Quantitative Probabilistic Modeling of Environmental Control and Life Support System Resilience for Long-Duration Human Spaceflight

by

Andrew Charles Owens

Submitted to the Department of Aeronautics and Astronautics on August 21, 2014, in partial fulfillment of the requirements for the degree of Master of Science in Aeronautics and Astronautics

Abstract

The future of human space exploration will see crews travel farther and remain in space for longer durations than ever before. For the first time in the history of human spaceflight, the Environmental Control and Life Support Systems (ECLSS) that sustain the crew in their habitat will have to function without rapid resupply or abort-to-Earth capability in the event of an emergency. In this environment, reliability and resilience will become more dominant design drivers, and will need to be considered alongside traditional system metrics such as mass and cost early in the design process in order to select the optimal ECLSS design for a given mission. This thesis presents the use of semi-Markov process (SMP) models to quantify the resilience of long-duration ECLSS. An algorithm is defined to translate ECLSS design data - including system architecture, buffer sizes, and component reliability information - into an SMP and then use that SMP to calculate resilience metrics such as the probability of system failure before the end of mission and the number of spares for each component that are required to achieve a certain probability of success. This methodology is demonstrated on a notional ECLSS, and then used to determine logistics requirements for a Mars One surface habitat Life Support Unit and examine the trade between resupply mass and the probability that sufficient spares are supplied. This case study found that, if sparing is performed at the processor level, 10,410 kg of spares would have to be provided in each resupply mission in order to provide a probability greater than 0.999 that sufficient spares are available to complete all required repairs. This is equivalent to over 75% of the mass of consumables that would be required to sustain an open loop system for the same duration. When coupled with the increased uncertainty associated with regenerative systems, the low mass savings associated with the selection of regenerative rather than open loop indicate that, at current reliability levels and with spares implemented at the processor level, regenerative ECLSS may not be the optimal design choice for a given mission. The SMP methodology described in this thesis provides an analytical means to quantify
system resilience based on system design data, thereby facilitating the use of formal multiobjective optimization methods and trade studies to create ECLSS with the appropriate balance between mass and resilience for a given mission.

Thesis Supervisor: Olivier L. de Weck
Title: Professor of Aeronautics and Astronautics and Engineering Systems
Acknowledgments

I would like to acknowledge all of the people that have made this thesis possible, and give thanks to:

My advisor, Prof. Olivier de Weck, who welcomed me into his research group here at MIT two years ago and has given me the guidance, insight, and resources to thrive in this research while simultaneously giving me the opportunity to explore and develop my own research topic.

Gordon Aaseng, who hosted me at Johnson Space Center during the summer of 2013 and helped me gain experience with fault management techniques and ECLSS technology.

Jennifer Green, who made my time at JSC possible and provided several references and resources for reliability analysis.

Sydney, Sam, and Koki, who provided the idea for the Mars One case study and helped with the development of processor data and other system information.

Margaret, Ioana, and Sydney - my friends and labmates - who have provided countless hours of collaboration and discussion, bouncing ideas around and giving feedback and suggestions on everything from overall research directions to low-level code structures.

My friends in AeroAstro and across the MIT community, who make this the amazing, exciting, intellectually stimulating, and fun place that it is.

Finally, I would like to thank my family - especially my parents and brother - for all their encouragement and support over the years. I could not have done this without you.

This research has been supported by funding from the Lemelson Minority Engineering Fellowship, ODGE Diversity Fellowship, MIT Skoltech Initiative, Department of Aeronautics and Astronautics Unified Engineering Teaching Assistantship, and Sustainable Solar Desalination Project.
Contents

1 Introduction ................................................................. 17
  1.1 Motivation ............................................................... 17
    1.1.1 Reliability and Resilience .................................. 18
    1.1.2 Environmental Control and Life Support Systems ........ 19
    1.1.3 The Need for Resilient ECLSS ............................... 21
  1.2 Literature Review: Risk Assessment ............................... 23
    1.2.1 Risk Matrices ................................................. 24
    1.2.2 Fault Trees ................................................... 25
    1.2.3 Reliability Block Diagrams ................................. 27
    1.2.4 Failure Modes, Effects, and Criticality Analysis ....... 29
    1.2.5 Probabilistic Risk Assessment .............................. 30
    1.2.6 Expected Productivity Analysis ............................ 31
    1.2.7 Systems-Theoretic Process Analysis ....................... 33
    1.2.8 Limitations of Current Methods ............................ 35
  1.3 Research Objectives and Thesis Outline ......................... 37
  1.4 Chapter 1 Summary .................................................. 38

2 Methodology .............................................................. 41
  2.1 Overview of Semi-Markov Processes ............................... 41
    2.1.1 Relation to Markov Chains .................................. 42
    2.1.2 History ....................................................... 42
  2.2 SMP Formulation and Solution ..................................... 43
    2.2.1 Nomenclature ............................................... 43
Chapter 2: Model Formulation

2.2.2 Model Formulation ............... 43
2.2.3 SMP Solution in the Laplace Domain ............... 45

2.3 Application to ECLSS ............... 47

2.3.1 ECLSS Generalization ............... 48
2.3.2 Modeling Assumptions ............... 50
2.3.3 ECLSS SMP Model Formulation ............... 51
2.3.4 Interpretation of Results ............... 58

2.4 Chapter 2 Summary ............... 62

3 Demonstration of Analysis

3.1 System Description and Analysis Goals ............... 63

3.1.1 Goal ............... 64
3.1.2 Assumptions and Parameters ............... 64

3.2 Model Formulation ............... 65

3.2.1 Basic Event Definition ............... 65
3.2.2 State Network Definition ............... 67
3.2.3 Transition PDF Definition ............... 68

3.3 Analysis and Results ............... 72

3.3.1 WT Sizing ............... 72
3.3.2 Number of Spares ............... 74
3.3.3 Final System ............... 76

3.4 Discussion ............... 77

3.4.1 Combined Effects ............... 77

3.5 Incorporation of Contingency Responses ............... 79

3.5.1 Background: ISS Pump Module Failure ............... 80
3.5.2 SMP Representation ............... 81

3.6 Chapter 3 Summary ............... 82

4 Case Study: Mars One

4.1 Background ............... 85
4.2 Analysis Goal ............... 86
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Normalized cost as a function of normalized MTBF [1].</td>
<td>19</td>
</tr>
<tr>
<td>1-2</td>
<td>Effect of increased mission duration on probability of failure. Data are shown using 3 component families - civil aviation, 2010s Unmanned Aerial Vehicles (UAVs), and 1990s UAVs - representing high, intermediate, and low reliability respectively [2].</td>
<td>20</td>
</tr>
<tr>
<td>1-3</td>
<td>Functional diagram of the ISS ECLSS [3].</td>
<td>22</td>
</tr>
<tr>
<td>1-4</td>
<td>Risk matrix template [4].</td>
<td>24</td>
</tr>
<tr>
<td>1-5</td>
<td>Example of basic fault tree structure [4].</td>
<td>26</td>
</tr>
<tr>
<td>1-6</td>
<td>Modified Reliability Block Diagram, showing blocks 10 though 13 as buffers for subsystems A through D [5].</td>
<td>28</td>
</tr>
<tr>
<td>1-7</td>
<td>Structure of a PRA event sequence diagram [6].</td>
<td>31</td>
</tr>
<tr>
<td>1-8</td>
<td>Example of Markov chain used in EPA, showing the various states and transition rates leading to system failure [7].</td>
<td>33</td>
</tr>
<tr>
<td>1-9</td>
<td>Feedback control structure used in STPA, showing a control algorithm (which includes a model of the process it is controlling), the process to be controlled, and the exchange of control actions and feedback [8].</td>
<td>34</td>
</tr>
<tr>
<td>1-10</td>
<td>Thesis roadmap, showing interactions between the six chapters and three appendices.</td>
<td>38</td>
</tr>
<tr>
<td>2-1</td>
<td>A three-state system with three transitions. States are denoted by numbered circles, while transitions are indicated by arrows.</td>
<td>43</td>
</tr>
<tr>
<td>2-2</td>
<td>Diagram of generalized ECLSS elements, showing processors (rectangles), buffers (cylinders), and flow of consumables (arrows).</td>
<td>49</td>
</tr>
</tbody>
</table>
2-3 SMP diagram for the system shown in figure 2-2. A corresponding state network outline is shown in table 2.2. Processor failures are shown in black, and repairs are shown in blue. Buffer depletion is shown in red.

2-4 Notional Markov renewal process probabilities, showing the CDF of the number of spares required for a component over a mission timeline.

3-1 Block diagram of system used in this analysis.

3-2 SMP diagram for this system. Processor failures are shown in black, and repairs are shown in blue. Buffer depletion is shown in red. States and transitions correspond to the state network outline shown in table 3.2.

3-3 Decrease in the probability of failure due to increased buffer size.

3-4 CDF of the number of spares required for the OGS.

3-5 CDF of the number of spares required for the WRS.

3-6 CDF of the number of spares required for the PDU.

3-7 Combined effect of increasing buffer size and the number of WRS spares.

3-8 SMP diagram showing the use of contingency operations to stabilize the ISS after the 2010 ammonia pump module failure. Processor failures are shown in black, repairs are shown in blue, buffer depletions are shown in red, and contingency operations are shown in green.

4-1 Artist’s rendering of the Mars One surface colony [9].

4-2 Major components of the Mars One LSU air and water management systems (based on ISS ECLSS) and their interactions [10].

4-3 CDF of the number of SPWE spares required.

4-4 CDF of the number of 4BMS spares required.

4-5 CDF of the number of Sabatier spares required.

4-6 CDF of the number of VCD spares required.

4-7 CDF of the number of CCAA spares required.

4-8 CDF of the number of MF spares required.
4-9 Tradespace of spares probability and mass of spares. The minimum mass architectures that achieve a spares probability of 0.95, 0.99, and 0.999 are shown by the blue, green, and red dots, respectively. The minimum mass and the number of spares for each processor for the three probability thresholds are shown in table 4.4.

4-10 Changes to tradespace from a 10% increase in MTBF for all processors. Baseline data are plotted as before; the increased MTBF case is shown in gray, with minimum mass points represented by blue, green, and red triangles for spares probability of 0.95, 0.99, and 0.999, respectively.

5-1 Bathtub curve failure rate model, showing changes in the failure rate of a component over its lifetime [11].

6-1 Graphical system representation used for OPDs, with corresponding verbal syntax [12].
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Symbols, names, and descriptions of SMP metrics [13]</td>
<td>46</td>
</tr>
<tr>
<td>2.2</td>
<td>State network outline for the system shown in figure 2-2. A corres-</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>ponding SMP diagram is shown in figure 2-3.</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Basic events and their distributions. All distribution parameters are</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>based on a timescale of hours.</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>State network outline for the system shown in figure 3-1. The corre-</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>sponding SMP diagram is shown in figure 3-2. The transition ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number indicates the distribution describing the transition, referring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to table 3.3</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Transitions in the state network, showing the combinations of RVs</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>used to create them and their ID number (as used in table 3.2). Each</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PDF can be generated by taking the difference of the distributions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>indicated by $C_+$ and $C_-$ and truncating to positive values of $t$</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Air and water processor data, including mass and MTBF.</td>
<td>88</td>
</tr>
<tr>
<td>4.2</td>
<td>Basic events and their distributions. All distribution parameters are</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>based on a timescale of days.</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>RV combinations used to generate transitions in the state network</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>(with transition ID number).</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Minimum mass architectures above a set of spares probability thresholds.</td>
<td>97</td>
</tr>
<tr>
<td>4.5</td>
<td>Minimum mass architectures above a set of spares probability threshold</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>s for an increased MTBF system.</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Resupply spares mass savings for a 10% increase in MTBF.</td>
<td>98</td>
</tr>
</tbody>
</table>
B.1 Mars One state network outline and adjacency data.  . . . . . . . . . . 129
Chapter 1

Introduction

1.1 Motivation

In the 53 years since Yuri Gagarin first rode into space, over 500 others have followed in his footsteps - some spending as many as 438 consecutive days in orbit [14, 15]. Humanity has built space stations, conducted countless experiments in orbit, and even walked on the face of the Moon. However, all of these accomplishments have taken place in the immediate neighborhood of Earth; the farthest away any human crew has been is a quarter of a million miles away, just past the Moon [15]. For the International Space Station (ISS), resupply of consumables and replacement parts has been available on a regular basis, and the Earth has never been more than a few days’ journey away in the event of an emergency [16]. The safe return of the crew of Apollo 13 to the Earth after catastrophic failure of their spacecraft’s oxygen tank was only possible due to the skill and ingenuity of the crew and engineers on the ground and the fact that the spacecraft could return to Earth in only a few days’ time [17].

The future of human spaceflight will see crews travel farther and remain in space for longer durations than ever before. In particular, when humans first leave the Earth-Moon system - for example, to explore our nearest neighbor, Mars - resupply logistics will become much more difficult, and an abort-to-Earth in the event of an emergency may not be feasible [18, 19, 3]. As a result, the systems that support human exploration will face greater challenges than ever before, “requir[ing] lighter
mass while being subjected to higher loads and more extreme service conditions over longer time periods than present generation vehicles. The requirements placed on systems and subsystems... will be greater than previously experienced while demands on long-term reliability will increase” [20].

1.1.1 Reliability and Resilience

Here it is important to define two related but distinct characteristics of systems: reliability and resilience. Reliability is defined as the “probability that [an] item will perform its required function under given conditions for a stated time interval” [21]. This is often characterized with a Mean Time Between Failures (MTBF) value, which indicates the expected time that a component will operate before it fails. In contrast, resilience is defined as the capability of a system to adapt to changing circumstances - such as component failure or changes in external environment - in order to maintain functionality [22]. Effectively, reliability is concerned with the prevention of failure, while resilience is concerned with the optimization of performance in the presence of failure. Resilience enables the graceful degradation of systems - that is, the transition through a series of degraded but still partially functional states before complete system failure.

Both reliability and resilience are important in system design. However, as mission durations increase it becomes less feasible to design systems constructed from components of such high reliability that they will effectively never fail. Even at extremely high levels of reliability, the probability of failure increases as mission duration increases, as shown in figure 1-2 [2]. In addition, the cost of improving the reliability of a component increases asymptotically beyond a certain point. This effect, illustrated in figure 1-1, is due to the fact that beyond a certain point a fundamental limit exists to the reliability of a component as a result of technological, physical, and material limits. In addition, the complexity of a system has a strong impact on the cost of reliability - the cost of increasing the MTBF of a complex element increases much more quickly than the cost for a simple element [1]. In these conditions, partially failed states must be considered in the system design process and the resilience of
the system - its ability to maintain functionality in the face of component failure - becomes a key consideration [2].

1.1.2 Environmental Control and Life Support Systems

The resilience of space habitation and Environmental Control and Life Support Systems (ECLSS) are of particular importance, since they are directly responsible for sustaining the crew. The primary functions of ECLSS include [23]:

- **Atmosphere Management**: Oxygen (O₂) supply, carbon dioxide (CO₂) and trace contaminant removal, pressure/temperature/humidity regulation, ventilation, and gas storage/distribution
- **Water Management**: Potable water (H₂O) supply, water supply for hygiene and other purposes, wastewater collection/treatment/filtering, water storage/distribution
- **Waste Management**: trash/metabolic waste collection and waste product storage/processing/disposal
- **Food Management**: Food provision, storage, and preparation
Figure 1-2: Effect of increased mission duration on probability of failure. Data are shown using 3 component families - civil aviation, 2010s Unmanned Aerial Vehicles (UAVs), and 1990s UAVs - representing high, intermediate, and low reliability respectively [2].
As this list demonstrates, ECLSS involve a variety of physical, chemical, and biological processes acting on a variety of consumables such as O$_2$, CO$_2$, H$_2$O, and food. For many past missions, ECLSS have utilized an open loop configuration. In open loop systems, external resources (either stored in the initial system or brought by resupply vehicles) are consumed gradually over the course of the mission and waste products are discarded rather than being recycled. This paradigm can result in very simple and reliable systems, but involves a steeply increasing cost for longer missions - the longer a system must go without resupply, the higher the initial mass of that system will be. In contrast, closed loop systems employ regenerative technologies to recycle resources, thus reducing the overall mass of consumables required to support a given mission. Systems may employ different levels of loop closure; for example, the ISS ECLSS processes urine to produce water and can use that water for the production of O$_2$, as shown in figure 1-3, but does not have a fully closed system [3]. ECLSS with a greater degree of loop closure can reduce the logistical requirements with respect to consumable supply, but they also tend to be more complex and less reliable than open loop designs. As such, it is important to consider system reliability and the additional costs associated with spares parts when designing closed loop ECLSS, since the failure of a regenerative technology element could have disastrous consequences if not repaired in a timely manner [23, 3, 24].

1.1.3 The Need for Resilient ECLSS

Current state-of-the-art ECLSS technology was designed for operation in Low Earth Orbit (LEO), as exemplified by the ISS. This design paradigm has resulted in a system which, while effective, depends on regular (and even unplanned, if necessary) resupply missions and the ability to quickly return a crew to Earth in an emergency in order to mitigate risk [16]. System requirements will be significantly different for Deep Space Life Support (DSLS), especially with regard to reliability, resilience, and logistics demands [18, 3]. Reliability and resilience will become more dominant design drivers, and will need to be considered in system trades early in the design process in order to enable the creation of optimal systems for a given mission [18].
Figure 1-3: Functional diagram of the ISS ECLSS [3].
While system prototyping and testing are of course necessary steps in the design process, it can often be expensive and difficult to execute in a comprehensive manner during the early stages of system creation. Modeling and simulation tools, on the other hand, can be applied early in the design process in order to examine many different design options. In this way, engineers “can make better decisions and communicate those decisions early enough in the design process that changes are easy and quick, as opposed to during production when they are extremely costly and practically impossible” [20].

Additional redundancy, spare parts, and consumable margin can increase a system’s resilience, but they also increase cost and mass. A quantitative, objective means to characterize system resilience enables its consideration alongside traditional metrics such as cost, productivity, and Equivalent System Mass (ESM) [25]; in addition, such a metric facilitates the use of formal multidisciplinary design optimization techniques, which can enable a faster and more effective design process. However, the analysis of system resilience and fault management has been called “more of an ‘art’ than a ‘science,’” and work at the Jet Propulsion Laboratory (JPL) has noted that there is “significant benefit to be gleaned from applying greater rigor and more systematic approach” [26]. A key challenge is to “show quantitative benefits to support engineering trades,” particularly with regard to hardware redundancy [26]. As stated in the 2013 Keck Institute for Space Studies report, *Engineering Resilient Space Systems: in order to consider resilience properly in the set of engineering trades performed during the design, integration, and operation of space systems, the benefits and costs of resilience need to be quantified*” [22].

1.2 Literature Review: Risk Assessment

Several techniques are currently used for fault management and risk assessment. A literature review of the primary methodologies used in space systems design is presented in this section. For each technique, a brief description is given as well as a discussion of the benefits, limitations, and drawbacks of that specific technique. Once
1.2.1 Risk Matrices

Risk matrices (also called “fever charts”) are a common methodology in which teams of experts identify risks and characterize their likelyhood and severity of consequences. This two-dimensional characterization is used to plot each risk in a matrix such as the one shown in figure 1-4 and determine the level of criticality. This is based upon the definition of the concept of risk as “the product of the probability of a negative event occurring... and the impact, or consequence, of that event” [27].

Since they do not require the construction of any mathematical models and can be based directly upon the analysis team’s understanding of the system at hand, risk matrices are a risk assessment technique that can be applied at a fairly low level of effort. However, this pure dependence on the humans performing the analysis is also the primary limitation of risk matrices. The characterization of risks is subjective and usually qualitative (though it can be made quantitative), and it is up to the analysis
team to identify risks on their own. As a result, there is a risk that the analysis will be incomplete; a risk may not be identified and characterized, and therefore may not be addressed. This risk of incompleteness can be particularly dangerous for the analysis if low severity events are overlooked, as more frequent but less severe basic events can result in severe high-level accidents at higher frequency than less frequent, more severe events [6]. In addition, the characterization process is subject to the bias (conscious and/or unconscious) of the members of the analysis team. Finally, the qualitative and subjective system characterization used in risk matrix analysis can make it difficult to apply this methodology in large trade studies, since it may be infeasible for an analysis team to examine more than a handful of design options in order to characterize their risk.

1.2.2 Fault Trees

Fault trees are a common modeling tool used to decompose high-level system failures into subsets of subsystem and component failure modes. In the fault tree modeling process, diagrams such as the one shown in figure 1-5 are constructed, tracing pathways from high-level events (at the top of the tree) to low-level events. Basic failure events - those at the lowest points on given branches in the tree - are assigned a probability, and the structure of the fault tree indicates the logic by which probabilities are combined [28, 27, 29]. For example, the “AND” and “OR” gates shown in figure 1-5 represent the logical combination of probabilities. Assuming independent events $A$ and $B$, the mathematical results of an AND ($\cap$) and an OR ($\cup$) gate combining the events are, respectively [29]:

\[
P(A \cap B) = P(A)P(B) \tag{1.1}
\]

\[
P(A \cup B) = P(A) + P(B) \tag{1.2}
\]

Using these and other logic gate combinations, the probability of the top-level event can be calculated as a function of the probabilities of the basic events [4, 28, 27, 29].

The fault tree analysis structure focuses on component-based events, and does
Figure 1-5: Example of basic fault tree structure [4].
not readily incorporate buffers in a probabilistic capacity. While a delay block does exist and could be used to incorporate the delaying effect a buffer has on a system - that is, the fact that a buffered system does not fail immediately upon failure of its components - this delay is deterministic in nature and does not capture the distribution of delaying time that a buffer could provide if the level of the buffer or the rate at which it is consumed are variable or uncertain [28]. In addition, corrective maintenance can be difficult to model.

1.2.3 Reliability Block Diagrams

Reliability Block Diagrams (RBDs) are a method for representing a system as strings of blocks (representing system elements) forming a pathway from an input to an output. The structure of the RBD is based upon the interactions between the components, and this structure is used to combine the reliabilities of the system elements into an overall system reliability (similar to fault tree analysis). The overall reliability of a system can be thought of as the probability that continuous path exists through the block network that connects the input to the output.

In order to incorporate the impact of buffers within systems, Jiang et al. proposed Modified Reliability Block Diagrams (MRBDs), such as the one shown in 1-6 [5]. MRBDs have the same structure and logic as regular RBDs, but include blocks representing system buffers in parallel to the subsystems that have buffered capacity. For example, block 10 in figure 1-6 represents the buffer for subsystem A (consisting of blocks 1 and 2) and continues to provide a connection between the input and the next subsystem after failure of subsystem A for as long as the buffer lasts.

Once the RBD structure is developed and reliability functions $R_i(t)$ are defined for each block, these reliabilities are combined based upon the structure of the RBD. The example system shown in figure 1-6 includes four basic structures: series (subsystem A), parallel (subsystem B), $k$-out-of-$n$ (subsystem C), and passive redundancy (subsystem D). The rules for combining the reliabilities of the system elements in these configurations are given by equations (1.3) through (1.6), respectively. These correspond to equations 1, 2, 3, and 4 in Jiang [5].
Figure 1-6: Modified Reliability Block Diagram, showing blocks 10 though 13 as buffers for subsystems A through D [5].
\[ R(t) = \prod_{i=1}^{n} R_i(t) \] (1.3)

\[ R(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \] (1.4)

\[ R(t, n, k) = \sum_{r=k}^{n} \binom{n}{r} R(t)^r (1 - R(t))^{n-r} \] (1.5)

\[ R(t) = R_1(t) + \int_{T}^{t} f_1(T)R_2(t - T)dT \] (1.6)

Where \( R \) is the overall system reliability, \( R_i \) is the reliability of element \( i \), and \( f_i \) is the probability density function (PDF) corresponding to the reliability \( R_i \).

The primary weakness of RBDs and MRBDs appears when considering repairable systems and systems with buffers. While MRBDs provide a modeling solution that can include the delaying effect of a buffer in the system - the fact that a system’s buffering capacity can temporarily “absorb” a failure and prevent it from propagating forward in the system until the buffer is depleted - they still only capture this delay and not the repair possibilities that it facilitates. Both RBDs and MRBDs assume that no maintenance occurs, severely limiting their use in analysis of repairable systems [5].

1.2.4 Failure Modes, Effects, and Criticality Analysis

Failure Modes, Effects, and Criticality Analysis (FMECA) is a procedure in which a team of experts examines systems in order to identify potential failure modes, examines their effects, and ranks issues according to criticality. Then, corrective actions are identified, and the potential effects of these corrective actions are investigated. These actions and the discussions around them are documented in natural language, and the process is revisited periodically as an “ongoing procedure” [4, 30]. The goal of FMECA is to identify and address the most critical risks within a system design [4, 31].

The primary drawbacks of FMECA result from its dependence on a team of sys-
tem experts. The complex nature of many systems results in a need for an analysis team made up of expert engineers from wide, multidisciplinary backgrounds, and the analysis process can be very time-consuming and therefore expensive to execute [30, 32]. As a result, it is infeasible to conduct FMECA analyses on large numbers of system concepts for the purposes of tradespace exploration. In addition, the results of FMECA analyses can be ambiguous, inconsistent, and incomplete [30, 32]. In particular, since FMECA requires that the set of failures and failure modes be generated by the analysis team, an effective FMECA requires a team with extensive knowledge of how the elements of a system operate and can fail - and even then there is a risk that not all failure modes will be accounted for in the analysis [30]. Finally, as with all subjective analysis tools, FMECA runs the risk of incorporating conscious and/or unconscious bias from the members of the analysis team.

