3-21-2013

Determining Optimal Machine Replacement Events with Periodic Inspection Intervals

Theodore C. Shiveley

DETERMINING OPTIMAL MACHINE REPLACEMENT EVENTS WITH PERIODIC INSPECTION INTERVALS

THESIS

Theodore C. Shiveley, Captain, USAF

AFIT-ENC-13-M-17

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government.

AFIT-ENC-13-M-17

DETERMINING OPTIMAL MACHINE REPLACEMENT EVENTS WITH PERIODIC INSPECTION INTERVALS

THESIS

Presented to the Faculty

Department of Mathematics and Statistics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science (Operations Research)

Theodore C. Shiveley, BS

Captain, USAF

March 2013

AFIT-ENC-13-M-17

DETERMINING OPTIMAL MACHINE REPLACEMENT EVENTS WITH PERIODIC INSPECTION INTERVALS

Theodore C. Shiveley, BS
Captain, USAF

Approved:

Maj James Cordeiro, PhD (Chairman)	Date
Dr. Jeffery Cochran, PhD (Member)	Date

Abstract

This research will examine the optimal maintenance and replacement policies for a generic machine with periodic inspection intervals. The considered reliability models consist of a single machine that can fail during operation or else may be found to be inoperative during regularly-scheduled maintenance inspections. Most single-machine reliability models specify operational (degradation) modes for the machine to occupy between failures. However, since the process of determining the appropriate number of degradation states is subjective, and often results in unneeded complexity, we will instead utilize failure modes, each of which is associated to a system sojourn time that is distributed exponentially with distinct rate parameter $\mu_k, k = 1, 2, ..., r$, where r is the total number of failure modes. A distinction will be made between spontaneously-occurring failures during operation and those that are discovered during inspections. Since the elapsed time between inspections is constant, the resulting stochastic reliability process becomes non-Markovian, and thus a Semi-Markov Decision Process (SMDP) framework must be employed in order to determine the cost-optimal stationary policy consisting of repair and replace decisions and inspection intervals. Using the methodology developed here, a system controller will be able to readily develop an inspection-based strategy to optimize the overall costs of maintaining systems with a variety of failure characteristics over a finite time horizon.

I would like to dedicate this thesis to my mother; for inspiring me to pursue mathematics, but cautioning me about not getting carried away in the abstractions of pure mathematics.

Acknowledgements

I would like to thank everyone for the tremendous amount of patience they had with me in this process. Family, friends and especially teachers; your thoughtfulness, understanding, direction, and support made this possible.

My advisor, Maj Cordiero...It's been a journey to say the least; thank you for your dedication and careful guidance.

Everyone who helped me with understanding the nuances of different programming languages, in particular my father and wikibooks; Thanks.

Finally, Frankie Woods for telling me grad school ain't such a bad idea, and Kelsey Danner for keepin' it real!

Theodore C. Shiveley

Table of Contents

Page

Abstract iv

Acknowledgements vi

List of Figures ix

List of Tables x

I. Introduction 1

1.1 Background 1
1.2 Problem Statement 2
1.3 Research Objective and Scope 4
1.4 Summary 8

II. Literature Review 9

2.1 Overview 9
2.2 Markov Decision Processes 10
2.3 Remaining Useful Life Estimation 14
2.4 Other Influences 15
2.5 Summary 16

III. Methodology 17

3.1 Model Development 17
3.1.1 General Background 17
3.1.2 Inspection Process Design 20
3.1.3 MDP Construction 23
3.1.4 The Transition Time Matrices 25
3.1.5 The Transition Probability Matrices 28
3.1.6 The Transition Reward Matrices 31
3.1.7 Inspection Process Comparisons 32
3.2 Model Exploration and Process 39
3.2.1 Cost Structures 39
3.2.2 Probability Structures 41
3.2.3 Time Structures 42
3.2.4 Simulation Process 42
3.2.5 Scope and Exploration of Models 43

Page

IV. Implementation and Analysis 45

4.1 Matrix Construction 45
4.2 Policy Iteration Results 46
4.3 Simulation Results and CDFs 46
4.4 Discussion 47

V. Conclusions and Recommendations 67

5.1 Review 67
5.2 Insights 67
5.3 Potential Future Research 68
5.4 Conclusion 70

A. SMDP Policy Iteration Procedure 71

B. Java Simulation Code 74

Bibliography 84

List of Figures

Figure		Page
1	2-State Machine	5
2	Machine with Inspections and Repair/Replace Decisions.	6
3	Machine with Successive Degradation States.	7
4	Machine with Inspections and Multiple Failure States	7
5	Machine with Inspections and Multiple Failure States	21
6	Illustration of Embedded Inspection Process	22
7	Machine 1 Long Process Matrices	53
8	Machine 1 Short Process Matrices	54
9	Machine 2 Long Process Matrices	55
10	Machine 2 Short Process Matrices	56
11	Machine 3 Long Process Matrices	57
12	Machine 3 Short Process Matrices	58
13	Machine 4 Long Process Matrices	59
14	Machine 4 Short Process Matrices	60
15	Machine 1 Empirical CDFs	61
16	Machine 2 Empirical CDFs	62
17	Machine 3 Empirical CDFs	63
18	Machine 4 Empirical CDFs	64
19	Availability vs Age for Machine 2.	65
20	Enhanced Availability vs Age for Machine 2.	65
21	Availability vs Expense Rate for Machine 2.	66

List of Tables

Table		Page
1	Machine Parameters and Descriptions	18
2	Inspection Process Parameters and Description	19
3	Survival Probability Development	25
4	Inspection Cost Structure Elements	36
5	Policy Iteration Results	46
6	Policy Iteration Results for Highly Productive Machine 1	51
7	Cost of Inspection Cycles	51
8	Policy Iteration Results for Altered Machine 3	52

DETERMINING OPTIMAL MACHINE REPLACEMENT EVENTS
WITH PERIODIC INSPECTION INTERVALS

I. Introduction

A good plan now is better than a perfect plan next week.

– General George S. Patton

1.1 Background

Many times in the application of systems, the operator has some idea of the characteristic reliability inherent to that system. From assembly lines, to computers, to aircraft, quantities such as mean time to failure, mean time to critical failure, and others have been quantified to a great extent. Further, any entity concerned with cost will also have an idea of the cost of typical repairs and the time those repairs take; therefore, the opportunity cost or profit forgone while an machine is under repair can be calculated. If examined more closely, data can sometimes be found that more rigorously characterizes the reliability of the system and perhaps gives something similar to the estimated time before a given proportion of failures will occur. The ideal situation is to have a completely characterized distribution with definite parameter values. This would provide a basis to model the cost/reliability tradeoff inherent to a system. Then, the next question is to determine whether, at a occurrence of a failure, it is more advantageous to repair or replace the subject of study.

If such data is known, or an estimate of such data can be constructed from operational experience or empirical sources, stochastic process modeling can provide a

framework for approaching the repair or replace question. Markov chains, for one, give a suitable though basic framework in which to employ this data. Indeed, the "two-state machine" is a well studied problem in which a machine will oscillate between working and broken states, and metrics such as proportion of up and down time can be readily calculated. The utility of metrics such as this are immediately obvious; these numbers can be simply employed to construct an estimate of blended cash flow from operation based upon this simple model. However, in complex systems, where a more intensive approach is warranted or where the cost of downtime is very great, it would be worthwhile to investigate the consequences of different choices more thoroughly. One method to do this is to include periodic inspections into the system and allowing for a variety of failures to occur. The goal of this is to plan downtime more accurately, minimize unexpected events, and maximize the return on investment by knowing the systems intrinsic characteristics.

The problem presented by the introduction of inspections is interesting because it requires making at least two decisions in the process. The first is what to do with the data the inspection reveals, i.e. what action, if any, should be taken upon the system. The second is when to inspect; i.e. by time interval or some other relevant metric? The inclusion of a decision in the maintenance process requires examination beyond the framework of a basic Markov chain. A more suitable framework for examining this problem is the "Markov Decision Process" (MDP).

1.2 Problem Statement

In the particular case of regulating maintenance schedules and procedures, much effort has been focused on categorizing operational/degradation states of a machine, and then determining what should be done at the end of those states or when a failure is observed within one of those states. This convention has its definite attraction.

It is reasonable to assume that most systems exhibit some type of degradation with age and that their ability to perform at a designated capacity diminishes with time. For instance, if comparing a ten year old machine with a brand new machine, the newer machine might be assigned a better operational state just based on the age information. The operational state of the machine could be defined in many ways, such as use threshold (hours, miles, etc) or an expected time until an event, typical failure. The object of a MDP governing a process such as this is then to determine when it is optimal to replace the machine, and up until what point should repairs be made. Also in the operational states framework, the uniformity of identical repair actions is intuitive. At each decision point in the process having identical actions to choose from for each state simplifies the construction, but does so in a way that preserves the central ideas of when to repair and replace within a machine's lifetime.

Though this approach is not without merit, another approach to follow is to observe the system from the failure side, instead of the operational side. In this formalism, an operational machine is just an operational machine. The ambiguity of determining operational states is removed; if a machine works, it is capable of performing its designed task. Indeed this approach is not appropriate for all types of analysis, and it might be viewed as an overly crude simplification, but its value should not be lost. For example, there is no uncertainty in determining to what extent is a brand new machine is superior to a ten year old one. The failure side observation of the process allows this machine to break in any of several ways, each with different consequences to the system. This allows the MDP's policy to choose the best action at each of those failure events. In this way specific failure types that warrant a replacement, and the ones that only require a repair will be determined.

1.3 Research Objective and Scope

An MDP, as the name suggests, allows for the inclusion of decisions into a Markov chain. Once a decision point is reached, an action by the governing entity is required, that will then result in the system assuming a different state. Also concurrent with any action is a reward or cost for choosing that action. The ultimate goal of these different actions is to attain the optimal value of some objective function, usually cost or rate of cost, by varying the actions chosen. The collection of decisions undertaken in an MDP is known as a "policy." Though rules of thumb such as having your air conditioner serviced every spring can go a long way torward approximating an efficient maintenance/inspection routine, those types of nonspecific policies leave at least two questions unanswered:

1. Is the risk to the system measured in the right units, i.e. is miles the right metric or might hours in operation be more relevant to knowing the 'intrinsic' age of the system.

2. What is the best inspection period to optimize the desired metric; be it system availability, cash flow, or minimizing some punitive aspect.

Further, if the cost of downtime to the system is excessively large, or if the cost of downtime is generally negligible, it may be in the best interest of the controlling entity to determine a repair or replace policy that weights the costs associated with failure events in a way that reflects this risk aversion or acceptance.

The next question is how to define the state space of the MDP. The answer to this question is less straightforward. The basic idea of the machine is a relatively simple two state machine, which at every occurrence of failure either choose to repair or replace (Figure 1). This results in two possible policies, and a straightforward conclusion for either adopted policy.

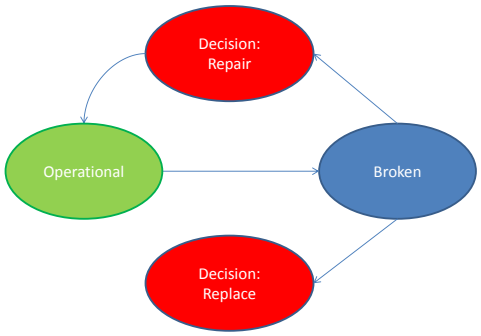


Figure 1. Generic Machine with Inspections and Repair/Replace Decision.

One approach that could be used to improve the framework in Figure 1 would be the inclusion of imperfect repairs, where subsequent repairs are a progressive degenerative process. An approach to this would be to have the degradation state of the system follow a Markov chain. This would imply that the system now possesses some intrinsic age indicated by the degradation state the system finds itself in. The goal of progressive degradation states would be to induce faster and faster breakdowns after repairs are undertaken. This system is somewhat more interesting to study, but in a manner suffers from the same problem of determining and specifying the different degradation states and schedules.

The model being developed incorporates not only breakdowns, but also periodic inspections if a breakdown is not observed. From the operational state, the machine can either operate until an inspection is required, or spontaneously fail (Figure 2). Further, the broken state can be accessed from the inspection state by an unserviceable decision as well as from operation. It is also possible for a machine to experience several chained failures without ever returning to operation through an unsatisfactory post repair inspection. This model could also be expanded to accommodate more than one failure state.

This model allows considerably more flexibility and merits more consideration. However, with the added flexibility comes complexity. In addition to deciding re-

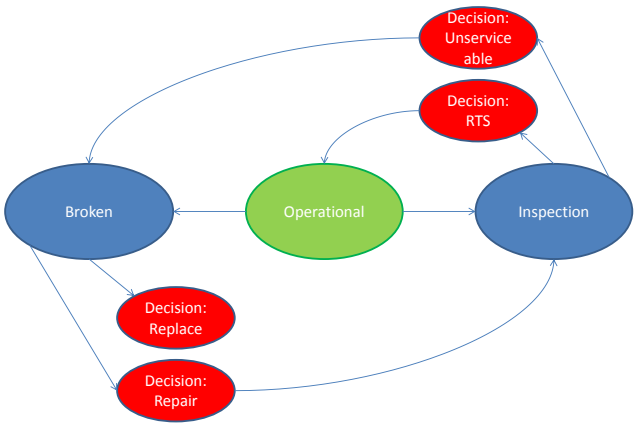


Figure 2. Machine with Inspections and Repair/Replace Decisions.

pair/replace costs and lifetime distributions, inspection intervals, criteria for inspection decisions, the benefit gained by preemptively catching a failure before it has the opportunity to degenerate into a more significant situation must also be specified. Also if using multiple failure types, those failures need to be characterized along with all of the associated costs, time penalties and probabilities of failure occurrence. This framework obviously presents a number of challenges to model, and numerous decisions and criteria need to be specified before this can be undertaken.

