations to be on the border between fault categories. For instance, a dual fault combination have an instantaneous failure rate to LOTC that is very close to the limit for LTD category faults and STD category faults. In this case the automatic setting of TLD criteria within the code might cycle between applying a STD category to the fault combination and a LTD category to the fault combination. As an example, imagine that the faults are categorised initially as LTD and initial tests show that they must next be classified as STD because the instantaneous failure rates to LOTC dictate that this should be the case. The TLD criteria are set and the system is simulated again. This time, because the exposure to the fault combination has reduced (from the LTD interval to the STD interval) the instantaneous failure rates to LOTC actually decrease back just below the limit for LTD faults and, as such, may be reclassified as LTD faults. This reclassification as LTD, in the absence of other changes to the TLD criteria, then leads to the requirement for an STD category to be applied, which then leads again to an LTD category to be applied and this cycle will continue ad infinitum if left unchecked. For this reason, in the TLD MCS code checks are carried out for this type of behaviour. If all TLD criteria remain unchanged apart from a number which cycle between categories and this behaviour continues for 3 successive cycles then the process is brought to a halt within the code by applying the tighter of the dispatch criteria to the relevant fault combinations and ending the iterative process of automatically setting dispatch criteria. In the case that this occurs it is a conservative way of setting dispatch criteria for a system. It is, of course, also possible that the dispatch criteria might cycle in a more complex pattern than that just described. For this reason if more than 10 iterations are performed by the TLD MCS code when

automatically setting dispatch criteria, the dispatch criteria for faults are set to the most restrictive of those that have been applied in the previous 5 simulations.

Appendix 3. TLD MCS Code Input Files for the 8 Component System

Fault Tree File for Eight Component Example System					
G:LOTC	OR	G:GATE_1	G:GATE_2	G:GATE_3	G:GATE_4
G:GATE_1	AND	G:F1_X_FAILS	G:F1_Y_FAILS		
G:GATE_2	AND	G:F2_X_FAILS	G:F2_Y_FAILS		
G:GATE_3	AND	E:H	G:F1_X_FAILS	G:F2_Y_FAILS	
G:GATE_4	AND	E:H	G:F2_X_FAILS	G:F1_Y_FAILS	
G:F1_X_FAILS	AND	E:A		E:B	
G:F1_Y_FAILS	AND	E:A		E:E	
G:F2_X_FAILS	OR	E:C		E:D	
G:F2_Y_FAILS	OR	E:F		E:G	
endoftree					

Failure Rate File for Eight Component Example System	
E:A	FAIL EXP 0.000075
E:B	FAIL EXP 0.00004
E:C	FAIL EXP 0.00003
E:D	FAIL EXP 0.00002
E:E	FAIL EXP 0.00004
E:F	FAIL EXP 0.00003
E:G	FAIL EXP 0.00002
E:H	FAIL EXP 0.00002
ENDOFDATA	

Fault List File for Eight Component Example System
E:H
G:F1_X_FAILS
G:F1_Y_FAILS
G:F2_X_FAILS
G:F2_Y_FAILS

Appendix 4. Sample Output Files for the 8 Component System

System LOTC Rate Output File for Eight Component Example System

```
flight length : 10.000000
System is in service for 130000.000000 hours in each simulation
Maintenance strategy: 4. repair ALL failures present in system

First check of rates after 1000000 simulations
Rates checked every 50000 simulations thereafter
Convergence required to 3 significant figures (sig_fig)
Convergence assumed after 3 identical rates at checks (consec_vals)

STD/LTD     500.0    1000.0    1500.0    2000.0    2500.0
  250.0 5.67e-06 8.15e-06 1.03e-05 1.23e-05 1.41e-05
  375.0 5.66e-06 8.17e-06 1.04e-05 1.23e-05 1.41e-05
  500.0 5.68e-06 8.17e-06 1.04e-05 1.24e-05 1.41e-05
```

Instantaneous LOTC Rates Output File for Eight Component Example System

```
flight length : 10.000000
System is in service for 130000.000000 hours in each simulation
Maintenance strategy: 4. repair ALL failures present in system

First check of rates after 1000000 simulations
Rates checked every 50000 simulations thereafter
Convergence required to 3 significant figures (sig_fig)
Convergence assumed after 3 identical rates at checks (consec_vals)

STD  500.0     LTD 2500.0
E:H,2.02e-05
G:F1_X_FAILS,4.95e-05
G:F1_Y_FAILS,4.95e-05
G:F2_X_FAILS,5.62e-05
G:F2_Y_FAILS,5.62e-05
E:H G:F1_X_FAILS,8.96e-05
E:H G:F1_Y_FAILS,8.98e-05
E:H G:F2_X_FAILS,7.27e-05
E:H G:F2_Y_FAILS,7.26e-05
G:F1_X_FAILS E:H,8.96e-05
G:F1_X_FAILS G:F2_X_FAILS,9.01e-05
G:F1_X_FAILS G:F2_Y_FAILS,1.10e-04
G:F1_Y_FAILS E:H,8.98e-05
G:F1_Y_FAILS G:F2_X_FAILS,1.10e-04
G:F1_Y_FAILS G:F2_Y_FAILS,9.00e-05
G:F2_X_FAILS E:H,7.27e-05
G:F2_X_FAILS G:F1_X_FAILS,9.01e-05
G:F2_X_FAILS G:F1_Y_FAILS,1.10e-04
G:F2_Y_FAILS E:H,7.26e-05
G:F2_Y_FAILS G:F1_X_FAILS,1.10e-04
G:F2_Y_FAILS G:F1_Y_FAILS,9.00e-05
E:H G:F1_X_FAILS G:F2_X_FAILS,8.99e-05
E:H G:F1_Y_FAILS G:F2_Y_FAILS,8.96e-05
E:H G:F2_X_FAILS G:F1_X_FAILS,8.99e-05
E:H G:F2_Y_FAILS G:F1_Y_FAILS,8.96e-05
G:F1_X_FAILS G:F2_X_FAILS E:H,8.99e-05
G:F1_Y_FAILS G:F2_Y_FAILS E:H,8.96e-05
```

Updated TLD Criteria Output File for Eight Component Example System

```
E:H LTD
G:F1_X_FAILS LTD
G:F1_Y_FAILS LTD
G:F2_X_FAILS LTD
G:F2_Y_FAILS LTD
E:H G:F1_X_FAILS STD
E:H G:F1_Y_FAILS STD
E:H G:F2_X_FAILS STD
E:H G:F2_Y_FAILS STD
G:F1_X_FAILS G:F2_X_FAILS STD
G:F1_X_FAILS G:F2_Y_FAILS DND
G:F1_Y_FAILS G:F2_X_FAILS DND
G:F1_Y_FAILS G:F2_Y_FAILS STD
E:H G:F1_X_FAILS G:F2_X_FAILS STD
E:H G:F1_Y_FAILS G:F2_Y_FAILS STD
endofdata
```

Sample Simulation Output for Eight Component Example System

```
-----------------------------------------------------------------------------
        3083.40   | * E:C fails
                  |       checking new status of parent G:F2_X_FAILS, OR gate
                  | * G:F2_X_FAILS fails
                  | * Adding 70000 to the schedule, time:   4090.00
                  |       checking new status of parent G:GATE_2, AND gate
                  | * G:GATE_2 works
                  |       checking new status of parent G:GATE_4, AND gate
                  | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
        4090.00   | *** Maintenance ***
                  | G:F2_X_FAILS now works
                  | E:C                 next failure:        46788.16
-----------------------------------------------------------------------------
        5627.39   | * E:E fails
                  |       checking new status of parent G:F1_Y_FAILS, AND gate
                  | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
        6112.40   | * E:B fails
                  |  E:E is also failed
                  |       checking new status of parent G:F1_X_FAILS, AND gate
                  | * G:F1_X_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
        8706.50   | * E:H fails
                  |  E:B is also failed
                  |  E:E is also failed
                  | * Adding 70000 to the schedule, time:   9710.00
                  |       checking new status of parent G:GATE_3, AND gate
                  | * G:GATE_3 works
                  |       checking new status of parent G:GATE_4, AND gate
                  | * G:GATE_4 works
TLD faults present: E:H
-----------------------------------------------------------------------------
        9710.00   | *** Maintenance ***
-----------------------------------------------------------------------------
       12405.77   | * E:A fails
                  |       checking new status of parent G:F1_X_FAILS, AND gate
                  | * G:F1_X_FAILS works
                  |       checking new status of parent G:F1_Y_FAILS, AND gate
                  | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       13183.33   | * E:G fails
                  |  E:A is also failed
                  |       checking new status of parent G:F2_Y_FAILS, OR gate
                  | * G:F2_Y_FAILS fails
```

Sample Simulation Output for Eight Component Example System

```
                    | * Adding 70000 to the schedule, time:  14190.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_3, AND gate
                    | * G:GATE_3 works
TLD faults present: G:F2_Y_FAILS
-----------------------------------------------------------------------------
       14190.00     | *** Maintenance ***
                    | G:F2_Y_FAILS now works
                    | E:G
-----------------------------------------------------------------------------
       21305.73     | * E:E fails
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       23079.31     | * E:A fails
                    |  E:E is also failed
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS fails
                    | * Adding 70000 to the schedule, time:  24080.00
                    |        checking new status of parent G:GATE_1, AND gate
                    | * G:GATE_1 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F1_Y_FAILS
-----------------------------------------------------------------------------
       24080.00     | *** Maintenance ***
                    | G:F1_Y_FAILS now works
                    | E:A                 next failure:        24732.65
                    | E:E                 next failure:        26268.15
-----------------------------------------------------------------------------
       24732.65     | * E:A fails
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       26268.15     | * E:E fails
                    |  E:A is also failed
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS fails
                    | * Adding 70000 to the schedule, time:  27270.00
                    |        checking new status of parent G:GATE_1, AND gate
                    | * G:GATE_1 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F1_Y_FAILS
-----------------------------------------------------------------------------
       27270.00     | *** Maintenance ***
                    | G:F1_Y_FAILS now works
                    | E:A                 next failure:        33310.13
                    | E:E                 next failure:        35576.63
-----------------------------------------------------------------------------
       28195.17     | * E:B fails
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       33310.13     | * E:A fails
                    |  E:B is also failed
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS fails
                    | * Adding 70000 to the schedule, time:  34320.00
                    |        checking new status of parent G:GATE_1, AND gate
                    | * G:GATE_1 works
                    |        checking new status of parent G:GATE_3, AND gate
                    | * G:GATE_3 works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present: G:F1_X_FAILS
-----------------------------------------------------------------------------
```