1.2.5 Probabilistic Risk Assessment

Probabilistic Risk Assessment (PRA) is a "state-of-the-art assessment technique used by NASA in the later [spacecraft and space mission] design phases" [27]. Developed in the aerospace sector in the early 1960s and adopted by the nuclear industry for reactor safety assessment, PRA provides a "logical, systematic, and comprehensive approach" to analysis of risk in complex technical systems [6, 33, 4]. Since the turn of the century, PRA has been applied to the analysis of human systems including the Space Shuttle, the ISS, and the Constellation Program, and it is also applied for any mission involving radioactive material [6]. The PRA process involves the identification of events could have an adverse effect on the system. Event sequence diagrams (following the structure shown in figure 1-7) are constructed to show potential pathways from initiating events to consequences, with intervening events providing branches in the pathway. Following a similar process to fault tree analysis, the probabilities of the various events can be utilized to calculate the probabilities of each of the consequences [6, 27].

PRA is similar to fault tree and RBD analysis in that chains of low-level event probabilities are combined to determine the probabilities of high-level effects. In fact,
both fault trees and RBDs are tools that can be implemented as part of a larger PRA in order to determine the probabilities associated with events. The framework of PRA is much more general, however, and can include repair processes as events in the sequence [6].

PRA focuses on the probability that the system will enter some end state, starting from a certain initiating event. This provides powerful insight into the system and can be used to determine, for example, which events are of greatest importance for the system’s overall probability of success [6]. However, end states tend to describe the overall result of a mission - such as determining the probability of Loss of Mission (LoM) or Loss of Crew (LoC). The event sequences used in PRA do not provide insight into the potential state a system may be in (nominal, degraded, or otherwise) at a given point in the mission timeline, nor do they provide a robust means to probabilistically examine the number of spares that may be required for a given component.

1.2.6 Expected Productivity Analysis

Expected Productivity Analysis (EPA) is a method of system evaluation that combines the capability of a system to produce value in a given state with the probability that a system will be in that state in order to calculate an expected productivity [27]. First, a set of possible system states are identified, along with the transitions between
them, and used to create a Markov model of the system, such as the one shown in figure 1-8. In this model, the transitions are characterized by transition rates; in the case of failure analysis, these are often the failure rates of the various components in the system. This Markov model can then be characterized by a transition matrix \( A \) encoding the state diagram structure and the transition rates. Each off-diagonal entry \( A_{i,j} \) in the transition matrix indicates the transition rate from state \( j \) to state \( i \), and the diagonal entries \( A_{i,i} \) are equal to the opposite of the sum of the transition rates out of state \( i \). Once this transition matrix is determined, the state probability vector \( P(t) \) - with entries \( P_i(t) \) indicating the probability that the system will be in state \( i \) at time \( t \) - can be calculated based on an initial condition \( P(0) \):

\[
P(t) = e^{At}P(0)
\] (1.7)

Next, a productivity function \( C(t) \) is defined, with entries \( C_i(t) \) capturing the productivity of the system if it is in state \( i \) at time \( t \). This productivity could be, for example, the rate at which an observation system can take measurements [27]. Once the state probabilities and productivities are calculated, the expected productivity \( E[Prod]_{Total} \) of the system over its lifetime can be calculated as the sum of the state productivities weighted by the state probabilities:

\[
E[Prod]_{Total} = \sum_{t=0}^{life} C(t)P(t)
\] (1.8)

This expected productivity value provides a metric for the expected return from a mission, thus combining considerations of mission productivity and risk into one metric [27, 7, 34, 35].

EPA is a very powerful analysis, but its basis in Markov chains restricts it to the use of exponential distributions to describe state transitions. While this effectively applies the constant failure rate model to the entire system, if there are events within the system that do not occur at a constant rate - such as repair or buffer depletion - they can only be approximated as exponential at best. In addition, EPA requires
the use of a productivity function to give the system a “score” depending on its state. While this approach is suitable for systems that provide some form of value returned - such as scientific data - it is not as well suited to critical systems such as ECLSS, where the cumulative sum of the system’s score over time may not be the best metric (since even a very brief failure at any point in the mission timeline could have catastrophic consequences).

1.2.7 Systems-Theoretic Process Analysis

The methods described in the previous six sections have been based primarily on the examination of component failure accidents - that is, accidents caused by the failure of one or more components within a system. However, system failures can also occur as a result of unsafe interactions between functional components. As such, failure modes can exist within a system that are not captured by traditional analyses. Systems-Theoretic Process Analysis (STPA) is a hazard analysis technique designed to take into account potential system failure modes that are not captured in traditional component-based analysis in order to expand the scope of the analysis.
Figure 1-9: Feedback control structure used in STPA, showing a control algorithm (which includes a model of the process it is controlling), the process to be controlled, and the exchange of control actions and feedback [8].

beyond simple reliability and into system safety [8, 36].

STPA is based upon Systems-Theoretic Accident Model and Processes (STAMP), an accident modeling paradigm based upon systems theory as opposed to traditional reliability theory. STAMP models system safety as an emergent property of the interactions between the components within a system and between that system and its external environment. These interactions take the form of control loops such as the one shown in figure 1-9, and the goal is to enforce safe constraints on the system via these control structures (as opposed to the goal of preventing failure, used in many other traditional techniques) [8, 36, 37].

The STPA process has two main steps: identification of potential means by which inadequate control could lead to a hazard, and determination of how that inadequate control action could occur. Once these potentially hazardous control actions are identified, the control structure of the system can be restructured to mitigate them if possible. This process can be applied to existing designs in order to identify potential hazards, but it is most powerful when used to guide the development of new systems, since it can be applied in a top-down, hierarchical fashion. The end result is a system control structure that mitigates potentially hazardous conditions as much as possible [36, 38, 37].
STPA is an outlier among the methods presented here, in that it is not meant as a reliability assessment tool - instead, it focuses on providing information on potential hazards, their causes, and ways they could be mitigated during the system design process. The results of STPA are qualitative by design; the process does not yield a probabilistic assessment of system hazards. This is because many elements of complex systems - such as software or human controllers - are difficult to model probabilistically with any accuracy, and therefore any probabilistic information generated for systems including these elements would be inaccurate and potentially “dangerously misleading” [38]. Instead, STPA assists designers in the creation and analysis of a system control structure - effectively, a system architecture - with identified and mitigated hazardous scenarios [38].

This qualitative nature of STPA is very powerful when applied to the detailed examination of an individual system, and it can provide excellent guidance to assist designers in the mitigation of possible hazardous scenarios through changes in the control structure of the system. However, it is not a method that can be applied during trade studies to examine different system options in a quantitative manner. In a way, STPA is an optimization algorithm for system architecture - it is a technique by which a system’s architecture can be developed in such a way as to mitigate potential hazards.

Once this architecture (in the form of the system control structure) is developed, however, it still must be populated with the actual elements that will execute processes, order control actions, implement control actions, and provide measured feedback. These elements have important quantitative values associated with them - cost, mass, processing rates, and reliability - which must be considered in quantitative trade studies to select the appropriate system design for a given mission.

1.2.8 Limitations of Current Methods

The seven methods described above provide a suite of assessment tools that been applied effectively to risk and hazard analysis and provide powerful insight into systems. Each tool has its strengths, and for a given application the appropriate tool must be
selected. However, there are some limitations to be addressed in general.

Most notably, most of these methods tend to be focused on modeling systems based on component failures, effectively combining component reliability data into system reliability data. However, this paradigm does not take into account the impact of buffers and repair within the systems. Particularly in human spaceflight - where by definition someone is on hand to implement a repair if a failure occurs within the system - the use of preventative and corrective maintenance can greatly increase the survival time of systems. In addition, as described in section 1.1.2, ECLSS are complex systems involving the processing and supply of consumables. This type of system structure inherently includes buffers - for example, in the form of consumable storage tanks, the habitat atmosphere, or even the humans themselves - and the impact of these buffers on overall system resilience can be very high.

As noted in section 1.1.1, the examination of long-duration systems requires consideration of system resilience - the consideration of partially failed states and the recovery of the system from them. For ECLSS, partially failed states are often facilitated by buffers, and repair of components to recover from those states is a key consideration. The methods presented above tend to focus on reliability rather than resilience, looking mainly at the probability that the system will accomplish its mission without failure. Logistical demands that may appear - such as resupply requirements or the number of spares used - do not appear directly in these methods. In addition, some of the methods above are qualitative and subjective in nature - specifically risk matrices and FMECA. This limits the usefulness of such methods during trade studies, which benefit from quantitative, objective analyses.

As discussed in section 1.2.7, STPA is different from the other presented methods in that it is specifically designed to investigate system safety as opposed to reliability. It intentionally does not include a quantitative assessment, and as such cannot be used in quantitative trade studies. Instead, the output of the STPA process is a system control structure that has mitigated as many hazards as possible. This can be thought of as a system architecture optimization process, and as a result STPA serves a somewhat different purpose than the other assessment techniques.
1.3 Research Objectives and Thesis Outline

The goal of this research is to develop and implement a risk assessment technique that facilitates the quantitative consideration of system resilience during the design of ECLSS for long-duration human spaceflight. This technique will not only examine the overall probability of system failure, but will also include consideration of partially failed states. It will allow the examination of the interplay between system design variables such as buffer sizes, repair times, component reliabilities, and the number of spares supplied - as well as operational considerations such as contingency responses - and their resulting impact on system resilience and logistical demands over the system’s lifetime. This quantitative analysis of system resilience will enable consideration of resilience in high-level system trades and optimization in order to support the development of the highly resilient systems required for the expansion of human spaceflight beyond the Earth-Moon system.

This thesis is organized into six chapters, as shown in figure 1-10. Chapter 1 - this chapter - has presented the motivation behind this work, as well as an overview of existing methods and the goals of this thesis. Chapter 2 describes the semi-Markov process methodology applied in this thesis, both as a general probabilistic analysis framework and in the context of ECLSS design. Chapter 3 presents a demonstration of the methodology for buffer sizing and selection of the appropriate number of spares for a simplified ECLSS design problem, and utilizes a historical case study to examine the inclusion of contingency response actions in the analysis process. Chapter 4 presents a case study analyzing the logistical requirements of the Mars One mission plan, focusing on the mass of spares that must be resupplied on a regular basis for the ECLSS implemented in that system architecture; a multiobjective optimization is performed to examine resupply options, and the sensitivity of the result to changes in component reliability is examined. Chapter 5 includes a discussion of the methodology, noting its strengths, assumptions, limitations, and relationship to the methods described in the literature review. Finally, Chapter 6 gives a summary of the thesis, discusses potential future work to expand on this methodology, and presents
Figure 1-10: Thesis roadmap, showing interactions between the six chapters and three appendices.

conclusions. Three appendices present the numerical methods used to transform to and from the Laplace domain, the state network outline for the Mars One case study, and MATLAB code used in chapters 3 and 4.

1.4 Chapter 1 Summary

This chapter introduced the work of this thesis, providing background and motivation. Specifically, the challenges of future human spaceflight and the need for quantitative analysis of system resilience were addressed. A brief overview of ECLSS was presented in section 1.1.2, and the need for increased ECLSS resilience was discussed in section 1.1.3. Section 1.2 presented a literature review of existing risk assessment methods,
including risk matrices, fault trees, RBDs/MRBDs, FMECA, PRA, EPA, and STPA. The limitations of these methods - particularly when applied to buffered, repairable systems - were described in section 1.2.8. Finally, research goals and an outline of this thesis were presented in section 1.3.
Chapter 2

Methodology

This chapter describes the framework proposed by this thesis for analysis of the resilience of ECLSS systems. First, an overview of Semi-Markov Processes (SMPs) is given in section 2.1. Section 2.2 describes the process of model definition and the method used to solve for values of interest. Section 2.3 describes the specific application of SMPs to ECLSS, addressing both the translation of ECLSS design data into an SMP and the interpretation of results.

2.1 Overview of Semi-Markov Processes

SMPs are a framework that enables the probabilistic prediction of the behavior of a system. An SMP model represents a system using a network of states such as the one shown in figure 2-1. The nodes represent states, which may be for example a description of the functionality of the system (e.g. which components are functional and which have failed). Edges represent transitions between states, often caused by some event (e.g. the failure of a component), and are described probabilistically using PDFs. The combination of adjacency data from the state network and the PDFs of the various transitions enables the calculation of several system metrics, such as the probability of the system being in a given state at a given point in the future [13, 39, 40].
2.1.1 Relation to Markov Chains

Markov chains - such as those used in EPA (see Chapter 1) - are a specialized form of SMP, and have very similar structure. The key difference, however, is the flexibility of the analysis. For Markov chains, the transitions between states must be described by exponential distributions, and the transition rate is the key value for analysis. In SMPs, however, transitions may be described by any PDF [13, 39]. In addition, the amount of time spent in a given state is a function of both the current state and the set of possible destination states; effectively, SMPs can include “competing” transitions within their structure [39]. This capability is particularly useful for the analysis of repairable systems, as it allows the model to capture the probability of a successful repair before total system failure (this will be described in more detail in section 2.3). Flexibility with regard to transition PDF, coupled with the ability to include competing transitions, makes SMPs a very powerful tool for the probabilistic modeling of system resilience [13, 39, 40].

2.1.2 History

SMPs were independently introduced by Lévy [41, 42], Takács [43], and Smith [44] in 1954-55 (as referenced in [13, 39, 40]). General solutions for many values of interest were presented in two papers by Pyke in 1961 [45, 46]; however, these solutions were computed in the Laplace domain, and Laplace transform inversion to obtain time-domain solutions was not possible without numerical algorithms and computational power that did not exist in the early 1960s. As a result, SMPs remained a theory without application in their early years and “have not been used as widely as one would expect, given their generality” [13]. Warr and Collins ascribe this lack of application of SMPs to the fact that they are perceived to be too complex to solve; however, utilizing numerical algorithms and the ample computing power available to modern engineers, they present a methodology whereby “processes that have many states with any smooth transition distributions can be solved in a matter of seconds” [13].
2.2 SMP Formulation and Solution

2.2.1 Nomenclature

The network of states in an SMP results in a matrix representation of the mathematics of the system. In some cases, it is more effective to present equations utilizing matrices, which are be represented by bold letters; in other cases, examination at the level of individual matrix entries is more effective - these are represented by roman letters with subscript indices. For all values, indexing convention follows that of adjacency matrices: the row index represents the source of a transition and the column index represents the destination [13]. For example, the adjacency matrix $A$ for the three-state SMP shown in figure 2-1 is

$$A = \begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix} \quad (2.1)$$

with entries $A_{1,2}$, $A_{2,1}$, and $A_{2,3}$ equal to 1 to indicate the transitions shown in the diagram. In addition, the SMP in figure 2-1 includes a Markov renewal process, shown in the cycle between states 1 and 2. This type of structure enables the calculation of Markov renewal process probabilities, which provide very useful information for resilience studies, as will be discussed further in this section.

2.2.2 Model Formulation

As described in section 2.1, the inputs used to formulate an SMP are the structure of the state network and the PDFs describing the transitions between states. These
are stored in the form of the holding time density matrix \( f(t) \). The entries \( f_{i,j}(t) \) in this matrix encode PDFs describing the amount of time spent in state \( i \) before a transition occurs, given that the following state is state \( j \). Once \( f(t) \) is defined, a corresponding matrix of Cumulative Distribution Functions (CDFs) \( F(t) \) can be calculated, with entries

\[
F_{i,j}(t) = \int_0^t f_{i,j}(t) dt
\]  

(2.2)

The CDF matrix entries encode the probability that a transition out of state \( i \) has occurred by time \( t \), given that the next state is \( j \). In this formulation, \( f(t) \) is used to define \( F(t) \), but this process could easily be reversed and a derivative could be used to define \( f(t) \) using \( F(t) \). Once \( f(t) \) and \( F(t) \) are defined, they can be used to calculate two key matrices for the solution of SMPs: the kernel matrix \( Q(t) \) and the unconditional waiting time density matrix \( H(t) \) \[13, 39, 40\].

The kernel matrix \( Q \) contains entries \( Q_{i,j}(t) \), which are PDFs encoding the time until a transition from state \( i \) to state \( j \) occurs, given that the last transition was to state \( i \) at time \( t = 0 \). Effectively, while the entries in \( f(t) \) were PDFs for specific transitions in isolation, the entries in \( Q(t) \) take into account all possible transitions from a given state in order to derive PDFs of the transition times. There are two ways that the entries \( Q_{i,j}(t) \) can be defined, which may be thought of as two different decision processes that can be taken when multiple destination states are present.\(^1\) In the first algorithm, the system makes a random selection of which state is next based upon the various transition probabilities; then, once the successor state is selected, a random draw from the PDF describing the transition to that state is used to determine the time that passes before the transition is made. In the second algorithm, a random draw is made from each transition PDF, and the transition whose draw results in the lowest transition time “wins” \[39, 40\]. The latter is the algorithm for competing transitions described in section 2.1.1, and is the algorithm

\(^1\)It is important to emphasize here that SMPs are probabilistic models of system behavior and are not simulations; however, it is useful and intuitive to imagine the system moving through states sequentially to investigate the factors that influence the system’s behavior when multiple destination states are possible.
utilized in this thesis. Kernel matrices need not be exclusive to one algorithm or the other, however - each entry \( Q_{i,j}(t) \) may be formulated individually using whichever algorithm is most appropriate [39].

The kernel matrix entries are defined as

\[
Q_{i,j}(t) = f_{i,j}(t) \prod_{k \neq j} (1 - F_{i,k}(t)) \tag{2.3}
\]

Equation (2.3) shows mathematically the competitive transition logic via the combination of the PDF of the transition from state \( i \) to state \( j \) and the CDFs of all other possible transitions; effectively, the product of the complement of the other possible transitions conditions the PDF of the transition from \( i \) to \( j \). The resulting distribution \( Q_{i,j}(t) \) describes the probability of a transition from state \( i \) to state \( j \) at time \( t \), given that the system has not already transitioned to some other state.

The unconditional waiting time density matrix \( H(t) \) is a diagonal matrix with entries defined as

\[
H_{i,i}(t) = \sum_j Q_{i,j}(t) \tag{2.4}
\]

As the name suggests, the entries in \( H(t) \) encode the PDF describing the amount of time spent in state \( i \) (given that the last transition was into state \( i \) at time \( t = 0 \)) unconditioned by destination state. The kernel matrix \( Q(t) \) and unconditional waiting time density matrix \( H(t) \) fully describe the SMP [13, 39, 40].

### 2.2.3 SMP Solution in the Laplace Domain

Warr and Collins [13] present a methodology for the solution of SMPs for several values of interest, based upon earlier work by Pyke [45, 46]. This methodology makes use of the Laplace domain, which greatly simplifies calculations; most importantly, convolution in the time domain is the same as multiplication in the Laplace domain [47, 13]. In order to reduce the clutter in equations, the Laplace transform of a
Table 2.1: Symbols, names, and descriptions of SMP metrics [13].

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
<th>Eqn.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\phi_{i,j}(t))</td>
<td>Time-dependent state probability</td>
<td>Probability that the system will be in state (j) at time (t)</td>
<td>(2.6)</td>
</tr>
<tr>
<td>(E_{i,j}(t))</td>
<td>Expected time spent in a state</td>
<td>Expected amount of time the system will have spent in state (j) up to time (t)</td>
<td>(2.7)</td>
</tr>
<tr>
<td>(g_{i,j}(t))</td>
<td>PDF of first passage time</td>
<td>PDF describing the time of taken to reach state (j)</td>
<td>(2.8)</td>
</tr>
<tr>
<td>(G_{i,j}(t))</td>
<td>CDF of first passage time</td>
<td>CDF giving the probability that the system has reached state (j) by time (t)</td>
<td>(2.9)</td>
</tr>
<tr>
<td>(V_{i,j}(k,t))</td>
<td>Markov renewal process probability</td>
<td>CDF giving the probability that the system has reached state (j) a total of (k) or fewer times by time (t)</td>
<td>(2.10)</td>
</tr>
</tbody>
</table>

The function will be indicated by a tilde as follows:

\[
\mathcal{L}\{Q(t)\} = \tilde{Q}(s)
\]  

(2.5)

To avoid the need for closed-form representations of the Laplace Transform (LT) and Inverse Laplace Transform (ILT) of each function in the analysis, numerical methods are used. LTs are computed via application of numerical integration to the definition of an LT; ILTs are computed using the EULER method, as described by Abate and Whitt [13, 48]. The numerical calculation of LTs and ILTs is described in more detail in Appendix A. These numerical methods expand the flexibility of the analysis, but they do require that the input PDFs be smooth and continuous [13, 48].

Five metrics may be calculated from the kernel and unconditional waiting time density matrices. Symbols, names, and descriptions of each metric (in the time domain) are presented in table 2.1. All values are in matrix form, with the entry \((i, j)\) indicating the value of the metric for state \(j\) given that the system began in state \(i\) at time \(t = 0\). Note that for the first four metrics \((\phi, E, g, G)\) are calculated under the assumption that a Markov renewal process can be completed as many times as is necessary. For example, the state probabilities in \(\phi\) do not take into account the concept of some transitions only being available a certain number of times before a system must default to some other transition. This has important implications for
reliability studies, and will be discussed further in section 2.3.4.

Once $Q(t)$ and $H(t)$ are generated via the process described in subsection 2.2.2, the respective LTs $\tilde{Q}(s)$ and $\tilde{H}(s)$ are calculated. Equations (2.6)-(2.10) are then used to calculate metrics for the system in the Laplace domain.\(^2\)

\[
\tilde{\phi}(s) = \frac{1}{s} \left( I - \tilde{Q}(s) \right)^{-1} \left( I - \tilde{H}(s) \right) \tag{2.6}
\]

\[
\tilde{E}(s) = \frac{1}{s} \tilde{\phi}(s) \tag{2.7}
\]

\[
\tilde{g}(s) = \tilde{Q}(s) \left( I - \tilde{Q}(s) \right)^{-1} \left[ I \circ \left( I - \tilde{Q}(s) \right)^{-1} \right]^{-1} \tag{2.8}
\]

\[
\tilde{G}(s) = \frac{1}{s} \tilde{g}(s) \tag{2.9}
\]

\[
\tilde{V}(k,s) = \frac{1}{s} \left( 1 - \tilde{g}(s) \circ \left[ 1 \left( I \circ \tilde{g}(s) \right)^k \right] \right) \tag{2.10}
\]

Here $I$ is the identity matrix, $1$ is a matrix of ones, and $\circ$ is the Hadamard product (element-wise matrix multiplication). Once the solution is found in the Laplace domain, an ILT is used in order to obtain the value of the metric in the time domain [13].

### 2.3 Application to ECLSS

In order to utilize SMP analysis to investigate ECLSS resilience, two steps must be taken. First, system design data must be translated into a state network and PDF (or CDF) descriptions of state transitions that can be input into the SMP model. This model can then be solved for the metrics described in table 2.1. Then, meaning of these SMP metrics must be interpreted in the context of ECLSS resilience. These two interfaces - the input and output side of the SMP model - are described in this section. Each of these steps are based upon a generalization of ECLSS, which is presented in section 2.3.1.

\(^2\)Equation (2.10) corresponds to equation (18) in Warr and Collins, where a typo places the exponent at the incorrect location; however, rederivation of this equation from Warr and Collins equation (17) produced the correct version shown here [13].
2.3.1 ECLSS Generalization

As described in Chapter 1, ECLSS are complex systems involving many different processes acting on consumables such as air, water, and power. In general, however, these systems can be represented as interactions between processors and buffers acting on consumables, as shown in figure 2-2. These three constituents are defined as follows:

- **Consumable**: an element that is consumed or produced by processors in order to perform some function in the system. Examples include O$_2$, H$_2$O, food, and power.
- **Processor**: an element that consumes and/or produces consumables to accomplish a system function.$^3$ Examples include water distillation assemblies, CO$_2$ removal systems, and power production systems.
- **Buffer**: an element that stores consumables. Examples include water and gas tanks, as well as the atmosphere within a habitat.

At the highest level, ECLSS requirements are based upon the need to supply appropriate levels of consumables to the crew - whether that means ensuring that adequate amounts are available (as in the case of O$_2$) or ensuring that levels do not go above a certain amount (as in the case of CO$_2$). As such, system level failure modes can be defined by the depletion (or filling) of the buffers that the crew draw resources from or output waste products to. Processors are system elements that work to maintain buffers at their appropriate levels, and the examination of the failure and repair of processors in the context of the larger system provides information about the resilience of that system. For simplicity, from this point forward buffers will be described in general in the context of depletion - that is, the situation where processors work to keep a buffer full and system failure is related to the emptying of a buffer. However, the case of buffer filling can also be implemented using the same framework.

$^3$Processors that bring consumables into or take them out of the system boundary may have inputs without corresponding outputs, or vice versa. In these cases, the processor can be thought of as drawing from or venting to an “infinite buffer” beyond the system boundary. For simplicity, these external buffers are not represented in system diagrams and are assumed to be limitless.
Depending on the level of fidelity desired in the analysis, the crew may be modeled as a set of processors and buffers representing various metabolic processes. The states of the crew processes and buffers could then be used to define system failure modes - for example, depletion of a buffer representing the amount of water in the crew could represent crew death as a result of thirst. For this thesis, however, the crew are considered to be a single process with consumable inputs and outputs corresponding to the net metabolic processes of all crewmembers, and failure modes are defined using consumable buffers that interact with the crew. The development of a crew model is discussed further as future work in Chapter 6.