To make the discussion of maintenance and inspection models complete, consider a model with multiple operational states, periodic inspections, and spontaneous failures (Figure 3). This is a very general framework, and has very great complexity and flexibility introduced by accommodating it. This framework preserves the feature of allowing an entity to fail spontaneously or be found unserviceable at an inspection. It also permits a machine to in effect become "younger" after a repair by allowing for the possibility of return to a higher operational state after the repair. Moreover, the failure states could also be expanded to accommodate a variety of failures. This model has definite attractive qualities, but it also possesses some shortcomings. This

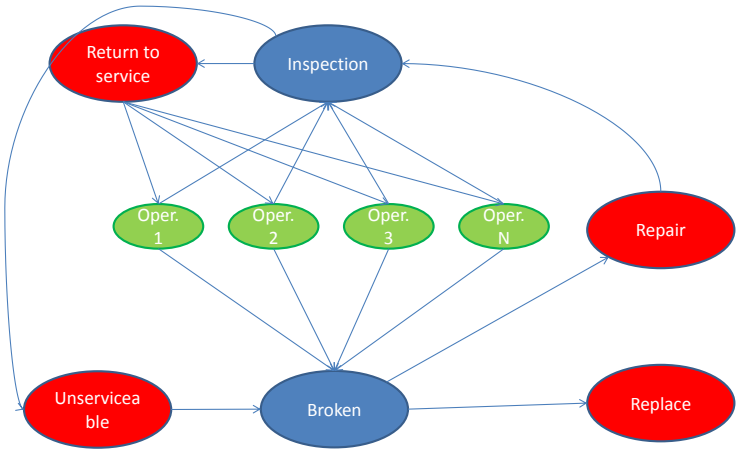


Figure 3. Machine with Successive Degradation States.

model still requires the specification of several operational states and the cumbersome parameter specification required to characterize it sacrifices objectivity.

In chapter 3, a model which has one operational state and a variety of failure states is developed (Figure 4).

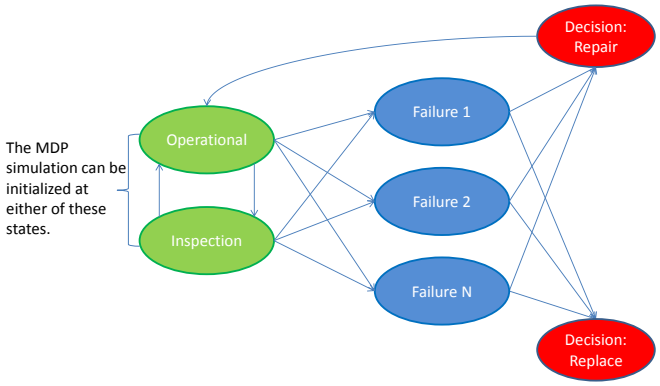


Figure 4. Machine with Embedded Inspection Process and Multiple Failure States.

This model possesses the advantage of avoiding imprecise specification of multiple operational states, and preserves the feature of defining failure states based on observable events. The MDP developed will seek a policy which not only specifies re-

pair/ replace actions for failure states, but is also able to determine which inspection scheme results in the lowest average operational cost among defined options.

1.4 Summary

In our studies of MDPs for the considered reliability models, average cost per time will be used to determine optimal policy rather than discounted cost per cycle. The reason for choosing this approach is that it removes speculation on asset price decay and the intrinsic value of a near cost versus a far cost. In contrast, an average cost paradigm allow the development of a policy based purely on value. Also, in the discounted cost paradigm, the varying lengths of time between decision epochs make it more difficult to deal with the discounting factor, and leaving the form of the problem as simple as possible preserves accuracy.

The end goal of this is to study not only understanding the nature of maintenance processes and MDPs, but also the merits and drawbacks of each approach to the various machine maintenance problems presented. Beyond mechanical applications, one can envision the utility of this work to fields such as healthcare, finance, and even customer service. Many parameters are considered: what values of inspection interval length for repair/replace decisions lead to dramatic changes of behavior; what type of cost schedule for a repair/replace decisions cause a change in behavior; the severity of downtime cost penalties; and the appropriate length of downtime. Rather than exhaustively characterizing a narrowly defined problem, we seek to understand the underlying dynamics of how an adopted maintenance process can affect various measures of performance is the desired outcome. The results will aid in predicting costs, reliability, and general performance of infrastructure as well as yield an idea as to what forms of infrastructure are best suited to adopting a particular machine model.

II. Literature Review

2.1 Overview

The emphasis of this research involves the merger of two central ideas. While each idea has its own strengths, their union possesses unique potential. The first idea is Markov Decision Processes. An MDP is akin to a Markov chain but differs in that a controller can make decisions at transitions of the process. This in turn, affects the evolution of the system over time. These decisions are chosen so that, in the long, run the behavior of the Markov chain formed by the collection of decisions will be optimal with respect to the predetermined critical metric of the system. This is done as a maximization or minimization of some quantity, be it money, production, or a rate. The second idea is the maintenance and reliability of systems; in this case the emphasis is optimal maintenance policy. Using MDPs to control a maintenance process is not a new idea, but the extent to which it has been applied and explored still leaves room for expansion.

Much of the work accomplished in this field has been done since the 1950's with respect to the dynamic programming part, and much more rapid growth has occurred with the rise of computers and the increase in computational complexity which machines are now able to tackle. The first two sections in this chapter will address the main focus areas already mentioned; MDPs and dynamic programming, and maintenance and reliability studies which emphasized Markov models; the third section will make mention of other sources consulted that were useful in this study and contributed some key aspect to this work.

2.2 Markov Decision Processes

Markov Decision Processes occupy an interesting spot in mathematics. They simultaneously represent an elaboration of a basic structure, the Markov chain, and also are a special case of a much more complex structure, the multi-player stochastic game [17]. In maintenance scenarios the objectives of other players/entities are not of consequence, so the appropriateness of only considering a single player is obvious; hence the complexities introduced by considering multiple players is not a concern. Similarly, understanding the optimal decision to make under a given set of assumptions is also desirable, so the use of an MDP is appropriate.

The genesis of the MDP structure can best be traced to the 1950's when Bellman was doing work on multistage decision making problems which he would later be called Dynamic Programming (DP). In these problems, the object was the maximization or minimization of some value function. A functional equation was used to determine the best path through the network an entity should take. This functional equation is applied iteratively at each step in the decision process to determine what action the entity should take. DPs come in two broad types: deterministic, and stochastic. In deterministic DPs, the optimal path is completely dictated by the selected action, where as in stochastic DPs, the action selected is only the probabilistic best outcome. Included with transition in dynamic programs are associated costs or rewards, which can also be of definite of variable value. Since Markov processes model how an entity may probabilistically travel through a known state space, if one includes decisions in this process, the result is an MDP.

MDPs, of course, rely on understanding Markov chains and Semi-Markov processes in order to fully leverage their benefits. The text referenced for Markov chains and semi-Markov processes was Ross [15]. Some aspects of this understanding were also

augmented by sections from the text by Phillips, Ravindran, and Solberg [14]. Both books cover much of the same material, though each has its merits.

Also before MDPs are addressed directly, an understanding of the concepts driving DPs is advantageous. A succinct introduction to dynamic programming and its motivations and techniques can be found in Denardo [4]. Bellman also authored several books on the subject, with the most known being simply called "Dynamic Programming" [1]. Other authors who have contributed notable texts on the subject include Puterman [13] and Bertskas [2]. All these books offer their own unique approaches and merits, though Denardo's book [4] offers the best introduction to the novice.

Finally in MDPs, the set of equations which are used to obtain optimal policies are known as the Bellman equations, after the same Bellman previously discussed. Though all of the above sources make mention of this set of equations, the most is made in Gosavi [6]. The strength of this text lies in the extensiveness of the discussion of the Bellman equations. Development is begun from an understanding of the underlying Markov chains. From there both policy and value iteration are discussed, these are the two ways the Bellman equations can be used to obtain an optimal policy. Policy iteration relies on solving a system of linear equations for each step, while value iteration only solves one equation and relies on epsilon convergence. The pitfalls of each approach and specific cautions about when one method is more appropriate than the other are also discussed. Methods for making value iteration converge faster are explored as is how to adapt each of the methods for both discounted reward and average reward approaches. The option of using either approach is somewhat a matter of preference though problem specific considerations come into play as well. Another useful aspect of this text is that it includes computer coding suggestions and code sections for most of the covered techniques, complete with implementable examples.

As one might imagine, the range of applications for MDPs is broad. A particularly instructive source on the MDP technique was by Tao et al. [18]. In this paper the authors thoroughly cover the thought process behind their decisions and examine the problem of bridge design and maintenance. Starting with the rudimentary elements of MDPs, the authors develop this to application on their problem. The result from their study is an optimal bridge maintenance policy and an optimal bridge design based on that policy.

Another paper that was found to be especially relevant to this study is by Tomasevicz et al. [19]. This paper is a numerical study on optimal maintenance upon air blast circuit breakers. The circuit breaker work is particularly meaningful to this research due to the following aspects. First, the authors' SMDP design allows for both spontaneous failure and also an inspection failure. In addition, the circuit breaker may also suffer a degradation failure. The degradation failure is a result of the circuit breaker traversing all of its degradation states without suffering a spontaneous failure. Another feature of this model is that at the end of every degradation state, if a spontaneous failure is not observed, then the MDP has the option to choose between two levels of maintenance to carry out, with each possessing different costs and different operational implications. Finally, spontaneous failures always require replacement and return to the highest degradation state; service at the end of a degradation state does not necessarily result in a "like-new" machine (replacement), but instead, returns the machine to the beginning of the just sojourned degradation state, or possibly a higher state depending on the type of maintenance selected. The end result for this work is construction of optimal policies for overall availability, and an optimal policy for minimum cost given a required availability threshold.

Another paper worthy of mention comes from Crabhill [3]. Here, conditions are set for when policies may be trimmed from consideration in an MDP. The number of

possible policies in an MDP is the product of the number of actions available from each decision state, $P = \prod_{s \in S} a_s$, where P is the number of policies, s is a single state and the state space S, and a_s is the number of actions available from state s. Since this number can grow to be quite large quite fast, it is advantageous to be able to rapidly eliminate policies so that the policy space over which the Bellman equations must iterate is smaller.

White [21] authored a survey paper that does an excellent job of reviewing published applications of MDPs. The application of MDPs and SMDPs are wide ranging in both implementation and construction. White reviews some 80 papers that discuss specific applications of MDPs and SMDPs. Broadly, the areas covered by this collection are:

- Harvesting (agriculture and population control)
- Water resource management
- Inspection, maintenance and repair policy
- Purchasing, inventory, and production
- Queues
- Finance and investments
- Several other smaller areas of interest

This is quite a spectrum of applications and conveniently White also includes brief discussion of all of these papers; details include a description of the goal, the strategy employed, and notes that are pertinent to the specific application being discussed. The methods used in applications cover finite and infinite horizons, discounted profits and losses, average returns, etc. Topics vary from hydroelectric power generation to how much money an insurance company should optimally invest each day.

2.3 Remaining Useful Life Estimation

After the MDP technique is employed, and a stable policy is decided upon, the resulting realization of the MDP/SMDP is reduced to a Markov Chain or semi Markov process. This is because, once a policy is decided upon, the transition probabilities associated with the chosen actions become static, and hence a transition matrix will be able to be formed because a specific action is known for each decision state. The relevance of understanding the contributions of remaining useful life is clear if time until absorption is treated as the end of life.

In Gebrraeel's paper [5] he presents an intriguing discussion about predicting remaining useful life in time-varying environments. The particular issue under study is bearing life, but its primary strength is providing a good discussion and results of remaining useful life in censored machines. One of the papers that Gebrraeel uses is Kharoufeh and Cox [7], which also focuses on censored machines. They propose a method to determine the appropriate number of degradation states a machine should possess if no other method is available to determine this number. It should be noted that the method proposed is dependent upon the person developing the model having actual data and is very similar to the familiar one-way ANOVA. Numerical results from this paper are somewhat constrained by the assumptions necessary to obtain them, so the most intriguing part is the method proposed to classify degradation states. Gabrraeel's bearing study also makes use of another Kharoufeh paper [8]; that also discusses remaining useful life estimation, but this time incorporates periodic inspections.

The survey paper by Si et al. [16] does a good job of examining the breadth of this subject. The authors divide the body of work into two broad categories, directly observed state processes, and indirectly observed state processes. Within

each of these categories are appropriate subheadings depending upon the variety of approaches that have been pursued in the literature.

2.4 Other Influences

Other works not immediately pertaining to MDPs or remaining useful life are still relevant to this research. Some of the highlights in particular deal with results from Queueing Theory, Renewal Theory, and methods for dealing with reliability data.

In Phillips, Ravindran and Solberg's book [14], they mention a little known theorem with a powerful result which they refer to as Khintchine's Theorem. However, upon further investigation, in a work by Khintchine, a proof is given for what Khintchine refers to as"Palm's theorem"[10]. This powerful result claims that if several processes are ongoing simultaneously, the inter-event time of the super-positioned processes will be exponential, regardless of the inter-event distributions of the member processes. This result may be thought of as being analogous to the Central Limit Theorem, but for inter-event time distributions. This result will be used as a simplifying assumption and justification.

Meeker and Escobar's text on reliability [12] covers a lot of ground and invests most of its focus in topics of fitting distributions to observed data and methods of parametric estimation. However, a particularly interesting chapter in this text involves nonparametric CDF construction and estimation from empirical data.

Finally, two other works find relevance to this thesis: the first is Kulkarni [11], and the second is another White paper [20], which happens to be a seminal work in the area of optimal replacement. Kulkarni's book deals with a number of topics, but its key contribution is in detailing and explaining the kernel matrix, which is the time-dependent transition probability matrix which describes a renewal process. In particular, his review and discussion of the kernel matrix for semi-Markov processes

is illuminating. The White paper, which is an exclusively analytical work, examines when it is optimal to repair or replace a system whose cost of operation is always increasing. Results are developed for rectangular, exponential, and normal distributions. It is worth noting that the original paper was submitted for publication by Bellman, who was previously mentioned in the MDP section, and was a major contributor in the development of dynamic programming. This work would be particularly instructive if examining a system that had a lifetime distribution with an increasing hazard rate, but undertaking that type of complexity was not possible. Instead, an increasing hazard rate could, in a manner, be simulated by allowing for ever increasing costs of operation.

2.5 Summary

This research will rely on ideas from several areas, in particular dynamic programming using Markov Decision Processes, reliability, and remaining useful life estimation. The mutual reinforcement among these areas, and deeper insight gained by examining them from different perspectives, is essential to understanding the goals of this study and the novelty in the approach.

III. Methodology

3.1 Model Development

This section will reiterate the motivations behind choices made in developing the form of the model, and go into detail on how the calculations of various quantities pertaining to the model were carried out.

3.1.1 General Background.

As discussed in Chapter One, the construction of the MDP model carries with it implications on the information it will reveal about the system. In much of the literature, the emphasis is on assigning different degradation/ operational states as the system progresses to failure. This formalism, while appealing to operations where a clear definition of different degradation or operational states exists, does not lend itself to explaining the characteristics of a system whose operation is effectively monotone, or where the operational parameters of the system are so disparate that attempting to assign operational or degradation states would just complicate the model and do little to elucidate the true characteristics of the system.