Sample Simulation Output for Eight Component Example System

```
        34320.00    | *** Maintenance ***
                    | G:F1_X_FAILS now works
                    | E:A                 next failure:        60383.08
                    | E:B                 next failure:        46801.94
-----------------------------------------------------------------------------
        35576.63    | * E:E fails
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
        36432.00    | * E:F fails
                    |  E:E is also failed
                    |        checking new status of parent G:F2_Y_FAILS, OR gate
                    | * G:F2_Y_FAILS fails
                    | * Adding 70000 to the schedule, time:  37440.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_3, AND gate
                    | * G:GATE_3 works
TLD faults present: G:F2_Y_FAILS
-----------------------------------------------------------------------------
        37440.00    | *** Maintenance ***
                    | G:F2_Y_FAILS now works
                    | E:F                 next failure:       174658.50
-----------------------------------------------------------------------------
        39242.24    | * E:H fails
                    | * Adding 70000 to the schedule, time:  40250.00
                    |        checking new status of parent G:GATE_3, AND gate
                    | * G:GATE_3 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: E:H
-----------------------------------------------------------------------------
        40250.00    | *** Maintenance ***
-----------------------------------------------------------------------------
        46788.16    | * E:C fails
                    |        checking new status of parent G:F2_X_FAILS, OR gate
                    | * G:F2_X_FAILS fails
                    | * Adding 70000 to the schedule, time:  47790.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
        46801.94    | * E:B fails
                    |  E:C is also failed
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
        47790.00    | *** Maintenance ***
                    | G:F2_X_FAILS now works
                    | E:C                 next failure:        55054.54
-----------------------------------------------------------------------------
        55054.54    | * E:C fails
                    |        checking new status of parent G:F2_X_FAILS, OR gate
                    | * G:F2_X_FAILS fails
                    | * Adding 70000 to the schedule, time:  56060.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
        56060.00    | *** Maintenance ***
                    | G:F2_X_FAILS now works
                    | E:C                 next failure:        86524.42
-----------------------------------------------------------------------------
        60383.08    | * E:A fails
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
```

Sample Simulation Output for Eight Component Example System

```
-----------------------------------------------------------------------------
       64612.43    | * E:G fails
                   |  E:A is also failed
                   |        checking new status of parent G:F2_Y_FAILS, OR gate
                   | * G:F2_Y_FAILS fails
                   | * Adding 70000 to the schedule, time:  65620.00
                   |        checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |        checking new status of parent G:GATE_3, AND gate
                   | * G:GATE_3 works
TLD faults present: G:F2_Y_FAILS
-----------------------------------------------------------------------------
       65620.00    | *** Maintenance ***
                   | G:F2_Y_FAILS now works
                   | E:G
-----------------------------------------------------------------------------
       72819.00    | * E:E fails
                   |        checking new status of parent G:F1_Y_FAILS, AND gate
                   | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
       81087.83    | * E:D fails
                   |  E:E is also failed
                   |        checking new status of parent G:F2_X_FAILS, OR gate
                   | * G:F2_X_FAILS fails
                   | * Adding 70000 to the schedule, time:  82090.00
                   |        checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |        checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
       82090.00    | *** Maintenance ***
                   | G:F2_X_FAILS now works
                   | E:D                 next failure:        82143.80
-----------------------------------------------------------------------------
       82143.80    | * E:D fails
                   |        checking new status of parent G:F2_X_FAILS, OR gate
                   | * G:F2_X_FAILS fails
                   | * Adding 70000 to the schedule, time:  83150.00
                   |        checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |        checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
       83150.00    | *** Maintenance ***
                   | G:F2_X_FAILS now works
                   | E:D                 next failure:       116530.56
-----------------------------------------------------------------------------
       86524.42    | * E:C fails
                   |        checking new status of parent G:F2_X_FAILS, OR gate
                   | * G:F2_X_FAILS fails
                   | * Adding 70000 to the schedule, time:  87530.00
                   |        checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |        checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
       87530.00    | *** Maintenance ***
                   | G:F2_X_FAILS now works
                   | E:C                 next failure:        93377.03
-----------------------------------------------------------------------------
       93377.03    | * E:C fails
                   |        checking new status of parent G:F2_X_FAILS, OR gate
                   | * G:F2_X_FAILS fails
                   | * Adding 70000 to the schedule, time:  94380.00
                   |        checking new status of parent G:GATE_2, AND gate
                   | * G:GATE_2 works
                   |        checking new status of parent G:GATE_4, AND gate
                   | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
       94128.89    | * E:B fails
```

Sample Simulation Output for Eight Component Example System

```
                    |  E:C is also failed
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
       94380.00     | *** Maintenance ***
                    | G:F2_X_FAILS now works
                    | E:C                 next failure:       110712.60
-----------------------------------------------------------------------------
       95195.51     | * E:A fails
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
      110712.60     | * E:C fails
                    |  E:A is also failed
                    |        checking new status of parent G:F2_X_FAILS, OR gate
                    | * G:F2_X_FAILS fails
                    | * Adding 70000 to the schedule, time: 111720.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
      111720.00     | *** Maintenance ***
                    | G:F2_X_FAILS now works
                    | E:C                 next failure:       126723.97
-----------------------------------------------------------------------------
      116530.56     | * E:D fails
                    |        checking new status of parent G:F2_X_FAILS, OR gate
                    | * G:F2_X_FAILS fails
                    | * Adding 70000 to the schedule, time: 117540.00
                    |        checking new status of parent G:GATE_2, AND gate
                    | * G:GATE_2 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
      117540.00     | *** Maintenance ***
                    | G:F2_X_FAILS now works
                    | E:D                 next failure:       152843.21
-----------------------------------------------------------------------------
      120142.06     | * E:E fails
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
      120329.54     | * E:A fails
                    |  E:E is also failed
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS fails
                    | * Adding 70000 to the schedule, time: 121330.00
                    |        checking new status of parent G:GATE_1, AND gate
                    | * G:GATE_1 works
                    |        checking new status of parent G:GATE_4, AND gate
                    | * G:GATE_4 works
TLD faults present: G:F1_Y_FAILS
-----------------------------------------------------------------------------
      121330.00     | *** Maintenance ***
                    | G:F1_Y_FAILS now works
                    | E:A                 next failure:       121698.03
                    | E:E                 next failure:       152060.83
-----------------------------------------------------------------------------
      121698.03     | * E:A fails
                    |        checking new status of parent G:F1_X_FAILS, AND gate
                    | * G:F1_X_FAILS works
                    |        checking new status of parent G:F1_Y_FAILS, AND gate
                    | * G:F1_Y_FAILS works
TLD faults present:
-----------------------------------------------------------------------------
```

Sample Simulation Output for Eight Component Example System

```
      124812.23   | * E:B fails
                  |  E:A is also failed
                  |        checking new status of parent G:F1_X_FAILS, AND gate
                  | * G:F1_X_FAILS fails
                  | * Adding 70000 to the schedule, time: 125820.00
                  |        checking new status of parent G:GATE_1, AND gate
                  | * G:GATE_1 works
                  |        checking new status of parent G:GATE_3, AND gate
                  | * G:GATE_3 works
TLD faults present: G:F1_X_FAILS
-----------------------------------------------------------------------------
      125820.00   | *** Maintenance ***
                  | G:F1_X_FAILS now works
                  | E:A                next failure:       129281.37
                  | E:B                next failure:       141845.97
-----------------------------------------------------------------------------
      126723.97   | * E:C fails
                  |        checking new status of parent G:F2_X_FAILS, OR gate
                  | * G:F2_X_FAILS fails
                  | * Adding 70000 to the schedule, time: 127730.00
                  |        checking new status of parent G:GATE_2, AND gate
                  | * G:GATE_2 works
                  |        checking new status of parent G:GATE_4, AND gate
                  | * G:GATE_4 works
TLD faults present: G:F2_X_FAILS
-----------------------------------------------------------------------------
      127730.00   | *** Maintenance ***
                  | G:F2_X_FAILS now works
                  | E:C                next failure:       136445.89
-----------------------------------------------------------------------------
      129281.37   | * E:A fails
                  |        checking new status of parent G:F1_X_FAILS, AND gate
                  | * G:F1_X_FAILS works
                  |        checking new status of parent G:F1_Y_FAILS, AND gate
                  | * G:F1_Y_FAILS works
TLD faults present:
Simulation over
```