Processors and buffers are linked by consumable flows into pathways that show dependency. Elements depend on a supply of consumables from upstream components; if this supply is interrupted, an element will be unable to perform its function. It is, however, not failed - as soon as supply is restored, it will return to full functionality. As a result, both processors and buffers may take on three states:

- **Processor states:**
  - *Nominal:* the processor is performing its function as designed.
  - *Offline:* an upstream failure has resulted in a lack of consumable supply, meaning that functionality has temporarily stopped. Functionality will be restored as soon as the input consumable flow is restored.
- **Failed**: a failure has occurred within the processor, and the element must be repaired/replaced.

**Buffer states:**

- **Nominal**: all processors on either side of the buffer are functioning nominally, and the buffer is functioning as designed.
- **Active**: an upstream failure has stopped input to the buffer, and consumption of the stored consumable from downstream processors (or the crew) means that the buffer is being depleted without being replenished.
- **Depleted**: the buffer has emptied, causing downstream components to go offline and/or failure of the system.

As this list shows, buffer states are dependent upon the status of the processors around them. If a processor upstream of a buffer fails or goes offline, the buffer becomes active. At this point, the race begins - either the failed processor is repaired, or the system fails due to depletion or filling of the buffer.

### 2.3.2 Modeling Assumptions

Two assumptions are made that simplify the analysis process and facilitate the use of Markov renewal processes to inform the selection of the appropriate number of spares for a given component. The implications of both of these assumptions are discussed further in Chapter 5.

First, it is assumed that elements in the system have a constant failure rate - that is, the PDF describing the time to failure for a given element in the system is exponential, with parameter \( \lambda = \text{MTBF}^{-1} \). The constant failure rate model is a commonly used first-order model of component failure behavior, and provides a good estimate of the time to failure for a random process [49]. In addition, exponential PDFs are “memoryless,” meaning that the remaining time to failure follows the same exponential distribution as before, no matter how much time has passed [50]. This memoryless property of the distribution enables a repair process to return the system to the nominal state without having to account for the time that was spent in another
state - if all transitions out of a given state follow an exponential distribution, that state can be returned to at any point in time and the transition PDFs are still accurate probabilistic descriptions of the time remaining until the transition occurs. By allowing a return to the initial state, Markov renewal process structures such as the one shown in figure 2-1 can be utilized along with equation (2.10) to calculate Markov renewal process probabilities for each component, providing very useful information for the determination of the appropriate number of spares.

Second, it is assumed that - except for dependencies captured in the system diagram - failure, repair, and buffer depletion processes are independent. For example, the failure of one processor does not change the failure rate of another processor. This assumption simplifies the analysis by reducing the required number of basic events. Dependencies within the system are still captured within the SMP structure; as described in section 2.3.1, when failure of a processor or depletion of a buffer results in a lack of required resources for a downstream component, that component goes offline. However, processor failures or buffer depletions are assumed to not have an impact beyond their consumable stream.

2.3.3 ECLSS SMP Model Formulation

This section presents a 3-step algorithm for the generation of a holding time density matrix $f(t)$ for an ECLSS SMP. First, events that can change the state of the system - called basic events - are identified and characterized with PDFs. Then, the state network is defined by expanding a tree of possible events, state by state, until an end state is reached or a loop is created. Finally, PDFs for each transition in the state network are created via combination of the basic event PDFs. The adjacency structure of the state network and the PDFs describing each transition combine to yield the holding time density matrix, which can be used to calculate the kernel matrix $Q(t)$ and unconditional waiting time density matrix $H(t)$ and solve for SMP metrics following the procedure described in section 2.2.2.
1) Basic Event Definition

Basic events are the building blocks of an SMP. These are events that can occur to cause change in the system’s state. At a basic level, these events are processor failure, processor repair, and buffer depletion. Other events - such as processor degradation or buffer replenishment - can also be used, but these are not considered in this thesis. The number of basic events varies depending on the system, but in general there are two basic events for each processor (failure and repair) and one for each buffer (depletion). Sequences of these events produce the state network.

Once identified, the basic events must be characterized probabilistically in order to be used in an SMP analysis. This involves the definition of a Random Variable (RV) representing the time until an event occurs, after some initiating event. An initiating event is the event that “starts the clock” on a basic event - for example, activation of a buffer is the initiating event for the RV describing buffer depletion. The symbols used for the RVs representing processor failures, processor repairs, and buffer depletion events are $X$, $R$, and $D$, respectively. These RVs are described by PDFs $f_X(t)$, $f_R(t)$, and $f_D(t)$ indicating the distribution of time before the event occurs, given that the initiating event occurs at time $t = 0$. The basic event PDFs should be constructed in a parametric fashion whenever possible to enable a direct connection between the system design variables and parameters and probabilistic system behavior.

As described in section 2.3.2, processor failures ($X$) must follow an exponential distribution

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

(2.11)

Here, $\lambda$ is equal to the failure rate, or MTBF$^{-1}$.

Repair processes $R$ can be described by any distribution, but the lognormal distribution is recommended as it provides a good estimate of the distribution of corrective
repair times [51, 52]. This distribution follows the form

\[
f_R(t) = \frac{1}{t\sqrt{2\pi}\sigma} e^{-\frac{(\ln(t) - \mu)^2}{2\sigma^2}}
\]  

(2.12)

where the shape parameter \(\sigma\) and log-scale parameter \(\mu\) can be obtained from the arithmetic mean (m) and standard deviation (s.d.) using equations (2.13) and (2.14).

\[
\sigma = \sqrt{\ln \left( 1 + \frac{s.d.}{m^2} \right)}
\]  

(2.13)

\[
\mu = \ln(m) - \frac{1}{2}\sigma^2
\]  

(2.14)

Using these equations, a lognormal distribution for repair time can be determined from operational data using the Mean Time To Repair (MTTR) and a standard deviation representing the uncertainty in repair time.

The time until buffer depletion can be a function of many parameters, but in general it depends upon the level of the buffer at the time it is activated and the rate at which it is depleted. Either of these parameters - as well as any others - can be RVs in and of themselves, and appropriate combination of of their PDFs can be used to derive the PDF describing buffer depletion \(f_D(t)\). A PDF describing the fill level of a buffer could, for example, be obtained from simulation or operational data. If the only available description of buffer level and consumption rate are single values (rather than distributions - this suggests deterministic knowledge of these values, which may or may not be a reasonable assumption), then \(f_D(t)\) could take the form of a distribution with a small standard deviation and mean equal to the deterministic depletion time, or the buffer level divided by the consumption rate.

2) State Network Definition

Once the basic events are defined, the state network is constructed state by state by enumerating the possible events that can occur in a given state, creating the resulting destination states, and repeating the process for all created states. Starting from the nominal state (the state in which all system elements are functioning as designed),
the process of enumerating possible events and defining new states based on those events is iterated to expand the state tree until it loops back on itself or reaches an end state - a state from which there are no transitions. Loops are created when repairs return the system to its nominal state, and end states represent system failure due to depletion of buffers.

An outline format provides a good framework for this process. For example, a state network outline for the generalized ECLSS shown in figure 2-2 (identifying the two processors as A and B, where B is inside the loop and A supplies the consumable to B) is shown in table 2.2. Beginning in the nominal state (state 1), two events can occur - failure of processor A, or failure of processor B, resulting in transitions to states 2 and 3, respectively. The destination state of these transitions is indicated by the indented structure beneath the transition description. Looking at state 2, two events can occur - either processor A is repaired (resulting in a return to the nominal state), or the buffer is depleted (resulting in a transition to an end state). Processor B cannot fail because it is offline, since it depends on processor A for resources to operate as indicated in figure 2-2. Since both transitions out of state 2 ended with a loop or an end state, this branch of the tree is complete. For the next branch (state 3, reached via failure of processor B), there are three possible events: repair of processor B, failure of processor A, or depletion of the buffer. The repair and depletion events result in loops and end states. However, failure of processor A creates a new state - state 4. As before, this new state creates a structure indented a level from the previous state and nested under the event that led to it. This process continues on, creating states 5 and 6 and eventually either closing loops back to state 1 or ending when the buffer is depleted.

The corresponding SMP diagram for this state network outline is shown in figure 2-3. Here the transitions are shown as arrows, with states represented by circles or rounded rectangles (states 1 and 7 are represented as larger rounded rectangles due to the large numbers of transitions into/out of them). The SMP diagram and the state network outline encode the same information. The SMP diagram provides a more visually appealing and intuitive depiction of the structure of the system, but
Table 2.2: State network outline for the system shown in figure 2-2. A corresponding SMP diagram is shown in figure 2-3.

1 | Processor A failure
2 | Processor A repair (return to 1)
   | Buffer depletion (to end state)
3 | Processor B failure
3 | Processor B repair (return to 1)
4 | Processor A failure
   | Processor A repair
   | Processor B repair (return to 1)
   | Buffer depletion (to end state)
5 | Processor B repair
   | Buffer depletion (to end state)
   | Buffer depletion (to end state)
6 | Processor A repair (return to 1)
   | Buffer depletion (to end state)
   | Buffer depletion (to end state)

Figure 2-3: SMP diagram for the system shown in figure 2-2. A corresponding state network outline is shown in table 2.2. Processor failures are shown in black, and repairs are shown in blue. Buffer depletion is shown in red.
the state network outline provides a good format for the iterative application of the state generation procedure and a more compressed representation of the resulting state network structure.

For a complete analysis, this state network generation process should be applied to all transitions for all states until every possible event sequence is included in the network. However, for systems with a large number of elements, this may result in very large state networks. In order to maintain the network at a reasonable size, it may be prudent to constrain the state generation process - for example, by specifying the number of concurrent failures that will be included in the analysis. Depending on the system at hand, the probability of three independent failures being present in the system at the same time may be extremely low. For example, if the timescale of repair operations is much lower than the timescale of failures, then the probability of an additional failure occurring before repair can be implemented on the first failure is very low, and the probability of a third failure may be negligible. However, this constraint should be practiced with caution. Effectively, constraining the state network trades model complexity for accuracy, and the model must be tailored to the goals of the analysis.

3) Transition PDF Definition

By defining basic events and using them to construct a state network, the structure of the SMP - that is, the adjacency of the states - is determined. The final step is the definition of the PDFs that describe each transition. The transitions in the state network are caused by basic events, but the basic event PDFs cannot always be used directly to describe the transition times. Recall that the basic events are defined as RVs describing the time until the event occurs, given that the initiating event occurred at time \( t = 0 \). As such, the basic event PDF does not describe the time until the basic event occurs after some intervening event has occurred.

As an example, consider state 4 in table 2.2 and figure 2-3. One of the transitions exiting this state corresponds to depletion of the buffer. However, the initiating event for buffer depletion - failure of processor B, in this case - occurred before state 4 was
entered. The time required for the intervening event - failure of processor A - must be accounted for. In terms of basic event RVs, this is accomplished by taking the difference of the two distributions, $D - F_A$. The resulting difference distribution can be computed via convolution [50]:

\[
  f_{D-F_A}(t) = \int_{-\infty}^{+\infty} f_D(x) f_{F_A}(x-t) dx = f_D(t) * f_{F_A}(-t)
\]

(2.15) \hspace{1cm} (2.16)

$f_{D-F_A}$ is a difference distribution, a PDF describing the difference between $D$ and $F_A$. In this form the distribution includes both positive and negative values; to generate the transition distribution $f_{4,7}(t)$ the difference distribution must be truncated to the positive values. This is accomplished by restricting the support of the function to positive values of $t$ and scaling the distribution using the CDF of the difference distribution, $F_{D-F_A}(t)$:

\[
  f_{4,7}(t) = \frac{f_{D-F_A}(t)}{1 - F_{D-F_A}(0)}
\]

(2.17)

The truncated difference distribution effectively encodes the amount of time remaining before buffer depletion, given that failure of processor A has occurred in the time between the initiating event and the depletion of the buffer. In the MATLAB code utilized for this thesis (see Appendix C), discrete convolution is applied to vector sample of the two functions in order to speed computation of the difference distribution.

In order to define the transition PDFs in the state network, the truncated difference distribution is applied to any transition that does not occur immediately after its initiating event, with the exception of processor failures. The truncated difference calculation does not need to be applied to processor failures because they follow an exponential distribution, which is memoryless - the remaining portion of the distribution after another event occurs is still exponential with the same parameter, no matter how much intervening time has passed [50]. For each non-exponential distri-
bution, however, the RV describing the transition time is determined by subtracting the RVs for any intervening events from the RV describing the transition at hand and truncating the resulting distribution to values of $t$ greater than 0.

As they are generated, the transition PDFs are inserted into the holding time density matrix $f(t)$ at the location corresponding to their position in the state network. From here, $f(t)$ can be used to generate the kernel matrix $Q(t)$ and unconditional waiting time density matrix $H(t)$, which fully describe the SMP. Once these matrices are generated, the model can be solved for the metrics described in section 2.2.3, and these values can be interpreted in the context of ECLSS design.

### 2.3.4 Interpretation of Results

The metrics described in table 2.1 have several uses in ECLSS design. The most immediate application is the use of the state probabilities $\phi(t)$. By calculating state probabilities at the time corresponding to the end of the mission, a designer can determine the probability that the system will enter a failed state before the end of the mission due to a failed repair. In general, this probability $P_{FR}$ can be calculated by adding the probabilities for all failed states:

$$P_{FR} = \sum_{j \in F} \phi_{1,j}(t) \quad (2.18)$$

where $P_{FR}$ is the probability of failure due to failed repair (that is, the probability that a buffer depletes before a repair can be implemented) and $F$ is the set of failed states. In addition, if there is concern about the system being in a partially failed state at a specific point in the mission timeline, that probability can be calculated as well.

The Markov renewal process probabilities $V$ are a powerful tool to inform the decision of how many spares to include in a system. By calculating $V(k, t)$ for increasing integer values of $k$ (starting at 0), CDFs of the number of times a particular state is visited can be quickly generated. The process for SMP generation described in section 2.3.3, however, does not necessarily result in a state network where the number
of visits to a single state represents the total number of failures for a given element. For example, in the SMP diagram shown in figure 2-3, both state 2 and state 4 are entered via a failure of processor A. As a result, the total number of spares required is the sum of the number of entries in to state 2 and the number of entries in to state 4, obtained via convolution of the Probability Mass Functions (PMFs) of the number of entries into each state. These PMFs $v_{i,j}(k)$ can be generated from CDFs $V_{i,j}(k)$ via discrete differentiation, or they can be calculated directly by removing the $\frac{1}{s}$ term in equation (2.10). In general, the PMF describing the number of spares required is generated via convolution of all the PMFs of the different state renewal probabilities for the set of states that are entered via failure of the component in question

$$s_i(k) = \prod_{j \in S_i} v_j(k) \quad (2.19)$$

where $s_i(k)$ is the PMF of the number of spares required for component $i$, $S_i$ is the set of states that are entered via failure of component $i$, $v_j(k)$ is the PMF of the number of entries into state $j$, and $\prod$ is a convolution product, indicating the convolution of all the elements within the given set (effectively, a summation of the RVs in the set). The corresponding CDF $S_i(k)$ can be obtained via a cumulative sum of $s_i(k)$.

Using this method, plots of CDFs such as the one shown in figure 2-4 can be quickly generated. These CDFs provide a direct quantitative connection between the cost of adding more spare parts to a system and the benefits of additional spares from a risk management perspective. For example, by setting a threshold on the probability that not enough spares are included, a minimum number of spare components can be determined - and therefore a minimum mass of spares required to reach the probability threshold. In addition, the marginal rate of return - the change in the probability of having sufficient spares resulting from the addition of another spare - can be utilized in cost-benefit analyses to support sparing decisions. As figure 2-4 shows, the Markov renewal process probabilities approach 1 asymptotically. As a result, after a certain point they exhibit a diminishing rate of return; put another way, once a critical number of spares is reached, each additional spare contributes less to the system.
than the one before it. By coupling this diminishing return with a model of the cost (financial, mass, or otherwise) of additional spare parts, designers can determine the appropriate balance for a given system.

As mentioned in section 2.2.3, the state probabilities are calculated under the assumption that each Markov renewal process can be repeated as many times as is necessary. In the formulation presented here, Markov renewal processes represent repair, which consumes a spare part. As such, the probability $P_{FR}$ only includes entries into failed states due to a failure to repair a processor before the buffer representing that specific system failure mode is depleted or filled. To determine the true probability of system failure - including failures due to failed repairs and failures due to lack of components - the probability of failed repair $P_{FR}$ must be combined with the spare CDFs $S_i(k)$ for each component, as shown in equation (2.20).

$$P_{fail} = 1 - (1 - P_{FR}) \prod_i S_i(k_i)$$ (2.20)

Here $P_{fail}$ is the net probability of system failure and $k_j$ is the number of spares available for the element whose failure is represented by partially failed state $i$.

The expected time spent in a state $E$ can be used to inform the amount of contingency supplies that may be required over the course of the mission, if supplies cannot be renewed. If supplies can be recovered or produced in-situ, then the expected time in state over a given time period can inform the rate at which supplies must be produced in order to meet demand. The expected time in state can also inform operations by giving a sense of how much time the crew will need to spend on repairs and how much time will be free for other activities, such as science and sleep. This is based on the assumption that repairs are enacted immediately upon failure of an element, and thus time spent in a partially failed state is time spent on repair. If repairs are not enacted immediately, however, this metric can still be used - the repair time PDF can be updated to include a delay (accounting for failure detection, for example). That delay, along with a Markov renewal process probability for that state, can then be used to determine the proportion of expected time in that state.
that is actually spent on repair. Whether or not repairs are implemented immediately, the expected time spent on repairs can inform designers as to which components are expected to require the most crew time for maintenance. In addition, the expected time in nominal state, $E_{0,0}(t)$, represents free time where no preventative maintenance is occurring. By comparing this number to the total mission time - for example, by calculating the percentage of mission time that is available for crew science activities - a metric for crew productivity could be developed to help evaluate systems [53].

The PDF $g(t)$ and CDF $G(t)$ describing first passage times are primarily used as stepping-stones to the Markov renewal probabilities, but can have value on their own as well. For example, if spares or repair capability for a certain event will not be available immediately at the start of the mission, the first passage time CDF $G(t)$ can be used to determine the probability of that event occurring before a certain time. This information can be used to determine how long a system should be allowed to function without repair capability.

Each of these uses for the values of interest calculated from an SMP has assumed that a mission timeline is fixed before the analysis. However, if the mission duration...
(or time between resupply) is a variable in the analysis then these values can be adapted to inform selection of that value. For example, when defining a resupply campaign the Markov renewal probabilities $V(k, t)$ and resulting spares CDFs $S_i(k)$ can be used to connect the time between resupply $t$ and the number of spares brought by the resupply ship $k$ to the probability of that many spares being needed to replenish stores used by repairs since the previous resupply mission.

### 2.4 Chapter 2 Summary

This chapter presented a brief overview of SMPs, including their relation to Markov chains and a history of their development. The general procedure for SMP definition was then described. Section 2.2.3 presented five system metrics that can be obtained from SMPs, as well as a method to solve for them using LTs and ILTs.

Section 2.3 presented an algorithm for the translation of ECLSS design data into SMPs and the translation of the resulting SMP metrics back into system metrics that can inform design decisions. A generalized abstraction of ECLSS was presented in section 2.3.1, and section 2.3.2 described two assumptions used in the analysis. Using this ECLSS generalization and set of assumptions, section 2.2.2 presented a 3-step process - consisting of the definition of basic events, the state network, and transition PDFs - for the definition of an SMP. Finally, section 2.3.4 described the interpretation of SMP metrics in the context of ECLSS.
Chapter 3

Demonstration of Analysis

This chapter presents a demonstration of the methodology described in Chapter 2 on a simple example problem. All systems, parameters, and functions described in this example are notional and intended only for illustrative purposes. MATLAB code for the calculations performed in this chapter may be found in Appendix C. In addition, a historical example of a failure event in the ISS External Thermal Control System (ETCS) is used to demonstrate how contingency response actions can be incorporated into the SMP model framework.

3.1 System Description and Analysis Goals

Figure 3-1 shows a diagram of the example system analyzed in this chapter. This system represents a simplified version of an ECLSS, and includes four processors and two buffers. The crew is shown in the center, receiving a supply of O\textsubscript{2} and water from two tanks. Each system consists of a processing component and a buffer, and is powered by a common Power Distribution Unit (PDU). For O\textsubscript{2} supply, the Oxygen Generation System (OGS) produces O\textsubscript{2} and passes it to a 4.5 kg Oxygen Tank (OT) for storage and distribution to the crew; the OGS has an identical redundant backup that comes online in the event of failure of the primary component. Likewise, the Water Recovery System (WRS) recycles wastewater from the crew into potable water and passes it to the Water Tank (WT) for storage and distribution back to the crew.
3.1.1 Goal

This system is to be used to support a crew of 6 in space for 1 year with no resupply. The goal of this analysis is to determine a quantitative relationship between the size of the WT and the number of spares for each processor brought and the overall probability of system failure before the end of the mission. In this case, system failure is defined as depletion of either buffer. This information will be used to suggest an appropriate WT capacity and number of spares for each component, and could in the future be combined with mass data for spare parts in order to perform system-level trades between mass and risk.

3.1.2 Assumptions and Parameters

The system is assumed to operate at steady state, and the processing rates of the OGS and WRS are assumed to be high enough that the buffers are always full when the system is in a nominal state. External spares are utilized, meaning that time is required to complete any repair. Repair consists of removal of the failed processor and replacement with a spare, and is assumed to return the system to good-as-new health. The tanks are assumed to not fail. It is assumed that analysis up to two
levels of failure is sufficient - that is, no more than two independent failures will occur before the system is fully repaired. Finally, it is assumed that neither the OGS nor the WRS can fail while they are offline due to a failure of the PDU.

The crew consumes $O_2$ and $H_2O$ at rates of 0.835 kg CM$^{-1}$ day$^{-1}$ and 3.909 kg CM$^{-1}$ day$^{-1}$, respectively, or a net consumption rate of about 0.21 kg h$^{-1}$ for $O_2$ and 0.98 kg h$^{-1}$ for $H_2O$ [54]. Historical data indicate that the Mean Time Between Failures (MTBF) for the OGS is 2,160 h$^{-1}$ [5], and OGS repair is estimated to have a Mean Time To Repair (MTTR) 8 h, with a standard deviation (s.d.) of 1 h. Likewise, the WRS has an estimated MTBF of 4,320 h [5] and an MTTR of 12 h (s.d. 3 h). The MTBF of the PDU is estimated to be 69,536 h [55], and PDU repair has an MTTR of 10 h (s.d. 3 h).

### 3.2 Model Formulation

This section describes the formulation of the SMP model for this system, following the procedure outlined in Chapter 2.

#### 3.2.1 Basic Event Definition

There are eight basic events in this system, corresponding to failure and repair for each of the three different types of processor and depletion for each buffer. These basic events, their RV representation, distribution type, and parameter values are presented in table 4.2. The general form of the equations for exponential and log-normal distributions are shown in chapter 2 (equations (2.11) and (2.12), respectively). The failure rates $\lambda$ are obtained through inversion of the MTBF for each processor, and the shape and log-scale parameters $\sigma$ and $\mu$ are obtained from the mean and standard deviation of the repair time, using equations (2.13) and (2.14).

A lognormal distribution is also used to model the time until buffer depletion. Since in this case no random variables were given describing the fill level of the tanks or the processing rates, the deterministic mean time required for buffer depletion can be obtained by dividing the buffer capacity by the consumption rate. To model this
Table 3.1: Basic events and their distributions. All distribution parameters are based on a timescale of hours.

<table>
<thead>
<tr>
<th>Basic Event</th>
<th>RV</th>
<th>PDF Form</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGS failure</td>
<td>( X_{OGS} )</td>
<td>Exponential</td>
<td>( \lambda_{OGS} )</td>
<td>( 4.6296 \times 10^{-4} )</td>
</tr>
<tr>
<td>OGS repair</td>
<td>( R_{OGS} )</td>
<td>Lognormal</td>
<td>( \sigma_{OGS} )</td>
<td>0.1245</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \mu_{OGS} )</td>
<td>2.0717</td>
</tr>
<tr>
<td>WRS failure</td>
<td>( X_{WRS} )</td>
<td>Exponential</td>
<td>( \lambda_{WRS} )</td>
<td>( 2.3148 \times 10^{-4} )</td>
</tr>
<tr>
<td>WRS repair</td>
<td>( R_{WRS} )</td>
<td>Lognormal</td>
<td>( \sigma_{WRS} )</td>
<td>0.2462</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \mu_{WRS} )</td>
<td>2.4545</td>
</tr>
<tr>
<td>PDU failure</td>
<td>( X_{PDU} )</td>
<td>Exponential</td>
<td>( \lambda_{PDU} )</td>
<td>( 1.4381 \times 10^{-5} )</td>
</tr>
<tr>
<td>PDU repair</td>
<td>( R_{PDU} )</td>
<td>Lognormal</td>
<td>( \sigma_{PDU} )</td>
<td>0.2936</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \mu_{PDU} )</td>
<td>2.2595</td>
</tr>
<tr>
<td>OT depletion</td>
<td>( D_{OT} )</td>
<td>Lognormal</td>
<td>( \sigma_{OT} )</td>
<td>0.0117</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \mu_{OT} )</td>
<td>3.0647</td>
</tr>
<tr>
<td>WT depletion</td>
<td>( D_{WT} )</td>
<td>Lognormal</td>
<td>( \sigma_{WT} )</td>
<td>[see eqn. (3.1)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \mu_{WT} )</td>
<td>[see eqn. (3.2)]</td>
</tr>
</tbody>
</table>

as a distribution, a small standard deviation (0.25 h) is used along with this mean time to depletion in order to generate a lognormal distribution to approximate the repair occurring at the mean depletion time, following the same procedure that was used for the repair time distributions.