With this in mind, the model put forward will have one operational state and any number of failure states. This difference allows for the reduction of speculation on the operational state of the system and instead focuses on observable failure events. If the type of failures a machine or process can experience are known then it would be beneficial to have this represented in the model formulation. If this data is not known, or the types of failures that can be experienced are so numerous that it would only complicate analysis to try to exhaustively include all failures, then an approach similar to that of Kharoufeh [9] may be used to determine the best number of failure states.

For the aforementioned reasons, this approach is worthy of further consideration as a contrast to the multiple operational states model. While this removes some of the purity of only estimating the remaining useful life of the system, it will be able to produce both a cost and lifetime CDF of the modeled system.

This approach requires rather intensive parameter specification before it is undertaken. In order to facilitate this process Table 1 and Table 2 list the parameters that need to be specified. The first table lists the values which define the machine under inspection and the second table lists the parameters which define the inspection process. Most of these values are quantities that need to be specified by the machine operator; however, several values rely on other parameters but are included here due to their centrality.

Table 1. Machine Parameters and Descriptions

Notation	Element Name	Description
$C_{replace}$	Replacement cost	The cost imposed when a replacement decision is made from a node
C_{F_n}	Failure Cost	There are as many of these as there are failure types
T_{F_n}	Failure Time	There are as many of these as there are failure types
V	Production rate	The rate at which the machine produces reward per unit time; included to calculate opportunity costs for inspections and other down-time

There is some amount of interplay involved in these parameters, and that in particular the inspection parameter, cost of down time, depends upon the machine parameter, production rate. This entanglement is even more substantial in future calculations. This should not cause concern; the tables should serve as a guideline for the mental allocation and not as absolute divisions. Further notation that is involved

Table 2. Inspection Process Parameters and Description

Notation	Element Name	Description
T_O	Length of operational period	If a failure is observed, the last period is ended at that point
T_I	Length of inspection period	Failures are accumulated at the end of inspection periods and they always last the whole interval
$\overline{T}$	Average length of operation/inspection cycle	This is a calculated quantity from other parameters
P_{OF_n}	Probability of operational failure	There will be as many of these as there are failure states
P_{IF_n}	Probability of inspection failure discovery	There will be as many of these as there are failure states
P_{OI}	Probability of transition from operation to inspection	Also equal to $1 - \sum P_{OF_n}$
P_{IO}	Probability of transition from inspection to operation	Also equal to $1 - \sum P_{IF_n}$
C_I	Total inspection cost	The total cost incurred per visit to inspection node
C_M	Inspection material cost	The material cost incurred per visit to inspection node
C_{B_n}	Cost bonus for inspection failure discovery	Incentivizes failure discovery while it is advantageous
C_D	Cost of downtime due to inspection	The product of the machine's "production rate" and the length of inspection period, opportunity cost

in the application of particular techniques will be explained when appropriate. These tables are an exhaustive list.

The remainder of this chapter is broken up into two main sections: Model Development and Model Exploration and Process. The first section will discuss the techniques employed and the derivation of quantities necessary to characterize the model. The discussion in the latter section will focus on the parameter levels being investigated, the reasons why such parameters were chosen, and the methods and procedure used to realize results.

3.1.2 Inspection Process Design.

In this model, an inspection scheme will be implemented as part of the routine operation of the machine. Because of the inspections and MDP models must necessarily be defined in discrete steps, a semi-Markov process is embedded into the ongoing decision process to simulate this inspection process. This is required because every node in an MDP/SMDP must be a decision node, and the passive ongoing inspection process does not require any decisions to be made. No decisions are made at inspections because the object of the MDP is not to optimize what is a failure and what is not, rather it is to determine what the optimal action is in consideration of the just observed failures. This process could be likened to a hidden Markov process where the emissions of the process are generating the failures that the system observes.

The illustration in Figure 5 should serve to clarify the explanation just given. Figure 5 illustrates how the embedded process of operation and inspection conceptually interact. To interpret the figure correctly, failures are the decision states of the MDP, the ovals labeled with “decision” are the possible actions from each decision state, and the ovals labeled “operation” and “inspection” represent the embedded operation and

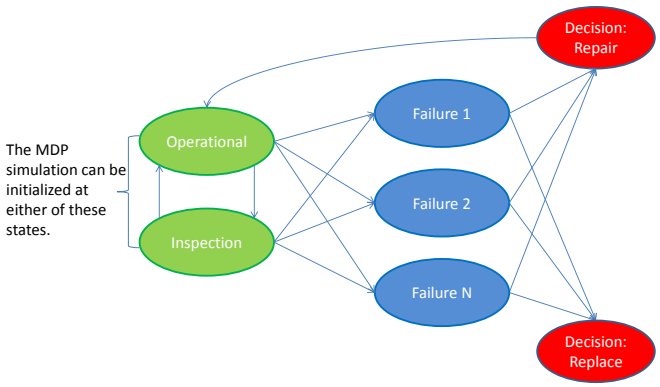


Figure 5. Machine with Embedded Inspection Process and Multiple Failure States.

inspection process from which any failure state is attainable from either operation or inspection.

In this model, the time between operational failures of the entity will be assumed to be exponential. This is done for the following reasons. First, this is done to simplify the model and corresponding analysis. Because the exponential distribution is memoryless, after every operation and inspection cycle the remaining time until failure possesses the same distribution as that of the entire lifetime, which enables the use of the Markov process. This allows the decisions to remain time stationary, which is a requirement when using a MDP because transition probabilities cannot change with time or iteration. Also, because of this assumption a series representation of how the process transitions through the operation/inspection process to the failure states is able to be constructed. Finally, this assumption can be justified when Palm's Theorem (see [10]), and the generally observed convention of good-as-new repairs are considered. Palm's Theorem states that the superposition of the interevent times of a sufficient number of randomly occurring processes, regardless of the individual interevent distributions, will be exponential. When taken together, this theorem and the good-as-new convention allows the assumption of a static operation and inspection process.

The assumption of exponentially distributed failure times also allows for construction of a series which describes the time until absorption of the inspection process. From this series it will also be possible to calculate other quantities necessary for the use of the Bellman equations, such as the time until absorption distribution, and in particular its mean. The series is constructed relying upon the symmetrical behavior between cycles that the inspection process possesses (Figure 6).

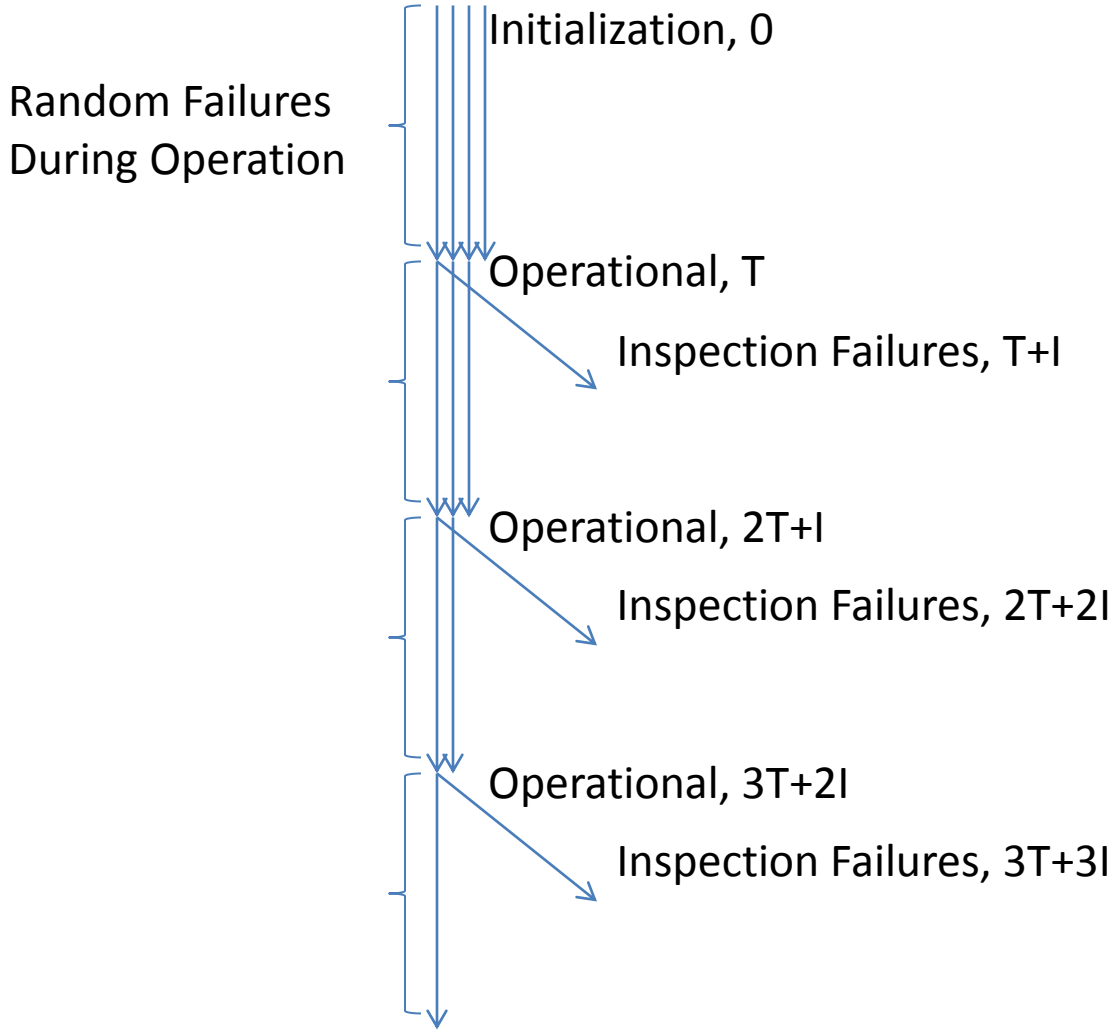


Figure 6. Illustration of Embedded Inspection Process Time Progression and Survivorship.

3.1.3 MDP Construction.

The actual MDP problem will be solved using the Bellman equations, before this is done several considerations need to be weighed. First, there are two approaches on using the Bellman equations; one is to look at average reward, the other is to use discounted reward. This study will focus on using the average reward method of solving the Bellman equations, which is motivated by removing speculation on the value of near dollars versus far dollars. This removes speculation on inflation, interest rates, and the business performance of the enterprise controlling the machine. It is necessary to use one or the other of these methods to adjust the costs and reward of the MDP because without them there is no guarantee that on an infinite time horizon the costs and rewards of the MDP would converge.

The other choice that must be made is whether to use the techniques known as policy or value iteration; policy iteration searches for optimal policy by repetitively solving systems of linear equations, while value iteration does so by using a form of ϵ-convergence. This analysis uses policy iteration. For a thorough understanding of the advantages and disadvantages of each approach see Gosavi [6].

The Bellman equation for the average reward semi-Markov decision process using Gosavi's notation takes the following form:

$$h_k(i) = \bar{r}(i, \mu_k(i)) - \rho_k \bar{t}(i, \mu_k(i)) + \sum_{j=1}^{|S|} p(i, \mu_k(i), j) h_k(j) \tag{3.1}$$

Where:

- h is the value function being optimized. Its argument, i, is the state of the system, its subscript, k, is the iteration number. The h's are the unknowns, there are as many of these as there are states.

- $\bar{r}$ is the average reward. The arguments of it should be interpreted as the average reward from state i, under the policy(μ) used in the k iteration.
- μ is the policy. If it is accompanied by a specific argument, it is referencing the action for that state. μ used without an argument represents the policy vector, or the collection of actions for all states.
- ρ_k is the average reward of the k iteration. This is one additional unknown.
- j is the next state the system may transition to.
- S is the state space of the system.
- $p(i, \mu_k(i), j)$ is the probability of going from state i to state j under policy μ_k.
- $\bar{t}(i, \mu(i))$ is the average time spent in the state i under the current policy μ_k.

It should be evident that this system is under determined by one unknown. This problem is handled by setting one of the $h_k(i)$ terms to zero at each iteration and using that states value as the benchmarked relative value from which the other states average reward is determined. The system of equations that results from the Bellman equations are solved iteratively until the recommended policy becomes stable. In this system specific symmetry intrinsic to the model will ensure very fast convergence to optimal policy. The previously introduced Figure 5 can be referenced as a diagram of the process that policy iteration will be performed upon.

To use the Bellman equations it is necessary to first define three sets of Matrices for the Semi Markov Decision process (SMDP). These are the transition time matrix (TTM), the transition probability matrix (TPM), and the transition reward matrix (TRM). In the framework being used, sometimes individual elements of these matrices cannot be solved for under the conventions set out. Instead rows or entire matrices are solved for in one step to provide the requisite information necessary to use the Bellman

equations, this results in no loss of information and actually helps to streamline the process.

3.1.4 The Transition Time Matrices.

In order to calculate the average time of transition out of this process to a failure type for the MDP, refer to Table 2 (particularly T_O, T_I, TP_{OI}, P_{IO}). These variables are not intrinsic characteristics of the system, and their values need to be assigned by the operator as required.

To begin calculating the lifetime distribution of the inspection process survivorship is used to build up the survival function of the inspection process. The survival function can then be converted into the CDF by taking its complement. This is done to simplify the expressions that result as the survival function contains the same information as the CDF. Using the variables just declared the probability of the original entity surviving operation/inspection cycles is developed in Table 3. Note the pattern that is developing as an entity successively survives operation and inspection. By use of induction it can be shown that the following two equations can be used to express the survival probability, P_n, for any number, n, transitions:

$$P_n = \begin{cases} (1-P_O)^{\frac{n+1}{2}}(1-P_I)^{\frac{n-1}{2}} & n \text{ is odd} \\ (1-P_O)^{\frac{n}{2}}(1-P_I)^{\frac{n}{2}} & n \text{ is even} \end{cases}$$

Table 3. Survival Probability Development

Transition(n)	Survival Probability Expression
0	1
1	$(1-P_O)$
2	$(1-P_O)(1-P_I)$
3	$(1-P_O)^2(1-P_I)$
4	$(1-P_O)^2(1-P_I)^2$

Using this result allows for calculating the number of transitions which take place on average before the mean survival percentage of the distribution is reached. $\overline{N}$ is reserved to denote the average number of transitions that occur in a given type of inspection scheme. Given an n, it becomes straightforward to calculate the time until P_n percentage is reached:

$$T_n = \begin{cases} \frac{n-1}{2}(T_O + T_I) + T_O & n \text{ is odd} \\ \frac{n}{2}(T_O + T_I) & n \text{ is even} \end{cases} \tag{3.2}$$

To solve the Bellman equations, mean transition time is required. The mean time of transition will specifically be denoted by the quantity $\overline{T}$. When the mean transition time occurs during an inspection, Equation 3.2 is still valid, since all failures are aggregated at the end of the inspection period. However, when the mean time of transition takes place during an operational period (note, n will be odd), the following adjustment is necessary:

$$\begin{aligned} \overline{T} &= \frac{\overline{N}-1}{2}(T_O + T_I) + E(P_{\overline{N}-1} - .5) \\ &= \frac{\overline{N}-1}{2}(T_O + T_I) - \frac{1}{\lambda_{\text{implicit}}} \ln\left(1 - (1 - P_O)\frac{P_{\overline{N}-1} - .5}{P_{\overline{N}-1} - P_{\overline{N}}}\right) \end{aligned} \tag{3.3}$$

Here it is observed that in the case the mean or other designated percent, will occur during an operational interval, to compensate for the fractional interval time that will pass between the end of inspection and the mean the additional term which can be shown to be equal to $E(P_{\overline{N}-1} - .5)$ is included. This result relies upon properties of the exponential distribution. Finally, the quantity $\lambda_{\text{implicit}}$, is a machine property that represents the exponential lambda implied by the assigned inspection process

and can be calculated as follows:

$$\lambda_{\text{implicit}} = -\frac{\ln(1 - P_{OI})}{T_O} \tag{3.4}$$

This quantity will differ between inspection processes even when comparing the same machines. This represents relative differences in the effectiveness/efficiency of inspection processes. λ is derived by inversing an exponential CDF where time and cumulative percent are known.