Appendix 5. TLD MCS Code Input Files for the Large Scale System

System Fault Tree Input File

```
G:PWR:A:LOSS    AND     G:PWR1:A:LOSS  G:PWR2:A:LOSS
G:PWR1:A:LOSS   OR      E:PWR1:A:FAULT_1       E:PWR1:CM_FAULT_1      E:PWR1:A:FAULT_2
G:PWR2:A:LOSS   OR      E:PWR2:A:FAULT_1       E:PWR2:CM_FAULT_1      E:PWR2:A:FAULT_2
G:A:UNAVAIL     OR      G:PWR:A:LOSS   G:EEC:A:LOSS
G:PWR:B:LOSS    AND     G:PWR1:B:LOSS  G:PWR2:B:LOSS
G:PWR1:B:LOSS   OR      E:PWR1:CM_FAULT_1      E:PWR1:B:FAULT_1       E:PWR1:B:FAULT_2
G:PWR2:B:LOSS   OR      E:PWR2:CM_FAULT_1      E:PWR2:B:FAULT_1       E:PWR2:B:FAULT_2
G:B:UNAVAIL     OR      G:PWR:B:LOSS   G:EEC:B:LOSS
G:EEC_FD_WORKS_A        NOT     E:EEC:A:FAULT_3
G:EEC_FD_WORKS_B        NOT     E:EEC:B:FAULT_3
G:EEC_INDET_A   AND     E:EEC:A:FAULT_2        E:EEC:A:FAULT_3
G:EEC_INDET_B   AND     E:EEC:B:FAULT_2        E:EEC:B:FAULT_3
G:EEC:A:LOSS    OR      G:FAULT_DET_A  E:EEC:A:FAULT_1
G:EEC:B:LOSS    OR      G:FAULT_DET_B  E:EEC:B:FAULT_1
G:LOTC:CAUSE_1:FADEC    AND     G:A:UNAVAIL    G:B:UNAVAIL
G:FAULT_DET_A   AND     E:EEC:A:FAULT_2        G:EEC_FD_WORKS_A
G:FAULT_DET_B   AND     E:EEC:B:FAULT_2        G:EEC_FD_WORKS_B
G:LOTC:CAUSE_2:5        AND     E:FN1:A:FAULT  G:LOTC:CAUSE_2:7
G:LOTC:CAUSE_2:6        AND     E:FN1:B:FAULT  G:LOTC:CAUSE_2:8
G:LOTC:CAUSE_2:7        OR      G:FN4_VALIDATED_ERRONEOUS_A
        G:FN2_VALIDATED_ERRONEOUS_A
G:LOTC:CAUSE_2:8        OR      G:FN4_VALIDATED_ERRONEOUS_B
        G:FN2_VALIDATED_ERRONEOUS_B
G:LOTC:CAUSE_2:2        AND     G:FN3_VALIDATED_ERRONEOUS_A
        G:FN3_VALIDATED_ERRONEOUS_B
G:LOTC:CAUSE_2:1        OR      G:LOTC:CAUSE_2:3       G:LOTC:CAUSE_2:4
G:LOTC:CAUSE_2:3        OR      G:FN1_VALIDATED_ERRONEOUS_A   G:LOTC:CAUSE_2:5
G:LOTC:CAUSE_2:4        OR      G:FN1_VALIDATED_erroneous_B   G:LOTC:CAUSE_2:6
G:LOTC:CAUSE_3:HMU      OR      E:HMU:FAULT_1  E:HMU:FAULT_2  E:HMU:FAULT_3
        E:HMU:FAULT_4
G:LOTC  OR      G:LOTC:CAUSE_1:FADEC   G:LOTC:CAUSE_2:FN      G:LOTC:CAUSE_3:HMU
G:LOTC:CAUSE_2:FN       OR      G:LOTC:CAUSE_2:1       G:LOTC:CAUSE_2:2
G:FN3_A_ERR_FROM_B      OR      G:FN3_ERR_A    G:COMMS_ERR_A
G:FN3_A_UNAVAIL/INV     OR      G:FN3_INV_A    G:COMMS_ERR/LOST_A
G:FN3_AVE_ERR_A         AND     G:FN3_ERR_A    G:FN3_B_ERR_FROM_A
G:FN3_AVE_ERR_B         AND     G:FN3_A_ERR_FROM_B     G:FN3_ERR_B
G:FN3_B_ERR_FROM_A      OR      G:FN3_ERR_B    G:COMMS_ERR_B
G:FN3_B_UNAVAIL/INV     OR      G:FN3_INV_B    G:COMMS_ERR/LOST_B
G:FN3_ERR_A     OR      E:FN3:A:FAULT_3        E:FN3_FN4:A:CM_FAULT_1 E:FN3:A:FAULT_1
        E:FN3_FN4:CM_FAULT_1
G:FN3_ERR_B     OR      E:FN3_FN4:CM_FAULT_1   E:FN3_FN4:B:CM_FAULT_1 E:FN3:B:FAULT_1
        E:FN3:B:FAULT_3
G:FN3_INV_A     OR      E:FN3_FN4:CM_FAULT_2   E:FN3_FN4:A:CM_FAULT_2 E:FN3:A:FAULT_2
        E:FN3:A:FAULT_4
G:FN3_INV_B     OR      E:FN3_FN4:CM_FAULT_2   E:FN3_FN4:B:CM_FAULT_2 E:FN3:B:FAULT_2
        E:FN3:B:FAULT_4
G:FN3_MOD_ERR_A         AND     G:FN3_MOD_SEL_A        E:FN3:CM_FAULT_1
G:FN3_MOD_ERR_B         AND     G:FN3_MOD_SEL_B        E:FN3:CM_FAULT_1
G:FN3_MOD_SEL_A         AND     G:FN3_B_UNAVAIL/INV    G:FN3_INV_A
G:FN3_MOD_SEL_B         AND     G:FN3_A_UNAVAIL/INV    G:FN3_INV_B
G:FN3_OTH_ERR_A         AND     G:FN3_INV_A    G:FN3_B_ERR_FROM_A
G:FN3_OTH_ERR_B         AND     G:FN3_A_ERR_FROM_B     G:FN3_INV_B
G:FN3_OWN_ERR_A         AND     G:FN3_B_UNAVAIL/INV    G:FN3_ERR_A
G:FN3_OWN_ERR_B         AND     G:FN3_A_UNAVAIL/INV    G:FN3_ERR_B
G:FN3_VALIDATED_ERRONEOUS_A     OR      G:FN3_AVE_ERR_A        G:FN3_OWN_ERR_A
        G:FN3_OTH_ERR_A         G:FN3_MOD_ERR_A
G:FN3_VALIDATED_ERRONEOUS_B     OR      G:FN3_AVE_ERR_B        G:FN3_OWN_ERR_B
        G:FN3_OTH_ERR_B         G:FN3_MOD_ERR_B
G:FN4_A_ERR_FROM_B      OR      G:COMMS_ERR_A  G:FN4_ERR_A
G:FN4_A_UNAVAIL/INV     OR      G:COMMS_ERR/LOST_A     G:FN4_INV_A
G:FN4_AVE_ERR_A         AND     G:FN4_ERR_A    G:FN4_B_ERR_FROM_A
G:FN4_AVE_ERR_B         AND     G:FN4_A_ERR_FROM_B     G:FN4_ERR_B
G:FN4_B_ERR_FROM_A      OR      G:FN4_ERR_B    G:COMMS_ERR_B
G:FN4_B_UNAVAIL/INV     OR      G:FN4_INV_B    G:COMMS_ERR/LOST_B
G:FN4_ERR_A     OR      E:FN3_FN4:CM_FAULT_1   E:FN3_FN4:A:CM_FAULT_1
        E:FN4_FN2:A:CM_FAULT_1 E:FN4:A:FAULT_1         E:FN4:A:FAULT_3
G:FN4_ERR_B     OR      E:FN3_FN4:CM_FAULT_1   E:FN3_FN4:B:CM_FAULT_1
```

System Fault Tree Input File

```
        E:FN4_FN2:B:CM_FAULT_1 E:FN4:B:FAULT_1        E:FN4:B:FAULT_3
G:FN4_INV_A    OR      E:FN3_FN4:CM_FAULT_2   E:FN3_FN4:A:CM_FAULT_2
        E:FN4_FN2:A:CM_FAULT_2 E:FN4:A:FAULT_2        E:FN4:A:FAULT_4
G:FN4_INV_B    OR      E:FN3_FN4:CM_FAULT_2   E:FN3_FN4:B:CM_FAULT_2
        E:FN4_FN2:B:CM_FAULT_2 E:FN4:B:FAULT_2        E:FN4:B:FAULT_4
G:FN4_MOD_ERR_A        AND     G:FN4_MOD_SEL_A        E:FN4:CM_FAULT_1
G:FN4_MOD_ERR_B        AND     G:FN4_MOD_SEL_B        E:FN4:CM_FAULT_1
G:FN4_MOD_SEL_A        AND     G:FN4_B_UNAVAIL/INV    G:FN4_INV_A
G:FN4_MOD_SEL_B        AND     G:FN4_A_UNAVAIL/INV    G:FN4_INV_B
G:FN4_OTH_ERR_A        AND     G:FN4_INV_A    G:FN4_B_ERR_FROM_A
G:FN4_OTH_ERR_B        AND     G:FN4_A_ERR_FROM_B     G:FN4_INV_B
G:FN4_OWN_ERR_A        AND     G:FN4_B_UNAVAIL/INV    G:FN4_ERR_A
G:FN4_OWN_ERR_B        AND     G:FN4_A_UNAVAIL/INV    G:FN4_ERR_B
G:FN4_VALIDATED_ERRONEOUS_A   OR      G:FN4_AVE_ERR_A        G:FN4_OWN_ERR_A
        G:FN4_OTH_ERR_A        G:FN4_MOD_ERR_A
G:FN4_VALIDATED_ERRONEOUS_B   OR      G:FN4_AVE_ERR_B        G:FN4_OWN_ERR_B
        G:FN4_OTH_ERR_B        G:FN4_MOD_ERR_B
G:FN2_1_A_ERR_FROM_B   OR      G:COMMS_ERR_A  G:FN2_1_ERR_A
G:FN2_1_AVE_ERR_A      AND     G:FN2_1_ERR_A  G:FN2_1_B_ERR_FROM_A
G:FN2_1_AVE_ERR_B      AND     G:FN2_1_A_ERR_FROM_B   G:FN2_1_ERR_B
G:FN2_1_B_ERR_FROM_A   OR      G:FN2_1_ERR_B  G:COMMS_ERR_B
G:FN2_1_ERR_A  OR      E:FN4_FN2:A:CM_FAULT_1 E:FN2_1:CM_FAULT_1     E:FN2_1:A:FAULT_1
        E:FN2_1:A:FAULT_3
G:FN2_1_ERR_B  OR      E:FN4_FN2:B:CM_FAULT_1 E:FN2_1:CM_FAULT_1     E:FN2_1:B:FAULT_1
        E:FN2_1:B:FAULT_3
G:FN2_1_INV_A  OR      E:FN4_FN2:A:CM_FAULT_2 E:FN2_1:CM_FAULT_2     E:FN2_1:A:FAULT_2
        E:FN2_1:A:FAULT_4
G:FN2_1_INV_B  OR      E:FN4_FN2:B:CM_FAULT_2 E:FN2_1:CM_FAULT_2     E:FN2_1:B:FAULT_2
        E:FN2_1:B:FAULT_4
G:FN2_1_OTH_ERR_A      AND     G:FN2_1_INV_A  G:FN2_1_B_ERR_FROM_A
G:FN2_1_OTH_ERR_B      AND     G:FN2_1_A_ERR_FROM_B   G:FN2_1_INV_B
G:FN2_1_OWN_ERR_A      AND     G:FN2_1_ERR_A  G:FN2_1_UNAVAIL_B
G:FN2_1_OWN_ERR_B      AND     G:FN2_1_UNAVAIL_A      G:FN2_1_ERR_B
G:FN2_1_UNAVAIL_A      OR      G:COMMS_ERR/LOST_A     G:FN2_1_INV_A
G:FN2_1_UNAVAIL_B      OR      G:FN2_1_INV_B  G:COMMS_ERR/LOST_B
G:FN2_1_UNAVAIL_TO_A   AND     G:FN2_1_INV_A  G:FN2_1_UNAVAIL_B
G:FN2_1_UNAVAIL_TO_B   AND     G:FN2_1_UNAVAIL_A      G:FN2_1_INV_B
G:FN2_2_A_ERR_FROM_B   OR      G:COMMS_ERR_A  G:FN2_2_ERR_A
G:FN2_2_AVE_ERR_A      AND     G:FN2_2_B_ERR_FROM_A   G:FN2_2_ERR_A
G:FN2_2_AVE_ERR_B      AND     G:FN2_2_A_ERR_FROM_B   G:FN2_2_ERR_B
G:FN2_2_B_ERR_FROM_A   OR      G:FN2_2_ERR_B  G:COMMS_ERR_B
G:FN2_2_ERR_A  OR      E:FN4_FN2:A:CM_FAULT_1 E:FN2_2:CM_FAULT_1     E:FN2_2:A:FAULT_1
        E:FN2_2:A:FAULT_3
G:FN2_2_ERR_B  OR      E:FN4_FN2:B:CM_FAULT_1 E:FN2_2:CM_FAULT_1     E:FN2_2:B:FAULT_1
        E:FN2_2:B:FAULT_3
G:FN2_2_INV_A  OR      E:FN4_FN2:A:CM_FAULT_2 E:FN2_2:CM_FAULT_2     E:FN2_2:A:FAULT_2
        E:FN2_2:A:FAULT_4
G:FN2_2_INV_B  OR      E:FN4_FN2:B:CM_FAULT_2 E:FN2_2:CM_FAULT_2     E:FN2_2:B:FAULT_2
        E:FN2_2:B:FAULT_4
G:FN2_2_OTH_ERR_A      AND     G:FN2_2_INV_A  G:FN2_2_B_ERR_FROM_A
G:FN2_2_OTH_ERR_B      AND     G:FN2_2_A_ERR_FROM_B   G:FN2_2_INV_B
G:FN2_2_OWN_ERR_A      AND     G:FN2_2_ERR_A  G:FN2_2_UNAVAIL_B
G:FN2_2_OWN_ERR_B      AND     G:FN2_2_UNAVAIL_A      G:FN2_2_ERR_B
G:FN2_2_UNAVAIL_A      OR      G:FN2_2_INV_A  G:COMMS_ERR/LOST_A
G:FN2_2_UNAVAIL_B      OR      G:FN2_2_INV_B  G:COMMS_ERR/LOST_B
G:FN2_2_UNAVAIL_TO_A   AND     G:FN2_2_UNAVAIL_B      G:FN2_2_INV_A
G:FN2_2_UNAVAIL_TO_B   AND     G:FN2_2_UNAVAIL_A      G:FN2_2_INV_B
G:FN2_EITHER_ERR_A     OR      G:FN2_1_AVE_ERR_A      G:FN2_2_AVE_ERR_A
G:FN2_EITHER_ERR_B     OR      G:FN2_1_AVE_ERR_B      G:FN2_2_AVE_ERR_B
G:FN2_MOD_ERR_A        AND     G:FN2_MOD_SEL_A        E:FN2:CM_FAULT_1
G:FN2_MOD_ERR_B        AND     G:FN2_MOD_SEL_B        E:FN2:CM_FAULT_1
G:FN2_MOD_SEL_A        AND     G:FN2_1_UNAVAIL_TO_A   G:FN2_2_UNAVAIL_TO_A
G:FN2_MOD_SEL_B        AND     G:FN2_1_UNAVAIL_TO_B   G:FN2_2_UNAVAIL_TO_B
G:FN2_OTH_ERR_A        OR      G:FN2_1_OTH_ERR_A      G:FN2_2_OTH_ERR_A
G:FN2_OTH_ERR_B        OR      G:FN2_1_OTH_ERR_B      G:FN2_2_OTH_ERR_B
G:FN2_OWN_ERR_A        OR      G:FN2_1_OWN_ERR_A      G:FN2_2_OWN_ERR_A
G:FN2_OWN_ERR_B        OR      G:FN2_1_OWN_ERR_B      G:FN2_2_OWN_ERR_B
G:FN2_VALIDATED_ERRONEOUS_A   OR      G:FN2_EITHER_ERR_A     G:FN2_OWN_ERR_A
        G:FN2_OTH_ERR_A        G:FN2_MOD_ERR_A
G:FN2_VALIDATED_ERRONEOUS_B   OR      G:FN2_EITHER_ERR_B     G:FN2_OWN_ERR_B
        G:FN2_OTH_ERR_B        G:FN2_MOD_ERR_B
G:FN1_A_ERR_FROM_B     OR      G:COMMS_ERR_A  G:FN1_ERR_A
G:FN1_A_UNAVAIL/INV    OR      G:COMMS_ERR/LOST_A     G:FN1_INV_A
G:FN1_AVE_ERR_A        AND     G:FN1_ERR_A    G:FN1_B_ERR_FROM_A
G:FN1_AVE_ERR_B        AND     G:FN1_A_ERR_FROM_B     G:FN1_ERR_B
```