One of the goals of the analysis is the selection of WT capacity, \( c_{WT} \). In order to enable investigation of a range of values for \( c_{WT} \), the shape and log-scale parameters for the WT depletion PDF are parametrized as a function of WT capacity, using equations (3.1) and (3.2).

\[
\sigma_{WT} = \sqrt{\ln \left( 1 + \frac{0.25^2}{c_{WT}^{2}} \right)} \tag{3.1}
\]

\[
\mu_{WT} = \ln \left( \frac{c_{WT}}{0.98} \right) - \frac{1}{2} \sigma_{WT}^2 \tag{3.2}
\]

As with the distribution describing OT depletion, the mean of this distribution is determined by the buffer capacity divided by the processing rate, and the standard deviation is 0.25 h.
3.2.2 State Network Definition

The state network outline for this system is shown in table 3.2, and the resulting SMP diagram is displayed in figure 3-2. In addition, the start and end states for each transition, as well as the transition ID number, are listed in the right three columns. These data encode the adjacency structure of the SMP and indicates the indices of the transitions within the holding time density matrix \( f(t) \), as well as which transition should be used, using the index to select from the transitions presented in table 3.3 in section 3.2.3.

As described in Chapter 2, this network is generated by examining the set of possible basic events that can cause a transition away from a given state, starting in the nominal state (state 1). Through sequences of processor failure, processor repair, and/or buffer depletion events, branches of the state network are expanded until the system loops back to the nominal state or enters a failed state. There are two possible failed states in this system: depletion of the OT (state 18) and depletion of the WT (state 19). In order to constrain the number of states, the assumption that a maximum of two concurrent failures may occur (from section 3.1.2) is applied.

The structure of the system (from figure 3-1) incorporates two notable characteristics. First, an identical, redundant OGS is positioned in the \( O_2 \) supply line, and comes online immediately upon failure of the primary OGS. As a result, a single OGS failure does not result in activation of the OT buffer. This fact is reflected in the state network in that state 2 (the state where one OGS has failed) does not have a transition out of it corresponding to OT depletion. From this state, the only transition options are repair of the OGS (return to state 1), failure of the second OGS (transition to state 3), failure of the WRS (transition to state 4), or failure of the PDU (transition to state 7). In addition, since the two OGSs are identical, it is assumed that if a failure of the second OGS occurs then repairs continue on the first OGS without starting repairs on the second. This is because it will most likely be faster to complete repairs on the first OGS than to start and complete repairs on the second. Since only one OGS is required for the system to be completely functional (albeit with
a loss of redundancy on one element) and there is no danger of entering a failed state directly from state 2, there is no reason to split resources and repair both OGSs at the same time. As a result, the repair process exiting state 3 returns directly to state 2, where a single OGS has failed. Once repair on the first OGS is complete, repair on the second begins (which results in a transition back to state 1 when completed). Structuring the state network in this way simplifies it and eliminates the need for additional states to examine different orders in which the two identical components could have been repaired.

The second notable characteristic is the dependence of the OGS and WRS on the PDU for functionality. Both the OGS and WRS receive power from the PDU; if that power is not supplied, they both go offline. As stated in section 3.1.2, neither the OGS nor the WRS can fail when they are offline. As a result, no additional failure events can occur after failure of the PDU. This is shown in state 17, where the only exit transitions are repair of the PDU or depletion of either buffer. In addition, since failure of the PDU causes both the OGS and WRS to go offline, it also results in the activation of both the OT and the WT. In any state, if the PDU is failed then transitions to both failed states are available.

3.2.3 Transition PDF Definition

The final step in the SMP definition process is the definition of the transition PDFs based on the basic events given in table 4.2 and the state network described in table 3.2 and shown in figure 3-2. These PDFs are described in table 3.3, which shows the transition ID number (corresponding to the ones shown in table 3.2), the RV algebra the is used to generate the distribution from sums and differences of basic events. Each of the distributions can be obtained via applying the truncated difference distribution to a combination of two of the other transitions (as described in Chapter 2). The indices of the two transition distributions that are to be combined are shown in the last two columns of table 3.3, where the resulting distribution comes from the truncation of $C_+ - C_-$ to positive values of $t$. For cases where no subtraction is required, the index of $C_-$ is left blank.
Table 3.2: State network outline for the system shown in figure 3-1. The corresponding SMP diagram is shown in figure 3-2. The transition ID number indicates the distribution describing the transition, referring to table 3.3

<table>
<thead>
<tr>
<th>State Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 OGS failure</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2 OGS repair (return to 1)</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>OGS failure</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3 OGS repair (return to 2)</td>
<td>3</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>3</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>WRS failure</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4 OGS repair</td>
<td>4</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>5 WRS repair (return to 1)</td>
<td>5</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>5</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>WRS repair</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>6 OGS repair</td>
<td>6</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>6 WRS repair (return to 1)</td>
<td>6</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>6</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>PDU failure</td>
<td>7</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>7 OGS repair</td>
<td>7</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>8 PDU repair (return to 1)</td>
<td>8</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>8</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>8</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>PDU repair</td>
<td>7</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>9 OGS repair</td>
<td>8</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>9 WRS repair (return to 1)</td>
<td>9</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>9</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>9</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>WRS failure</td>
<td>1</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>10 WRS repair (return to 1)</td>
<td>10</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>OGS failure</td>
<td>10</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>11 OGS repair</td>
<td>11</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>12 WRS repair (return to 1)</td>
<td>12</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>12</td>
<td>19</td>
<td>24</td>
</tr>
<tr>
<td>WRS repair</td>
<td>11</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>13 OGS repair</td>
<td>13</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>13 WRS repair (return to 1)</td>
<td>13</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>13</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>PDU failure</td>
<td>10</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>14 WRS repair</td>
<td>14</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>14 PDU repair (return to 1)</td>
<td>15</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>15</td>
<td>18</td>
<td>27</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>15</td>
<td>19</td>
<td>28</td>
</tr>
<tr>
<td>PDU repair</td>
<td>14</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>15 WRS repair (return to 1)</td>
<td>16</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>16</td>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>16</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>16</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>PDU failure</td>
<td>1</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>16 PDU repair (return to 1)</td>
<td>17</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>OT depletion (to failed state 18)</td>
<td>17</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>WT depletion (to failed state 19)</td>
<td>17</td>
<td>19</td>
<td>8</td>
</tr>
</tbody>
</table>
Figure 3-2: SMP diagram for this system. Processor failures are shown in black, and repairs are shown in blue. Buffer depletion is shown in red. States and transitions correspond to the state network outline shown in table 3.2.
Table 3.3: Transitions in the state network, showing the combinations of RVs used to create them and their ID number (as used in table 3.2). Each PDF can be generated by taking the difference of the distributions indicated by $C_+$ and $C_-$ and truncating to positive values of $t$.

<table>
<thead>
<tr>
<th>ID</th>
<th>RV Combination</th>
<th>$C_+$ Index</th>
<th>$C_-$ Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X_{OGS}$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$R_{OGS}$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$X_{WRS}$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$R_{WRS}$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$X_{PDU}$</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$R_{PDU}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$D_{OT}$</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$D_{WT}$</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$R_{OGS} - X_{OGS}$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>$R_{OGS} - X_{WRS}$</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>$R_{OGS} - X_{PDU}$</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>$R_{WRS} - X_{OGS}$</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>$D_{WT} - X_{OGS}$</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>$R_{WRS} - X_{PDU}$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>$D_{WT} - X_{PDU}$</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>$R_{WRS} - (R_{OGS} - X_{WRS})$</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>$D_{WT} - (R_{OGS} - X_{WRS})$</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>$(R_{OGS} - X_{WRS}) - R_{WRS}$</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>$R_{PDU} - (R_{OGS} - X_{PDU})$</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>20</td>
<td>$D_{OT} - (R_{OGS} - X_{PDU})$</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>21</td>
<td>$D_{WT} - (R_{OGS} - X_{PDU})$</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>$(R_{OGS} - X_{PDU}) - R_{PDU}$</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>$(R_{WRS} - X_{OGS}) - R_{OGS}$</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>$(D_{WT} - X_{OGS}) - R_{OGS}$</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>$R_{OGS} - (R_{WRS} - X_{OGS})$</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>26</td>
<td>$R_{PDU} - (R_{WRS} - X_{PDU})$</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>27</td>
<td>$D_{OT} - (R_{WRS} - X_{PDU})$</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>28</td>
<td>$(D_{WT} - X_{PDU}) - (R_{WRS} - X_{PDU})$</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>29</td>
<td>$(R_{WRS} - X_{PDU}) - R_{PDU}$</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>$(D_{WT} - X_{PDU}) - R_{PDU}$</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
The sequence of events leading to a transition from its initiating event can be read from the RV representation of that transition. For example, transition 10 is represented by \( R_{OGS} - X_{WRS} \), meaning that it is defined as the time remaining before OGS repair \( R_{OGS} \) is complete when an intervening failure of the WRS \( X_{WRS} \) has occurred. Table 3.2 indicates that transition distribution 10 is used to describe the transition from state 4 to state 5. The basic event for this transition is OGS repair, but state 4 was entered via failure of the WRS after the OGS failed. Thus, the time taken for the WRS to fail (while the OGS repair is underway) is taken into account in the distribution, as shown in the RV representation.

3.3 Analysis and Results

The goal of this analysis is to examine the interplay between the capacity of the WT, the number of spares for each processor, and the overall probability of system failure. Section 3.2 set up the SMP for the analysis; following the procedures outlined in Chapter 2, this SMP is used to solve for the state probabilities \( \phi(t) \) and the Markov renewal process probabilities \( V(k, t) \), where the mission time \( t \) is 8,760 h (one year).

The goal is tackled in two parts. First, a range of WT capacities is investigated in order to determine the impact on WT capacity on the probability that the system will enter a failed state because a buffer depletes before repair can be implemented. This analysis is used to suggest a value for WT capacity. Then, CDFs for the number of spares for each processor that will be required over the course of the mission are generated and used to inform selection of the appropriate number of spares.

3.3.1 WT Sizing

The parametric definition of the WT depletion RV \( D_{WT} \) described in table 4.2 is used to examine a range of WT capacities between 0.25 and 50 kg, in increments of 0.25 kg. At each WT capacity, the probability of the system being in either failed state is calculated using equation (3.3). This represents the probability that the system will fail due to depletion of either tank before a repair can be implemented.
Figure 3-3: Decrease in the probability of failure due to increased buffer size.

The results of this analysis are shown in figure 3-3.

\[ P_{FR} = \phi_{1.18} + \phi_{1.19} \]  \hspace{1cm} (3.3)

As the WT capacity increases, the probability that the system fails due to failed repairs decreases in an S-curve shape. Initially, when the buffer size is very low (less than 5 kg), the probability of failure is quite high - nearly 0.9. This makes sense; based on the consumption rate of 0.98 kg h\(^{-1}\) given in section 3.1.2, a 5 kg water buffer would only allow the system to survive for a little over 5 h. The two processors whose failure results in activation of the WT are the WRS and PDU, which have MTTRs of 12 h and 10 h, respectively. As a result, it is unlikely that a repair could be implemented successfully for a 5 kg WT. The probability of failure drops sharply between 5 kg and 25 kg; beyond this point the probability approaches zero. This is due to the fact that after 25 kg (over 25.5 h of buffer time), the probability that a repair of either the WRS or PDU is successful is very high.

Plots such as this provide designers with a direct, quantitative connection between the size of the WT and the probability of system failure (assuming sufficient spares
are available). Since the probability tails off to approach zero at large tank sizes, this plot can be used to look at the marginal rate of return of an increase tank size. This enables designers to not only select a tank size high enough to provide sufficient confidence that repairs will be completed successfully, but also to ensure that the tank size is not unnecessarily high. In this way a balance is struck between the mass of the buffer and the benefits it gives to system resilience. For this case study, a WT capacity of 35 kg is recommended; at this size the probability of failed repairs $P_{FR}$ is approximately 0.0025.

### 3.3.2 Number of Spares

To determine the relationship between the number of spares for each processor and the resilience of the system, the Markov renewal process probabilities for this system are used to generate CDFs relating the number of spares for each processor to the probability that that many spares will be sufficient. This is accomplished using the methodology outlined in Chapter 2 - convolution of the PMFs for the set of states that represent different ways a spare for the processor can be consumed. For the OGS, this set of states includes states 2 and 11. Though state 3 is also entered via failure of an OGS, the repair process from state 3 goes back to state 2 since the OGS redundancy is identical. As a result, the counting process for state 2 captures returns from this state, and to include state 3 would be to double-count failures of the secondary OGS. For the WRS, the renewal states are 4 and 10. The applicable states for the PDU are 7, 14, and 17. The overall PMF for each processor can be obtained using equations (3.4) through (3.6). The cumulative sum of each PMF gives the CDF for that component; these are displayed in figures 3-4 through 3-6.

$$s_{OGS}(k) = v_{0,2}(k) * v_{0,11}(k)$$  \hspace{1cm} (3.4)

$$s_{WRS}(k) = v_{0,4}(k) * v_{0,10}(k)$$  \hspace{1cm} (3.5)

$$s_{PDU}(k) = v_{0,7}(k) * v_{0,14}(k) * v_{0,17}(k)$$  \hspace{1cm} (3.6)

Using these CDFs, the number of spares required to achieve a certain level of
Figure 3-4: CDF of the number of spares required for the OGS.

Figure 3-5: CDF of the number of spares required for the WRS.
confidence for that particular spare can be read directly - for example, at least 4 spares are required for the WRS in order to have a probability greater than 0.9 that sufficient spares are available (see figure 3-5). The overall probability that sufficient spares are available for the system, however, is the product of the probability for each processor. Using the data contained in these CDFs along with data for the mass cost of each additional spare, the probability of having sufficient spares can be balanced against the cost of those spares.

### 3.3.3 Final System

For this notional study, suppose the system designers select the WT capacity of 35 kg recommended above and, based on a cost-benefit analysis of the CDFs presented in section 3.3.2, decide to include 10, 6, and 1 spare for the OGS, WRS, and PDU, respectively. Using the data presented in figures 3-3 through 3-6 and equation (2.20)
from Chapter 2, the overall probability of system failure can be calculated as

\[ P_{\text{fail}} = 1 - (1 - P_{\text{FR}}) \times S_{\text{OGS}}(10) \times S_{\text{WRS}}(6) \times S_{\text{PDU}}(1) \]  
(3.7)

\[ = 1 - (1 - 0.0025) \times 0.9969 \times 0.9951 \times 0.9927 \]  
(3.8)

\[ = 0.0177 \]  
(3.9)

Thus, the resilience of the system has been characterized quantitatively as a function of its parameters and design variables.

### 3.4 Discussion

The process described above demonstrated the application of the methodology described in Chapter 2 to a simplified ECLSS, tracing the path from design variables to resilience metrics. Specifically, the effect of the size of a buffer in the system on that system’s resilience was quantified and used to select a tank capacity, and the Markov renewal probabilities for each processor were used to determine how many spares should be taken for each. The probability of failure for the resulting system was then characterized.

While this analysis varied the capacity of the WT in order to select a size, the same process could be applied to repair times instead. For example, if the WT capacity is limited or has already been set for other reasons, the repair time distribution for the WRS and PDU could be varied to produce plots similar to the one shown in figure 3-3. The data generated could then be used to define requirements on the repairability for the system - for example, to specify that repair of the WRS must take no more than 9 h.

#### 3.4.1 Combined Effects

The analysis in this chapter examined tank sizing and spares selection independently, then combined their results to determine the characteristics of the overall system. Buffer sizing and spares selection can be examined concurrently, however. Figure
Figure 3-7: Combined effect of increasing buffer size and the number of WRS spares

3-7 shows the overall probability of system failure - due to both failed repairs and insufficient spares - for this system, varying the WT capacity between 0.25 and 35 kg and the number of WRS spares between 0 and 7. It is assumed that the OGS and PDU have sufficient spares.

Figure 3-7 shows an important relationship between buffer size and the number of spares. As expected, an increase in either the buffer size or the number of spares lowers the probability of system failure. However, each is limited by the other. At low buffer sizes, increasing the number of spares can lower the probability of failure slightly, but it remains close to 0.9; this is a result of the low probability of successful repair if the buffer is too small. It does not matter how many spares are available if there is no time to implement them. Similarly, at high buffer sizes the benefits obtained depend on the number of spares that are available. A high probability of successful repair only benefits the system if the spares needed to complete that repair are available.

Note that, while continued increases of the WT capacity would continue to lower the probability of failure due to the delaying effect of the buffer in the system, the
returns are gained slowly and at a high cost. Under the assumptions of the above analysis, for example, a system with 0 spares could achieve a very low probability of failure by increasing the tank capacities to the point where the crew can survive on the stored consumables alone - effectively becoming an open-loop system. The mass of $O_2$ and water required for this would be approximately 1,840 kg and 8,585 kg, respectively. Tank capacities of this size may not be the optimal solution for the mission. Combined analysis of the buffer sizes and the number of spares can allow designers to create a system with similar resilience for much lower mass cost.

3.5 Incorporation of Contingency Responses

The above example examined the impact of system design variables on the overall probability of failure of the system. In this analysis, the possible system events examined were based upon responsive repair - that is, the solution to a failure was considered to be repair of the failed component, and once a failure occurred the only options (other than additional failures) were for the system to be repaired or enter a failed state. However, more options are often available to a system for contingency responses that do not directly repair the system. For example, possible contingency responses could be actions that seek to stabilize the system and give the crew more time to complete a repair. In cases where a processor failure takes the system to a state where the transition to a failed state can be very short, an attempt to directly return to the nominal state may be very difficult to execute before system failure, and thus may have a low probability of success. In these cases, a more prudent response is to take some action that transitions the system to a state that provides more time to complete repairs on the initial failure, even if that state lowers the overall capability of the system.

This section presents the structuring of an SMP diagram to account for events other than failures, repairs, and buffer depletions - with a focus on contingency responses. The inclusion of contingency response options in the SMP framework enables consideration of operational responses as design variables alongside buffer sizes and
the number of spares, expanding the scope of the analysis. In this way, analysis of the resilience of the system can take into account operational solutions that - rather than increasing the mass of the system - may place the system in a reduced-capability mode temporarily in order to increase the overall probability of survival. This section describes this process using the ISS cooling loop A ammonia pump module failure event of 2010 as an example of the use of contingency responses.

3.5.1 Background: ISS Pump Module Failure

On the evening of July 31, 2010, one of two redundant ISS External Thermal Control System (ETCS) coolant loops shut down, resulting in a loss of half of the station cooling capability. The shutdown was the result of a failure of the loop A ammonia pump module, which circulates ammonia through a set of radiators to reject heat from the station’s electronic systems. With the loss of one of the cooling loops, the ETCS was unable to dissipate the entire heat load of the station electronics in their nominal state. As a result, immediately after the failure (i.e. before any response action was taken), the system was in a state where there was a risk of overheating [56, 57, 58].

In the time immediately after the cooling loop failure, the station crew and ground support team implemented contingency responses to stabilize the system and ensure crew safety. These included the power-down of several systems - including scientific experiments - in order to reduce the heat load, as well as the connection of jumper cables from the U.S. segment to the Russian Zarya module to reconfigure the power system and shift thermal loads to other cooling systems [59, 56, 57]. These power-downs placed the system in a stable configuration after 8 h of work by the team [58]. By reducing and redistributing the thermal load on the system, the crew and ground support team reduced the time pressure on repairs, helping to ensure system survival until the repairs could be implemented. However, in the intervening period the system remained in a degraded state. Some portions of the system were operating at 0-fault-tolerance - meaning that there was no redundancy in the system - and many scientific experiments were placed on hold [59, 56].
The pump module (as well as two spares) was mounted on the outside of the station, on the main power truss [57]. In order to implement a repair and replace the failed pump module with a spare, extravehicular activity (EVA) by the crew was required. Initially, two EVAs were estimated to be required to complete repairs, to take place on August 5 and 7 [56]. However, a schedule slip pushed these EVAs to the 7th and 11th and an ammonia leak during the first EVA prevented removal of the pump module on schedule; as a result, a third EVA was required and cooling loop A was not fully restored until August 19th - 19 days after the initial failure - after which the station resumed its normal operating configuration [59, 60, 61].

3.5.2 SMP Representation

The sequence of events described in section 3.5.1 represents a historical example of the implementation of a contingency response to stabilize a system before implementing a full repair. An SMP diagram depicting a simplified version of these events is shown in figure 3-8. In this case, the threat of system overheating is modeled as a buffer, which can be thought of as the thermal energy in the habitat. Electronic components within the habitat increase thermal energy by injecting heat; the ETCS is a processor that removes heat. When the ETCS is not functioning, the heat in the system builds up until some critical failure point is reached and the system overheats - unless a contingency action is taken.

In figure 3-8, the system transitions out of the nominal state (state 1) into state 2 upon failure of the pump module. In this state, the system is at risk of overheating, and will do so if no further action is taken. A direct repair of the pump is possible from this state (shown by the blue arrow returning to state 1), but it is unlikely that this could be accomplished before the system overheats and transitions to the failed state (state 4). Alternatively, contingency operations can be used to transition the system into state 3, which is stable but degraded. State 3 may have some cost to the system’s productivity (such as loss of scientific data), but the stabilizing effect of the contingency operations means that there is no risk of transition to the failed state in this case. Once the system has reached state 3, repair operations can continue with
Figure 3-8: SMP diagram showing the use of contingency operations to stabilize the ISS after the 2010 ammonia pump module failure. Processor failures are shown in black, repairs are shown in blue, buffer depletions are shown in read, and contingency operations are shown in green.

a much higher probability of success due to the reduced time pressure.

In this case the contingency operations lead to a state with no transition to failed state, indicating a full stabilization of the system. However, there may be cases where contingency operations cannot completely eliminate the transition to a failed state, but instead delay it. For example, in the notional system presented in section 3.1 a potential contingency operation after WRS failure could be to ration water while the WRS is being repaired. The WT would still be depleted after a certain amount of time, but it would last longer and thereby provide more time for repairs to be implemented. In these cases, a contingency response state would still include a transition to a failed state, but that transition PDF would be altered based on the impact of the contingency response.

3.6 Chapter 3 Summary

This chapter presented a demonstration of the methodology described in Chapter 2. First, a notional ECLSS with three processors and two buffers was analyzed to determine an appropriate buffer size and number of spares for each component. The process of defining basic events, state network structure, and transition PDFs
was carried out using the design data, including a parameterization of transition PDFs based on tank capacity. The resulting SMP was then solved, and the results interpreted to determine the relationship between buffer size and probability of failed repairs, followed by the relationship between the number of spares for each processor and the probability that sufficient spares are available for a fixed buffer size. A final system structure was selected and characterized, and the overall process discussed. In particular, the combined effects of buffer size and number of spares was discussed, noting the need for both.

In section 3.5, the event sequence of a real-world failure event was recreated to show how this analysis could have been applied to inform contingency response actions. The failure of the ISS cooling loop A ammonia pump module in 2010 was described as a brief case study, showing the process of initial failure, contingency operation, and final repair. Figure 3-8 presented an SMP diagram representing the use of contingency operations, and section 3.5.2 the incorporation of contingency operations into SMP models as transitions to states that either prevent or delay transition into failed states.
Chapter 4

Case Study: Mars One

In this chapter, a case study is presented wherein the semi-Markov techniques described in Chapter 2 are applied to an analysis of the Mars One ECLSS architecture. Data and analysis in this chapter were compiled and executed in collaboration with Sydney Do, Sam Schreiner, and Koki Ho, as part of a larger analysis of the Mars One mission plan [62].

4.1 Background

The Mars One Foundation is a not-for-profit organization seeking to establish the first permanent human settlement on Mars. Announced in 2012, the Mars One program intends to utilize existing technology to support a colony, claiming that “no new technology developments are required to establish a human settlement on Mars” [63]. The first crew of 4 is scheduled to depart Earth in 2024, arriving on Mars the following year. Every two years following this first landing, 4 more crewmembers will join the growing colony, bringing with them consumables and hardware to expand the habitat and resupply the colony. The supplies brought by additional crews will augment consumables generated via In-Situ Resource Utilization (ISRU), and will need to include spare parts and other components to support maintenance activities [64, 9, 62].

The Mars One ECLSS architecture is based around Life Support Units (LSUs), which are independent ECLSS packaged within landers that are hooked up to inflat-
able habitats in order to create a habitable environment for the crew. Figure 4-1 shows an artist’s rendering of what the Mars One colony might look like, showing the line of landers - some of which contain LSUs - connected to inflatable habitats (buried beneath Martian soil mounds behind the landers). In keeping with the Mars One Foundation’s claim that no new technology will be required, these LSUs will “be very similar to those units which are fully functional on-board the International Space Station” [10, 9, 63].

In order to minimize the probability of ECLSS failure, the Mars One ECLSS architecture includes dual redundancy on the LSUs; two independent LSUs are available to support each crew of 4 [9]. However, an important question remains: how many spares will be required to maintain these systems during the two years between crew arrivals? The number of spares utilized will affect the mass of spares that need to be brought by subsequent missions, and must be included in consideration of the costs of the system.

4.2 Analysis Goal

The goal of this analysis is to examine the air and water management systems of a Mars One LSU and determine what mass of spares will be required over the two-year duration between crew arrivals. Using ISS ECLSS data along with the SMP methodology presented in Chapter 2, CDFs of the number of spares required for
each processor in the system will be developed. These CDFs will be utilized along with mass data for each processor to examine the trade between mass of spares and the probability that sufficient spares are supplied. The end goal is to identify the minimum required mass of spares to achieve 0.95, 0.99, and 0.999 probability of sufficient spares.