The average time of transition just found can be inserted into the Bellman Equations [3.1], for both repair and replacement since the machine possesses identical parameters under either repair or replace actions. Simply stated:

$$\overline{T}_{\mu_{repair}} = \overline{T}_{\mu_{replace}} = \overline{T}$$

TTM matrices are square matrices that describe the time distribution it takes for an entity to sojourn from one decision node to the next. With three failure states, this will be a 3x3 matrix. Because $\overline{T}$ was calculated previously it is not necessary to develop expressions for each element of the TTM matrix for each policy, rather all row elements can be declared identical, possessing the quantity $T_{F_n} + \overline{T}$.

$$T_{\mu_{replace}} = \begin{bmatrix} T_{F_1} + \overline{T} & T_{F_1} + \overline{T} & T_{F_1} + \overline{T} \\ T_{F_2} + \overline{T} & T_{F_2} + \overline{T} & T_{F_2} + \overline{T} \\ T_{F_3} + \overline{T} & T_{F_3} + \overline{T} & T_{F_3} + \overline{T} \end{bmatrix} \quad T_{\mu_{repair}} = \begin{bmatrix} T_{F_1} + \overline{T} & T_{F_1} + \overline{T} & T_{F_1} + \overline{T} \\ T_{F_2} + \overline{T} & T_{F_2} + \overline{T} & T_{F_2} + \overline{T} \\ T_{F_3} + \overline{T} & T_{F_3} + \overline{T} & T_{F_3} + \overline{T} \end{bmatrix} \tag{3.5}$$

The same matrix for repair and replacement decisions can be used because the operation/inspection process that the machine returns to after a failure event is the same in either case; it only makes sense that both actions possess the same TTM.

Further, because sojourn time from a specific failure state to a different one it is not necessary, it is not necessary to specify each element uniquely. Only the overall average is required by the Bellman equations, therefore all the entries, regardless of originating failure state and ending failure state, are identical.

3.1.5 The Transition Probability Matrices.

The next pieces of information which is needed for the Bellman equations are the TPMs. Again symmetry between repair and replace actions will be leveraged so that the transition matrix for each action is identical. These matrices possess the transition probabilities between failure types, and again they will be 3x3 matrices. In this case the transition probabilities can be represented by the absorption probabilities from the operation/inspection process to one of the various failure states. These are analogous to the absorption probabilities of absorbing states in a SMP. Equation 3.6 describes a generic P-matrix which will describe the inspection process:

$$P = \begin{bmatrix} 0 & P_{OI} & P_{OF_1} & P_{OF_2} & \dots & P_{OF_n} \\ P_{IO} & 0 & P_{IF_1} & P_{IF_2} & \dots & P_{IF_n} \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{3.6}$$

This P-Matrix describes a generic process in which n failures can occur and no specifications are given to the frequency of failures other than the assigned probabilities. The features that are particularly descriptive are the zero probabilities assigned to the P_{OO} and P_{II} terms, indicating guaranteed transition, and the probability of 1 given to each failure state, indicating absorption; all of these elements are located on

the matrices diagonal. The model under development possess 3 failure modes, so the specific form used to calculate the absorption probabilities is:

$$P = \begin{bmatrix} 0 & P_{OI} & P_{OF_1} & P_{OF_2} & P_{OF_3} \\ P_{IO} & 0 & P_{IF_1} & P_{IF_2} & P_{IF_3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Using a process similar to Ravindran et al. [14], which uses a system of equations approach, this matrix cab be used to calculate the conditional absorption probabilities by each failure state, conditioned upon the process commencing in the operational state. Here a_{ij} will be used to denote the probability of absorption in a Markov chain which originates at node i and is absorbed by node j. This system is defined:

$$a_{ij} = P_{ij} + \sum_k P_{ik} a_{kj}, \forall k \text{ accessible from } i$$

Where a separate system of equations must be solved to determine the absorption probabilities for each different initializing state. This equation describes the probability of absorption, a_{ij}, as being the sum of the probabilities of immediate absorption by node j from node i and the sum of the absorption probabilities from other nodes, multiplied by the probability of transition to those nodes. For this scenario specific absorption probabilities for each failure state; a_{OF_1}, a_{OF_2}, and a_{OF_3}. Using the above, both a_{OF_1} and a_{IF_1} are needed to solve for a_{OF_1}:

$$a_{OF_1} = P_{OF_1} + P_{OO} a_{OF_1} + P_{OI} a_{IF_1}$$
$$a_{IF_1} = P_{IF_1} + P_{IO} a_{OF_1} + P_{II} a_{IF_1}$$

Noting that P_{OO} and P_{II} are both zero (transition is forced at the end of operation or inspection states), and substituting in for a_{IF_1} gives:

$$a_{OF_1} = P_{OF_1} + P_{OI}[P_{IF_1} + P_{IO}a_{OF_1}]$$

And with manipulation this becomes:

$$a_{OF_1} = \frac{P_{OF_1} + P_{OI}P_{IF_1}}{1 - P_{OI}P_{IO}} \tag{3.8}$$

The other absorption probabilities follow similarly:

$$a_{OF_2} = \frac{P_{OF_2} + P_{OI}P_{IF_2}}{1 - P_{OI}P_{IO}} \tag{3.9}$$

$$a_{OF_3} = \frac{P_{OF_3} + P_{OI}P_{IF_3}}{1 - P_{OI}P_{IO}} \tag{3.10}$$

The absorption probabilities beginning from the operational node should all sum to 1, which can be verified:

$$\begin{aligned}
1 &= a_{OF_1} + a_{OF_2} + a_{OF_3} \\
1 &= \frac{P_{OF_1} + P_{OI}P_{IF_1}}{1 - P_{OI}P_{IO}} + \frac{P_{OF_2} + P_{OI}P_{IF_2}}{1 - P_{OI}P_{IO}} + \frac{P_{OF_3} + P_{OI}P_{IF_3}}{1 - P_{OI}P_{IO}} \\
1 &= \frac{[P_{OF_1} + P_{OF_2} + P_{OF_3}] + P_{OI}[P_{IF_1} + P_{IF_2} + P_{IF_3}]}{1 - P_{IO}P_{OI}} \\
1 &= \frac{[1 - P_{OI}] + P_{OI}[1 - P_{IO}]}{1 - P_{OI}P_{IO}} \\
1 &= \frac{1 - P_{OI}P_{IO} + [P_{OI} - P_{OI}]}{1 - P_{OI}P_{IO}}
\end{aligned} \tag{3.11}$$

At this point, time until absorption and absorption probabilities have been developed for each failure type. These values can now be inserted into the respective TPMs .

$$P_{\mu_{replace}} = P_{\mu_{repair}} = \begin{bmatrix} \frac{P_{OF_1}+P_{OI}P_{IF_1}}{1-P_{OI}P_{IO}} & \frac{P_{OF_2}+P_{OI}P_{IF_2}}{1-P_{OI}P_{IO}} & \frac{P_{OF_3}+P_{OI}P_{IF_3}}{1-P_{OI}P_{IO}} \\ \frac{P_{OF_1}+P_{OI}P_{IF_1}}{1-P_{OI}P_{IO}} & \frac{P_{OF_2}+P_{OI}P_{IF_2}}{1-P_{OI}P_{IO}} & \frac{P_{OF_3}+P_{OI}P_{IF_3}}{1-P_{OI}P_{IO}} \\ \frac{P_{OF_1}+P_{OI}P_{IF_1}}{1-P_{OI}P_{IO}} & \frac{P_{OF_2}+P_{OI}P_{IF_2}}{1-P_{OI}P_{IO}} & \frac{P_{OF_3}+P_{OI}P_{IF_3}}{1-P_{OI}P_{IO}} \end{bmatrix} \tag{3.12}$$

Again, the symmetry of the situation has been taken advantage of and the transition probabilities are identical between failure states. This is again explained by the choice of only using one operational state and the convention of good-as-new repairs, which forces identical failure probability distributions after each failure event, regardless of action chosen.

3.1.6 The Transition Reward Matrices.

The TRMs which define this problem will be very similar to the TTMs, but now the matrices are different between the policies. This is due to the cost/reward of the action chosen being the only quantity associated with a decision that can be affected by the decision. Since the Bellman equations (3.1) only require an average reward for each state/policy pair, each row of the TRM for the repair policy will hold the same value, the repair cost induced from that node. Similarly for the TRM which describes the replacement policy, every entry in it will be identical since the cost to replace after any failure will always be the same.

When only considering a replacement decision versus one inspection (not several competing processes), the ongoing cost of the inspection process does not need to be considered in the construction of the TRMs for each policy. Because the cost imposed from the inspection process upon each element of the repair policy matrix is the same

as the cost imposed by the inspection process to the same node in the replace policy matrix, the cost/benefit of inspection process will be nullified. In light of this, fully defined TRMs take the form:

$$R_{\mu_{repair}} = \begin{bmatrix} C_{F_1} & C_{F_1} & C_{F_1} \\ C_{F_2} & C_{F_2} & C_{F_2} \\ C_{F_3} & C_{F_3} & C_{F_3} \end{bmatrix} \qquad R_{\mu_{replace}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.13}$$

Where the various repair costs associated with each node, denoted C_{F_n} are operator specified, and the replacement has a cost of 1 in order to emphasize the relative nature of the comparisons. Repair costs should be scaled appropriately to the replacement cost.

3.1.7 Inspection Process Comparisons.

In order to compare different inspection schemes to determine which one delivers the most value, every TTM, TPM, and TRM require adjustment to account for the differences which altering inspection process will impose. The construction of the inspection process is user defined and can take any form which one wishes to investigate; however, the conventions of exponential interevent times, good-as-new repairs, and the general inspection process format need to be preserved in order for these calculations to hold.

The simplest adjustment to be made is the TTM. Since the underlying nature of the process is unchanged, so it the process to calculate $\overline{T}$. The same series process is still applicable, but it requires recalculations for each process that one wishes to examine. As long as regular recurring interval lengths are consistently used, no adaptation will be required. If one wishes to use another type of inspection scheme, that method would need to be developed separately.

The next adjustments to be made are to the TPMs. The adjustments to these matrices can take a variety of forms, and the method used to change the TPMs warrants careful consideration. It is desirable that, when changes are made to the TPMs, the changes made do not imply an entirely different machine is undergoing the inspection process, only that the process is different.

Unfortunately accomplishing this in a completely uncompromised manner is very difficult. The method chosen to ensure that competing processes preserve machine characteristics is to check for consistency in the time normalized cyclic rate of failures between processes. This means that for every operation/inspection cycle, the number of expected failures of any given type should be consistent on a per unit time basis. The following steps are proposed to construct competing processes. The steps are designed for the comparison of two candidate process, with expansion to a greater number of processes requiring analogous steps.

1. First, with original process, calculate the percentage of surviving machines after one operation/inspection cycle. Using the notation from equation [3.7], it is clear that this is equal to

$$S_{\text{cycle}} = P_{OI}P_{IO}$$

   Where S_{cycle} represents the survival percentage.

2. The normalized percentage of failures expected to be observed per cycle in the new process should be the product of the cycle survival percentage from the original process multiplied by ratio of process cycle lengths. (note quantities corresponding to the new cycle are denoted as primes)

$$S'_{\text{cycle}} = P_{OI}P_{IO}\frac{T'_O + T'_I}{T_O + T_I} = P'_{OI}P'_{IO} \tag{3.14}$$

This equation is underdetermined since it has 2 unknowns. This leaves it open to the discretion of the operator to determine the relative efficiency of the new inspection process. To simulate a rigorous inspection process, proportionally more failures should be discovered during inspection than occur during operation. In contrast, in a less rigorous inspection process most failures should occur from operation, with relatively few found during inspection.

3. Next, an analogous procedure should be performed for the failure blends. Failure blends have already been calculated for the original distribution, called absorption probabilities (equation (3.8) and (3.9)). These absorption probabilities should be the same for the new process as it is for the legacy process, since both processes are applying to the same machine. This part of constructing the new process requires subjective judgment on the part of the operator. It is required that:

$$\begin{aligned} a_{OF_n} &= \frac{P_{OF_n} + P_{OI}P_{IF_n}}{1 - P_{OI}P_{IO}}, \forall n \\ &= \frac{P'_{OF_n} + P'_{OI}P'_{IF_n}}{1 - P'_{OI}P'_{IO}}, \forall n \end{aligned}$$

The only term known in this equation is the absorption probability from the original inspection process. This system has n equations and $2n$ unknowns. 2 unknowns can be removed since it is know that

$$\begin{aligned} P_{OI} &= 1 - (P_{IF_1} + P_{IF_2} + \ldots + P_{IF_n}) \\ P_{IO} &= 1 - (P_{OF_1} + P_{OF_2} + \ldots + P_{OF_n}) \end{aligned}$$

However this still leaves an underdetermined system in all cases except $n = 2$. This again will require the expertise of the operator to construct realistic failure

probabilities that conserve the essential properties of the inspection process and ensure the different processes are describing the same machine.