System Fault Tree Input File

```
G:FN1_B_ERR_FROM_A      OR      G:FN1_ERR_B     G:COMMS_ERR_B
G:FN1_B_UNAVAIL/INV     OR      G:FN1_INV_B     G:COMMS_ERR/LOST_B
G:FN1_MOD_ERR_A         AND     G:FN1_MOD_SEL_A         E:FN1:CM_FAULT_1
G:FN1_MOD_ERR_B         AND     G:FN1_MOD_SEL_B         E:FN1:CM_FAULT_1
G:FN1_MOD_SEL_A         AND     G:FN1_INV_A     G:FN1_B_UNAVAIL/INV
G:FN1_MOD_SEL_B         AND     G:FN1_A_UNAVAIL/INV     G:FN1_INV_B
G:FN1_OTH_POS_ERR_A     AND     G:FN1_INV_A     G:FN1_B_ERR_FROM_A
G:FN1_OTH_ERR_B         AND     G:FN1_A_ERR_FROM_B      G:FN1_INV_B
G:FN1_OWN_ERR_A         AND     G:FN1_B_UNAVAIL/INV     G:FN1_ERR_A
G:FN1_OWN_ERR_B         AND     G:FN1_A_UNAVAIL/INV     G:FN1_ERR_B
G:FN1_ERR_A     OR      E:FN1:A:FAULT_1         E:FN1:A:FAULT_3
G:FN1_ERR_B     OR      E:FN1:B:FAULT_1         E:FN1:B:FAULT_3
G:FN1_INV_A     OR      E:FN1:A:FAULT_2         E:FN1:A:FAULT_4
G:FN1_INV_B     OR      E:FN1:B:FAULT_2         E:FN1:B:FAULT_4
G:FN1_VALIDATED_ERRONEOUS_A     OR      G:FN1_AVE_ERR_A         G:FN1_OWN_ERR_A
        G:FN1_OTH_POS_ERR_A     G:FN1_MOD_ERR_A
G:FN1_VALIDATED_erroneous_B     OR      G:FN1_AVE_ERR_B         G:FN1_OWN_ERR_B
        G:FN1_OTH_ERR_B         G:FN1_MOD_ERR_B
G:COMMS_AVAIL_A         NOT     G:COMMS_LOST_A
G:COMMS_AVAIL_B         NOT     G:COMMS_LOST_B
G:COMMS_CORRUPT_A       OR      G:EEC_INDET_A   E:COMMS_CORRUPTED
G:COMMS_CORRUPT_B       OR      E:COMMS_CORRUPTED       G:EEC_INDET_B
G:COMMS_ERR_A   AND     G:COMMS_AVAIL_A         G:COMMS_CORRUPT_A
G:COMMS_ERR_B   AND     G:COMMS_AVAIL_B         G:COMMS_CORRUPT_B
G:COMMS_ERR/LOST_A      OR      G:COMMS_ERR_A   G:COMMS_LOST_A
G:COMMS_ERR/LOST_B      OR      G:COMMS_ERR_B   G:COMMS_LOST_B
G:COMMS_LOST_A NULL     G:PWR:A:LOSS
G:COMMS_LOST_B NULL     G:PWR:B:LOSS
endofdata
```

Basic event failure distribution input file

E:FN3_FN4:CM_FAULT_1	FAIL	EXP	0.0000006
E:FN3_FN4:A:CM_FAULT_1	FAIL	EXP	0.0000004
E:FN3_FN4:B:CM_FAULT_1	FAIL	EXP	0.0000004
E:FN3_FN4:CM_FAULT_2	FAIL	EXP	0.0000008
E:FN3_FN4:A:CM_FAULT_2	FAIL	EXP	0.000002
E:FN3_FN4:B:CM_FAULT_2	FAIL	EXP	0.000002
E:PWR1:CM_FAULT_1	FAIL	EXP	0.000001
E:PWR2:CM_FAULT_1	FAIL	EXP	0.000001
E:FN4_FN2:A:CM_FAULT_1	FAIL	EXP	0.0000005
E:FN4_FN2:B:CM_FAULT_1	FAIL	EXP	0.0000005
E:FN4_FN2:A:CM_FAULT_2	FAIL	EXP	0.000001
E:FN4_FN2:B:CM_FAULT_2	FAIL	EXP	0.000001
E:FN2_1:CM_FAULT_1	FAIL	EXP	0.000001
E:FN2_1:CM_FAULT_2	FAIL	EXP	0.000005
E:FN2_2:CM_FAULT_1	FAIL	EXP	0.000001
E:FN2_2:CM_FAULT_2	FAIL	EXP	0.000005
E:FN3:A:FAULT_3	FAIL	EXP	0.0000004
E:FN3:B:FAULT_3	FAIL	EXP	0.0000004
E:FN3:A:FAULT_4	FAIL	EXP	0.000008
E:FN3:B:FAULT_4	FAIL	EXP	0.000008
E:FN4:A:FAULT_3	FAIL	EXP	0.0000003
E:FN4:B:FAULT_3	FAIL	EXP	0.0000003
E:FN4:A:FAULT_4	FAIL	EXP	0.000008
E:FN4:B:FAULT_4	FAIL	EXP	0.000008
E:FN1:A:FAULT_3	FAIL	EXP	0.0000004
E:FN1:B:FAULT_3	FAIL	EXP	0.0000004
E:FN1:A:FAULT_4	FAIL	EXP	0.000008
E:FN1:B:FAULT_4	FAIL	EXP	0.000008
E:FN2_1:A:FAULT_3	FAIL	EXP	0.0000003
E:FN2_1:B:FAULT_3	FAIL	EXP	0.0000003