4.3 Model Structure

4.3.1 System Description and Assumptions

The assumed architecture of the LSU air and water management systems is shown in figure 4-2. This architecture is based on current ISS ECLSS, and utilizes ISS ECLSS components. This system incorporates six processors to manage air and water ECLSS requirements:

- **Solid Polymer Water Electrolysis (SPWE):** produces O\textsubscript{2} and H\textsubscript{2} via electrolysis of water. The O\textsubscript{2} is injected into the habitat atmosphere, and the H\textsubscript{2} is supplied to the Sabatier system for CO\textsubscript{2} reduction [65].

- **Four Bed Molecular Sieve (4BMS):** removes CO\textsubscript{2} from the habitat atmosphere via adsorption onto a molecular sieve. The collected CO\textsubscript{2} is passed on to the Sabatier system for reduction [65].

- **Sabatier:** reduces CO\textsubscript{2} into water and methane, using the H\textsubscript{2} generated by the SPWE. The water is injected into waste water storage, and the methane is vented or stored for use as fuel [66].

- **Vapor Compression Distillation (VCD):** distills urine to recover water, which is injected into waste water storage [65].

- **Common Cabin Air Assembly (CCAA):** dehumidifies the habitat atmosphere to recover water vapor using a condensing heat exchanger. Recovered condensate is injected into waste water storage [65].

- **Multifiltration (MF):** filters waste water into potable water [66].

Reliability and mass data for these processors are presented in table 4.1.
Figure 4-2: Major components of the Mars One LSU air and water management systems (based on ISS ECLSS) and their interactions [10].

Table 4.1: Air and water processor data, including mass and MTBF.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SPWE</td>
<td>O₂ generation</td>
<td>113</td>
<td>476 [66]</td>
<td></td>
</tr>
<tr>
<td>4BMS</td>
<td>CO₂ removal</td>
<td>201</td>
<td>335 [66]</td>
<td></td>
</tr>
<tr>
<td>Sabatier</td>
<td>CO₂ reduction</td>
<td>18</td>
<td>365 [66]</td>
<td></td>
</tr>
<tr>
<td>VCD</td>
<td>Urine processing</td>
<td>128</td>
<td>269 [66]</td>
<td></td>
</tr>
<tr>
<td>CCAA</td>
<td>Dehumidification</td>
<td>96</td>
<td>922 [54]</td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>Water processing</td>
<td>476</td>
<td>228 [66]</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4-2 shows the interactions between these six processors, as well as six buffers (O\textsubscript{2}, CO\textsubscript{2}, water vapor, urine, waste water, and potable water) that serve as interfaces between the processors and the crew. The habitat atmosphere is modeled as a set of three buffers - one each for O\textsubscript{2}, CO\textsubscript{2}, and water vapor. These buffers represent the primary atmosphere constituents that are acted upon by the crew and processors. This diagram shows a dependency of the Sabatier on the SPWE and 4BMS. The Sabatier requires H\textsubscript{2} from the SPWE and CO\textsubscript{2} from the 4BMS to reduce CO\textsubscript{2} into water; if either of these supply lines are interrupted, the Sabatier goes offline. The processor is assumed to not fail while offline. Failure of the Sabatier does not, however, take either the SPWE or the 4BMS offline. They both continue to function, producing O\textsubscript{2} and removing CO\textsubscript{2}, and the waste products H\textsubscript{2} and CO\textsubscript{2} are vented overboard until the Sabatier is brought back online.

This analysis will focus on a single LSU, looking into the logistical supply requirements on a per-LSU basis. For the purposes of this analysis, it is assumed that all buffers are large enough that they completely isolate individual processor failures. Effectively, the buffers are assumed to be sufficiently large that the probability of a buffer depleting before a repair can be successfully implemented is negligible.

Sparing is assumed to be done at the processor level, and in the event of a failure the failed processor is removed and replaced with an identical replacement. This replacement process returns the system to a good-as-new state. In addition, for simplicity it is assumed that all processors’ repair times can be characterized by the same lognormal distribution, with an MTTR of 24 h and a standard deviation of 4 h. Finally, in order to constrain the number of states in the SMP, it is assumed that no more than 2 concurrent failures occur.

### 4.3.2 SMP Generation and Analysis

In order to determine what mass of spares will be required, the Markov renewal process probabilities are calculated for each processor. The basic events consist of failures of the processors and the repair process (which is the same for all processors), as listed in table 4.2. Using these basic events, the dependencies in the system, and
Table 4.2: Basic events and their distributions. All distribution parameters are based on a timescale of days.

<table>
<thead>
<tr>
<th>Basic Event</th>
<th>RV</th>
<th>PDF Form</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPWE failure</td>
<td>$X_{SPWE}$</td>
<td>Exponential</td>
<td>$\lambda_{SPWE}$</td>
<td>$2.101 \times 10^{-3}$</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>$X_{4BMS}$</td>
<td>Exponential</td>
<td>$\lambda_{4BMS}$</td>
<td>$2.985 \times 10^{-3}$</td>
</tr>
<tr>
<td>Sabatier failure</td>
<td>$X_{Sab}$</td>
<td>Exponential</td>
<td>$\lambda_{Sab}$</td>
<td>$2.740 \times 10^{-3}$</td>
</tr>
<tr>
<td>VCD failure</td>
<td>$X_{VCD}$</td>
<td>Exponential</td>
<td>$\lambda_{VCD}$</td>
<td>$3.717 \times 10^{-3}$</td>
</tr>
<tr>
<td>CCAA failure</td>
<td>$X_{CCAA}$</td>
<td>Exponential</td>
<td>$\lambda_{CCAA}$</td>
<td>$1.085 \times 10^{-3}$</td>
</tr>
<tr>
<td>MF failure</td>
<td>$X_{MF}$</td>
<td>Exponential</td>
<td>$\lambda_{MF}$</td>
<td>$4.386 \times 10^{-3}$</td>
</tr>
<tr>
<td>Repair</td>
<td>$R$</td>
<td>Lognormal</td>
<td>$\sigma_{WT}$</td>
<td>0.2462</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\mu_{WT}$</td>
<td>-0.0303</td>
</tr>
</tbody>
</table>

the limitation to two concurrent failures, a state network of 91 states with 20 different transitions is generated. This state network is too large to be displayed here; the state network outline is instead presented in Appendix B. Table 4.3 shows the combinations of basic event RVs used to generate the 20 state transition distributions.

Using this state network and set of transition distributions, the kernel matrix $Q(t)$ and unconditional waiting time density matrix $H(t)$ are generated according to the methodology described in Chapter 2, thereby defining the SMP for this analysis. This SMP is then solved for the Markov renewal processes probabilities for each state, given that the system began in state 1. The PMF $s(k)$ and CDF $S(k)$ of the number of spares required for each component is generated using equation (2.19), where the set of renewal states for each processor is given by

$$S_{SPWE} = \{2, 16, 29, 45, 61, 77\} \quad (4.1)$$
$$S_{4BMS} = \{3, 15, 32, 48, 64, 80\} \quad (4.2)$$
$$S_{Sab} = \{28, 51, 67, 83\} \quad (4.3)$$
$$S_{VCD} = \{6, 19, 35, 44, 70, 86\} \quad (4.4)$$
$$S_{CCAA} = \{9, 22, 38, 54, 60, 89\} \quad (4.5)$$
$$S_{MF} = \{12, 25, 41, 57, 73, 76\} \quad (4.6)$$

Note that all processors have six renewal states except for the Sabatier, which only has four. This is due to the dependence of the Sabatier on the SPWE and 4BMS. Since
Table 4.3: RV combinations used to generate transitions in the state network (with transition ID number).

<table>
<thead>
<tr>
<th>ID</th>
<th>RV Combination</th>
<th>$C_+$ Index</th>
<th>$C_-$ Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X_{SPWE}$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$X_{4BMS}$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$X_{Sab}$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$X_{VCD}$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$X_{CCAA}$</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$X_{MF}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$R$</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$R - X_{4BMS}$</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>$R - X_{VCD}$</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>$R - X_{CCAA}$</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>$R - X_{MF}$</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>$R - X_{SPWE}$</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>$R - X_{Sab}$</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>$R - R$</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>$R - (R + X_{4BMS})$</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>$R - (R + X_{VCD})$</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>$R - (R + X_{CCAA})$</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>$R - (R + X_{MF})$</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>$R - (R + X_{SPWE})$</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>$R - (R + X_{Sab})$</td>
<td>14</td>
<td>3</td>
</tr>
</tbody>
</table>

Failure of either of these processors takes the Sabatier offline, the Sabatier cannot fail while the SPWE or 4BMS are failed. As a result, there are two partially failed states in which every component can fail except for the Sabatier (states 2 and 15).

### 4.4 Results

The results of this case study are in two parts. First, the CDFs for the number of spares required for each processor individually are presented in figures 4-3 through 4-8. Second, a multiobjective tradespace of spares mass vs. system reliability is shown in figure 4-9, and the minimum mass architecture that achieves an overall probability of sufficient spares greater than three threshold values (0.95, 0.99, and 0.999) is identified for each threshold value in table 4.4.
4.4.1 Spares CDFs

Figures 4-3 through 4-8 show the relationship between the number of spares for a given processor and the probability that sufficient spares are supplied. For these plots, the CDFs are only shown for probabilities above 0.95. The highest probability on each plot is the first probability higher than 0.99999 - it is assumed that beyond this point any gain in reliability through the addition of more spares is negligible. The x-axis for each plot is scaled accordingly; the lowest value on the x-axis is the number of spares of that component required to have a probability higher than 0.95, and the highest value is the number of spares required to have a probability higher than 0.99999.

The values shown in these plots track well with intuition based on the MTBF values for each processor. For example, the CCAA has the highest MTBF of the six processors in this analysis at 922 d. It requires 3 spares to achieve a probability higher than 0.95 (see figure 4-7). With 8 spares, the CCAA has a greater than 0.99999 probability of having sufficient spares. In contrast, the MF (figure 4-8) - which has the lowest MTBF at 228 d - requires 7 spares to achieve a probability higher than 0.95, and 14 spares for a probability higher than 0.99999.

4.4.2 Multiobjective Optimization

By constraining the number of possible spares for each processor to the values giving probabilities from 0.95 to just above 0.99999 (the values shown in figures 4-3 through 4-8), a total of just under 85,000 possible resupply architectures are identified, with each architecture characterized by the number of spares resupplied for each processor. Using this information, the total mass of spares per resupply $M$ for each architecture can be identified, as well as the spares probability $P$ (i.e. the probability that the number of spares supplied is greater than the number required before the next
Figure 4-3: CDF of the number of SPWE spares required.

Figure 4-4: CDF of the number of 4BMS spares required.
Figure 4-5: CDF of the number of Sabatier spares required.

Figure 4-6: CDF of the number of VCD spares required.
Figure 4-7: CDF of the number of CCAA spares required.

Figure 4-8: CDF of the number of MF spares required.
Figure 4-9: Tradespace of spares probability and mass of spares. The minimum mass architectures that achieve a spares probability of 0.95, 0.99, and 0.999 are shown by the blue, green, and red dots, respectively. The minimum mass and the number of spares for each processor for the three probability thresholds are shown in table 4.4.

resupply) using equations (4.7) and (4.8):

\[
M = \sum k_i m_i \tag{4.7}
\]

\[
P = \prod S_i(k_i) \tag{4.8}
\]

where \(k_i\) is the number of spares supplied for processor \(i\), \(m_i\) is the mass of that processor, and \(S_i\) is the CDF for that processor.

A scatter plot of all of these resupply architectures that provide a spares probability greater than 0.9 is shown in figure 4-9. A distinct Pareto frontier is visible in the direction of utopia - high probability of having enough spares and low spares mass. This Pareto front is expected for this sort of trade - in order to increase the spares probability, the mass of spares must be increased, and a reduction in the mass of spares supplied will result in a reduction of the probability that the system will have enough spares.
Table 4.4: Minimum mass architectures above a set of spares probability thresholds.

<table>
<thead>
<tr>
<th>Spares Probability</th>
<th>Mass of Spares [kg]</th>
<th>SPWE</th>
<th>4BMS</th>
<th>Sabatier</th>
<th>VCD</th>
<th>CCAA</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>7427</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>0.99</td>
<td>8701</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>0.999</td>
<td>10410</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>7</td>
<td>11</td>
</tr>
</tbody>
</table>

Three points are picked out of the tradespace corresponding to the minimum-mass resupply architecture that provides a spares probability greater than 0.95, 0.99, and 0.999 (shown in blue, green, and red, respectively). The mass of spares in each of these architectures - as well as the number of spares resupplied for each processor - is shown in table 4.4. In order to achieve a probability greater than 0.95, 7,427 kg of spares are required for every two years of operation. To achieve a probability of 0.99, the required mass is 8,701 kg - an increase of 1,274 kg for an increase of 0.04 in probability. To reach 0.999, 10,410 kg of spares must be supplied - an increase of 1,709 kg for an increase of only 0.009 in probability. These numbers illustrate the diminishing return on investment for additional spares, and give designers a good basis upon which to make system trades with respect to system resilience and resupply mass.

4.5 Case Study Discussion

This case study demonstrated the use of the SMP methodology outlined in Chapter 2 to inform logistics campaigns by identifying the number of replacement spare parts that must be brought to in each resupply mission to account for spares that were used in the time since the last mission. The analysis took into account the structure of the system (including dependence of the Sabatier on the SPWE and 4BMS) as well as the combined effects of the sparing levels of all components. The output of the analysis was more than just a point design; instead, an entire tradespace of options was developed, with a clear Pareto frontier that can be used by system designers to trade between Pareto-optimal spares manifesting options.
Table 4.5: Minimum mass architectures above a set of spares probability thresholds for an increased MTBF system.

<table>
<thead>
<tr>
<th>Spares Probability</th>
<th>Mass of Spares [kg]</th>
<th>SPWE</th>
<th>4BMS</th>
<th>Sabatier</th>
<th>VCD</th>
<th>CCAA</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>6918</td>
<td>6</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>0.99</td>
<td>8258</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>0.999</td>
<td>9952</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.6: Resupply spares mass savings for a 10% increase in MTBF.

<table>
<thead>
<tr>
<th>Spares Probability</th>
<th>Baseline Mass [kg]</th>
<th>Increased MTBF Mass [kg]</th>
<th>Mass Savings [kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>7427</td>
<td>6918</td>
<td>509</td>
</tr>
<tr>
<td>0.99</td>
<td>8701</td>
<td>8258</td>
<td>443</td>
</tr>
<tr>
<td>0.999</td>
<td>10410</td>
<td>9952</td>
<td>458</td>
</tr>
</tbody>
</table>

4.5.1 Sensitivity to MTBF

The MTBF values for the processors listed in table 4.1 are based upon the technology implemented on the ISS today. Even though the Mars One Foundation states that “no new technology developments are required” for their mission, it is logical to assume that some improvements to processor reliability will be made in the ten years between now and the launch of the first Mars One crew in 2024 [63, 9]. In order to examine the possible effect of an increase in processor reliability, the analysis presented above was repeated with a 10% increase applied to all processor MTBF values. The resulting tradespace is shown in comparison to the baseline in figure 4-10. The baseline data are plotted as black points, and the data for increased MTBF are shown in gray. The minimum mass architectures for 0.95, 0.99, and 0.999 are once again determined and plotted as the blue, green, and red triangles, respectively. The minimum mass architectures for the baseline case are shown with circles, as before. The resulting architectures and their spares mass are displayed in table 4.5, and the minimum mass architectures for the baseline and increased MTBF cases are compared in table 4.6.

As expected, increased component reliability resulted in a decrease in the overall mass of spares required. Interestingly, however, the actual number of spares utilized for each component did not necessarily decrease. For example, the baseline minimum mass architecture for 0.999 spares probability utilized 10 spares for the Sabatier and
Figure 4-10: Changes to tradespace from a 10% increase in MTBF for all processors. Baseline data are plotted as before; the increased MTBF case is shown in gray, with minimum mass points represented by blue, green, and red triangles for spares probability of 0.95, 0.99, and 0.999, respectively.
11 for the MF. For the increased MTBF case, however, these numbers were switched; 11 Sabatier spares were included, and only 10 spares were included for the MF. All other component sparing levels remained the same. This result, as well as the results for the other spares probability thresholds, show that an across-the-board increase in processor reliability does not simply translate into a proportional change in the number of spares for each processor included in the system. Instead, the mass cost of spares probability can be redistributed across the different system elements. In the case of the Sabatier and MF described here, a 476 kg MF spare was exchanged for an 18 kg Sabatier spare in order to take advantage of the 10% increase in processor MTBFs and achieve the same spares probability for a lower mass. This kind of non-intuitive redistribution of spares in order to adapt to changes in processor MTBFs is easily captured using the multiobjective methods described here.

4.5.2 Comparison to Open Loop

The mass of consumable resupply required to enable the use of an open loop system provides a good baseline against which to evaluate the cost of spares resupply for a regenerative system such as the one examined here. The first generation Mars One surface habitat architecture supports a crew of four for two years before the second crew arrives; in this time, based on consumption rates of 0.835 kg CM\(^{-1}\) d\(^{-1}\) and 3.909 kg CM\(^{-1}\) d\(^{-1}\) for O\(_2\) and H\(_2\)O, respectively, an open loop air and water management system would need to supply 2438.2 kg of O\(_2\) and 11,414.28 kg of water, for a total consumable mass of approximately 13,852 kg [54]. This does not account for CO\(_2\) and humidity control processes, which would still necessitate some form of atmospheric processing - though this system could likely be implemented in a simpler form in an open-loop capacity, spares of some form would still be required. However, examination of the open loop consumable mass requirements provides a good basis for comparison of resupply architectures.

Put in context of the open loop resupply requirements, the minimum mass of spares required for 0.999 probability of success is equivalent to over 75% of the open loop mass requirement. Put another way, a consumable mass equivalent to the mass
of spares in this resupply architecture could support the crew in open loop mode for over 548 d - three quarters of the required duration. Even with a 10% increase in processor reliability, the spares mass required for 0.999 probability of success is nearly 72% of the open loop requirement; the equivalent consumable mass would support the crew for over 524 d.

If the probability requirements are relaxed to 0.99, the required spares mass is reduced to approximately 63% and 60% of the open loop consumable requirement for the baseline case and the case with a 10% increase in reliability. Even with the accepted risk of one failure per 100 missions (a fairly high risk to accept for a human space program), the mass of spares required to maintain the regenerative system is still well over half of the mass of consumables that would be required to support an open loop system. Given the higher complexity and greater uncertainty of regenerative systems compared to their open-loop counterparts, the mass savings associated with the use of a regenerative system may not be worth the additional risk [3, 23, 24].

4.5.3 Level of Discretization

The analysis applied above discretized the system at the processor level; that is, reliability and mass data was given for the processors, and when a failure occurred the full processor was replaced. As a result, any failure of any kind within the processor requires an additional spare at a cost of the mass of a full processor. While this maintenance paradigm may reduce operational complexity by encapsulating complex systems into modules, it is likely not the most mass-efficient way to supply spare parts to the system. Repair at a lower level could reduce the mass of spares used, but would require more advanced diagnostic and repair capability on-site in order to identify and replace the specific failed component. In addition, replacement of lower-level components may increase the time required to implement repairs [16].

Many of these processors are made up of smaller subassemblies that can be designed to be replaceable. For example, the CCAA includes a condensing heat exchanger, a water separator, and various sensors and electronic subassemblies, each of which makes up only a portion of the mass of the overall system and has its own
MTBF. If replacement occurs at the subassembly (or even component) level rather than at the processor level, significant mass savings could be realized in spares re-supply. If a 100 kg processor fails every 50 days because of a 5 kg subassembly but has other portions that could last for years, replacement of that subassembly rather than of the entire processor at each failure will result in a significantly lower mass of spares used.

As a result of the processor-level discretization of the system, it is likely that the spares mass values presented in figure 4-9 and table 4.4 are overestimates. This depends, however, on what level spares and repairs will be implemented for the actual Mars One LSU. With more detailed design data on the constituent elements of each processor, a more complex SMP could be constructed that analyzes the sparing problem at a subassembly level and likely determines a lower-mass solution for each of spares probabilities presented in table 4.4.

4.5.4 Optimization

In this analysis, the multiobjective optimization process took the form of a partial enumeration of the design space. The individual processor CDFs were used to identify ranges on the number of spares for each processor, then the full factorial combination of these ranges was used to produce a tradespace. Enumeration of this kind was facilitated by the fact that there are only a few discrete design variables with small ranges of values, resulting in less than 85,000 options. In this small design space, enumeration is a quick and simple method that enables examination of every possibility in order to determine the Pareto-optimal architectures.

Enumeration is, however, a rather blunt optimization tool, and becomes less feasible as the size of the design space increases. For example, if this case study sought to determine the overall probability of system failure - including buffer depletion and insufficient spares - and included the urine, waste water, and potable water tank capacities as design variables in addition to the number of spares for each component, the design space would consist of 9 design variables, 3 of which are continuous. In this case, more elegant optimization tools - such as Normal Boundary Intersection
Adaptive Weighted Sum [68], or multiobjective genetic algorithms [69] - can provide a more effective and timely way to examine the tradespace and find the set of Pareto-optimal solutions.

4.6 Chapter 4 Summary

In this chapter, an application of the SMP methodology to Mars One LSU resupply logistics was presented. Sections 4.1 and 4.2 described the Mars One project and presented the goal of this analysis: the determination of the mass of spares required to keep the system running during the two-year interval between resupply missions. Section 4.3 constructed an SMP model of the system, listing modeling assumptions and basing the system design on ISS ECLSS (as shown in figure 4-2). Basic events were defined and used to construct a state network and define transition PDFs. This SMP was then solved for the Markov renewal probabilities, which were used to construct the CDFs of the number of spares required in section 4.4.

A multiobjective tradespace characterizing a set of options in terms of the mass of spares required and the resulting probability that sufficient spares are supplied was developed and shown in figure 4-9. This tradespace enabled identification of a Pareto front, and the minimum-mass architectures for a set of threshold probabilities were identified and presented in table 4.4. Section 4.5 presented a discussion of the case study, including an analysis of the sensitivity of the results to changes in processor reliabilities and a comparison of the required mass of spares to the mass of consumables that would be required to operate the system in open loop mode for the same duration. It was also noted that the spares masses obtained in this analysis are likely overestimates and that a more detailed, lower-level analysis that utilizes spares at the subassembly level would likely result in lower minimum-mass solutions for a given spares probability. In addition, while the optimization algorithm applied here was a brute-force enumeration of all design options within a region of the design space, more elegant optimization algorithms could be applied to more complex formulations of the SMP analysis.
THIS PAGE INTENTIONALLY LEFT BLANK
Chapter 5

Discussion

This chapter discusses the methodology described in Chapter 2, drawing from the experience gained in the demonstration and case studies of Chapters 3 and 4. The strengths of the analysis technique are discussed, as well as assumptions and limitations. In addition, a brief comparison of the method to the risk analysis methods described in the literature review (Chapter 1) is given.

5.1 Strengths

As demonstrated in Chapters 3 and 4, the application of SMP modeling to ECLSS resilience enables the calculation of probabilistic system characteristics based directly on the system design variables. Application to the notional ECLSS of the Chapter 3 demonstrated the ability of the analysis to incorporate information about both buffer size and the number of available spares in order to determine the overall probability of a mission’s success or failure, and discussion of the failure of the ISS ETCS ammonia pump module and the resulting response by NASA showed how contingency response actions could be included as well.

More important than the methodology’s use as a pure analysis tool - that is, for the calculation of specific system metrics from specific design variables - is its ability to serve as a design tool. Chapter 4 in particular demonstrated this capability with the development of a two-dimensional trade study examining the mass of spares that
must be resupplied at a given interval and the probability that the number of spares supplied will be sufficient until the next resupply mission. In addition, Chapter 3 included the examination of a range of tank capacities and their impact on the probability of failure in order to inform selection of a size for the water tank; in this case study a set of CDFs describing the number of spares required for each processor were also developed and used to inform sparing.

In both cases, the output of the analysis was a tradespace rather than a point design. This capability of the methodology to quickly and easily develop a set of system options using a single model framework representing the system’s overall architecture - exemplified by the spares CDFs for each processor in the system, which can be calculated for a wide range of number of spares in a matter of seconds - provides a key resource for designers. Using trade studies such as the one shown in figure 4-9, system designers can identify and characterize the set of Pareto-optimal architectures and use that information to trade among metrics and select a design that best suits the needs of the mission at hand. In addition, the format of the SMP analysis enables the consideration of many design variables within a single framework, including:

- Component reliabilities
- Component costs (mass, power, volume, financial, etc.)
- Individual buffer capacities
- Number of spares per component
- Contingency responses

In addition, the state network generation algorithm (which is currently executed manually) has the potential to be automated, which would enable design variables to include the architecture and connectivity of the system itself. Automation of state network generation is further discussed as future work in Chapter 6.

The structure of the SMP model - with clear, quantitative inputs and outputs in the form of system design variables and metrics - lends itself well to the application of formal optimization algorithms. This is discussed briefly at the end of Chapter 4 in the context of the tradespace exploration that was performed for the Mars One case study. In that case, a partial enumeration of the tradespace was utilized; however, other
methods such as Normal Boundary Intersection [67], Adaptive Weighted Sum [68], or multiobjective genetic algorithms [69] could also be utilized, incorporating either the SMP model on its own or as part of a larger system analysis. Optimization methods such as these have been applied in single- or multi-objective form to ECLSS studies before, usually seeking to minimize system cost or mass or maximize productivity [70, 53]. Implementation of this SMP methodology has the potential to add risk and/or resilience to the set of metrics that can be included in an optimization.