Notice that keeping the total proportion of failures observed from operation/ inspection consistent between different processes is not desired, as this is self defeating. If this were done, outside of manipulating the costs of the processes, there would be no way distinguish between the failure discovery efficiency of the different inspection processes being considered. For instance, if minor failures are not of any concern during inspection their probabilities can be set to zero. Likewise, if major failures are not allowed to happen during inspection, that probability could also be set to zero. Simply stated for this model: $P_{IF_1} = 0$ and $P_{IF_3} = 0$. Under these assumptions, it is possible to construct an inspection process which only searches for the medium level failures, even though all types of failures may be considered in the competing inspection schemes. This may lead to a very different percentage of failures contributed from operation than from inspection. This would in turn drive higher implicit error rate ($\lambda_{\text{implicit}}$) during the operational period in the process. However, it is conceivable that the tradeoff could be productive one, since proportionally more of those failures would be failures of little consequence.

Though the process just outlined is not without its flaws, it arguably is the best approach to ensure consistency between proposed competing inspection processes. A demonstration of this method will be given in chapter 4 to illuminate the process. These quantities compose the identity, or fingerprint, of the machine, and keeping them consistent between processes ensures similar machines are being compared under different inspection schedules.

The cost structure of the inspection process could assume many forms. Table 4 describes the elements for this process.

Table 4. Inspection Cost Structure Elements

Notation	Element Name	Description
C_I	Total inspection cost	The total cost incurred per visit to inspection node
C_M	Inspection material cost	The material cost incurred per visit to inspection node
C_{B_n}	Cost bonus for inspection failure discovery	Incentivizes failure discovery while it is advantageous
C_D	Cost of downtime due to inspection	The product of the machine's "production rate" and the length of inspection period, opportunity cost

With this structure in mind, costs or benefits, $\overline{r_\mu}(i)$, of the various inspection processes can be calculated. Because the Bellman equations only require an average reward for a policy and state, and not distinct quantities for each possible transition, $\overline{r_\mu}(i)$ can calculated as follows for this process:

$$\begin{aligned}\overline{r_\mu}(i) = &- (\text{cost of repair i}) \\ &- (\text{Average num. of inspections})(\text{Cost of inspection} + \text{Cost of downtime}) \\ &+ a_{OF_1} C_{B_1} P(\text{inspection}|F_1) \\ &+ a_{OF_2} C_{B_2} P(\text{inspection}|F_2) + \dots \\ &+ a_{OF_n} C_{B_n} P(\text{inspection}|F_n)\end{aligned} \tag{3.15}$$

Where the grouping of the last three terms, the absorption probability times the cost of that failure times the likelihood that the failure of that type was discovered during an inspection, describes the average bonus yielded by the inspection process due to its efficiency in discovering failures. The conditional probability term can be derived

as follows:

$$P(\text{inspection}|F_i) = \frac{P(F_i|\text{inspection})P(\text{inspection})}{P(F_i|\text{inspection})P(\text{inspection}) + P(F_i|\text{operational})P(\text{operational}))}$$
$$= \frac{P_{IF_i}\frac{P_{IF_i}}{P_{IF_i}+P_{OF_i}}}{P_{IF_i}\frac{P_{IF_i}}{P_{IF_i}+P_{OF_i}} + P_{OF_i}\frac{P_{OF_i}}{P_{IF_i}+P_{OF_i}}}$$

The check of this conditional probability can be performed and validated in the same manner as it was for the absorbing probabilities, earlier discovered in equation (3.11). Further, the average cost of the inspection process is identical for each failure type and for replacement and repair actions, so the net result is that the terms of the TRM only need modification by the addition of the term C_I, the cost of inspection, which is expressed as:

$$C_I = \frac{\overline{N}}{2}(C_D + C_M) + \sum_{i=1}^{n} a_{OF_i} C_{B_i} P(\text{inspection}|F_i) \ \forall \text{ states } n$$

With these developments the relevant TTMs, TPMs, and TRMs can now be described that compare two or more inspection processes for the 3 node system. The TTMs are very similar:

$$T_{\mu_{replace}} = T_{\mu_{repair}} = \begin{bmatrix} T_{F_1} + \overline{T} & T_{F_1} + \overline{T} & T_{F_1} + \overline{T} \\ T_{F_2} + \overline{T} & T_{F_2} + \overline{T} & T_{F_2} + \overline{T} \\ T_{F_3} + \overline{T} & T_{F_3} + \overline{T} & T_{F_3} + \overline{T} \end{bmatrix}$$
$$T'_{\mu_{replace}} = T'_{\mu_{repair}} = \begin{bmatrix} T_{F_1} + \overline{T}' & T_{F_1} + \overline{T}' & T_{F_1} + \overline{T}' \\ T_{F_2} + \overline{T}' & T_{F_2} + \overline{T}' & T_{F_2} + \overline{T}' \\ T_{F_3} + \overline{T}' & T_{F_3} + \overline{T}' & T_{F_3} + \overline{T}' \end{bmatrix}$$

The TPMs also have the same form as they did in Equation [(3.12)], but will be populated with quantities unique to the new process. The TPM is displayed

again here, but to avoid visual frustration, only the equations corresponding to the new system, whose transition probabilities were developed following the guidelines previously discussed, is displayed.

$$P'_{\mu_{replace}} = P'_{\mu_{repair}} = \begin{bmatrix} \frac{P'_{OF_1}+P'_{OI}P'_{IF_1}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_2}+P'_{OI}P'_{IF_2}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_3}+P'_{OI}P'_{IF_3}}{1-P'_{OI}P'_{IO}} \\ \frac{P'_{OF_1}+P'_{OI}P'_{IF_1}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_2}+P'_{OI}P'_{IF_2}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_3}+P'_{OI}P'_{IF_3}}{1-P'_{OI}P'_{IO}} \\ \frac{P'_{OF_1}+P'_{OI}P'_{IF_1}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_2}+P'_{OI}P'_{IF_2}}{1-P'_{OI}P'_{IO}} & \frac{P'_{OF_3}+P'_{OI}P'_{IF_3}}{1-P'_{OI}P'_{IO}} \end{bmatrix}$$

Finally the recalculation of the TRMs under differing inspection schemes needs to be performed. As noted, the change required is simply the addition of the appropriate C_I or C'_I term.

$$R_{\mu_{repair}} = \begin{bmatrix} C_{F_1}+C_I & C_{F_1}+C_I & C_{F_1}+C_I \\ C_{F_2}+C_I & C_{F_2}+C_I & C_{F_2}+C_I \\ C_{F_3}+C_I & C_{F_3}+C_I & C_{F_3}+C_I \end{bmatrix} \quad R_{\mu_{replace}} = \begin{bmatrix} 1+C_I & 1+C_I & 1+C_I \\ 1+C_I & 1+C_I & 1+C_I \\ 1+C_I & 1+C_I & 1+C_I \end{bmatrix}$$

$$R'_{\mu_{repair}} = \begin{bmatrix} C_{F_1}+C'_I & C_{F_1}+C'_I & C_{F_1}+C'_I \\ C_{F_2}+C'_I & C_{F_2}+C'_I & C_{F_2}+C'_I \\ C_{F_3}+C'_I & C_{F_3}+C'_I & C_{F_3}+C'_I \end{bmatrix} \quad R'_{\mu_{replace}} = \begin{bmatrix} 1+C'_I & 1+C'_I & 1+C'_I \\ 1+C'_I & 1+C'_I & 1+C'_I \\ 1+C'_I & 1+C'_I & 1+C'_I \end{bmatrix}$$

It is worth noting, that whenever another inspection process is added for comparison, the number of matrices required to define the system increases by 6. This pattern would continue for every additional inspection scheme one wished to test; 3 matrices are required to describe repairs for each inspection process, and 3 are required to describe replacements for each inspection process. Also, the parameters pertaining to the machine when additional inspection processes are introduced remain unchanged

(i.e. costs of failures, times of failures, replacement costs, etc). This is done to ensure that the machines are identical, and that the effects of the different inspection processes are isolated.

With these quantities and matrices it is possible to determine optimal inspection processes and maintenance actions for a properly specified machine using MDP policy iteration.

3.2 Model Exploration and Process

Numerous factors play into the values that are selected for the various parameter levels. Certainly the situation/machine being investigated is the primary driver, but understanding the selected parameter levels and the implications those levels carry is vital to achieving useful results. Further understanding the tools selected to implement the method described is also a necessary to the integrity of the results

3.2.1 Cost Structures.

There are seven costs which need to be declared before the model can be completely characterized. They are:

- Cost of each repair type, C_{F_n}, $n = 1, 2, 3$.
- Cost of replacement, 1 unit cost
- Cost of inspection, C_I
- Production rate per inspection period, V
- Inspection discovery reward, C_B

Since the primary interest is on relative effects, all costs are stated relative to the cost of replacement, 1. This is an intuitive place to be able to draw conclusions from, and will impart some mandatory structure to the costs.

The cost of the various failure types will be between 0 to values greater than 1. Though several values will be used, small failures (type 1) are intended to represent an event which is inconvenient, but not debilitating. Values for this will range from .01 to .1. Moderate failures (type 2) are failures which represent an event of meaningful significance. Values for this could be thought of from .1 to less than 1. Catastrophic failures (type 3) are always meant to result in replacement. These are the major accidents or mechanical failures.

The cost of inspections offers an interesting question of interpretation because it needs to accurately capture a couple effects. First, it is reasonable to assume that an entity under inspection is not performing its designed task, so there should be a cost of downtime included in the cost of an inspection. Also, a failure discovered during an inspection potentially does not result in as much unscheduled downtime, and therefore should receive a benefit relative to a failure during operation. For this reason as described in Table 4, the cost of inspection encompasses more than one element, and defining the cost of inspection really requires defining the perceived importance one wishes to place on the value of the machine operating. For this study, the C_B elements were all given a value of one half the cost of a failure. This was done because if a failure was discovered during an inspection, the machine would likely already be in a disassembled state for the inspection, and hence ready to expediently process the necessary repair. Therefore, it is estimated that this would result in saving equal to roughly half the value of the same failure if it had occurred from operation.

Finally, the production rate of the asset is intended to reflect how useful the asset is perceived to be. This is best illustrated via examples. Consider the example of taxi

cab; the cab itself may be worth, say, $10,000, but generates $120,000 in fares over the course of an annual inspection cycle. This would give a production rate of 12. Next, consider a rental house with a value of 100,000 USD. Using the rule of thumb that the monthly cost of a house is about one percent of the purchase price (in a normal interest rate environment), and allowing for 10% profit margin, the production rate of the house would be $.01 * 12 * 1.1 = .132$ per annual inspection period. Though these are rough estimates, they illustrate the effect and scope the production rate term is intended to capture.

3.2.2 Probability Structures.

Assigning probabilities to the various transitions results in a number of interesting tradeoffs. Discussion of the effects of this were covered in section 3.1.7, which developed the TPMs for competing inspection regimes. The constituent probabilistic elements required to be declared for this process are:

- Probability of failure during operation for each failure type, P_{OF_n}
- Probability of failure discovery during inspection for each failure type, P_{IF_n}

When examining competitive inspection processes for an asset under study, the probabilities necessary to ensure total consistency between machine identity results in an inherently underdetermined system of linear equations. This allows for great flexibility in constructing the consequences of the inspection process to fit the requirements of the scenario under study.

The guidelines for assigning probabilities to these elements should be weighed against the cost and time which each failure type incurs. For realism, catastrophic type failures will not be paired with high probabilities.

3.2.3 Time Structures.

The time element of an SMDP is what makes the process so relevant to so many applications. For this scenario, the following parameters concern time:

- Operational period length, T_O
- Inspection period length, T_I
- Downtime presumed for each failure type, T_{F_n}
- Production rate, V, considered as a function of time.

These elements are the fewest in number out of all the cost, probability and time categories , but the effect that time can have on the rate of cost flow is what is of primary concern to someone managing a asset. Again the same considerations that were observed with cost and probability parameters should be observed here, and the design of these elements is largely free formed to the specification of the user.

3.2.4 Simulation Process.

The simulation process uses three overarching steps. First, after all of the required parameters are specified, matrices for each TPM, TTM, and TRM need to be constructed. Next, these matrices need to be applied to the policy iteration algorithm. Finally, after a policy is assigned, the resulting semi Markov process then needs to be simulated to characterize the time and expense rate and availability CDFs.

This first step in calculating the matrices from the assigned parameters takes place in a computer spreadsheet. Using the assigned parameters the spreadsheet automatically populates the matrices with the appropriate values as derived. Microsoft Excel was used in this step, which, aside from being accessible, is easily formatted and useful for verifying the correctness of the resulting matrix.

The next step uses a C-program designed by Gosavi [6]. C is a good choice not only for this application but also more broadly for other SMDPs that one may wish to construct, since compiling C code results in an executable file which runs very fast without the overhead of other programs. Due to the large sizes of the systems of equations that result when solving SMDPs, and the numerous computations entailed in solving them necessitates a code that runs with minimal impedance. The code was edited in the NetBeans® Integrated Development Environment (IDE) version 7.0.1, and was run from both the NetBeans® console and the command line in Windows 7®. The output of this code results in an optimal action assigned to each failure state, and also calculates the long run average reward of adhering to that policy relative to the node which was set to zero to initialize the computation.

The final step of simulation takes place in a Java program designed specifically for the policy that results from the C-programs policy iteration. The java simulation was also written within the NetBeans® IDE, and ran from the same. The "object-oriented" nature of Java made well suited to the nature of the problem. The console output of the simulation was captured and stored in Excel spreadsheets to be sorted and analyzed later.

3.2.5 Scope and Exploration of Models.

The primary consideration in the design and construction of is selecting parameter levels of the design factors at which there is thought to be a reasonable expectation of interesting or revealing behavior. In particular, the levels of cost and rewards and their associated probabilities profoundly affect the behavior and output of the SMDP. The process described in the "Simulation Process" subsection will be run four different times, representing four different machines. Each run will compare two competing inspection processes; one process being a more effective but less frequent

inspection, and the other being a less effective, but more frequent inspection. The four different machines that will be examined will be:

1. A highly productive machine that has low cost of repairs and low probability of failure.

2. A highly productive machine that has high cost of repairs and high probability of failure.

3. A less productive machine that has low cost of repairs and low probability of failure.

4. A less productive machine that has high cost of repairs and high probability of failure.

The motivation is to observe the effects of highly productive and less productive machines separately since those quantities both effect the "reward" side of the process, and in this case indirect costs; but consider repair probability and cost together since both of those quantities affect the "cost" side of the process.

IV. Implementation and Analysis

This chapter will deal with the implementation, results, and discussion of the procedure outlined in Chapter III. The construction of this chapter largely follows the procedure previously described in the same chapter.