Basic event failure distribution input file			
E:FN2_1:A:FAULT_4	FAIL	EXP	0.000008
E:FN2_1:B:FAULT_4	FAIL	EXP	0.000008
E:FN2_2:A:FAULT_3	FAIL	EXP	0.0000004
E:FN2_2:B:FAULT_3	FAIL	EXP	0.0000004
E:FN2_2:A:FAULT_4	FAIL	EXP	0.000009
E:FN2_2:B:FAULT_4	FAIL	EXP	0.000009
E:PWR1:A:FAULT_2	FAIL	EXP	0.000002
E:PWR1:B:FAULT_2	FAIL	EXP	0.000002
E:PWR2:A:FAULT_2	FAIL	EXP	0.000002
E:PWR2:B:FAULT_2	FAIL	EXP	0.000002
E:EEC:A:FAULT_2	FAIL	EXP	0.0000035
E:EEC:B:FAULT_2	FAIL	EXP	0.0000035
E:EEC:A:FAULT_3	FAIL	EXP	0.0000022
E:EEC:B:FAULT_3	FAIL	EXP	0.0000022
E:EEC:A:FAULT_1	FAIL	EXP	0.000002
E:EEC:B:FAULT_1	FAIL	EXP	0.000002
E:FN3:A:FAULT_1	FAIL	EXP	0.0000002
E:FN3:B:FAULT_1	FAIL	EXP	0.0000002
E:FN3:A:FAULT_2	FAIL	EXP	0.000008
E:FN3:B:FAULT_2	FAIL	EXP	0.000008
E:FN4:A:FAULT_1	FAIL	EXP	0.00000025
E:FN4:B:FAULT_1	FAIL	EXP	0.00000025
E:FN4:A:FAULT_2	FAIL	EXP	0.000008
E:FN4:B:FAULT_2	FAIL	EXP	0.000008
E:FN2_1:A:FAULT_1	FAIL	EXP	0.00000023
E:FN2_1:B:FAULT_1	FAIL	EXP	0.00000023
E:FN2_1:A:FAULT_2	FAIL	EXP	0.000005
E:FN2_1:B:FAULT_2	FAIL	EXP	0.000005
E:FN2_2:A:FAULT_1	FAIL	EXP	0.00000023
E:FN2_2:B:FAULT_1	FAIL	EXP	0.00000023
E:FN2_2:A:FAULT_2	FAIL	EXP	0.000005
E:FN2_2:B:FAULT_2	FAIL	EXP	0.000005
E:FN1:B:FAULT_1	FAIL	EXP	0.00000025
E:FN1:A:FAULT_1	FAIL	EXP	0.00000025
E:FN1:A:FAULT_2	FAIL	EXP	0.000008
E:FN1:B:FAULT_2	FAIL	EXP	0.000008
E:HMU:FAULT_2	FAIL	EXP	0.000001
E:HMU:FAULT_3	FAIL	EXP	0.000001
E:HMU:FAULT_4	FAIL	EXP	0.000001
E:HMU:FAULT_1	FAIL	EXP	0.000001
E:FN3:CM_FAULT_1	FAIL	EXP	0.000001
E:PWR1:A:FAULT_1	FAIL	EXP	0.000009
E:PWR1:B:FAULT_1	FAIL	EXP	0.000009
E:PWR2:A:FAULT_1	FAIL	EXP	0.0000095
E:PWR2:B:FAULT_1	FAIL	EXP	0.0000095
E:FN4:CM_FAULT_1	FAIL	EXP	0.000001
E:FN2:CM_FAULT_1	FAIL	EXP	0.000001
E:FN1:A:FAULT	FAIL	EXP	0.000005
E:FN1:B:FAULT	FAIL	EXP	0.000005
E:FN1:CM_FAULT_1	FAIL	EXP	0.000001
E:COMMS_CORRUPTED	FAIL	EXP	0.0000016
Endofdata			

System Fault List File
G:CH_A_PWR_LOSS
G:CH_B_PWR_LOSS
G:EEC_LOSS_A
G:EEC_LOSS_B
G:VSV_POS_INV_A
G:VSV_POS_INV_B
G:OILP_INV_A
G:OILP_INV_B
G:SPEED_INV_A
G:SPEED_INV_B
G:TEMP_1_INV_A
G:TEMP_1_INV_B
G:TEMP_2_INV_A
G:TEMP_2_INV_B
endofdata

System Fault Tree 'Associated' Repairs File
G:FN1_INV_A
G:FN1_ERR_A
E:FN1:CM_FAULT_1
E:FN1:A:FAULT
G:COMMS_CORRUPT_A
endforfault
G:FN1_INV_B
G:FN1_ERR_B
E:FN1:CM_FAULT_1
E:FN1:B:FAULT
G:COMMS_CORRUPT_B
endforfault
G:FN3_INV_A
G:FN3_ERR_A
E:FN3:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:FN3_INV_B
G:FN3_ERR_B
E:FN3:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
G:FN4_INV_A
G:FN4_ERR_A
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:FN4_INV_B
G:FN4_ERR_B
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
G:FN2_1_INV_A
G:FN2_1_ERR_A
G:FN2_2_INV_A
G:FN2_2_ERR_A
E:FN2:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:FN2_1_INV_B
G:FN2_1_ERR_B

System Fault Tree 'Associated' Repairs File
G:FN2_2_INV_B
G:FN2_2_ERR_B
E:FN2:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
G:FN2_2_INV_A
G:FN2_2_ERR_A
G:FN2_1_INV_A
G:FN2_1_ERR_A
E:FN2:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:FN2_2_INV_B
G:FN2_2_ERR_B
G:FN2_1_INV_B
G:FN2_1_ERR_B
E:FN2:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
G:PWR:A:LOSS
G:EEC:A:LOSS
G:FN1_INV_A
G:FN1_ERR_A
E:FN1:CM_FAULT_1
E:FN1:A:FAULT
G:FN2_1_INV_A
G:FN2_1_ERR_A
G:FN2_2_INV_A
G:FN2_2_ERR_A
E:FN2:CM_FAULT_1
G:FN3_INV_A
G:FN3_ERR_A
E:FN3:CM_FAULT_1
G:FN4_INV_A
G:FN4_ERR_A
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:PWR:B:LOSS
G:EEC:B:LOSS
G:FN1_INV_B
G:FN1_ERR_B
E:FN1:CM_FAULT_1
E:FN1:B:FAULT
G:FN2_1_INV_B
G:FN2_1_ERR_B
G:FN2_2_INV_B
G:FN2_2_ERR_B
E:FN2:CM_FAULT_1
G:FN3_INV_B
G:FN3_ERR_B
E:FN3:CM_FAULT_1
G:FN4_INV_B
G:FN4_ERR_B
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
G:EEC:A:LOSS
G:PWR:A:LOSS

System Fault Tree 'Associated' Repairs File
G:FN1_INV_A
G:FN1_ERR_A
E:FN1:CM_FAULT_1
E:FN1:A:FAULT
G:FN2_1_INV_A
G:FN2_1_ERR_A
G:FN2_2_INV_A
G:FN2_2_ERR_A
E:FN2:CM_FAULT_1
G:FN3_INV_A
G:FN3_ERR_A
E:FN3:CM_FAULT_1
G:FN4_INV_A
G:FN4_ERR_A
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_A
endforfault
G:EEC:B:LOSS
G:PWR:B:LOSS
G:FN1_INV_B
G:FN1_ERR_B
E:FN1:CM_FAULT_1
E:FN1:B:FAULT
G:FN2_1_INV_B
G:FN2_1_ERR_B
G:FN2_2_INV_B
G:FN2_2_ERR_B
E:FN2:CM_FAULT_1
G:FN3_INV_B
G:FN3_ERR_B
E:FN3:CM_FAULT_1
G:FN4_INV_B
G:FN4_ERR_B
E:FN4:CM_FAULT_1
G:COMMS_CORRUPT_B
endforfault
endofdata

System Initial TLD Criteria File
G:PWR:A:LOSS DND
G:PWR:B:LOSS DND

Appendix 6. Results For the Large Scale System

Importance Measures for dual fault dispatchable combinations for STD MEL 2500, LTD MEL 5000, repair strategy 1

Dual Fault Combination		Importance Measure
G:EEC:A:LOSS	G:FN3_INV_A	2.04×10^{-3}
G:EEC:A:LOSS	G:FN3_INV_B	2.02×10^{-3}
G:EEC:A:LOSS	G:FN4_INV_A	2.05×10^{-3}
G:EEC:A:LOSS	G:FN4_INV_B	2.03×10^{-3}
G:EEC:A:LOSS	G:FN2_1_INV_A	1.88×10^{-3}
G:EEC:A:LOSS	G:FN2_1_INV_B	1.87×10^{-3}
G:EEC:A:LOSS	G:FN2_2_INV_A	2.00×10^{-3}
G:EEC:A:LOSS	G:FN2_2_INV_B	1.99×10^{-3}
G:EEC:A:LOSS	G:FN1_INV_A	1.86×10^{-3}
G:EEC:A:LOSS	G:FN1_INV_B	1.95×10^{-3}
G:EEC:B:LOSS	G:FN3_INV_A	2.02×10^{-3}
G:EEC:B:LOSS	G:FN3_INV_B	2.01×10^{-3}
G:EEC:B:LOSS	G:FN4_INV_A	2.07×10^{-3}
G:EEC:B:LOSS	G:FN4_INV_B	2.03×10^{-3}
G:EEC:B:LOSS	G:FN2_1_INV_A	1.85×10^{-3}
G:EEC:B:LOSS	G:FN2_1_INV_B	1.87×10^{-3}
G:EEC:B:LOSS	G:FN2_2_INV_A	1.99×10^{-3}
G:EEC:B:LOSS	G:FN2_2_INV_B	1.99×10^{-3}
G:EEC:B:LOSS	G:FN1_INV_A	1.88×10^{-3}
G:EEC:B:LOSS	G:FN1_INV_B	1.89×10^{-3}
G:FN3_INV_A	G:FN3_INV_B	4.68×10^{-3}
G:FN3_INV_A	G:FN4_INV_A	1.24×10^{-2}
G:FN3_INV_A	G:FN4_INV_B	5.07×10^{-3}
G:FN3_INV_A	G:FN2_1_INV_A	4.67×10^{-3}
G:FN3_INV_A	G:FN2_1_INV_B	4.65×10^{-3}
G:FN3_INV_A	G:FN2_2_INV_A	5.14×10^{-3}
G:FN3_INV_A	G:FN2_2_INV_B	5.06×10^{-3}
G:FN3_INV_A	G:FN1_INV_A	4.08×10^{-3}
G:FN3_INV_A	G:FN1_INV_B	4.17×10^{-3}
G:FN3_INV_B	G:FN4_INV_A	5.22×10^{-3}
G:FN3_INV_B	G:FN4_INV_B	1.24×10^{-2}
G:FN3_INV_B	G:FN2_1_INV_A	4.71×10^{-3}
G:FN3_INV_B	G:FN2_1_INV_B	4.77×10^{-3}
G:FN3_INV_B	G:FN2_2_INV_A	5.10×10^{-3}
G:FN3_INV_B	G:FN2_2_INV_B	5.08×10^{-3}
G:FN3_INV_B	G:FN1_INV_A	4.11×10^{-3}
G:FN3_INV_B	G:FN1_INV_B	4.17×10^{-3}
G:FN4_INV_A	G:FN4_INV_B	5.28×10^{-3}
G:FN4_INV_A	G:FN2_1_INV_A	6.44×10^{-3}
G:FN4_INV_A	G:FN2_1_INV_B	4.40×10^{-3}
G:FN4_INV_A	G:FN2_2_INV_A	6.78×10^{-3}
G:FN4_INV_A	G:FN2_2_INV_B	4.70×10^{-3}
G:FN4_INV_A	G:FN1_INV_A	4.06×10^{-3}