5.2 Assumptions

Chapter 2 described two assumptions that are required to enable the use of the SMP structures utilized in this methodology. The implications of these assumptions are discussed here, and potential strategies to eliminate the need for them are presented as future work in Chapter 6.

5.2.1 Exponential Failure Distributions

In order to facilitate the use of Markov renewal processes to count the number of visits to each state and thereby develop distributions of the numbers of spares, the PDFs describing component failures must be exponential. This is known as the constant failure rate model, and is a common first-order model of the time to failure for randomly-failing components [49]. Exponential distributions are “memoryless”, and as a result they enable state networks where the initial nominal state can be returned to whenever all failed components are repaired [50]. This looping structure is key to the calculation of spares CDFs, and cannot be implemented without a memoryless failure distribution.

Since the exponential distribution is already used a first-order model for component failures, it enables the direct application of the SMP technique described in this thesis for a first-order analysis of the failure characteristics of a system. When a system is in the early phases of design, and the only data available to describe component reliability are MTBF values, the constant failure rate model may be sufficient.
Figure 5-1: Bathtub curve failure rate model, showing changes in the failure rate of a component over its lifetime [11].

However, for some components an exponential distribution may not be an accurate representation of the time to failure, or may not capture the full richness of data that are available.

For example, a common model for component failure that includes changes in failure rate due to burn-in of new components and wear out towards the end of component lifetime is the bathtub curve, shown in figure 5-1 [11, 71]. In this model, the failure rate of a particular component is higher at the beginning of life due to “infant mortality” caused by design faults or manufacturing defects; this high failure rate decreases as the component ages and confidence that it is defect-free increases. Once a component survives beyond the burn-in period, the failure rate stabilizes at a constant level representing random failures during the component’s useful lifetime. At the end of life, as the component begins to wear out, the failure rate increases again. Though the validity of decreasing failure rate at the beginning of life has been debated, the increasing failure rate at the end of life provides a good model in particular for mechanical components, such as gearboxes, that experience material fatigue that weakens their structure [11, 71, 72]. Other components, such as filters, may be designed for specific lifetimes and experience sharply increasing failure rates due to clogging or other effects once they reach their design life.

For components with non-constant failure rates, multiparameter distributions such
as the Weibull distribution can be used to model the PDF of time to failure [47, 71]. For limited-life items that are replaced on a regular basis even if they have not failed (i.e. scheduled repair), bounded PDFs such as the Beta distribution could be used [47]. However, these distributions are not memoryless. As a result, the use of looping structures in the state network would inaccurately describe the behavior of the system since a return to the nominal state due to the repair of one component would reset all of the failure distributions exiting that state. For components that follow the bathtub curve model, this would mean that anytime any repair is executed all components would go through the burn-in period again, and any component that was experiencing increased failure rates due to wearout would have its transition reset, even though it has not been repaired. Without these loops, Markov renewal process probabilities cannot be calculated, and as a result the CDF of the number of spares required for a given component cannot be calculated directly using the method described in Chapter 2.

5.2.2 Independent Event Distributions

The second assumption is that the PDFs describing different failure, repair, and depletion processes in the system are independent. This facilitates the use of basic events and the truncated difference distribution to construct the PDFs describing each transition in the state network and limits the number of basic events. Dependence between components is still captured based on connections in the system diagram, but the basic events themselves are independent.

This assumption must be applied with care, since there are several situations in which it may not be correct. For example, if the temperature or humidity of a component’s environment impacts its failure rate, then a failure of the component controlling the temperature and humidity of that environment would impact the failure distribution for that component. In this case, the failure distribution of the second component is not independent from the first, and the failure distribution describing the second failure would have to be described with a conditional probability distribution.
5.3 Limitations

Two primary limitations exist to the application of this method, based on the potentially limited availability of PDFs describing events within a system and the requirement of smooth continuous distributions for implementation of the EULER numerical ILT algorithm used in this model. These limitations - and potential means to mitigate them - are described here.

5.3.1 Availability of Probability Distributions

The SMP modeling technique described in this thesis depends upon probability distributions describing the basic events in a system - failure, repair, and buffer depletion. As with any model, the outputs are only as good as the inputs, and so the objectivity and accuracy of the distributions used to describe the basic events have a direct impact on the objectivity and accuracy of the model results overall. As such, it is important that the distributions be based in empirical data as much as possible, either from testing or operational history.

Since the failure distributions are required to be exponential for the analysis, the only value required to describe them is the MTBF (or alternatively failure rate $\lambda$). These values are available from various sources, and are often based on ISS operational history in the form of the Modeling and Analysis Data Set (MADS) [54, 66, 5]. Repair distributions are more difficult to obtain, though MTTR data is also reported in some references [54]. The lognormal distribution provides a good representation of corrective repair processes, and an MTTR value coupled with a standard deviation representing uncertainty in that value can be used to construct a repair time distribution, as described in Chapter 3 [51, 52].

Parametrization of depletion distributions was utilized in Chapter 3 to allow the use of buffer size as a design variable. In this case the buffer size and consumption rate were used to determine the mean time to depletion, which was then used to construct a lognormal distribution. However, if the buffer size or consumption rate vary over the course of the mission (as would be the case in systems that employ batch
processing), they can be represented by an RV. These RVs can also be parametric, and the RV describing buffer depletion can then be derived via combination of the size and rate RVs.

### 5.3.2 Smooth Continuous PDFs

The method used to solve the SMP for system metrics (presented in Chapter 2) makes use of the Laplace domain, requiring the application of LTs and ILTs. Since the resulting solution in the Laplace domain may not have a closed-form representation for its ILT, a numerical method must be utilized. Based on the suggestion of Warr and Collins, this thesis has utilized the EULER method for numerical ILT, which is described in detail in Appendix A [13, 48]. The EULER method can provide accurate and rapid calculation of the ILT of a function at a given point, but it requires that the function be smooth (that is, the CDF of the function must have at least two continuous derivatives) [13, 48]. The exponential and lognormal distributions used in this thesis have sufficient smoothness, but if other, less smooth distributions are desired, then the EULER method cannot be applied directly. In this case, the non-smooth distribution could be approximated by a smooth one, or convolution smoothing (using a known transform) could be applied before applying EULER [48].

### 5.4 Comparison to Existing Methods

The SMP methodology described in Chapter 2 provides a quantitative, analytically derived resilience metric for a given system. This is not accomplished by risk matrices or FMECA, since their outputs are generally qualitative and subjective in nature. However, FMECA does provide a richer dataset than the SMP method if executed properly, since it includes criticality and examination of corrective actions. Fault trees, RBDs, and MRBDs provide quantitative outputs as a function of their inputs, but these are in the form of overall probabilities of success or failure. In addition, these analyses do not incorporate the ability to examine repair processes.

Whereas the SMP methodology presented here has focused on guiding the selection
of system design variables (i.e. buffer sizes and the number of spares) within the
context of an existing architecture, STPA guides the creation of a robust system
control architecture. Effectively, STPA embodies the process of determining what
system architecture (control structure) is required in order for a system to operate
safely, and the SMP method provides a means to examine design options for that
architecture and quantify their resilience. The combined application of these two
processes could provide very useful results during the system architecting and design
process; this possibility is discussed further as future work in Chapter 6.

The SMP method is most similar to PRA and EPA. The state network genera-
tion process is very similar to PRA’s event sequence diagram construction and the
creation of the Markov model in EPA. However, the SMP-based structure of the
state network enables the use of non-exponential PDFs to describe events such as
repair or buffer depletion, while the Markov model framework would require that all
transitions be exponential. In addition, whereas the PRA event sequence diagram
poses each event in the sequence as a yes/no question in order to determine end state
probabilities, the SMP method utilizes the transition PDFs and the structure of the
kernel matrix entries (see equation (2.3) in Chapter 2) in order to determine not only
which event occurs, but how long it takes to occur. Both PRA and EPA include a
means to determine the end state probabilities, and EPA includes the calculation of
state probabilities across the mission timeline, but neither methodology includes a
counting process such as the one used in the SMP method to generate spares CDFs.

5.5 A Note on System Safety

It is important to note that the SMP analysis technique described in this thesis is only
intended to examine the reliability and resilience of a system. The results generated
from this method do not provide a measure of how safe a system is, nor are they
intended to. Safety and reliability are two different characteristics; a system can be
reliable without being safe, and likewise a safe system is not necessarily reliable [36].
Both are, however, desireable characteristics in a critical system such as ECLSS. As
described in the literature review in Chapter 1, system safety is an emergent property of the interactions between elements within a system, and a system’s structure can be adapted to become more safe by implementing controls (with the appropriate action and feedback pathways) in order to mitigate possible hazardous states [8]. Reliability and resilience, however, are a property that emerges from the structure of a system as well as the characteristics of the components within it. By considering both of these portions of system design - the creation of a safe control structure and the instantiation of that control structure in a reliable and resilient manner - systems can be created that are reliably safe. This is discussed further as future work in Chapter 6.

5.6 Chapter 5 Summary

This chapter presented a discussion of the SMP methodology used in this thesis. Strengths of the methodology were described in section 5.1, which noted that the SMP analysis framework provides a clear analytical connection between the system design variables and resilience-related metrics. This results in a powerful analysis and design tool that can be used to quickly construct a tradespace of design options and even be implemented in formal multiobjective optimization algorithms. Section 5.2 discussed the assumptions behind the model - namely, exponential failure distributions and independent basic event PDFs - and their implications. Section 5.3 discussed limitations to the methodology, including the potential lack of detailed PDFs to describe basic events and the need for PDFs used in the analysis to be smooth and continuous. Section 5.4 presented a brief comparison of the SMP method to the risk analysis methods described in the literature review in Chapter 1, noting that it is most similar to PRA and EPA. Finally, section 5.5 noted that SMP models are only intended to examine the resilience of a system, and not system safety.
Chapter 6

Conclusions and Future Work

6.1 Thesis Summary

This thesis examined the application of SMP modeling techniques to investigate the resilience of long-duration ECLSS. Chapter 1 presented the motivation for this work, noting the challenges associated with the expansion of human exploration beyond the Earth-Moon system - particularly with regard to the reliability and resilience of ECLSS. In addition, an overview of ECLSS architecture and the primary functions of these systems was presented, along with a literature review of existing risk analysis methods. Chapter 1 ended with an outline of the thesis and a statement of the research objective: to develop and implement a risk assessment technique that facilitates the quantitative consideration of system resilience during the design of ECLSS for long-duration human spaceflight.

Chapter 2 described the SMP methodology applied in this thesis, starting with an overview of SMPs in general and their relation to Markov chains. The process of SMP definition was described, followed by a method to solve for system metrics using the Laplace domain. Next, a generalization of ECLSS and a set of modeling assumptions used to translate ECLSS design data into SMP models was described; this technique involved the definition of basic events, the use of those basic events to create the state network, and the definition of transition PDFs based upon combinations of the basic event PDFs. The process of translating SMP model outputs back into ECLSS
metrics was also described, focusing in particular on CDFs of the number of spares required by the system and the overall probability of system failure.

Chapter 3 presented a demonstration of the analysis on a simplified ECLSS. This demonstration included sizing of a water tank and the selection of the appropriate number of spares for each element in the system, as well as a final analysis of the resulting system probability of failure. A discussion of the combined effects of increasing buffer sizes and the number of spares was also presented. In addition, the 2010 ISS ETCS ammonia pump module failure was used as a historical example to demonstrate the application of contingency responses to increase the probability of successful repair.

Chapter 4 examined the Mars One LSU logistics requirements, using the SMP method to determine the CDFs of the required number of spares for each processor. Combined with mass data for each processor (and assuming replacement at the processor level), these CDFs were used to generate a tradespace of resupply architectures based on the number of spares for each processor brought at each 2-year resupply interval. Each architecture was characterized by the total mass of spares to be resupplied as well as the probability that the supplied spares are sufficient to repair processor failures that occur between resupply missions. Using this tradespace, the minimum mass resupply architectures were identified for three threshold probabilities and the sensitivity of this minimum mass to changes in processor reliability was examined. This chapter also noted that the level at which spares are implemented (i.e. processor, subassembly, component) has a strong impact on the resupply mass, and that the result of this trade study likely overestimates the minimum mass.

Chapter 5 discussed the SMP methodology, noting that it provides a powerful framework for a quantitative analysis and design tool that can be implemented in trade studies and multiobjective optimization. The assumptions and limitations of the analysis were discussed, and the SMP framework was compared to the risk assessment methods identified in Chapter 1.
6.2 Future Work

The framework described in this thesis provides several avenues for continued development, which are described here.

6.2.1 Validation Against Monte Carlo Simulation

The case studies presented in this paper generated reasonable results that matched intuitive prediction for these simple systems. However, a more rigorous validation should be conducted by comparing the results of an SMP analysis for a complex system with a Monte Carlo simulation of that system (or real-world data, if available). This validation should include sensitivity analysis with respect to SMP modeling parameters, such as the maximum number of concurrent failures allowed in the state network.

6.2.2 Incorporation of Non-Exponential Failure Distributions

One of the assumptions used for the development of SMP models is that all failure events are described by exponential PDFs. This assumption ensures that all failure transitions are memoryless, thus enabling the use of loop structures in the state network and the calculation of Markov renewal probabilities in order to develop probabilistic predictions of the number of spares required for each system element. However, as discussed in Chapter 5, exponential distributions do not always provide the most accurate or rich description of failure events. The development of extensions and adaptations of the SMP method presented in Chapter 2 that enable the use of non-exponential failure distributions would greatly expand the applicability of the analysis, allowing it to include the effects of burn-in and wear out, as well as scheduled repair.

One possible solution is to eliminate the looping structures from the state network and implement a one-way SMP model of the system. Instead of transitioning back to a nominal state, repair events would simply transition to a new state. The state network generation process described in Chapter 2 would then be repeated until an
end state is reached. In this formulation, the number of spares required for each component would have to be input as a design variable (rather than determined probabilistically, as was done in this thesis), since it would define the number of failure/repair events that could occur for a given component before a failure of that component is unrecoverable. Given a system description that includes failure/repair distributions and the number of spares supplied for each component as well as buffer depletion distributions for each failure mode, a one-way SMP could be generated that shows all possible system states as the system gradually transitions to failure, consuming spares along the way.

A one-way SMP representation for a system would be more complex than the looping SMPs presented in this thesis, and would require a more intensive state network generation and transition PDF process. The impact of additional spares could still be investigated, but it would require the addition of states to the network and recalculation of transition PDFs to examine each sparing option (as opposed to the CDFs used in this thesis, which are developed based on a static SMP structure). As a result, while analyses of the type shown in Chapters 3 and 4 are possible with non-exponential failure distributions, they may be very time-consuming. Automation of the SMP generation process could greatly speed up this process, however (see section 6.2.5).

6.2.3 Incorporation of Dependent PDFs

Another assumption behind the analysis is the independence of the basic event PDFs. This assumption is made in order to enable the use of a set of basic events and the truncated difference distribution to create all of the transition distributions in the SMP without needing to refer to conditional distributions. However, conditional distributions resulting from a dependence between two events could be utilized, albeit with a more complex SMP generation process. Effectively, the required conditional distributions for each element could be added to the list of basic events and used to generate additional distributions as necessary. Then, during state network generation, these conditional distributions can be used where they are needed. However, this
inclusion of conditional PDFs could result in a very complex system; the complexity of the model must be balanced against the number of factors that are included in it.

6.2.4 Interfacing with Object-Process Methodology

The SMP models used in this analysis present a view of the system with a focus on its reliability and resilience characteristics. This environment provides a good framework for the examination and optimization of system resilience, but can be difficult to interpret in the context of other system design considerations. In order to facilitate the application of this SMP method in collaboration with other system architecting and design tools, a Model-Based Systems Engineering (MBSE) approach - with system data captured in a centralized model to be utilized by different analyses and optimizations - may be particularly useful, for example by interfacing with Object-Process Methodology (OPM).

OPM is a graphical system representation language with a corresponding verbal syntax that enables the capture of system design data in Object-Process Diagrams (OPDs) using the terminology shown in figure 6-1 [73]. Descended from UML and related to SysML, OPM provides a powerful framework for system representation, enabling examination at various levels of abstraction [74]. The ability to interface the resilience analysis method described in this thesis with system design data in the OPD (and vice versa) would provide a powerful tool for the examination and optimization in many different contexts. A potential interface would involve both the ability to automatically generate SMP models based on OPD representations (discussed further in section 6.2.5) and the ability to examine each state in the SMP model as an OPD showing the status of each system element. This connection would facilitate the inclusion of resilience analysis in more general tradespace exploration and optimization studies, as well as assisting in the interpretation of the different states in the SMP network.
6.2.5 Automated SMP Generation

At this point, the most time-consuming portion of the analysis is the generation of the state network outline from the set of basic events. However, the algorithm by which this outline is generated is repetitive and clearly defined; implementation of an automatic process for generation of the state network and resulting list of transition PDFs could reduce the amount of time required for analysis setup and potentially expand the complexity of systems that can feasibly be examined. In addition, whereas the size of the state network is currently constrained by specifying the maximum number of concurrent failures that are allowed, an automated generation process could implement more complex constraints or operate unconstrained. For example, a threshold probability of transition could be specified rather than a number of concurrent failures. Then, as the automator generates the state network it examines each transition as it is created, in the context of all other transitions out of the state. If the probability of a certain transition is below a threshold value that transition could be neglected. Thus, the state network would expand organically to as many levels as are required before a nearly inevitable transition to a failed state or back to nominal occurs.

An automatic SMP generation algorithm based on system design data (such as the OPDs described in section 6.2.4) would allow for much more flexible implementation.
of SMP modeling to determine system resilience characteristics, since it would allow
the structure of the state network itself to be a function of system design variables.
For example, this would allow the consideration of dissimilar redundancy (via add-
dition of different components with different characteristics to accomplish the same
function), as well as system reconfigurations to change the dependency structure be-
tween processors. In addition, as described in section 6.2.2, automation of the SMP
generation process could be used to speed the examination of a variety of one-way
SMP models of a system to examine different sparing strategies. This would facili-
tate large-scale tradespace analysis of systems that incorporate scheduled repair or
components with non-exponential failure distributions.

6.2.6 Lifecycle Cost Analysis

The examples used in this thesis have utilized mass as a surrogate cost metric. This
provides a method to perform a trade study for fixed technology, where MTBF values
are known. However, if more variables are entered into the analysis, a more suitable
metric may be the overall (financial) lifecycle cost of a system. This would include the
cost of component development (a function of MTBF) and manufacturing (both of
the original and any spares, incorporating learning effects), as well as transportation
costs associated with the mass of processors and buffers, both for the original launch
and any resupply operations during the mission timeline. In addition, an opportunity
cost metric could be applied to resources such as crew time or storage volume that
are consumed by maintenance operations and spare parts. Formulation of the system
design problem in this manner would allow the determination of the minimum cost
ECLSS architecture that meets a specified resilience threshold, as well as sensitivity
analysis with respect to the various system design variables in order to determine
areas of potential improvement.
6.2.7 Examination of Level of Repair Implementation

The examination of the Mars One LSU in Chapter 4 assumed that repairs occurred via replacement at the processor level - when a processor failed, it was removed and replaced with an identical spare. As noted in that chapter, this may not be the most mass-efficient method. A lower level of sparing - for example, replacement of a failed subassembly or component rather than the entire processor - could significantly reduce the overall mass of spares required for a system. However, this reduced mass does come at the cost of increased repair time requirements due to the increased complexity of diagnostic and repair operations. As a result, the amount of crew time required for repairs, as well as system downtime during the repairs themselves, would likely increase for lower level sparing [16]. A trade study examining the relationship between these metrics and their sensitivity to changes in parameters could help determine the appropriate level at which to implement sparing for a given mission.

6.2.8 Interfacing with STPA: Reliably Safe Systems

Chapter 5 noted that reliability and safety are two different characteristics of a system, and that the SMP method described in this thesis is meant only to quantify how resilient a system is, not how safe it is. However, the combination of STPA and SMPs may present the opportunity to develop and examine a set of reliably safe systems.

As mentioned in the literature review in Chapter 1, STPA is effectively an algorithm for the development of system architectures that provide the necessary control and feedback pathways to mitigate hazards and maintain safe operation (that is, control of the system within constraints). The hierarchical control structure used in STPA (see figure 1-9) presents an abstracted view of the system as interactions between different controllers and processes via control actions and feedback pathways. In order to translate this abstracted system representation into an actual instantiation of a system, the controllers, processes, control actions, and feedback pathways must be translated into actual system components. While the control algorithm may
take the form of software, which is not amenable to probabilistic examination, the control action and feedback pathways can be thought of as actuators and sensors. These can be hardware elements that have a specific reliability and cost associated with them, and designers will have to select between a set of options for each position in the system architecture using quantitative system trades examining the costs and benefits of each potential system.

In short, STPA can be used to create a system architecture that is “safe” in that its structure mitigates possible hazards. However, the safety of this structure is only real if the elements within the structure can be relied on to function as designed. This process of populating a system architecture with design selections is effectively the same as the processes carried out in Chapters 3 and 4. Therefore, the implementation of SMP analysis and optimization on a system architecture generated with STPA has the potential to generate system design options that are reliably safe.

6.2.9 Comprehensive Comparison to Other Methods

This thesis discussed the relationship between the SMP method described here and the risk analysis methods found in the literature review in a qualitative way. A more comprehensive comparison - perhaps involving the application of each method to the same (relatively complex) system - could be undertaken to characterize the strengths, weaknesses, and limitations of each analysis method. A Monte Carlo simulation of the system and/or empirical data from system operation or testing could be used for comparison and validation. Some characteristics to gather for each method could include:

- Set of system metrics that can be calculated (e.g. state probabilities, renewal probabilities, etc.)
- Objectivity (or repeatability)
- Time required for analysis
- Perceived difficulty/complexity of analysis
- Accuracy (as compared to Monte Carlo or real-world data)
This analysis would enable benchmarking of the different methods and help inform the selection of the appropriate method to apply in a particular context.

### 6.2.10 Development of Human Physiological Failure Model

The analyses used in this thesis represented the crew as a processor, taking in consumables such as $O_2$ and outputting $CO_2$. However, as mentioned in Chapter 2 a higher-fidelity analysis could model the crew as a set of processors and buffers representing the different metabolic processes within the human body. For example, a processor could take in water from an external source and store it in a tank, while another processor takes water from that tank and outputs it as urine or sweat. A set of system failure modes could then be defined based on the buffers within the human. For example, depletion of the water buffer could signify crew death due to thirst, or overfilling of a $CO_2$ buffer could represent death due to $CO_2$ poisoning.

### 6.2.11 Application to Terrestrial Infrastructure Systems

This thesis has described the analysis process in the context of ECLSS for long-duration human spaceflight. However, the methodology is also applicable to terrestrial problems of resource distribution. For example, regional or national water, power, and gas infrastructure system resilience could be characterized. In addition, the results of an SMP analysis could be used to inform maintenance operational strategies, spares deployment, and contingency storage implementation during infrastructure expansion.

### 6.3 Conclusion

The ECLSS that support humans as they travel beyond the Moon and out into the solar system will face greater challenges than any current or previous system has faced. Mission durations will be longer, and the stakes will be higher. In the event of system failure, the crew might find themselves months away from resupply or return to Earth.
In these conditions, the demand for reliability and resilience in the ECLSS system requires the consideration of partially failed states and repairable systems during the design process - as well as all of the logistical demands that entails. System resilience will become a strong design driver, and will need to be traded against other system characteristics such as mass and cost.

Inclusion of resilience in system trades requires a quantitative, analytical method for quantifying resilience metrics as a function of system design variables. This thesis has presented the use of SMP models to several resilience-related metrics as a function of the system structure, buffer capacity, and reliability characteristics of the constituent components. This methodology enables the calculation of the overall probability of system failure, as well as the determination of logistical demands such as the required number of spares.

By providing these and other metrics, the techniques developed and described in this thesis facilitate the quantitative consideration of resilience directly alongside other system metrics. This capability was demonstrated in a case study examination of the Mars One surface habitat system logistics demands. Potential follow-on work to increase the fidelity and applicability of the analysis are suggested, and continued development of SMP modeling techniques for ECLSS and other long-duration systems has the potential to help in the design of the more effective, efficient, sustainable, and resilient systems required for the future.
Appendix A

Numerical Laplace and Inverse Laplace Transforms

This appendix presents the mathematical basis for numerical calculation of Laplace Transforms (LT) and Inverse Laplace Transforms (ILT); utilization of numerical methods such as these eliminates the need for closed-form representations of the function LTs and ILTs during the solution of semi-Markov models.

A.1 Laplace Transform: Using Euler’s Formula

The LT of a function $f(t)$ is defined by the complex integral

$$\tilde{f}(s) = \int_0^\infty e^{-s\alpha} f(\alpha) d\alpha$$  \hspace{1cm} (A.1)

Warr and Collins recommend the use of Euler’s formula to calculate this numerically, thereby avoiding complex integration [13]. After decomposing the complex variable into real and imaginary parts ($s = x + iy$), Euler’s formula can be applied to obtain

$$e^{(x+iy)\alpha} = e^{x\alpha}e^{i\alpha}$$  \hspace{1cm} (A.2)

$$= e^{x\alpha} (\cos(y\alpha) + i \sin(y\alpha))$$  \hspace{1cm} (A.3)
and therefore

\[ \tilde{f}(s) = \int_{0}^{\infty} e^{-z\alpha} \cos(y\alpha) f(\alpha) d\alpha - i \int_{0}^{\infty} e^{-z\alpha} \sin(y\alpha) f(\alpha) d\alpha \] (A.4)

Any numerical integration algorithm may then be applied to determine the real and imaginary parts of the Laplace transform [13].