4.1 Matrix Construction

The first step in developing an inspection policy and simulating the results is characterizing the necessary matrices. This is accomplished in Microsoft Excel®, via a spreadsheet expressly designed for this purpose. Figure 7 to Figure 14 contain the matrices for machines 1-4 and the two inspection processes discussed in Chapter III. Each of the machines were given names that might help identify the real world application in order to aid in identification. Parameters are not based on any physical observations.

The first two figures represent machine 1, the highly productive machine with low maintenance costs, a "generator." The ratio of short to long inspection is four to one. The next 2 figures represent Machine 2, the highly productive, high maintenance machine, the "NYC taxi." Short to long inspection ratio is twelve to one. Machine 3, the "solar panel," is a less productive machine, but with low maintenance costs. Its short process to long process ratio is also twelve to one. Finally, Machine 4 is a lower productivity machine that has high maintenance requirements, the "machine in the back of the shop." It has a short to long process ratio of four to one as well.

The entries that populate these matrices are calculated in accordance with the process described in Chapter III. The policy iteration algorithm for the determination of repair/replace and inspection policy will take 12 matrices per machine, 6 for each inspection process. This results in the 48 matrices defined in the figures.

4.2 Policy Iteration Results

The next step in the MDP process is running the policy iteration algorithm using the matrices defined for each machine. The results of policy iteration are captured in Table 5. As noted in the methodology chapter, the symmetry present helps policy

Table 5. Policy Iteration Results

Machine	1	2	3	4
Iterations Needed	2	3	2	2
Optimal value for failure A	0	0	0	0
Optimal value for failure B	-0.5412	-0.4976	-0.09921	-0.1652
Optimal value for failure C	-0.7413	-0.7414	-0.8576	-0.8616
Optimal policy for failure A	repair, long inspection			
Optimal policy for failure B	repair, long inspection			
Optimal policy for failure C	replace, long inspection			

iteration achieve convergence in minimal iterations, a maximum of only 3 in these cases. This was aided by the fact that the initial policy was set to choose "repair, long process" for each failure state, for all machines; this means only one change was needed from the initial policy in all cases. It was also observed, as designed, that the average reward for the chosen policy improved with each iteration; this indicated the algorithm was converging to a solution. Lastly, as assigned, the value for failure type A is always 0, and the relative values for type B and C are present as well. Here again, type C failures represents a considerably steeper penalty to the system than either state A or B, which makes sense in light of the fact that type C failures represent the most severe failure.

4.3 Simulation Results and CDFs

Performing a simulation of the prescribed optimal policy using Java, which is repair, repair, replace, using a long inspection process for all machines, gives similar results for all machines, as should be expected. The shape of the resulting life time

CDFs is decidedly exponential in nature, while the features of the expense rate and availability CDFs have more interesting features. Figure 15 to Figure 18, contain the results of these simulations; each displays life time, expense rate, and availability CDFs.

4.4 Discussion

The results revealed some behavior that was slightly unexpected and some that was in line with expectations. By one token the strictly exponential form of lifetime distributions is intuitive because the process is composed of all exponential distributions. However, by another token it is also surprising that the lifetime distributions did not tend to a mound shaped distribution with a clearly defined mode, as might be expected from the "averaging out" effects of the ongoing processes. This should be interpreted as indicative of the types of phenomena that this formalism is appropriate to model, and not a failure in design.

Mound shaped distributions are present in the expense rate and availability CDFs, though for all but Machine 1, the mound shape is relatively abrupt and near the origin when considered against how long the tails of the same distributions are (see Figures 15 thru 18). The nearness of the mode to the origin is explained by the specific combination of factors generating the CDF. The primary driver of this effect is the relative stratification of how the different machines fail from operation and inspection and the relative difference in those cost penalties, but the exact nature of the relationship is difficult, if not impossible, to specify. Time, production rate, failre costs, failure times, and failure probabilities for the various machines can be found in Figures 7 to 14.

While both expense rate CDFs and availability CDFs are indeed mound shaped, there are critical differences represented in their behavior. In the expense rate CDFs,

there is a theoretically limitless expense rate on the negative side, and extreme outliers are possible because of this. In fact, in all of the expense CDFs provided the data displayed only represents 99.9% of the 10000 samples observed. This is due to the existence of those extreme outliers and their skewing effect. These high expense rates would be caused by a quick type C failure resulting in replacement before the system had enough time to moderate out. This is only present on very few observations, but none-the-less is a feature of the tail to be aware of. The upper limit of the expense CDF is also bounded below zero by the cost of inspection and downtime. This might seem contradictory when later it is observed that inspection processes can add value, but this is only because the processes discover failures, which will inevitably induce costs greater than the benefit. The upper limit of the expense rate distribution can be shown to be equal to $\frac{-(C_D+C_I)}{T_O+T_I}$ by observing that a long lived machine which experiences no failures is the least cost inducing way a machine could exist, but even this machine will incur inspection costs every cycle. It may appear strange that the expense rate is always negative. This is an artifact of the choices made when the model was constructed. When it was decided to only count costs related to inspection and not from operation, it was implicitly decided that the reward of operational time would be zero. This simply means downtime is counted as opportunity cost. It would be equally valid to reward operational time and not count the opportunity cost of downtime; however, including both effects would result in double counting, and the results of such a simulation would not be erroneous.

The significant feature in the availability CDFs is the cluster of points with availability 1. This effect is not a testament to the superiority of the machine, but rather is due to the same effect which causes the extreme outliers in the expense rate CDFs, type C failures occurring before an inspection period. Therefore, "perfect availability" should be viewed negatively, and the natural threshold of availability should be

thought to be the point just before the series of availability one is observed. Unlike expense rate this upper limit is not clearly defined. However, the availability of an error free machine will be $\frac{T_I}{T_O+T_I}$, so it is prudent to view availabilities much beyond this value with some suspicion.

It is also interesting how availability is affected by longer machine lifetimes. Using only the first 1000 runs of our simulation (so as not to over populate the graph) the relationship seen in Figure 19 is observed. The tightening of the availability distribution with the passage of time is not an unexpected effect, and can provide an operator with another metric by which to evaluate their machine. This effect is present in all of the machines simulated; Machine 2 was chosen for this particular example due to its usefulness in illustrating this effect. Another interesting effect found in the same graph is the embedded failure boundaries displayed in Figure 20. This figure is a magnified portion of Figure 19 and the absorption boundaries within the main cluster of the graph. These formations are caused by specific patterns of failures before absorption, with the strongest series being that of one type A failure before being replaced, and the next most prominent being that of one type B failure before being replaced. Several other series are also visible in the graph, and each of them represents a specific path to replacement, with movement along the series caused by the different times until replacement when a machine follows that path. The graphs displayed also uses Machine 2's results for clarity.

A similar effect can be seen on the graph of Machine 2 when expense rate is plotted against availability (Figure 21). This time the lines are linear, and the same pattern holds; the most prominent series is the one type A failure and the next in prominence is the one type B failure and so on.

An interesting feature present in both of these figures is that there is a probabilistic like limit inherent in the display. The lower boundary of both graphs contains points

of varied failures of no especially discernible pattern other than they all have roughly the same failure type ratios (for a given machine). All of these points represent machines that have experienced many failures and repairs without being replaced. In both cases, Figure 19 and Figure 21, the blend of failures and number of failures which compose this lower limit are functions of the machines' parameters. In the case of availability against age (Figure 19)the lower limit is undesirable, but could be thought of as a minimum expectation for availability. The relevant information conveyed by the expense rate against availability graph is that even outside of the first operational period replacements, higher expense rates are correlated with lower availabilities, again confirming and emphasizing the healthy region of expense and availability expectation.

Finally selecting the optimum inspection interval and replacement failure events for all four machines seemed to be a very predictable process since all four machines selected identical policies. It can be observed from the construction of the MDP matrices, that in every case the algorithm is simply choosing the lesser of the cost of replacement, or the cost of the repair plus the associated downtime. The choice of which inspection scheme to use is independent of the repair/replace decision since all TRMs for a given inspection process add identical quantities to the TRMs associated with their repair and replace policies. This can be demonstrated by considering a machine identical to Machine 1 (Figure 7) except for assigning it a productivity rate of 25. When this machine is run through the policy iteration algorithm the result in Table 6 is obtained. With the added productivity, it is now more beneficial to replace the machine after a type B failure than to endure repair. The TPMs and TTMs do not play a role in the comparison since neither of them are affected by the change in productivity rate, so it is safe to conclude that this effect is isolated to the TRM. Further, in the event that changes were applied to the failure times, these changes

Table 6. Policy Iteration Results for Highly Productive Machine 1

Iterations Needed	2
Optimal policy for failure A	repair, long inspection
Optimal policy for failure B	replace, long inspection
Optimal policy for failure C	replace, long inspection
Optimal value for failure A	0
Optimal value for failure B	-0.7189
Optimal value for failure C	-0.5478

would impact matrices for both repair and replacement in the same manner, so the effect would be nullified.

Similarly, the choice optimum inspection process occurs independently of whether to repair or replace since the costs and downtime penalties of those actions will also be identical between inspection processes. The cost of the inspection process per transition in this scenario was defined as:

$$C_I = \frac{\overline{N}}{2}(C_D + C_M) + \sum_{i=1}^{n} a_{OF_i} C_{B_i} P(\text{inspection}|F_i) \ \forall \text{ states } n$$

The policy iteration algorithm then simply chooses the inspection process which results in the lowest anticipated cost per transition between failures; this term is appended to every element of the TRMs for each inspection process. This can also be verified by observing Table 7 where, for all four machines, the inspection cost penalty for the long process was less than that of the short process. Similar to the way Machine 1 was manipulated to affect its repair policy, Machine 3 can be manipulated to prefer a short inspection process to a long process by setting the cost of the short in-

Table 7. Cost of Inspection Cycles

Machine	Long Process Cost	Short Process Cost
Machine 1	-0.6767	-1.457
Machine 2	.05885	-0.642
Machine 3	.01545	-0.6271
Machine 4	.04387	-0.579

spection to zero. In this case the mean inspection reward for the long process remains the same, while the mean inspection reward for the short process becomes .02354, which is indeed greater than the mean reward for the long process. Policy iteration gives the results shown in Table 8 after this is done.

Table 8. Policy Iteration Results for Altered Machine 3

Iterations Needed	3
Optimal policy for failure A	repair, short inspection
Optimal policy for failure B	repair, short inspection
Optimal policy for failure C	replace, short inspection
Optimal value for failure A	0
Optimal value for failure B	-0.1012
Optimal value for failure C	-0.875947

Under this inspection cost paradigm it is misguided to make blanket statements about the relative worth of short or long period inspection processes. Given two competitive processes which one an MDP would choose can be determined by observing the cost of an inspection in the absence of the other information by calculating the anticipated cost of one inspection cycle. However, this still requires the operator to specify machine and inspection process parameters completely because of the computation involved in the cost of an inspection. Similarly the repair/replace decisions can be determined by calculating the cost of repair plus downtime and comparing that to the cost of replacement. This determination is simpler than the inspection determination since it can be performed without knowledge of the inspection process.

These results are intuitive and give insight to the process the MDP is performing. Though no concrete specifications or global recommendations can be made from these results about inherently superior process designs, the method demonstrated its ability to achieve actionable results across a variety of parameter configurations.

Parameter	"Generator"
replacement cost	-1
production rate	10
operational period length	0.916667
inspection length	0.083333
P(operational failure a)	0.05
P(operational failure b)	0.1
P(operational failure c)	0.01
Cum. Operational failures	0.16
P(inspection failure a)	0.1
P(inspection failure b)	0.2
P(inspection failure c)	0
Cum. Inspection failures	0.3
inspection cost	-0.05
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.83333
failure a cost	-0.15479
failure b cost	-0.71689
failure c cost	-3.66667
failure a time	0.005479
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.3252427	0.6504854	0.024272
b	0.3252427	0.6504854	0.024272
c	0.3252427	0.6504854	0.024272

TRM

	a	b	c
a	-0.831458	-0.8314576	-0.83146
b	-1.393558	-1.3935581	-1.39356
c	-4.34333	-4.3433298	-4.34333

TTM

	a	b	c
a	1.8578228	1.8578228	1.857823
b	1.8906995	1.8906995	1.890699
c	2.01901	2.01901	2.01901

Replace Action:

TPM

	a	b	c
a	0.325243	0.650485	0.024272
b	0.325243	0.650485	0.024272
c	0.325243	0.650485	0.024272

TRM

	a	b	c
a	-1.67666	-1.67666	-1.67666
b	-1.67666	-1.67666	-1.67666
c	-1.67666	-1.67666	-1.67666

TTM

	a	b	c
a	1.857823	1.857823	1.857823
b	1.890699	1.890699	1.890699
c	2.01901	2.01901	2.01901

Failure Blends

	operation	inspection	check	% of total failures
a	0.2	0.8	1	0.325242718
b	0.2	0.8	1	0.650485437
c	1	0	1	0.024271845
				1

Figure 7. Machine 1, "Generator," Long Process Matrices.

Parameter	"Generator"
replacement cost	-1
production rate	10
operational period length	0.229167
inspection length	0.020833
P(operational failure a)	0.02
P(operational failure b)	0.04
P(operational failure c)	0.0025
Cum. Operational failures	0.0625
P(inspection failure a)	0.015
P(inspection failure b)	0.03
P(inspection failure c)	0
Cum. Inspection failures	0.045
inspection cost	-0.05
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.20833
failure a cost	-0.15479
failure b cost	-0.71689
failure c cost	-3.66667
failure a time	0.005479
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.3253731	0.6507463	0.023881
b	0.3253731	0.6507463	0.023881
c	0.3253731	0.6507463	0.023881

TRM

	a	b	c
a	-1.611756	-1.6117556	-1.61176
b	-2.173856	-2.1738561	-2.17386
c	-5.123628	-5.1236278	-5.12363

TTM

	a	b	c
a	1.6107708	1.6107708	1.610771
b	1.6436475	1.6436475	1.643647
c	1.771958	1.771958	1.771958

Replace Action:

TPM

	a	b	c
a	0.325373	0.650746	0.023881
b	0.325373	0.650746	0.023881
c	0.325373	0.650746	0.023881

TRM

	a	b	c
a	-2.45696	-2.45696	-2.45696
b	-2.45696	-2.45696	-2.45696
c	-2.45696	-2.45696	-2.45696

TTM

	a	b	c
a	1.610771	1.610771	1.610771
b	1.643647	1.643647	1.643647
c	1.771958	1.771958	1.771958

Failure Blends

	operation	inspection	check	% of total failures	original failure blend
a	0.64	0.36	1	0.325373134	0.325243
b	0.64	0.36	1	0.650746269	0.650485
c	1	0	1	0.023880597	0.024272
				1	

Figure 8. Machine 1, "Generator," Short Process Matrices.