Dual Fault Combination		Importance Measure
G:FN4_INV_A	G:FN1_INV_B	4.84×10^{-3}
G:FN4_INV_B	G:FN2_1_INV_A	4.36×10^{-3}
G:FN4_INV_B	G:FN2_1_INV_B	6.39×10^{-3}
G:FN4_INV_B	G:FN2_2_INV_A	4.71×10^{-3}
G:FN4_INV_B	G:FN2_2_INV_B	6.70×10^{-3}
G:FN4_INV_B	G:FN1_INV_A	4.84×10^{-3}
G:FN4_INV_B	G:FN1_INV_B	4.94×10^{-3}
G:FN2_1_INV_A	G:FN2_1_INV_B	2.07×10^{-2}
G:FN2_1_INV_A	G:FN2_2_INV_A	6.33×10^{-3}
G:FN2_1_INV_A	G:FN2_2_INV_B	4.21×10^{-3}
G:FN2_1_INV_A	G:FN1_INV_A	4.47×10^{-3}
G:FN2_1_INV_A	G:FN1_INV_B	4.59×10^{-3}
G:FN2_1_INV_B	G:FN2_2_INV_A	4.25×10^{-3}
G:FN2_1_INV_B	G:FN2_2_INV_B	6.28×10^{-3}
G:FN2_1_INV_B	G:FN1_INV_A	4.44×10^{-3}
G:FN2_1_INV_B	G:FN1_INV_B	4.56×10^{-3}
G:FN2_2_INV_A	G:FN2_2_INV_B	2.11×10^{-2}
G:FN2_2_INV_A	G:FN1_INV_A	4.78×10^{-3}
G:FN2_2_INV_A	G:FN1_INV_B	4.91×10^{-3}
G:FN2_2_INV_B	G:FN1_INV_A	4.82×10^{-3}
G:FN2_2_INV_B	G:FN1_INV_B	4.91×10^{-3}
G:FN1_INV_A	G:FN1_INV_B	3.79×10^{-3}

Instantaneous failure rates to LOTC from the dual fault states for repair strategy 4 – used in the dual fault state Markov model for the system

Dual Fault Combination, i		$\lambda_{i,L}$
G:EEC:A:LOSS	G:FN3_INV_A	2.08×10-5
G:EEC:A:LOSS	G:FN3_INV_B	2.07×10-5
G:EEC:A:LOSS	G:FN4_INV_A	1.84×10-5
G:EEC:A:LOSS	G:FN4_INV_B	1.83×10-5
G:EEC:A:LOSS	G:FN2_1_INV_A	1.79×10-5
G:EEC:A:LOSS	G:FN2_1_INV_B	1.78×10-5
G:EEC:A:LOSS	G:FN2_2_INV_A	1.82×10-5
G:EEC:A:LOSS	G:FN2_2_INV_B	1.80×10-5
G:EEC:A:LOSS	G:FN1_INV_A	2.09×10-5
G:EEC:A:LOSS	G:FN1_INV_B	2.13×10-5
G:EEC:B:LOSS	G:FN3_INV_A	2.10×10-5
G:EEC:B:LOSS	G:FN3_INV_B	2.05×10-5
G:EEC:B:LOSS	G:FN4_INV_A	1.85×10-5
G:EEC:B:LOSS	G:FN4_INV_B	1.82×10-5
G:EEC:B:LOSS	G:FN2_1_INV_A	1.79×10-5
G:EEC:B:LOSS	G:FN2_1_INV_B	1.80×10-5
G:EEC:B:LOSS	G:FN2_2_INV_A	1.81×10-5
G:EEC:B:LOSS	G:FN2_2_INV_B	1.81×10-5
G:EEC:B:LOSS	G:FN1_INV_A	2.12×10-5
G:EEC:B:LOSS	G:FN1_INV_B	2.10×10-5
G:FN3_INV_A	G:FN3_INV_B	1.72×10-5
G:FN3_INV_A	G:FN4_INV_A	1.23×10-5
G:FN3_INV_A	G:FN4_INV_B	1.26×10-5

Dual Fault Combination, i		$\lambda_{i,L}$
G:FN3_INV_A	G:FN2_1_INV_A	1.22×10-5
G:FN3_INV_A	G:FN2_1_INV_B	1.21×10-5
G:FN3_INV_A	G:FN2_2_INV_A	1.24×10-5
G:FN3_INV_A	G:FN2_2_INV_B	1.24×10-5
G:FN3_INV_A	G:FN1_INV_A	1.53×10-5
G:FN3_INV_A	G:FN1_INV_B	1.53×10-5
G:FN3_INV_B	G:FN4_INV_A	1.29×10-5
G:FN3_INV_B	G:FN4_INV_B	1.22×10-5
G:FN3_INV_B	G:FN2_1_INV_A	1.23×10-5
G:FN3_INV_B	G:FN2_1_INV_B	1.23×10-5
G:FN3_INV_B	G:FN2_2_INV_A	1.25×10-5
G:FN3_INV_B	G:FN2_2_INV_B	1.25×10-5
G:FN3_INV_B	G:FN1_INV_A	1.52×10-5
G:FN3_INV_B	G:FN1_INV_B	1.52×10-5
G:FN4_INV_A	G:FN4_INV_B	1.07×10-5
G:FN4_INV_A	G:FN2_1_INV_A	9.30×10-6
G:FN4_INV_A	G:FN2_1_INV_B	9.70×10-6
G:FN4_INV_A	G:FN2_2_INV_A	9.36×10-6
G:FN4_INV_A	G:FN2_2_INV_B	9.84×10-6
G:FN4_INV_A	G:FN1_INV_A	1.33×10-5
G:FN4_INV_A	G:FN1_INV_B	1.30×10-5
G:FN4_INV_B	G:FN2_1_INV_A	9.68×10-6
G:FN4_INV_B	G:FN2_1_INV_B	9.22×10-6
G:FN4_INV_B	G:FN2_2_INV_A	9.70×10-6
G:FN4_INV_B	G:FN2_2_INV_B	9.29×10-6
G:FN4_INV_B	G:FN1_INV_A	1.29×10-5
G:FN4_INV_B	G:FN1_INV_B	1.32×10-5
G:FN2_1_INV_A	G:FN2_1_INV_B	9.52×10-6
G:FN2_1_INV_A	G:FN2_2_INV_A	9.13×10-6
G:FN2_1_INV_A	G:FN2_2_INV_B	9.53×10-6
G:FN2_1_INV_A	G:FN1_INV_A	1.25×10-5
G:FN2_1_INV_A	G:FN1_INV_B	1.26×10-5
G:FN2_1_INV_B	G:FN2_2_INV_A	9.47×10-6
G:FN2_1_INV_B	G:FN2_2_INV_B	9.13×10-6
G:FN2_1_INV_B	G:FN1_INV_A	1.25×10-5
G:FN2_1_INV_B	G:FN1_INV_B	1.25×10-5
G:FN2_2_INV_A	G:FN2_2_INV_B	9.45×10-6
G:FN2_2_INV_A	G:FN1_INV_A	1.26×10-5
G:FN2_2_INV_A	G:FN1_INV_B	1.27×10-5
G:FN2_2_INV_B	G:FN1_INV_A	1.27×10-5
G:FN2_2_INV_B	G:FN1_INV_B	1.27×10-5
G:FN1_INV_A	G:FN1_INV_B	1.61×10-5

Appendix 7. The Results Obtained Using the Genetic Algorithm

Genetic Algorithm Results File

Generation 0

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	0 0 1 0 1 1 1 0 1 0 1	7.74	12.80	24900.00	10790.85	605.00	6866.64
1	0 0 1 0 0 0 0 1 0 1 0	13.40	7.32	22400.00	13587.67	300.00	-23213.05
2	0 0 0 0 1 1 0 0 0 0 1	9.73	14.20	24000.00	12311.40	515.00	593.15
3	1 0 1 0 1 0 1 1 1 1 0	10.30	8.79	22950.00	11937.09	295.00	2043.55
4	0 1 1 1 0 0 0 1 0 1 0	12.40	7.12	22800.00	13391.41	280.00	-12901.36
5	0 0 1 1 0 1 1 0 1 0 1	7.72	12.80	24900.00	10770.14	605.00	6895.53
6	1 0 0 0 1 1 0 0 0 0 0	12.80	10.20	23000.00	14148.73	370.00	-19088.17
7	0 0 0 0 1 0 0 0 0 1 1	8.98	6.99	22950.00	10111.27	335.00	9997.58
8	0 1 0 0 0 1 1 0 1 0 0	15.50	7.56	23200.00	17089.58	350.00	-52003.46
9	1 1 1 1 1 0 1 1 0 0 0	9.49	8.61	23400.00	11972.99	300.00	6024.50

Overall Fittest Individual Occurred in Generation 0

7	0 0 0 0 1 0 0 0 0 1 1	8.98	6.99	22950.00	10111.27	335.00	9997.58

Generation 1

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 1 0 0 1 0 1 1	1.50	13.20	24950.00	7575.34	395.00	12125.38
1	0 1 1 0 1 0 1 1 0 0 0	12.80	7.09	22800.00	38657.71	280.00	-41777.23
2	1 1 1 1 1 0 0 0 1 0 1	3.11	9.75	24650.00	11606.93	375.00	10390.30
3	0 1 1 1 0 1 0 0 0 0 0	15.20	8.18	23200.00	46524.65	350.00	-77633.32
4	0 0 0 1 0 1 1 0 1 0 1	9.16	12.70	24600.00	30130.46	575.00	-14468.98
5	0 0 1 1 0 0 0 0 0 1 1	8.63	6.64	23250.00	26891.48	365.00	-5478.58
6	0 0 0 0 1 1 0 0 0 0 1	9.73	14.20	24000.00	12311.40	515.00	593.15
7	1 0 1 0 0 1 1 0 0 0 0	10.10	9.33	23400.00	31525.74	410.00	-16497.51
8	0 0 1 1 0 1 1 0 0 0 1	7.72	12.80	24900.00	10770.14	605.00	6895.53
9	0 0 1 0 0 1 1 1 1 0 1	2.26	10.00	25500.00	8864.20	600.00	11655.72