### A.2 Inverse Laplace Transform: EULER Method

The EULER method is a numerical method for inversion of LTs of probability density functions, described in detail in Abate and Whitt [48]. This method provides a stable inversion algorithm for non-negative functions with values between 0 and 1 (criteria met by PDFs and CDFs), given that the function is sufficiently smooth (i.e. has at least two continuous derivatives) [13, 48]. The time domain function \( f(t) \) is obtained from the Laplace domain representation \( \tilde{f}(t) \) using equations (A.5) and (A.6).

\[ f(t) = \sum_{j=0}^{m} \left( \frac{m!}{k!(m-j)!} \right) 2^{-m} g_{n+j}(t) \] (A.5)

\[ g_{b}(t) = \frac{e^{A/2}}{2t} \text{Re} \left[ \tilde{f} \left( \frac{A}{2t} \right) \right] + \frac{e^{A/2}}{t} \sum_{k=1}^{b} (-1)^k \text{Re} \left[ \tilde{f} \left( \frac{A + 2k\pi i}{2t} \right) \right] \] (A.6)

The parameters \( m, n, \) and \( A \) control the accuracy of the inversion. Abate and Whitt recommend values of 11 and 15 for \( m \) and \( n \), respectively [48]. \( A \) is related to the discretization error in the inversion and can be calculated based on the maximum desired discretization error, using equation (A.7) to determine the value of \( A \) required for a discretization error less than \( 10^{-\gamma} \):

\[ A = \gamma \ln(10) \] (A.7)

For this thesis, the recommended value of 18.4 is used, resulting in a discretization error less than \( 10^{-8} \) [48].

---

1These equations are a combination of equations (13), (14), and (15) in Abate and Whitt [48].
## Appendix B

### Mars One State Network Outline

Table B.1: Mars One state network outline and adjacency data.

<table>
<thead>
<tr>
<th>State Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SPWE failure</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2 SPWE repair (return to 1)</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3 SPWE repair</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4 4BMS repair (return to 1)</td>
<td>4</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4BMS repair</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5 SPWE repair (return to 1)</td>
<td>5</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>6 VCD failure</td>
<td>2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7 SPWE repair</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>8 VCD repair (return to 1)</td>
<td>7</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>8 SPWE repair (return to 1)</td>
<td>8</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>9 CCAA failure</td>
<td>2</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10 SPWE repair</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10 CCAA repair (return to 1)</td>
<td>10</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>11 CCAA repair</td>
<td>9</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>11 SPWE repair (return to 1)</td>
<td>11</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>12 MF failure</td>
<td>2</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>13 SPWE repair</td>
<td>12</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>13 MF repair (return to 1)</td>
<td>13</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>State Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF repair</td>
<td>12</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>1</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>4BMS repair (return to 1)</td>
<td>16</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>4BMS repair</td>
<td>16</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>SPWE failure</td>
<td>15</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>20</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>VCD repair</td>
<td>15</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>22</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>MF failure</td>
<td>15</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>26</td>
<td>11</td>
</tr>
<tr>
<td>26</td>
<td>26</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Sabatier repair (return to 1)</td>
<td>28</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>SPWE failure</td>
<td>28</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>29</td>
<td>30</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>Sabatier repair</td>
<td>29</td>
<td>31</td>
<td>12</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>28</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>33</td>
<td>7</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>State</th>
<th>Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Sabatier repair (return to 1)</td>
<td>33</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Sabatier repair</td>
<td>32</td>
<td>34</td>
<td>8</td>
</tr>
<tr>
<td>34</td>
<td>4BMS repair (return to 1)</td>
<td>34</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>VCD failure</td>
<td>28</td>
<td>35</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>35</td>
<td>Sabatier repair</td>
<td>35</td>
<td>36</td>
<td>9</td>
</tr>
<tr>
<td>36</td>
<td>VCD repair (return to 1)</td>
<td>36</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>VCD repair</td>
<td>35</td>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>37</td>
<td>Sabatier repair (return to 1)</td>
<td>37</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>CCAA failure</td>
<td>28</td>
<td>38</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>38</td>
<td>Sabatier repair</td>
<td>38</td>
<td>39</td>
<td>10</td>
</tr>
<tr>
<td>39</td>
<td>CCAA repair (return to 1)</td>
<td>39</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>CCAA repair</td>
<td>38</td>
<td>40</td>
<td>7</td>
</tr>
<tr>
<td>40</td>
<td>Sabatier repair (return to 1)</td>
<td>40</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>MF failure</td>
<td>28</td>
<td>41</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>41</td>
<td>Sabatier repair</td>
<td>41</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>42</td>
<td>MF repair (return to 1)</td>
<td>42</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>MF repair</td>
<td>41</td>
<td>43</td>
<td>7</td>
</tr>
<tr>
<td>43</td>
<td>Sabatier repair (return to 1)</td>
<td>43</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>VCD failure</td>
<td>1</td>
<td>44</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>44</td>
<td>VCD repair (return to 1)</td>
<td>44</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>SPWE failure</td>
<td>44</td>
<td>45</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>SPWE repair</td>
<td>45</td>
<td>46</td>
<td>7</td>
</tr>
<tr>
<td>46</td>
<td>VCD repair (return to 1)</td>
<td>46</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>VCD repair</td>
<td>45</td>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td>47</td>
<td>SPWE repair (return to 1)</td>
<td>47</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>44</td>
<td>48</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>4BMS repair</td>
<td>48</td>
<td>49</td>
<td>7</td>
</tr>
<tr>
<td>49</td>
<td>VCD repair (return to 1)</td>
<td>49</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>VCD repair</td>
<td>48</td>
<td>50</td>
<td>8</td>
</tr>
<tr>
<td>50</td>
<td>4BMS repair (return to 1)</td>
<td>50</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Sabatier failure</td>
<td>44</td>
<td>51</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>51</td>
<td>Sabatier repair</td>
<td>51</td>
<td>52</td>
<td>7</td>
</tr>
<tr>
<td>52</td>
<td>VCD repair (return to 1)</td>
<td>52</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>State Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCD repair</td>
<td>51</td>
<td>53</td>
<td>13</td>
</tr>
<tr>
<td>53 Sabatier repair (return to 1)</td>
<td>53</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>CCAA failure</td>
<td>44</td>
<td>54</td>
<td>5</td>
</tr>
<tr>
<td>54 VCD repair</td>
<td>54</td>
<td>55</td>
<td>10</td>
</tr>
<tr>
<td>55 CCAA repair (return to 1)</td>
<td>55</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>54</td>
<td>56</td>
<td>7</td>
</tr>
<tr>
<td>56 VCD repair (return to 1)</td>
<td>56</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>MF failure</td>
<td>44</td>
<td>57</td>
<td>6</td>
</tr>
<tr>
<td>57 VCD repair</td>
<td>57</td>
<td>58</td>
<td>11</td>
</tr>
<tr>
<td>58 MF repair (return to 1)</td>
<td>58</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>MF repair</td>
<td>57</td>
<td>59</td>
<td>7</td>
</tr>
<tr>
<td>59 VCD repair (return to 1)</td>
<td>59</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>CCAA failure</td>
<td>1</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td>60 CCAA repair (return to 1)</td>
<td>60</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>SPWE failure</td>
<td>60</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>61 SPWE repair</td>
<td>61</td>
<td>62</td>
<td>7</td>
</tr>
<tr>
<td>62 CCAA repair (return to 1)</td>
<td>62</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>61</td>
<td>63</td>
<td>12</td>
</tr>
<tr>
<td>63 SPWE repair (return to 1)</td>
<td>63</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4BMS failure</td>
<td>60</td>
<td>64</td>
<td>2</td>
</tr>
<tr>
<td>64 4BMS repair</td>
<td>64</td>
<td>65</td>
<td>7</td>
</tr>
<tr>
<td>65 CCAA repair (return to 1)</td>
<td>65</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>64</td>
<td>66</td>
<td>8</td>
</tr>
<tr>
<td>66 4BMS repair (return to 1)</td>
<td>66</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Sabatier failure</td>
<td>60</td>
<td>67</td>
<td>3</td>
</tr>
<tr>
<td>67 Sabatier repair</td>
<td>67</td>
<td>68</td>
<td>7</td>
</tr>
<tr>
<td>68 CCAA repair (return to 1)</td>
<td>68</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>67</td>
<td>69</td>
<td>13</td>
</tr>
<tr>
<td>69 Sabatier repair (return to 1)</td>
<td>69</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>VCD failure</td>
<td>60</td>
<td>70</td>
<td>4</td>
</tr>
<tr>
<td>70 VCD repair</td>
<td>70</td>
<td>71</td>
<td>7</td>
</tr>
<tr>
<td>71 CCAA repair (return to 1)</td>
<td>71</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>CCAA repair</td>
<td>70</td>
<td>72</td>
<td>9</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>State Name</th>
<th>Network Outline</th>
<th>Start</th>
<th>End</th>
<th>Transition ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>VCD repair (return to 1)</td>
<td>72</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>MF failure</td>
<td></td>
<td>60</td>
<td>73</td>
<td>6</td>
</tr>
<tr>
<td>73</td>
<td>CCAA repair</td>
<td>73</td>
<td>74</td>
<td>11</td>
</tr>
<tr>
<td>74</td>
<td>MF repair (return to 1)</td>
<td>74</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>73</td>
<td>75</td>
<td>7</td>
</tr>
<tr>
<td>75</td>
<td>CCAA repair (return to 1)</td>
<td>75</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>MF failure</td>
<td></td>
<td>1</td>
<td>76</td>
<td>6</td>
</tr>
<tr>
<td>76</td>
<td>MF repair (return to 1)</td>
<td>76</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>SPWE failure</td>
<td></td>
<td>76</td>
<td>77</td>
<td>1</td>
</tr>
<tr>
<td>77</td>
<td>SPWE repair</td>
<td>77</td>
<td>78</td>
<td>7</td>
</tr>
<tr>
<td>78</td>
<td>MF repair (return to 1)</td>
<td>78</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>77</td>
<td>79</td>
<td>12</td>
</tr>
<tr>
<td>79</td>
<td>SPWE repair (return to 1)</td>
<td>79</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>4BMS failure</td>
<td></td>
<td>76</td>
<td>80</td>
<td>2</td>
</tr>
<tr>
<td>80</td>
<td>4BMS repair</td>
<td>80</td>
<td>81</td>
<td>7</td>
</tr>
<tr>
<td>81</td>
<td>MF repair (return to 1)</td>
<td>81</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>80</td>
<td>82</td>
<td>8</td>
</tr>
<tr>
<td>82</td>
<td>4BMS repair (return to 1)</td>
<td>82</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Sabatier failure</td>
<td></td>
<td>76</td>
<td>83</td>
<td>3</td>
</tr>
<tr>
<td>83</td>
<td>Sabatier repair</td>
<td>83</td>
<td>84</td>
<td>7</td>
</tr>
<tr>
<td>84</td>
<td>MF repair (return to 1)</td>
<td>84</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>83</td>
<td>85</td>
<td>13</td>
</tr>
<tr>
<td>85</td>
<td>Sabatier repair (return to 1)</td>
<td>85</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>VCD failure</td>
<td></td>
<td>76</td>
<td>86</td>
<td>4</td>
</tr>
<tr>
<td>86</td>
<td>VCD repair</td>
<td>86</td>
<td>87</td>
<td>7</td>
</tr>
<tr>
<td>87</td>
<td>MF repair (return to 1)</td>
<td>87</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>86</td>
<td>88</td>
<td>9</td>
</tr>
<tr>
<td>88</td>
<td>VCD repair (return to 1)</td>
<td>88</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>CCAA failure</td>
<td></td>
<td>76</td>
<td>89</td>
<td>5</td>
</tr>
<tr>
<td>89</td>
<td>CCAA repair</td>
<td>89</td>
<td>90</td>
<td>7</td>
</tr>
<tr>
<td>90</td>
<td>MF repair (return to 1)</td>
<td>90</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>MF repair</td>
<td></td>
<td>89</td>
<td>91</td>
<td>10</td>
</tr>
<tr>
<td>91</td>
<td>CCAA repair (return to 1)</td>
<td>91</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>
Appendix C

MATLAB Code

This appendix presents the MATLAB code modules used for the calculations in this thesis. Code modules are presented first, followed by calculation scripts for the demonstration and the Mars One case study.

C.1 SMP Code Modules

```matlab
% difference_distribution.m
% Creator: Andrew Owens Last updated: 2014-08-19
% Inputs:
%   X - vector of sample points of the pdf f_x(x). The first entry
%       corresponds to x = 0, and the rest are evenly spaced with spacing
%       dt. The range of x is from 0 until a negligible tail can be
%       truncated.
%   Y - vector of sample points of the pdf f_y(y). The first entry
%       corresponds to y = 0, and the rest are evenly spaced with spacing
%       dt. The range of y is from 0 until a negligible tail can be
%       truncated.
%   dt - spacing of sample points.
%   cutoff - cutoff point for negligible tails. Values lower than this will
%            be trimmed out of the final distribution.
% Outputs:
%   Z - vector of sample points of the pdf f_z(z), which is the
%       distribution of the difference X - Y when z>=0. The first entry
%       corresponds to z = 0, and the rest are evenly spaced with a spacing
%       of dt. The range of z is from 0 until a negligible tail can be
%       truncated.
```

135
function Z = differenceDistribution(X,Y,dt,cutoff)

% pad the end of the shorter vector with zeros to make them the same length
% if they're the same length don't do anything
if length(X) < length(Y)
    X = [X, zeros(1,length(Y)-length(X))];
elseif length(Y) < length(X)
    Y = [Y, zeros(1,length(X)-length(Y))];
end

% pad the front of each one so that it is centered on 0, and flip Y to
% create Y_
padX = [zeros(1,length(X)-1), X];
npadY = fliplr([zeros(1,length(Y)-1), Y]);

% convolve the padded X distribution with Y_ to obtain the difference
% distribution D = X - Y
fullZ = dt*conv(padX,npadY,'same');

% find the index for z = 0
zero_index = (length(fullZ)+1)/2;

% find the last index before the cutoff tail

cutoff_index = find(fullZ>=cutoff,1,'last');

% if the cutoff is before 0, then effectively the distribution is 0.
% Returning that doesn't work for the rest of the calculations, so model it
% as an impulse at dt (the first index after zero). Effectively, in the
% remote chance that the previous transition occurs, the next one will
% occur immediately
if cutoff_index <= zero_index
    Z = [0 1];
else
    % truncate the distribution at 0 and take the upper half. Scale this
    % appropriately (ie divide by 1-cdf up to 0)
    Zlong = fullZ(zero_index:end)./(1-dt*sum(fullZ(1:zero_index-1)));

    % trim off negligible tail (beyond 1e-10)
    Z = Zlong(1:find(Zlong>=cutoff,1,'last'));
end

% EULERmachine.m
% Creator: Andrew Owens Last Updated: 2014-08-19
% Inputs:
% EULERparams - vector containing values for m, n, and a to use in
% this numerical ILT application. Recommended values are
% [11; 15; 18.4].
% LDvals - cell vector containing Laplace domain data for the function
% at the points required. Each cell contains the scalar, vector,
% or matrix of the data at that point in the Laplace domain.
% resultTime - time at which result is desired
% Outputs:
% output - the result in the time domain, given the values in LDvals
function output = EULERmachine(EULERparams, LDvals, resultTime)
% unpack the EULER parameters
m = EULERparams(1);
n = EULERparams(2);
a = EULERparams(3);
% outer summation level
output = zeros(size(LDvals{1})); % preallocate output to 0
for k = 0:m
    % inner summation
    innerSum = zeros(size(LDvals{1}));
    for j = 1:n+k
        innerSum = innerSum + (-1)^j * real(LDvals{j+1});
    end
    output = output + nchoosek(m,k)*2^(-m)*...
             ((exp(a/2)/(2*resultTime))*real(LDvals{1}) + ...
             (exp(a/2)/resultTime)*innerSum);
end
end

% getLaplacePoints.m
% Creator: Andrew Owens Last updated: 2014-08-19
% Inputs:
% resultTime - time at which final results are desired; used to determine
% which points in the Laplace domain are needed for EULER
% EULERparams - vector containing values for m, n, and a to use in
% this numerical ILT application.
% Outputs:
% sVals - matrix containing the complex values at which LT data are
% desired, in the form s(j) = sVals(j,1) + i*sVals(j,2)
function sVals = getLaplacePoints(resultTime, EULERparams)
% unpack the EULER parameters
m = EULERparams(1);
n = EULERparams(2);
a = EULERparams(3);
u = a/(2*resultTime);
v = zeros(m+n+1,1);
for j = 0:m
    v(j+1) = j*pi/resultTime;
end
sVals = [u.*ones(size(v)), v];
% getLT.m
% Creator: Andrew Owens Last updated: 2014-08-19
% Inputs:
% f - cell array of vectors encoding functions to be LT'd
% r,c - vectors indicating the location of entries in the f cell array
% sVals - matrix containing the complex values at which LT data are
% desired, in the form s(j) = sVals(j,1) + i*sVals(j,2)
% dt - discretization of the time vector
% Outputs:
% Lf - cell array containing the Laplace transforms of f at the points
% indicated by sVals. Each entry in the cell array corresponds to a
% row of sVals.
function Lf = getLT(f,r,c,sVals,dt)
% preallocate cell array to store values at each sVal
Lf = cell(size(sVals,1),1);
for j = 1:size(sVals,1) % cycle through each point in sVals
    % extract the sVal
    u = sVals(j,1);
    v = sVals(j,2);
    LV = zeros(size(r)); % preallocate a vector of values
    for k = 1:length(r) % cycle through each entry in f
        % extract the vector at this location in f, which is the function
        % sampled at the points 0:dt:dt*(length(funVec)-1)
        funVec = f{r(k),c(k)};
        % generate the vector of alpha values corresponding to the length
        % of this vector
        alpha = 0:dt:dt*(length(funVec)-1);
        % use the alpha vector, u, and v to generate the vectors
        % representing the integrand for the real and imaginary
        % coefficients
        realFunVec = exp(-u.*alpha).*cos(v.*alpha).*funVec;
        imagFunVec = exp(-u.*alpha).*sin(v.*alpha).*funVec;
        % numerically integrate with trapz
        realCoeff = trapz(alpha,realFunVec);
        imagCoeff = trapz(alpha,imagFunVec);
        % store the value in the LV vector
        LV(k) = realCoeff - 1i*imagCoeff;
    end
    % create a sparse matrix using r,c,LV and store it in the
    % appropriate cell in Lf. Use the size of Lf to pad out the matrix
    Lf{j} = sparse(r,c,LV,size(f,1),size(f,2));
end

% getLV.m
% getLV.m
% Creator: Andrew Owens Last updated: 2014-08-19
% Inputs:
% LQ - cell array of values of the Laplace transform of the kernel matrix
% at the values required for EULER inversion at the given time.
% Entry LQ{j} corresponds to the value of the Laplace transform at
% the point indicated by sVals(j,:)
% LH - cell array of values of the Laplace transform of the unconditional
% waiting time density matrix at the values required for EULER
% inversion.
% sVals - matrix encoding the complex values s = x + i*y at which the
% Laplace transform must be calculated for EULER inversion at the
% given time. The first column contains x values, the second
% contains y values.
% startState - state the system is in at time t = 0
% EULERparams - vector of parameters for the EULER numerical ILT
% algorithm. Entries are [m; n; a]
% resultTime - time at which probabilities are to be calculated
% Outputs:
function P = getP(LQ, LH, sVals, startState, EULERparams, resultTime)
LP = cell(size(LQ)); % preallocate LP
ident = speye(size(LQ{1})); % create a sparse identity matrix
% calculate the Laplace transform LP
for j = 1:length(LP)
    u = sVals(j,1);
    v = sVals(j,2);
    thisLP = (1/(u+1i*v))*inv(ident-LQ{j})*(ident-LH{j});
    LP{j} = thisLP(startState,:);
end
% use EULER to ILT back to the time domain
P = EULERmachine(EULERparams,LP,resultTime);

function VCDF = getVCDF(LQ, sVals, startState, EULERparams, resultTime,...
    renewalThreshold, renewalStates)
% preset the trigger p, which will be the minimum of the renewal
% probabilities of the states to watch
p = 0;
nSpares = 0; % set initial number of spares to 0
VCDF = []; % preallocate VCDF matrix
% Calculate renewal probabilities in a while loop until the minimum
% probability is above the threshold or 100 spares are used
while p <= renewThreshold && nSpares < 101
    LV = getLV(LQ,startState,nSpares,sVals,renewalStates); % calculate LV
    V = EULERmachine(EULERparams,LV,resultTime); % find probabilities
    VCDF = [VCDF; nSpares, V];% store probabilities and number of spares
    nSpares = nSpares + 1; % update the number of spares
    p = min(V); % check the trigger
end

% getLT.m
% Creator: Andrew Owens Last updated: 2014-08-19
% Inputs:
% f - cell array of vectors encoding functions to be LT'd
% r,c - vectors indicating the location of entries in the f cell array
% sVals - matrix containing the complex values at which LT data are
%     desired, in the form s(j) = sVals(j,1) + i*sVals(j,2)
% dt - discretization of the time vector
% Outputs:
% Lf - cell array containing the Laplace transforms of f at the points
%     indicated by sVals. Each entry in the cell array corresponds to a
%     row of sVals.
function Lf = getLT(f,r,c,sVals,dt)
    Lf = cell(size(sVals,1),1);
    for j = 1:size(sVals,1) % cycle through each point in sVals
        u = sVals(j,1);
        v = sVals(j,2);
        LV = zeros(size(r)); % preallocate a vector of values
        for k = 1:length(r) % cycle through each entry in f
            funVec = f{r(k),c(k)};
            alpha = 0:dt:*((length(funVec)-1));
            realFunVec = exp(-u.*alpha).*cos(v.*alpha).*funVec;
        end
    end

imagFunVec = exp(-u.*alpha).*sin(v.*alpha).*funVec;
% numerically integrate with trapz
realCoeff = trapz(alpha,realFunVec);
imagCoeff = trapz(alpha,imagFunVec);
% store the value in the LV vector
LV(k) = realCoeff - 1i*imagCoeff;
end
% create a sparse matrix using r,c,LV and store it in the
% appropriate cell in Lf. Use the size of Lf to pad out the matrix
Lf{j} = sparse(r,c,LV,size(f,1),size(f,2));
end

C.2 Demonstration Script

The script used to perform the calculations for the demonstration in Chapter 3 makes use of the subfunction transition_setup.m and the files adjacency.csv and transition_IDs.csv, which contain the adjacency matrix for the system (see table 3.2) and the table indicating combinations of RVs used to generate each transition (see table 3.3). The script demonstration.m sets up and solves the SMP, saving the result to a .mat file. The script demo_processing.m then performs postprocessing and produces the required plots and results.

% demonstration.m
% Creator: Andrew Owens Last updated: 2014-08-20
% This script performs the required calculations for the demonstration
% described in Chapter 3
addpath ../SMP_modules % add the SMP code modules to the path
adjMat = csvread('adjacency.csv',1,1); % load adjacency matrix
% set solution parameters
resultTime = 365*24; % end of mission [h] - all times based on hours
EULERparams = [11; 15; 18.4]; % parameters for EULER numerical ILT
startState = 1; % state the system is in at time t = 0
renewalStates = 1:size(adjMat,1); % renewal states
renewalThreshold = 0.9999; % threshold up to which to calculate renewals
cap_wt_vec = 0.25:.25:50; % WT capacity range [kg]
dt = 0.05; % timestep size
cutoff = 1e-9; % cutoff point for negligible tail
P_mat = zeros(length(cap_wt_vec),size(adjMat,1)); % preallocate storage
VCDFarray = cell(length(cap_wt_vec),1);
tic
for k = 1:length(cap_wt_vec) % cycle through each WT capacity
cap_wt = cap_wt_vec(k);
transitions = transition_setup(cap_wt,dt,cutoff); % set up transitions
[r,c,vals] = find(adjMat); % decompose the adjacency matrix

nStates = size(adjMat,1); % determine the number of states

[Q,H] = makeKernel(r,c,vals,adjMat,nStates,transitions); % get Q and H

sVals = getLaplacePoints(resultTime,EULERparams); % get Laplace points

LQ = getLT(Q,r,c,sVals,dt); % get LQ

Hindex = unique(r); % get LH and diagonalize

LH = getLT(H,Hindex,ones(size(Hindex)),sVals,dt);

for j = 1:length(LH)
    LH{j} = diag(LH{j});
end

P = getP(LQ,LH,sVals,startState,EULERparams,resultTime); % calculate P

% calculate Markov renewal CDFs VCDFs

VCDF = getVCDF(LQ,sVals,startState,EULERparams,resultTime,...
              renewalThreshold,renewalStates);

P_mat(k,:) = P; % store results

VCDFarray{k} = VCDF;

disp(['Completed ' num2str(cap_wt) ' kg; MET = ' num2str(toc) ' s'])

end

totalElapsedTime = toc;

% save the results into a .mat file, naming it using the date and time

filename = [datestr(now) '.mat'];

save(filename,'cap_wt_vec','P_mat','VCDFarray','totalElapsedTime')

% demo_processing.m
% Creator: Andrew Owens Last Updated: 2014-08-20
% This script loads result data from the demonstration problem and executes
% calculations / produces plots as required.