Parameter	"NYC-Taxi"
replacement cost	-1
production rate	10
operational period length	1
inspection length	0.038356
P(operational failure a)	0.333333
P(operational failure b)	0.333333
P(operational failure c)	0.166667
Cum. Operational failures	0.833333
P(inspection failure a)	0.4
P(inspection failure b)	0.25
P(inspection failure c)	0
Cum. Inspection failures	0.65
inspection cost	-0.05
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.38356
failure a cost	-0.10719
failure b cost	-0.63356
failure c cost	-3.66667
failure a time	0.008219
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.4247788	0.3982301	0.176991
b	0.4247788	0.3982301	0.176991
c	0.4247788	0.3982301	0.176991

TRM

	a	b	c
a	-0.048341	-0.0483413	-0.04834
b	-0.574711	-0.5747111	-0.57471
c	-3.607816	-3.6078162	-3.60782

TTM

	a	b	c
a	0.395072	0.395072	0.395072
b	0.425209	0.425209	0.425209
c	0.5535195	0.5535195	0.553519

Replace Action:

TPM

	a	b	c
a	0.424779	0.39823	0.176991
b	0.424779	0.39823	0.176991
c	0.424779	0.39823	0.176991

TRM

	a	b	c
a	-0.94115	-0.94115	-0.94115
b	-0.94115	-0.94115	-0.94115
c	-0.94115	-0.94115	-0.94115

TTM

	a	b	c
a	0.395072	0.395072	0.395072
b	0.425209	0.425209	0.425209
c	0.553519	0.553519	0.553519

Failure Blends

	operation	inspection	check	% of total failures
a	0.4098361	0.5901639	1	0.424778761
b	0.64	0.36	1	0.398230088
c	1	0	1	0.17699115
				1

Figure 9. Machine 2, "NYC Taxi," Long Process Matrices.

Parameter	"NYC-Taxi"
replacement cost	-1
production rate	10
operational period length	0.1
inspection length	0.01
P(operational failure a)	0.03
P(operational failure b)	0.03
P(operational failure c)	0.0175
Cum. Operational failures	0.0775
P(inspection failure a)	0.015
P(inspection failure b)	0.01
P(inspection failure c)	0
Cum. Inspection failures	0.025
inspection cost	-0.01
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.1
failure a cost	-0.10719
failure b cost	-0.63356
failure c cost	-3.66667
failure a time	0.008219
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.4359229	0.3900559	0.174021
b	0.4359229	0.3900559	0.174021
c	0.4359229	0.3900559	0.174021

TRM

	a	b	c
a	-0.750163	-0.7501628	-0.75016
b	-1.276533	-1.2765327	-1.27653
c	-4.309638	-4.3096377	-4.30964

TTM

	a	b	c
a	0.739168	0.739168	0.739168
b	0.769305	0.769305	0.769305
c	0.8976155	0.8976155	0.897615

Replace Action:

TPM

	a	b	c
a	0.435923	0.390056	0.174021
b	0.435923	0.390056	0.174021
c	0.435923	0.390056	0.174021

TRM

	a	b	c
a	-1.64297	-1.64297	-1.64297
b	-1.64297	-1.64297	-1.64297
c	-1.64297	-1.64297	-1.64297

TTM

	a	b	c
a	0.739168	0.739168	0.739168
b	0.769305	0.769305	0.769305
c	0.897615	0.897615	0.897615

Failure Blends

	operation	inspection	check	% of total failures	original failure blend
a	0.8	0.2	1	0.435922933	0.424779
b	0.9	0.1	1	0.390055935	0.39823
c	1	0	1	0.174021131	0.176991
				1	

Figure 10. Machine 2, "NYC Taxi," Short Process Matrices.

Parameter	"Solar Panel"
replacement cost	-1
production rate	0.25
operational period length	0.994521
inspection length	0.005479
P(operational failure a)	0.05
P(operational failure b)	0.1
P(operational failure c)	0.025
Cum. Operational failures	0.175
P(inspection failure a)	0.25
P(inspection failure b)	0.25
P(inspection failure c)	0
Cum. Inspection failures	0.5
inspection cost	-0.05
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.00137
failure a cost	-0.100685
failure b cost	-0.203425
failure c cost	-2.041667
failure a time	0.00274
failure b time	0.013699
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.4361702	0.5212766	0.042553
b	0.4361702	0.5212766	0.042553
c	0.4361702	0.5212766	0.042553

TRM

	a	b	c
a	-0.085234	-0.0852343	-0.085234
b	-0.187974	-0.1879741	-0.187974
c	-2.026216	-2.0262161	-2.026216

TTM

	a	b	c
a	1.0027397	1.0027397	1.00274
b	1.0136986	1.0136986	1.013699
c	1.1666667	1.1666667	1.166667

Failure Blends

	operation	inspection	check	% of total failures
a	0.0384615	0.9615385	1	0.436170213
b	0.137931	0.862069	1	0.521276596
c	1	0	1	0.042553191
				1

Replace Action:

TPM

	a	b	c
a	0.43617	0.521277	0.042553
b	0.43617	0.521277	0.042553
c	0.43617	0.521277	0.042553

TRM

	a	b	c
a	-0.984549	-0.984549	-0.984549
b	-0.984549	-0.984549	-0.984549
c	-0.984549	-0.984549	-0.984549

TTM

	a	b	c
a	1.00274	1.00274	1.00274
b	1.013699	1.013699	1.013699
c	1.166667	1.166667	1.166667

Figure 11. Machine 3, "Solar Panel," Long Process Matrices.

Parameter	"Solar Panel"
replacement cost	-1
production rate	0.25
operational period length	0.082192
inspection length	0.00274
P(operational failure a)	0.013
P(operational failure b)	0.014
P(operational failure c)	0.00225
Cum. Operational failures	0.02925
P(inspection failure a)	0.009
P(inspection failure b)	0.013
P(inspection failure c)	0
Cum. Inspection failures	0.022
inspection cost	-0.05
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.00068
failure a cost	-0.10068
failure b cost	-0.20342
failure c cost	-2.04167
failure a time	0.00274
failure b time	0.013699
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.4295249	0.5260144	0.044461
b	0.4295249	0.5260144	0.044461
c	0.4295249	0.5260144	0.044461

TRM

	a	b	c
a	-0.727811	-0.7278108	-0.72781
b	-0.830551	-0.8305506	-0.83055
c	-2.668793	-2.6687926	-2.66879

TTM

	a	b	c
a	1.1567747	1.1567747	1.156775
b	1.1677336	1.1677336	1.167734
c	1.3207017	1.3207017	1.320702

Replace Action:

TPM

	a	b	c
a	0.429525	0.526014	0.044461
b	0.429525	0.526014	0.044461
c	0.429525	0.526014	0.044461

TRM

	a	b	c
a	-1.62713	-1.62713	-1.62713
b	-1.62713	-1.62713	-1.62713
c	-1.62713	-1.62713	-1.62713

TTM

	a	b	c
a	1.156775	1.156775	1.156775
b	1.167734	1.167734	1.167734
c	1.320702	1.320702	1.320702

Failure Blends

	operation	inspection	check	% of total failures	original failure blend
a	0.676	0.324	1	0.429524863	0.43617
b	0.5369863	0.4630137	1	0.526014445	0.521277
c	1	0	1	0.044460692	0.042553
				1	

Figure 12. Machine 3, "Solar Panel," Short Process Matrices.

Parameter	"The Machine in the back of the shop"
replacement cost	-1
production rate	0.333333
operational period length	2
inspection length	0.019231
P(operational failure a)	0.25
P(operational failure b)	0.333333
P(operational failure c)	0.125
Cum. Operational failures	0.708333
P(inspection failure a)	0.333333
P(inspection failure b)	0.333333
P(inspection failure c)	0
Cum. Inspection failures	0.666667
inspection cost	-0.1
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.00641
failure a cost	-0.10183
failure b cost	-0.26279
failure c cost	-2.05556
failure a time	0.005479
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.3846154	0.4769231	0.138462
b	0.3846154	0.4769231	0.138462
c	0.3846154	0.4769231	0.138462

TRM

	a	b	c
a	-0.057962	-0.0579619	-0.05796
b	-0.218921	-0.2189208	-0.21892
c	-2.011691	-2.011691	-2.01169

TTM

	a	b	c
a	1.1305872	1.1305872	1.130587
b	1.1634639	1.1634639	1.163464
c	1.2917744	1.2917744	1.291774

Failure Blends

	operation	inspection	check	% of total failures
a	0.36	0.64	1	0.384615385
b	0.5	0.5	1	0.476923077
c	1	0	1	0.138461538
				1

Replace Action:

TPM

	a	b	c
a	0.384615	0.476923	0.138462
b	0.384615	0.476923	0.138462
c	0.384615	0.476923	0.138462

TRM

	a	b	c
a	-0.95614	-0.95614	-0.95614
b	-0.95614	-0.95614	-0.95614
c	-0.95614	-0.95614	-0.95614

TTM

	a	b	c
a	1.130587	1.130587	1.130587
b	1.163464	1.163464	1.163464
c	1.291774	1.291774	1.291774

Figure 13. Machine 4, "The Machine in the Back of the Shop," Long Process Matrices.

Parameter	The Machine in the back of the shop
replacement cost	-1
production rate	0.25
operational period length	0.244521
inspection length	0.005479
P(operational failure a)	0.03
P(operational failure b)	0.03
P(operational failure c)	0.015
Cum. Operational failures	0.075
P(inspection failure a)	0.015
P(inspection failure b)	0.025
P(inspection failure c)	0
Cum. Inspection failures	0.04
inspection cost	-0.1
inspection discovery reward	failure cost/2
Inspection downtime cost	-0.00137
failure a cost	-0.10137
failure b cost	-0.25959
failure c cost	-2.04167
failure a time	0.005479
failure b time	0.038356
failure c time	0.166667

Repair Action:

TPM

	a	b	c
a	0.3917411	0.4743304	0.133929
b	0.3917411	0.4743304	0.133929
c	0.3917411	0.4743304	0.133929

TRM

	a	b	c
a	-0.680386	-0.6803862	-0.68039
b	-0.838605	-0.8386054	-0.83861
c	-2.620683	-2.620683	-2.62068

TTM

	a	b	c
a	1.5054795	1.5054795	1.505479
b	1.5383562	1.5383562	1.538356
c	1.6666667	1.6666667	1.666667

Replace Action:

TPM

	a	b	c
a	0.391741	0.47433	0.133929
b	0.391741	0.47433	0.133929
c	0.391741	0.47433	0.133929

TRM

	a	b	c
a	-1.57902	-1.57902	-1.57902
b	-1.57902	-1.57902	-1.57902
c	-1.57902	-1.57902	-1.57902

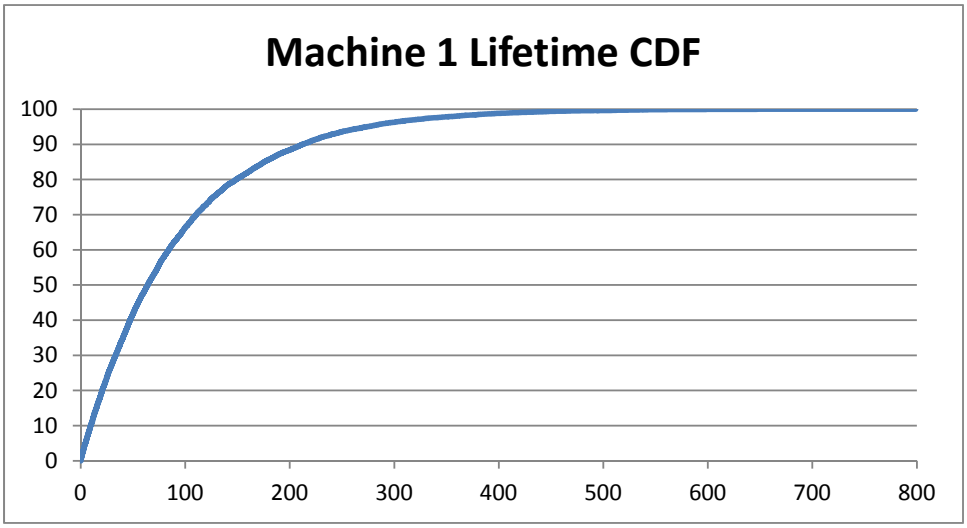
TTM

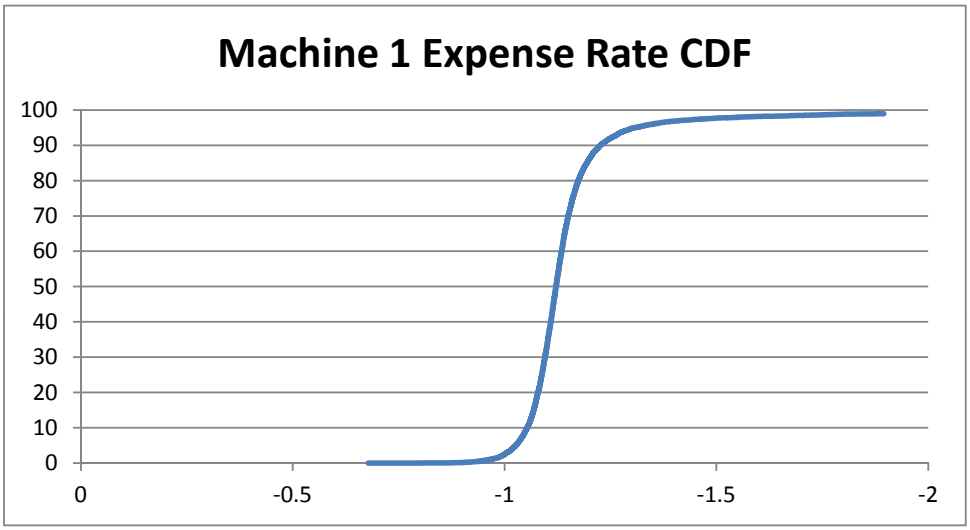
	a	b	c
a	1.505479	1.505479	1.505479
b	1.538356	1.538356	1.538356
c	1.666667	1.666667	1.666667

Failure Blends

	operation	inspection	check	% of total failures	original failure blend
a	0.8	0.2	1	0.391741071	0.384615
b	0.5901639	0.4098361	1	0.474330357	0.476923
c	1	0	1	0.133928571	0.138462
				1	

Figure 14. Machine 4, "The Machine in the Back of the Shop ," Short Process Matrices.