Overall Fittest Individual Occurred in Generation 1

0	1 1 1 0 1 0 0 1 0 1 1	1.50	13.20	24950.00	7575.34	395.00	12125.38

Generation 2

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 1 0 0 1 1 1 1	1.48	13.00	25250.00	7641.06	370.00	12519.21
1	1 1 1 0 1 0 0 1 0 1 0	10.40	8.64	23200.00	33102.03	280.00	-19279.58
2	0 0 1 1 1 1 0 1 1 1 1	2.14	10.50	25500.00	8614.65	600.00	11547.40
3	1 1 1 0 1 1 1 0 0 1 1	1.75	17.10	26000.00	7618.91	585.00	8006.80
4	1 0 1 1 0 1 1 1 1 0 1	0.32	10.90	26300.00	3341.17	670.00	15658.47
5	1 1 1 0 1 0 0 1 0 1 1	1.50	13.20	24950.00	7575.34	395.00	12125.38
6	1 1 1 0 1 0 0 1 1 0 1	1.48	13.00	25250.00	7643.06	370.00	12517.21
7	1 1 1 1 1 0 0 0 1 1 1	3.10	9.75	24650.00	11575.46	375.00	10421.77
8	1 0 1 1 0 0 0 0 0 1 1	3.42	10.00	23750.00	11846.03	405.00	9075.54
9	1 1 0 0 1 0 0 0 1 0 1	3.72	10.40	24050.00	13317.16	315.00	7618.79

Overall Fittest Individual Occurred in Generation 2

4	1 0 1 1 0 1 1 1 1 0 1	0.32	10.90	26300.00	3341.17	670.00	15658.47

Generation 3

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 1 0 0 1 1 1 1	1.48	13.00	25250.00	7641.06	370.00	12519.21
1	1 1 1 1 1 0 1 0 1 1 1	3.10	9.75	24650.00	11575.46	375.00	10421.77
2	0 0 1 1 1 1 0 1 1 1 1	2.14	10.50	25500.00	8614.65	600.00	11547.40
3	1 0 1 1 0 1 1 0 1 0 1	1.81	17.40	25400.00	7332.68	645.00	6075.50
4	1 1 0 0 1 0 0 0 1 0 1	3.72	10.40	24050.00	13317.16	315.00	7618.79
5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
6	1 0 1 0 1 0 1 1 0 1 1	1.52	13.70	24350.00	6904.56	455.00	11779.97
7	1 0 0 0 1 0 0 0 1 0 1	3.85	10.70	23450.00	13079.14	375.00	7039.08
8	1 1 1 0 1 0 1 1 1 0 1	1.48	13.00	25250.00	7643.06	370.00	12517.21
9	1 0 1 1 0 1 1 1 0 1 1	0.31	11.10	26000.00	3260.45	695.00	14621.55

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Genetic Algorithm Results File

Generation 4

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
1	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
2	1 0 1 1 0 1 1 0 0 1 1	1.82	17.40	25400.00	7360.28	645.00	6047.90
3	0 0 0 1 1 1 1 1 0 1 1	2.81	10.30	25500.00	10669.59	655.00	8350.59
4	0 0 1 1 1 0 0 1 1 1 1	2.31	10.50	24450.00	8542.87	420.00	12720.83
5	1 0 1 1 0 1 0 1 0 1 0	4.15	9.25	23500.00	14099.17	440.00	7092.39
6	1 0 0 0 1 0 0 0 0 0 1	3.85	10.70	23450.00	13079.14	375.00	7039.08
7	1 1 1 0 1 0 0 1 1 1 1	1.48	13.00	25250.00	7641.06	370.00	12519.21
8	1 1 1 0 1 0 0 0 0 1 1	3.41	9.72	24350.00	12412.63	345.00	9305.52
9	1 1 1 1 1 0 0 1 1 1 1	1.27	12.80	25550.00	7008.70	400.00	13610.12

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Generation 5

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 0 1 1 1 0 0 0	3.49	8.64	24300.00	12719.99	440.00	9678.01
1	1 1 1 1 0 0 1 1 1 0 1	1.41	12.90	25250.00	7414.05	370.00	12825.62
2	1 1 1 0 0 0 0 1 1 1 1	1.63	13.10	24950.00	8036.25	340.00	11744.37
3	1 1 1 1 1 1 1 1 1 0 1	0.28	10.50	27200.00	3809.51	640.00	17136.82
4	1 1 1 1 0 0 1 1 1 1 1	1.41	12.90	25250.00	7400.55	370.00	12839.12
5	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
6	1 1 1 1 0 1 1 1 0 0 1	0.28	10.90	26600.00	3711.10	635.00	16463.26
7	1 0 1 1 0 0 1 1 1 0 1	1.45	13.40	24650.00	6789.85	430.00	12450.35
8	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
9	1 1 0 1 1 1 0 1 1 0 1	0.36	11.30	26300.00	4257.29	550.00	17095.70

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Generation 6

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 1 0 1 1 1 0 1	1.28	12.80	25550.00	7022.96	400.00	13595.86
1	1 0 1 0 0 0 0 1 0 1 1	1.67	13.80	24050.00	7298.57	425.00	11017.72
2	1 1 1 1 1 1 1 0 0 0 1	1.71	17.00	26300.00	7538.56	615.00	7830.02
3	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
4	1 1 0 1 1 1 0 1 0 0 1	0.36	11.60	26000.00	4189.81	575.00	16171.51
5	1 0 1 1 0 0 1 1 1 1 1	1.44	13.40	24650.00	6780.16	430.00	12460.03
6	1 1 0 1 0 1 1 1 1 1 1	0.33	10.90	26600.00	4024.76	580.00	17360.69
7	1 1 1 1 0 0 1 1 1 1 1	1.41	12.90	25250.00	7400.55	370.00	12839.12
8	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
9	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Generation 7

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 0 1 0 1 1 1 0 1 1	0.33	11.10	26300.00	3943.82	605.00	16469.49
1	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
2	1 1 0 1 0 1 1 1 1 1 1	0.33	10.90	26600.00	4024.76	580.00	17360.69
3	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
4	1 1 1 1 0 1 1 0 1 1 1	1.73	17.10	26000.00	7554.30	585.00	8071.41
5	0 1 1 1 0 1 1 1 1 1 1	2.31	9.64	26000.00	9349.21	590.00	12134.47
6	1 0 0 0 1 0 0 1 0 1 1	1.58	14.10	24050.00	7045.02	425.00	11025.90
7	1 1 1 1 1 1 1 1 1 0 1	0.28	10.50	27200.00	3809.51	640.00	17136.82
8	1 0 1 1 0 0 1 1 1 1 1	1.44	13.40	24650.00	6780.16	430.00	12460.03
9	1 1 1 1 0 0 1 1 0 1 1	1.43	13.10	24950.00	7335.18	395.00	12445.44

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Generation 8

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	0 1 0 1 1 0 1 1 1 1 1	2.59	10.70	24350.00	9587.16	350.00	11431.06

Genetic Algorithm Results File

1	1 1 1 0 0 1 1 1 0 0 1	0.28	10.90	26300.00	3695.07	605.00	16866.25
2	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
3	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
4	1 1 0 1 0 1 0 1 0 1 1	0.36	11.70	25700.00	4166.33	545.00	16371.39
5	1 1 0 0 1 0 1 1 1 1 1	1.52	13.40	24950.00	7740.11	340.00	11800.09
6	0 1 0 1 1 1 1 0 1 1 1	9.45	12.00	25100.00	31791.94	565.00	-15823.62
7	1 1 0 1 0 1 1 1 1 1 1	0.33	10.90	26600.00	4024.76	580.00	17360.69
8	1 1 0 0 0 1 1 1 0 1 1	0.33	11.20	26000.00	3919.41	575.00	16742.74
9	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Overall Fittest Individual Occurred in Generation 3

5	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47

Generation 9

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	0 1 0 1 1 0 1 1 1 1 1	2.59	10.70	24350.00	9587.16	350.00	11431.06
1	1 1 1 1 0 1 1 1 1 0 1	0.28	10.60	26900.00	3788.29	610.00	17482.47
2	1 1 1 1 0 1 1 1 0 1 0	3.52	8.63	24300.00	12798.92	440.00	9605.62
3	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
4	1 1 1 0 0 0 1 0 0 0 1	3.63	9.70	24050.00	12952.52	315.00	8479.57
5	1 1 1 0 0 1 0 1 0 0 1	0.32	11.50	25700.00	3962.71	545.00	16726.59
6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06
7	0 1 0 1 1 0 1 0 0 1 1	10.10	6.13	23450.00	31731.00	325.00	-14730.63
8	0 1 1 0 0 1 1 1 0 0 1	2.18	10.00	25400.00	8656.15	585.00	12078.30
9	1 1 0 1 0 1 0 1 0 1 1	0.36	11.70	25700.00	4166.33	545.00	16371.39

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 10

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06
1	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
2	1 1 1 1 0 1 1 1 0 0 1	0.28	10.90	26600.00	3711.10	635.00	16463.26
3	1 1 1 0 1 0 0 1 0 0 1	1.50	13.20	24950.00	7563.51	395.00	12137.21
4	1 0 0 0 0 1 1 1 0 0 1	0.38	11.50	25400.00	3482.39	635.00	15044.30
5	1 0 0 1 1 1 0 1 1 1 1	0.40	11.60	25700.00	3811.46	610.00	15516.30
6	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
7	1 1 1 0 0 0 0 1 1 1 1	1.63	13.10	24950.00	8036.25	340.00	11744.37
8	1 1 1 1 0 0 1 1 1 1 1	1.41	12.90	25250.00	7400.55	370.00	12839.12
9	0 1 1 0 0 1 1 1 1 1 1	2.27	9.73	25700.00	9080.07	560.00	12629.90

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 11

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 1 0 1 1 1 1 1	1.48	13.00	25250.00	7641.06	370.00	12519.21
1	1 1 1 1 0 0 0 1 1 1 0	10.10	8.37	23350.00	32455.49	255.00	-16841.69
2	1 1 1 1 0 0 1 1 0 1 1	1.43	13.10	24950.00	7335.18	395.00	12445.44
3	1 1 1 0 0 0 0 1 1 1 1	1.63	13.10	24950.00	8036.25	340.00	11744.37
4	1 1 1 1 0 0 1 1 1 0 1	1.41	12.90	25250.00	7414.05	370.00	12825.62
5	1 1 1 1 0 1 1 1 1 0 0	3.59	8.35	24450.00	13143.29	415.00	9592.21
6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06
7	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
8	0 0 0 0 0 1 0 1 1 1 0	8.89	8.45	23050.00	27627.81	375.00	-8073.95
9	1 1 1 0 0 0 0 1 1 1 0	10.90	8.48	23150.00	34310.44	235.00	-23234.07