%% load data
% cap_wt_vec - vector of values of WT capacity [kg] that was examined
% P_mat - matrix of state probabilities. Rows correspond to the values of
%          WT capacity in cap_wt_vec, columns are states.
% totalElapsedTime - time that it took for the full analysis
% VCDFarray - cell array containing values for the Markov renewal
%          probabilities. Each cell corresponds to a WT capacity, and
%          contains a matrix; rows are number of visits (increasing
%          integers starting at 0) until all probabilities are above
%          the threshold used in the analysis. Column 1 is the number
%          of visits, and each subsequent column corresponds to states
load '16-Aug-2014 17:43:20'

%% Buffer sizing
figure
plot(cap_wt_vec,sum(P_mat(:,end-1:end),2),'b','linewidth',2)
title('Effect of WT Capacity on Probability of Failed Repairs',...
      'FontSize',16,'FontWeight','bold')
xlabel('WT Capacity [kg]', 'FontSize',16,...
% Number of Spares
WTcap = 35;
index = find(cap_wt_vec>=WTcap,1,'first'); % find this location index
VCDF = VCDFarray{index}; % pull out the VCDF for this WT capacity

% define required states for renewal calculations
needed = [2 11 4 10 7 14 17];

% pull out relevant vectors into a single matrix
% have to add one because first column is number of spares
spareCDFs = VCDF(:,needed+1);

% generate PMFs via diff. Add a 0 on top so diff captures the first jump
sparePMFs = diff([zeros(1,length(needed)); spareCDFs]);

% convolve appropriately to determine the number of spares required
spareOGS_PMF = conv(sparePMFs(:,1),sparePMFs(:,2));
spareWRS_PMF = conv(sparePMFs(:,3),sparePMFs(:,4));
sparePDU_PMF = conv(conv(sparePMFs(:,5),sparePMFs(:,6)),sparePMFs(:,7));

% use cumsum to get CDFs
spareOGS_CDF = cumsum(spareOGS_PMF);
spareWRS_CDF = cumsum(spareWRS_PMF);
sparePDU_CDF = cumsum(sparePDU_PMF);

% truncate these to a threshold level, 0.9999
thresh = 0.9999;
spareOGS_CDF = spareOGS_CDF(1:find(spareOGS_CDF>=thresh,1,'first'));
spareWRS_CDF = spareWRS_CDF(1:find(spareWRS_CDF>=thresh,1,'first'));
sparePDU_CDF = sparePDU_CDF(1:find(sparePDU_CDF>=thresh,1,'first'));

figure % plot
stairs(0:length(spareOGS_CDF)-1,spareOGS_CDF,'linewidth',2)
title('CDF of Number of OGS Spares Required','FontSize',16,'FontWeight','bold')
xlabel('Number of Spares','FontSize',16,'FontWeight','bold')
ylabel('CDF','FontSize',16,'FontWeight','bold')
set(gca,'xtick',0:length(spareOGS_CDF)-1,'fontsize',12)
axis([0 length(spareOGS_CDF)-1 0 1])
grid on

figure
stairs(0:length(spareWRS_CDF)-1,spareWRS_CDF,'linewidth',2)
title('CDF of Number of WRS Spares Required','FontSize',16,'FontWeight','bold')
xlabel('Number of Spares','FontSize',16,'FontWeight','bold')
ylabel('CDF','FontSize',16,'FontWeight','bold')
set(gca,'xtick',0:length(spareWRS_CDF)-1,'fontsize',12)
axis([0 length(spareWRS_CDF)-1 0 1])
grid on
% Relationship between buffer size and spares
buffSize = cap_wt_vec(1:index); % get buffer size vector
P_failedRepair_vec = zeros(length(buffSize),1); % preallocate storage
P_fail = zeros(length(buffSize),7+1); % add one for 0 spares
for j = 1:length(buffSize) % for each buffer size
    % grab probability of unsuccessful repair
    P_failedRepair_vec(j) = sum(P_mat(j,18:19));
    % determine the CDF for the number of spares, truncating to 7
    cdfs = VCDFarray{j}(:,[4,10]+1); % add 1 b/c first row is num spares
    pmfs = diff([zeros(1,2); cdfs]); %
    spareCDF = cumsum(conv(pmfs(:,1),pmfs(:,2))); %
    spareCDF = spareCDF(1:7+1); % add 1 to account for 0 spares
    % calculate and store vector of overall probability of failure
    P_fail(j,:) = 1 - (1 - P_failedRepair_vec(j)).*spareCDF;
end
figure % plot the results
plot(buffSize,P_fail,'linewidth',2)
title('Combined Effect of Buffer Size and Number of Spares','FontSize',16,...
    'FontWeight','bold')
xlabel('WT Capacity','FontSize',16,'FontWeight','bold')
ylabel('Probability of Failure','FontSize',16,'FontWeight','bold')
legend('0 Spares','1 Spares','2 Spares','3 Spares','4 Spares','5 Spares','6 Spares','7 Spares','location','southwest')
grid on
set(gca,'FontSize',12)
function transitions = transition_setup(cap_wt, dt, cutoff)

%% define basic event pdfs
%% start with parameters. All based on a timescale of hours.
mtbf_ogs = 2160; % OGS mean time between failures
mttr_ogs = 8; % OGS mean time to repair
sdr_ogs = 1; % OGS standard deviation in repair time
mtbf_wrs = 4320; % WRS mean time between failures
mttr_wrs = 12; % WRS mean time to repair
sdr_wrs = 3; % WRS standard deviation in repair time
mtbf_pdu = 69536; % PDU mean time between failures
mttr_pdu = 10; % PDU mean time to repair
sdr_pdu = 3; % PDU standard deviation in repair time
cap_ot = 4.5; % OT capacity [kg]
mdot_o2 = 0.21; % O2 consumption rate [kg h\(^{-1}\)]
mdot_h2o = 0.98; % H2O consumption rate [kg h\(^{-1}\)]
sd_dep = 0.25; % arbitrarily small standard deviation in depletion time

%% calculate distribution parameters
lam_ogs = 1/mtbf_ogs; % OGS failure rate
sig_ogs = sqrt(log(1+sdr_ogs^2/mttr_ogs^2)); % OGS repair shape parameter
mu_ogs = log(mttr_ogs)-(1/2)*sig_ogs^2; % OGS repair log-scale parameter
lam_wrs = 1/mtbf_wrs; % WRS failure rate
sig_wrs = sqrt(log(1+sdr_wrs^2/mttr_wrs^2)); % WRS repair shape parameter
mu_wrs = log(mttr_wrs)-(1/2)*sig_wrs^2; % WRS repair log-scale parameter
lam_pdu = 1/mtbf_pdu; % PDU failure rate
sig_pdu = sqrt(log(1+sdr_pdu^2/mttr_pdu^2)); % PDU repair shape parameter
mu_pdu = log(mttr_pdu)-(1/2)*sig_pdu^2; % PDU repair log-scale parameter
sig_ot = sqrt(log(1+sd_dep^2/(cap_ot/mdot_o2)^2)); % OT depl shape param
mu_ot = log(cap_ot/mdot_o2)-(1/2)*sig_ot^2; % OT depl log-scale param
sig_wt = sqrt(log(1+sd_dep^2/(cap_wt/mdot_h2o)^2)); % WT depl shape param
mu_wt = log(cap_wt/mdot_h2o)-(1/2)*sig_wt^2; % WT depl log-scale param
% Generate vectors for basic event random variables. Ensure that all of
% these are configured to take vector input.
% OGS failure
\[ t = 0:dt:-mtbf_ogs*\log(mtbf_ogs*cutoff); \]
\[ X_ogs = lam_ogs.*\exp(-lam_ogs.*t); \]
\[ X_ogs(isnan(X_ogs)) = 0; \]
\[ X_ogs = X_ogs/(dt*\text{sum}(X_ogs)); \]
% OGS repair
\[ t = 0:dt:mttr_ogs+15*sdr_ogs; \]
\[ R_ogs = (1./(t.*\text{sqrt}(2*pi).*sig_ogs)).*... \]
\[ \exp(-\text{log}(t)-\mu_ogs).^2./(2*\text{sig}_ogs^2)); \]
\[ R_ogs(isnan(R_ogs)) = 0; \]
\[ R_ogs = R_ogs(1:\text{find}(R_ogs>=cutoff,1,'last')); \]
\[ R_ogs = R_ogs/(dt*\text{sum}(R_ogs)); \]
% WRS failure
\[ t = 0:dt:-mtbf_wrs*\log(mtbf_wrs*cutoff); \]
\[ X_wrs = lam_wrs.*\exp(-lam_wrs.*t); \]
\[ X_wrs(isnan(X_wrs)) = 0; \]
\[ X_wrs = X_wrs/(dt*\text{sum}(X_wrs)); \]
% WRS repair
\[ t = 0:dt:mttr_wrs+15*sdr_wrs; \]
\[ R_wrs = (1./(t.*\text{sqrt}(2*pi).*sig_wrs)).*... \]
\[ \exp(-\text{log}(t)-\mu_wrs).^2./(2*\text{sig}_{wrs}^2)); \]
\[ R_wrs(isnan(R_wrs)) = 0; \]
\[ R_wrs = R_wrs(1:\text{find}(R_wrs>=cutoff,1,'last')); \]
\[ R_wrs = R_wrs/(dt*\text{sum}(R_wrs)); \]
% PDU failure
\[ t = 0:dt:-mtbf_pdu*\log(mtbf_pdu*cutoff); \]
\[ X_pdu = lam_pdu.*\exp(-lam_pdu.*t); \]
\[ X_pdu(isnan(X_pdu)) = 0; \]
\[ X_pdu = X_pdu/(dt*\text{sum}(X_pdu)); \]
% PDU repair
\[ t = 0:dt:mttr_pdu+15*sdr_pdu; \]
\[ R_pdu = (1./(t.*\text{sqrt}(2*pi).*sig_pdu)).*... \]
\[ \exp(-\text{log}(t)-\mu_pdu).^2./(2*\text{sig}_{pdu}^2)); \]
\[ R_pdu(isnan(R_pdu)) = 0; \]
\[ R_pdu = R_pdu(1:\text{find}(R_pdu>=cutoff,1,'last')); \]
\[ R_pdu = R_pdu/(dt*\text{sum}(R_pdu)); \]
% OT depletion
\[ t = 0:dt:cap_ot/mdot_o2+15*sd_dep; \]
\[ D_ot = (1./(t.*\text{sqrt}(2*pi).*sig_ot)).*... \]
\[ \exp(-\text{log}(t)-\mu_ot).^2./(2*\text{sig}_{ot}^2)); \]
\[ D_ot(isnan(D_ot)) = 0; \]
\[ D_ot = D_ot(1:\text{find}(D_ot>=cutoff,1,'last')); \]
\[ D_ot = D_ot/(dt*\text{sum}(D_ot)); \]
% WT depletion
\begin{verbatim}
t = 0:dt:cap_wt/mdot_h2o+15*sd_dep;
D_wt = (1./(t.*sqrt(2*pi).*sig_wt)).*... 
    exp(-(log(t)-mu_wt).^2./(2*sig_wt^2));
D_wt(isnan(D_wt)) = 0;
D_wt = D_wt(1:find(D_wt>=cutoff,1,'last'));
D_wt = D_wt./(dt*sum(D_wt));
\end{verbatim}

%% Construct the transition array
% read in the transition IDs from csv
% This is a two column vector - each row corresponds to a transition and 
% the entries in the columns indicate how that transition is created.
% First column is the base pdf, second column (if nonzero) is a pdf that 
% should be subtracted from it (using difference_distribution.m)
transition_IDs = csvread('transition_IDs.csv',1,1);
% preallocate cell array for transitions
transitions = cell(size(transition_IDs,1),1);
% Go ahead and put the basic event pdfs and cdfs in
transitions{1} = [X_ogs; dt.*cumsum(X_ogs)];
transitions{2} = [R_ogs; dt.*cumsum(R_ogs)];
transitions{3} = [X_wrs; dt.*cumsum(X_wrs)];
transitions{4} = [R_wrs; dt.*cumsum(R_wrs)];
transitions{5} = [X_pdu; dt.*cumsum(X_pdu)];
transitions{6} = [R_pdu; dt.*cumsum(R_pdu)];
transitions{7} = [D_ot; dt.*cumsum(D_ot)];
transitions{8} = [D_wt; dt.*cumsum(D_wt)];
% starting after this point, populate the cell array using transition_IDs
for j = 9:size(transition_IDs,1)
    % calculate Z = X - Y
    pdf = differenceDistribution(...
        transitions{transition_IDs(j,1)}(1,:),
        transitions{transition_IDs(j,2)}(1,:),dt,cutoff);
    pdf = pdf./(dt*sum(pdf)); % normalize
    transitions{j} = [pdf; dt.*cumsum(pdf)]; % insert with cdf
end

C.3 Mars One Script

The script used to perform the calculations for the Mars One case study in Chapter 9 
makes use of the subfunction M1_transitionSetup.m and the files M1_adjacency_data.csv 
and M1_transitionIDs.csv, which contain the adjacency matrix for the system (see 
table B.1) and the table indicating combinations of RVs used to generate each tran-
sition (see table 4.3). The script marsone.m sets up and solves the SMP, saving the 
result to a .mat file. The script marsone_MSDO.m then performs postprocessing and 
generates the multiobjective tradespace.
This script performs the calculations required to gather data for the Mars One case study described in Chapter 4.

```matlab
addpath ../SMP_modules % add the SMP code modules to the path
adjData = csvread('M1_adjacency_data.csv'); % load Mars One adjacency data
r = adjData(:,1);
c = adjData(:,2);
vals = adjData(:,3);
adjMat = sparse(r,c,vals);
nStates = size(adjMat,1); % determine number of states in system

% set solution parameters
resultTime = 365*2; % time between resupply [d] - all times based on days
EULERparams = [11; 15; 18.4]; % parameters for EULER numerical ILT
startState = 1; % state the system is in at time t = 0
renewalStates = 1:nStates; % states to watch for renewal probabilities
renewalThreshold = 0.99999; % threshold up to which to calculate renewals

% set up transitions
dt = 0.01; % sample spacing
cutoff = 1e-9; % cutoff probability for tails
transitions = M1_transitionSetup(dt,cutoff);

% get Q and H
sVals = getLaplacePoints(resultTime,EULERparams); % get Laplace points
LQ = getLT(Q,r,c,sVals,dt); % get LQ
Hindex = unique(r); % get LH and diagonalize
LH = getLT(H,Hindex,ones(size(Hindex)),sVals,dt);
for j = 1:length(LH)
    LH{j} = diag(LH{j});

% calculate Markov renewal probabilities
VCDF = getVCDF(LQ,sVals,startState,EULERparams,resultTime,...
    renewalThreshold,renewalStates);

% Postprocessing
Sarray = cell(6,1);
Sarray{1} = [2 16 29 45 61 77]; % SPWE renewal states
Sarray{2} = [3 15 32 48 64 80]; % 4BMS renewal states
Sarray{3} = [28 51 67 83]; % Sabatier renewal states
Sarray{4} = [6 19 35 44 70 86]; % VCD renewal states
Sarray{5} = [9 22 38 54 60 89]; % CCAA renewal states
Sarray{6} = [12 25 41 57 73 76]; % MF renewal states

% generate PMF matrix using diff
VPMF = diff([zeros(1,size(VCDF(:,2:end),2)); VCDF(:,2:end)]);
Spares = cell(6,1); % preallocate results storage
```

```matlab
...
loThreshold = 0.95; % set low end threshold
for j = 1:length(Sarray)
    pmf = VPMF(:,Sarray{j}(1)); % grab the first pmf for this processor
    % cycle through the rest of the renewal states, convolving
    for k = 2:length(Sarray{j})
        pmf = conv(pmf,VPMF(:,Sarray{j}(k)));
    end
    % store the PMF and CDF, cutting off at the renewal threshold
    cdf = cumsum(pmf);
    hicut = find(cdf>=renewalThreshold,1,'first');
    locut = find(cdf>=loThreshold,1,'first');
    Spares{j} = [locut:hicut; pmf(locut:hicut)' ; cdf(locut:hicut)'];
end

%% Plotting
% SPWE
figure
this = 1;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of SPWE Spares Required','fontsize',16,...
    'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on
% 4BMS
figure
this = 2;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of 4BMS Spares Required','fontsize',16,...
    'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on
% Sabatier
figure
this = 3;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of Sabatier Spares Required','fontsize',16,...
    'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on

% VCD
figure
this = 4;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of VCD Spares Required','fontsize',16,...
     'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on

% CCAA
figure
this = 5;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of CCAA Spares Required','fontsize',16,...
     'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on

% MF
figure
this = 6;
stairs(Spares{this}(1,:),Spares{this}(3,:),'linewidth',2)
title('CDF of Number of MF Spares Required','fontsize',16,...
     'fontweight','bold')
xlabel('Number of Spares','fontsize',16,'fontweight','bold')
ylabel('CDF','fontsize',16,'fontweight','bold')
set(gca,'xtick',Spares{this}(1,:),'fontsize',12)
axis([Spares{this}(1,1) Spares{this}(1,end) loThreshold 1])
grid on

% save outputs
save('M1_Spares_baseline.mat','Spares')
% create full factorial of the number of spares for each element, based on
% the length of the entry
options = zeros(1,length(Spares));
for j = 1:length(Spares)
    options(j) = length(Spares{j}(1,:));
end
% each row in archMat represents an architecture; entries in each column
% represent the index of the selection for each processor
archMat = fullfact(options);
% preallocate results storage
m = zeros(size(archMat,1),1); % system mass
r = m; % reliability
nSpares = zeros(size(archMat));
cdf_prob = zeros(size(archMat));
% for each architecture
for j = 1:size(archMat,1)
    nSpares(j,:) = [Spares{1}(1,archMat(j,1)),...
                    Spares{2}(1,archMat(j,2)),...
                    Spares{3}(1,archMat(j,3)),...
                    Spares{4}(1,archMat(j,4)),...
                    Spares{5}(1,archMat(j,5)),...
                    Spares{6}(1,archMat(j,6))];
    cdf_prob(j,:) = [Spares{1}(3,archMat(j,1)),...
                     Spares{2}(3,archMat(j,2)),...
                     Spares{3}(3,archMat(j,3)),...
                     Spares{4}(3,archMat(j,4)),...
                     Spares{5}(3,archMat(j,5)),...
                     Spares{6}(3,archMat(j,6))];
    m(j) = sum(m_vec.*nSpares(j,:)); % calculate the mass
    r(j) = prod(cdf_prob(j,:)); % calculate reliability
end
% find the minimum mass for 0.95, 0.99, and 0.999 reliability
rvals = [0.95; 0.99; 0.999];
% preallocate storage for the mass, reliability, and architecture of
% these points
points = zeros(length(rvals),2+length(options));
for j = 1:length(rvals)
    minmass = min(m(r>=rvals(j)));
    index = find(m==minmass);
    minmassr = max(r(index));
    pointind = find((m==minmass).*r==minmassr);
    points(j,:) = [minmass, minmassr, nSpares(pointind,:)];
end
%% Plotting
figure
% scatter(m(r>=0.9),r(r>=0.9),0.5,'k')
title('Spares Probability vs. Mass of Spares','fontsize',16,...
'fontweight','bold')
xlabel('Mass of Spares [kg]','fontsize',16,'fontweight','bold')
ylabel('Spares Probability [-]','fontsize',16,'fontweight','bold')
set(gca,'fontsize',12)
grid on

hold on

for j = 1:size(points,1)
    plot(points(j,1),points(j,2),colorvec{j},'markersize',25)
end

% save the discovered points and their architecture vectors, as well as the
% rvals, to a csv file
resultpoints = [rvals, points];
csvwrite('M1_resultpoints_baseline.csv',resultpoints)

%% Repeat for 10% MTBF increase

% save old data
m_base = m(r>=0.9);
r_base = r(r>=0.9);
points_base = points;

load M1_Spares_MTBFp10.mat % load data (Spares, a cell array)
m_vec = [113, 201, 18, 128, 96, 476]; % set mass vector [kg per component]

% create full factorial of the number of spares for each element, based on
% the length of the entry
options = zeros(1,length(Spares));
for j = 1:length(Spares)
    options(j) = length(Spares{j}(1,:));
end

% each row in archMat represents an architecture; entries in each column
% represent the index of the selection for each processor
archMat = fullfact(options);

% preallocate results storage
m = zeros(size(archMat,1),1); % system mass
r = m; % reliability
nSpares = zeros(size(archMat));
cdf_prob = zeros(size(archMat));

% for each architecture
for j = 1:size(archMat,1)
    nSpares(j,:) = [Spares{1}(1,archMat(j,1)),
                    Spares{2}(1,archMat(j,2)),
                    Spares{3}(1,archMat(j,3))];
end
Spares(4)(1,archMat(j,4)),...
Spares(5)(1,archMat(j,5)),...
Spares(6)(1,archMat(j,6));
cdf_prob(j,:) = [Spares(1)(3,archMat(j,1)),...
                Spares(2)(3,archMat(j,2)),...
                Spares(3)(3,archMat(j,3)),...
                Spares(4)(3,archMat(j,4)),...
                Spares(5)(3,archMat(j,5)),...
                Spares(6)(3,archMat(j,6))];
m(j) = sum(m_vec.*mSpares(j,:)); % calculate the mass
r(j) = prod(cdf_prob(j,:)); % calculate reliability
end

% find the minimum mass for 0.95, 0.99, and 0.999 reliability
rvals = [0.95; 0.99; 0.999];
% preallocate storage for the mass, reliability, and architecture of
% these points
points = zeros(length(rvals),2+length(options));
for j = 1:length(rvals)
    minmass = min(m(r>=rvals(j)));
    index = find(m==minmass);
    minmassr = max(r(index));
    pointind = find((m==minmass).*(r==minmassr));
    points(j,:) = [minmass, minmassr, nSpares(pointind,:)];
end

%% Plotting
figure
scatter(m_base,r_base,0.5,'k')
hold on
scatter(m(r>=0.9),r(r>=0.9),0.5,.6.*[1,1,1])
title('Spares Probability vs. Mass of Spares - Increased MTBF','fontsize',16,...
      'fontweight','bold')
xlabel('Mass of Spares [kg]','fontsize',16,'fontweight','bold')
ylabel('Spares Probability [-]','fontsize',16,'fontweight','bold')
set(gca,'fontsize',12)
grid on
% plot threshold points
% fadecolor_mat = [.5 .5 1; .5 1 .5; 1 .5 .5];
colorvec = {'b' 'g' 'r'};
for j = 1:size(points,1)
    plot(points_base(j,1),points_base(j,2),'.','color',colorvec{j},'
      'markersize',25)
    plot(points(j,1),points(j,2),'v',
      'MarkerFaceColor',colorvec{j},'MarkerEdgeColor',colorvec{j},
      'markersize',12)
end
% save the discovered points and their architecture vectors, as well as the
% rvals, to a csv file
resultpoints = [rvals, points];
csvwrite('M1_resultpoints_MTBFp10.csv',resultpoints)

% M1_transitionSetup.m
% Creator: Andrew Owens Last updated: 2014-08-20
% This function sets up the transitions for the Mars One case study
% presented in Chapter 4.
function transitions = M1_transitionSetup(dt,cutoff)
% read in transition IDs
transitionIDs = csvread('M1_transition_IDs.csv');
% set mean time to failures (all times in days)
mtbf_vec = [476; 335; 365; 269; 922; 228];
mttr = 1; % mean time to repair
sdr = 0.25; % standard deviation in repair time
% calculate failure and repair distribution parameters
lam_vec = 1./mtbf_vec; % vector of failure rates
sig = sqrt(log(1+sdr^2/mttr^2)); % repair shape parameter
mu = log(mttr)-(1/2)*sig^2; % GGS repair log-scale parameter
% Generate RV vectors and store them in the transitions array
% Preallocate cell array. Each cell contains a 2 x m matrix - the first row
% is the pdf and the second row is the cdf of the transition time, up to
% the negligible tail specified by cutoff
transitions = cell(size(transitionIDs,1),1);
for j = 1:6 % the first 6 entries are the failure distributions
% create a t vector that goes up to the cutoff value
    t = 0:dt:-mtbf_vec(j)*log(mtbf_vec(j)*cutoff);
% use this vector to populate the pdf
    pdf = lam_vec(j).*exp(-lam_vec(j).*t);
    pdf = pdf./(dt*sum(pdf)); % normalize
% generate cdf and store results
    transitions{j} = [pdf; dt.*cumsum(pdf)];
end
% entry 7 is the repair distribution
t = 0:dt:mttr+15*sdr; % go out to 15 standard deviations
pdf = (1./(t.*sqrt(2*pi).*sig)).*exp(-(log(t)-mu).^2./(2*sig^2));
pdf(isnan(pdf)) = 0; % remove any NaNs, replace with 0
pdf = pdf(1:find(pdf>cutoff,1,'last'));
pdf = pdf./(dt*sum(pdf)); % normalize
transitions(7) = [pdf; dt.*cumsum(pdf)]; % generate cdf and store results
% the rest of the transitions are populated using the transitionIDs data
for j = 8:length(transitions)
% calculate Z = X - Y
pdf = differenceDistribution(...
    transitions{transitionIDs(j,1)}(1,:),...
    transitions{transitionIDs(j,2)}(1,:),dt,cutoff);
pdf = pdf./(dt*sum(pdf)); % normalize
transitions{j} = [pdf; dt.*cumsum(pdf)]; % generate cdf and store
end
Bibliography