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 12

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 0 0 0 1 1 1 1	1.63	13.10	24950.00	8036.25	340.00	11744.37
1	0 0 1 0 0 1 0 1 1 1 1	2.09	10.70	24900.00	8246.51	540.00	12321.71
2	1 1 0 0 0 1 1 1 0 1 1	0.33	11.20	26000.00	3919.41	575.00	16742.74
3	0 0 1 1 0 0 0 1 1 1 1	2.41	10.60	24150.00	8742.63	390.00	12148.49
4	0 1 1 0 1 1 1 1 1 1 1	2.30	9.67	26000.00	9283.89	590.00	12178.95
5	1 1 1 1 0 0 1 1 0 1 1	1.43	13.10	24950.00	7335.18	395.00	12445.44
6	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
7	1 1 0 1 0 1 0 1 1 1 1	0.36	11.40	26000.00	4252.64	520.00	17210.56
8	1 1 1 1 1 1 1 1 1 1 0	3.55	8.31	24650.00	13134.99	435.00	9826.05

Genetic Algorithm Results File

9	0 0 1 0 1 1 0 1 1 0 1	2.11	10.60	25200.00	8420.82	570.00	11980.70

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 13

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 0 0 1 1 1 1 0 1 1	0.33	11.20	26300.00	3934.21	605.00	16404.63
1	1 1 0 1 0 1 0 1 0 1 1	0.36	11.70	25700.00	4166.33	545.00	16371.39
2	1 1 0 1 0 1 0 1 1 1 1	0.36	11.40	26000.00	4252.64	520.00	17210.56
3	1 1 1 1 0 1 1 1 1 1 1	0.28	10.60	26900.00	3795.42	610.00	17475.34
4	1 1 1 1 1 1 1 1 1 1 0	3.55	8.31	24650.00	13134.99	435.00	9826.05
5	0 1 0 0 0 0 1 1 0 1 1	2.82	11.30	23450.00	9876.50	315.00	9797.71
6	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
7	1 1 0 1 0 1 1 0 1 1 1	1.97	17.80	25700.00	8369.08	555.00	6911.84
8	1 1 0 0 0 1 1 1 0 1 1	0.33	11.20	26000.00	3919.41	575.00	16742.74
9	1 1 0 0 0 1 1 1 0 1 1	0.33	11.20	26000.00	3919.41	575.00	16742.74

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 14

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 0 0 1 1 0 1 1	1.43	13.10	24950.00	7335.18	395.00	12445.44
1	1 1 0 0 0 1 0 1 1 1 1	0.37	11.50	25700.00	4234.85	490.00	17242.64
2	1 1 0 1 0 1 0 1 0 1 1	0.36	11.70	25700.00	4166.33	545.00	16371.39
3	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
4	1 1 0 1 1 1 1 1 1 1 1	0.33	10.80	26900.00	4036.21	610.00	17088.41
5	1 1 0 1 0 1 0 1 1 1 0	4.32	8.99	23850.00	15248.52	355.00	6468.20
6	1 1 0 0 1 1 1 1 0 1 1	0.33	11.20	26300.00	3934.21	605.00	16404.63
7	1 0 0 1 0 1 0 1 1 1 1	0.41	11.70	25400.00	3803.26	580.00	15782.93
8	1 1 0 0 0 1 1 1 1 1 1	0.34	11.00	26300.00	4005.17	550.00	17571.22
9	1 0 1 1 0 1 1 1 1 1 1	0.32	10.90	26300.00	3343.78	670.00	15655.85

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 15

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 0 1 0 1 0 0 1 1 1	2.54	19.30	25100.00	10151.48	495.00	4075.29
1	1 0 0 1 0 1 0 0 1 1 1	2.66	19.70	24500.00	10003.18	555.00	2347.90
2	1 1 0 0 0 0 1 1 0 1 1	1.69	13.80	24350.00	8084.16	335.00	10532.12
3	0 1 0 1 0 1 0 1 1 1 1	2.64	10.40	25100.00	10280.32	500.00	11291.55
4	1 0 1 1 0 1 1 1 1 1 1	0.32	10.90	26300.00	3343.78	670.00	15655.85
5	0 1 0 0 0 0 1 1 1 1 1	2.82	11.00	23750.00	10053.43	290.00	10144.17
6	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
7	0 1 0 0 0 1 1 0 1 1 1	9.56	12.20	24500.00	31357.86	505.00	-15569.36
8	0 0 0 1 0 0 0 1 0 1 1	2.72	11.30	23550.00	9477.42	385.00	10296.79
9	1 1 1 0 1 1 1 1 1 1 1	0.28	10.60	26900.00	3790.59	610.00	17480.16

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 16

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 0 0 1 0 1 1 1 1 1 1	0.38	11.20	26000.00	3593.71	640.00	15637.89
1	0 1 0 1 0 1 0 1 1 1 1	2.64	10.40	25100.00	10280.32	500.00	11291.55
2	1 1 1 0 1 1 1 1 1 1 0	3.60	8.37	24450.00	13170.95	415.00	9551.74
3	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
4	1 0 1 1 0 1 1 0 0 1 1	1.82	17.40	25400.00	7360.28	645.00	6047.90
5	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
6	1 0 1 0 1 1 0 1 1 1 1	0.35	11.40	25700.00	3569.11	610.00	15909.67
7	1 0 1 1 0 0 1 1 0 1 1	1.46	13.60	24350.00	6690.94	455.00	12074.68
8	1 1 0 0 0 1 0 1 0 1 1	0.37	11.70	25400.00	4169.21	515.00	16539.22
9	1 1 1 1 0 0 1 1 1 0 1	1.41	12.90	25250.00	7414.05	370.00	12825.62

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Genetic Algorithm Results File

Generation 17

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 0 1 1 0 0 1 0 1 1 1	3.42	10.00	23750.00	11846.03	405.00	9075.54
1	1 1 1 1 0 1 0 1 1 0 1	0.31	11.10	26300.00	4039.23	550.00	17462.99
2	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
3	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
4	0 1 1 0 0 1 0 1 0 1 1	2.02	10.70	24800.00	8085.92	525.00	12621.58
5	1 1 0 1 0 1 0 1 0 1 1	0.36	11.70	25700.00	4166.33	545.00	16371.39
6	1 0 1 0 0 0 0 1 1 0 1	1.65	13.60	24350.00	7358.83	400.00	11419.88
7	1 0 1 0 0 1 0 1 1 1 0	4.28	9.05	23450.00	14531.21	395.00	6745.33
8	1 0 0 0 1 1 0 1 1 1 0	4.65	9.21	23450.00	15682.40	395.00	5486.28
9	1 0 1 1 0 1 0 1 0 1 1	0.35	11.60	25400.00	3490.63	635.00	14960.41

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 18

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 0 1 0 1 1 0 1	0.31	11.10	26300.00	4039.23	550.00	17462.99
1	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
2	1 1 0 1 0 1 1 1 1 1 1	0.33	10.90	26600.00	4024.76	580.00	17360.69
3	1 1 1 1 1 1 0 1 0 1 1	0.31	11.30	26300.00	3964.54	605.00	16299.54
4	0 1 1 1 0 1 1 1 0 1 1	2.22	9.91	25700.00	8886.12	615.00	11566.60
5	0 1 1 0 0 1 0 1 0 1 1	2.02	10.70	24800.00	8085.92	525.00	12621.58
6	1 0 1 1 0 1 1 1 0 1 1	0.31	11.10	26000.00	3260.45	695.00	14621.55
7	1 1 1 1 0 0 0 1 1 0 0	10.10	8.37	23350.00	32413.58	255.00	-16799.77
8	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
9	1 0 1 1 0 1 1 1 0 1 1	0.31	11.10	26000.00	3260.45	695.00	14621.55

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 19

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 1 0 1 0 0 1 0 1	2.26	18.60	25400.00	9260.97	525.00	5498.32
1	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
2	1 1 1 1 0 1 0 1 1 0 1	0.31	11.10	26300.00	4039.23	550.00	17462.99
3	1 1 1 1 0 0 0 1 1 1 1	1.41	12.90	25250.00	7400.55	370.00	12839.12
4	1 1 1 1 0 0 0 1 1 0 1	1.41	12.90	25250.00	7414.05	370.00	12825.62
5	1 1 1 0 0 1 0 1 1 0 1	0.32	11.20	26000.00	4036.98	520.00	17576.06
6	1 1 1 0 0 1 0 1 0 1 1	0.32	11.50	25700.00	3969.08	545.00	16720.22
7	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
8	1 1 1 1 0 1 0 1 0 1 1	0.31	11.40	26000.00	3959.45	575.00	16552.87
9	0 1 0 1 0 1 1 1 1 1 1	2.89	9.77	25700.00	11188.47	560.00	10493.58

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06

Generation 20

Individual	String	LOTC	Delays	Revenue	Mt Cost	Weight	Fitness
0	1 1 1 0 0 1 0 1 1 0 1	0.32	11.20	26000.00	4036.98	520.00	17576.06
1	1 0 1 1 0 0 0 0 1 0 1	3.42	10.00	23750.00	11860.62	405.00	9060.95
2	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
3	1 1 1 1 0 1 0 0 1 1 1	2.25	18.60	25400.00	9233.33	525.00	5525.96
4	0 1 1 0 0 1 0 1 1 0 1	2.08	10.40	25100.00	8426.85	500.00	13145.02
5	1 1 1 1 0 1 0 1 1 0 1	0.31	11.10	26300.00	4039.23	550.00	17462.99
6	1 1 1 1 0 1 0 1 1 1 1	0.31	11.10	26300.00	4034.86	550.00	17467.36
7	1 1 0 1 0 1 0 1 1 1 1	0.36	11.40	26000.00	4252.64	520.00	17210.56
8	1 1 1 0 0 1 0 1 1 0 1	0.32	11.20	26000.00	4036.98	520.00	17576.06
9	1 1 1 0 0 0 0 1 1 0 1	1.63	13.10	24950.00	8064.11	340.00	11716.51

Overall Fittest Individual Occurred in Generation 9

6	1 1 1 0 0 1 0 1 1 1 1	0.32	11.20	26000.00	4028.98	520.00	17584.06