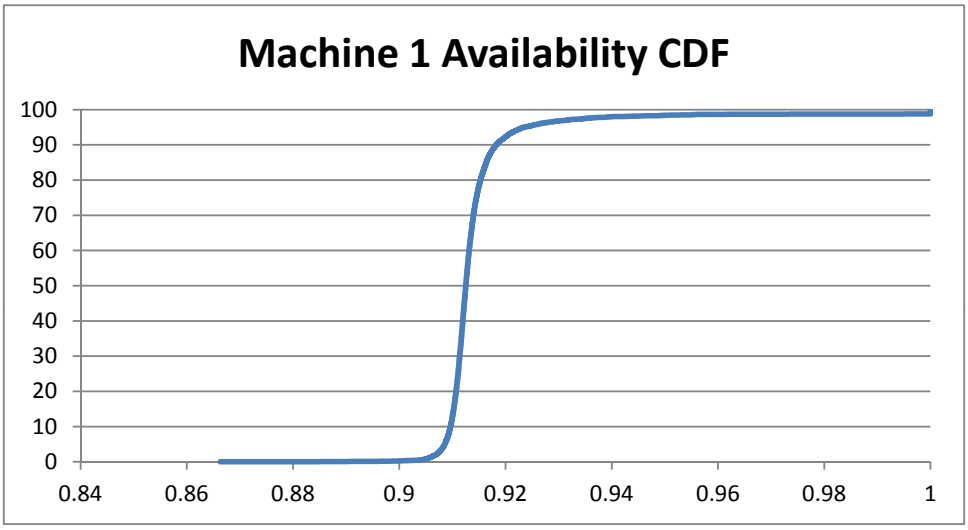
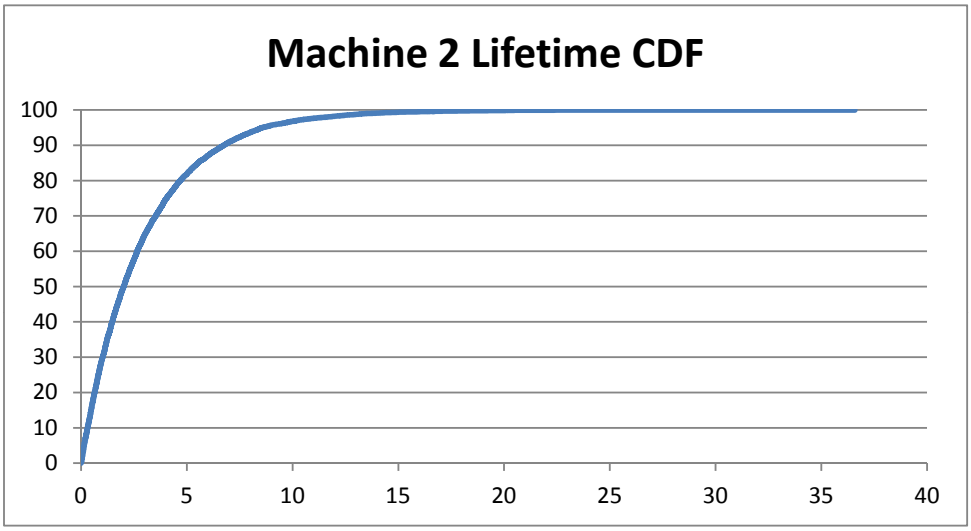
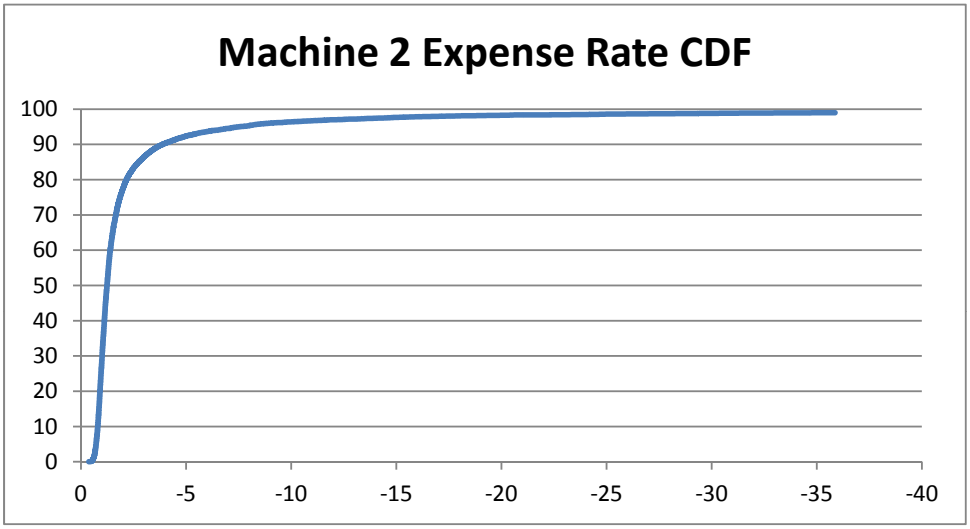


Figure 15. Machine 1 Lifetime, Expense Rate, and Availability CDFs





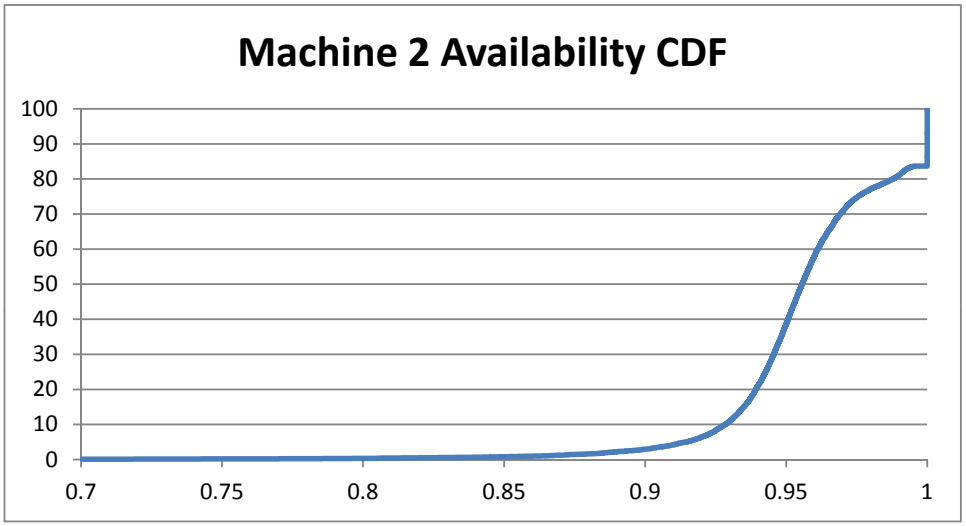


Figure 16. Machine 2 Lifetime, Expense Rate, and Availability CDFs

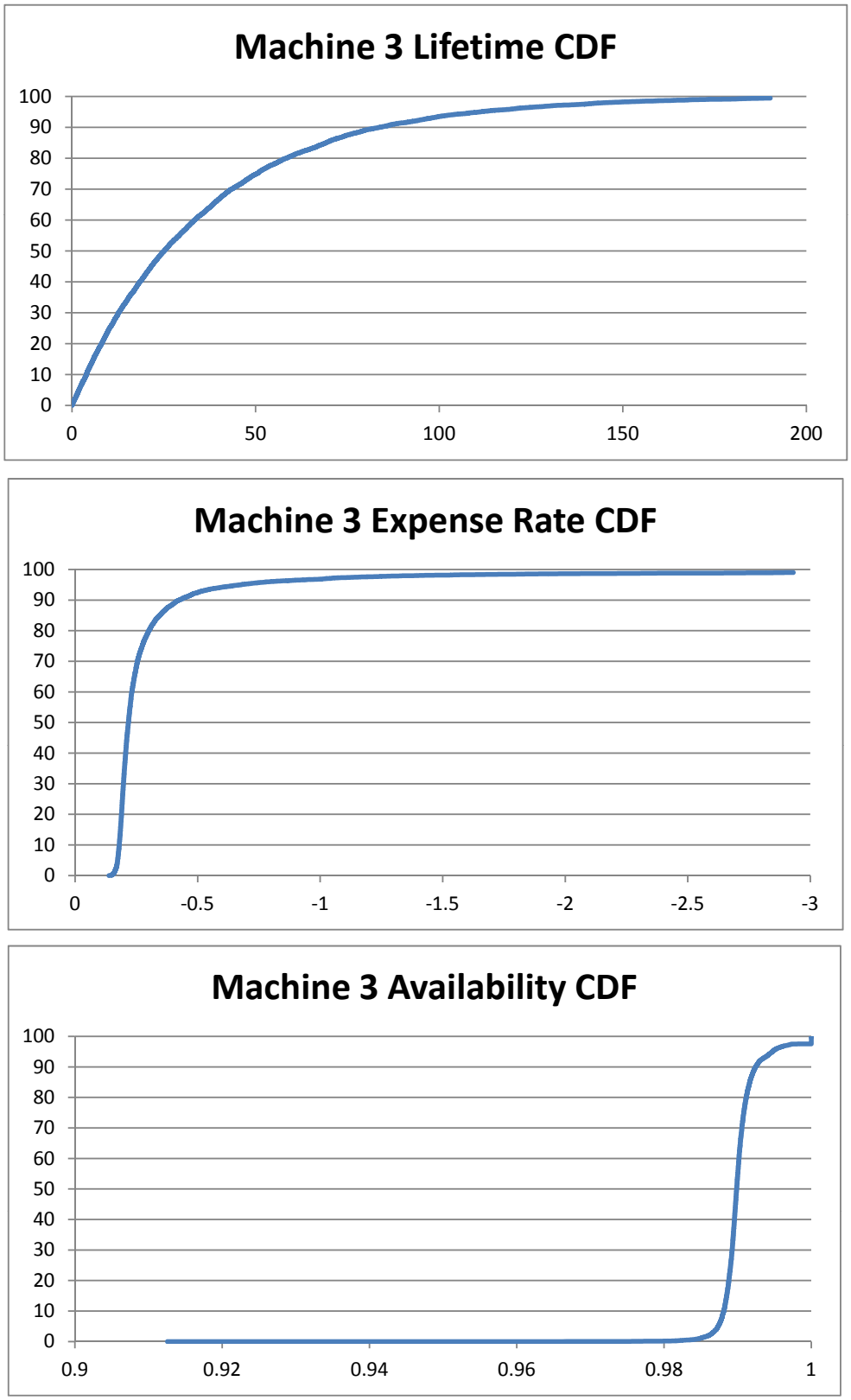


Figure 17. Machine 3 Lifetime, Expense Rate, and Availability CDFs

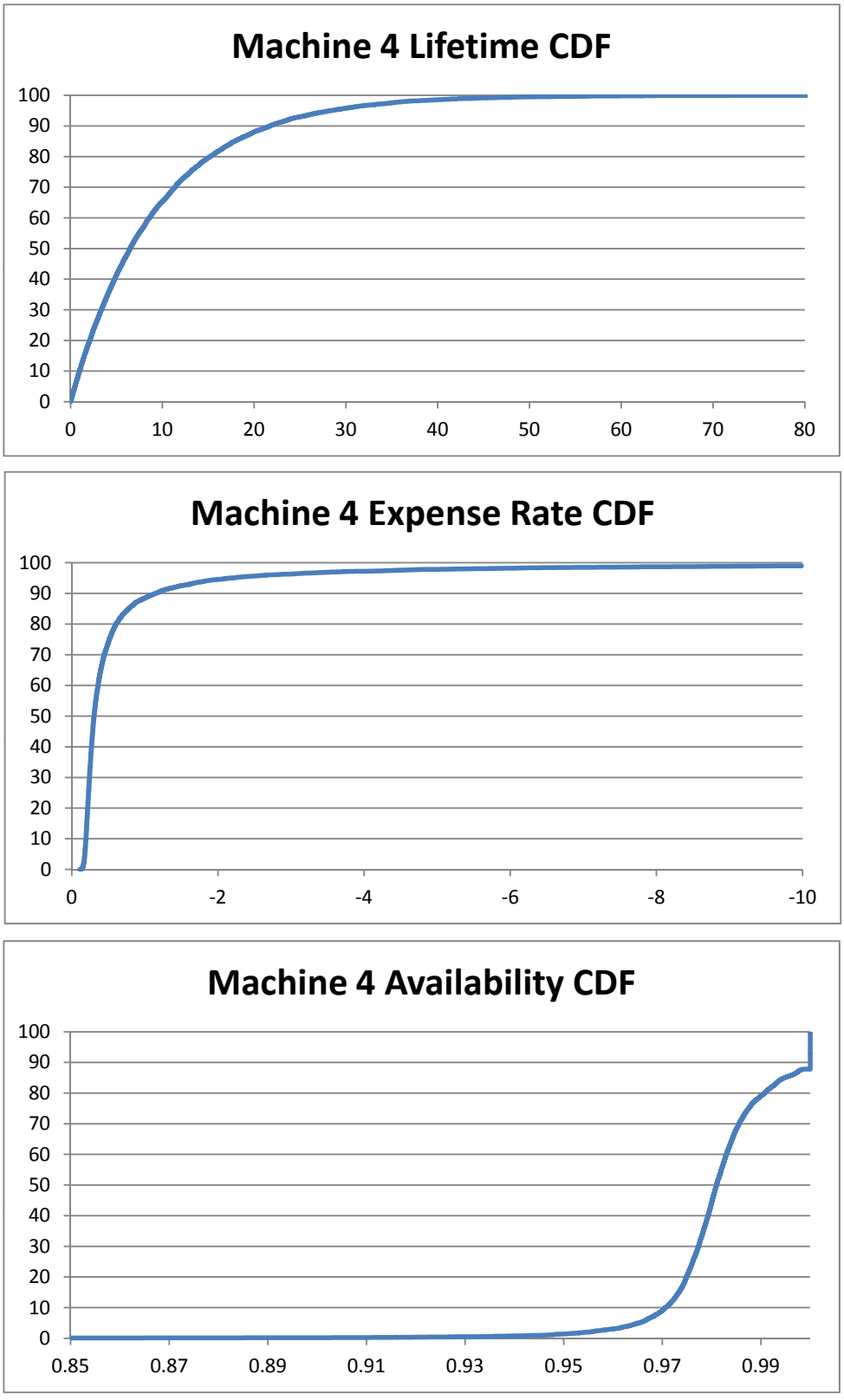


Figure 18. Machine 4 Lifetime, Expense Rate, and Availability CDFs

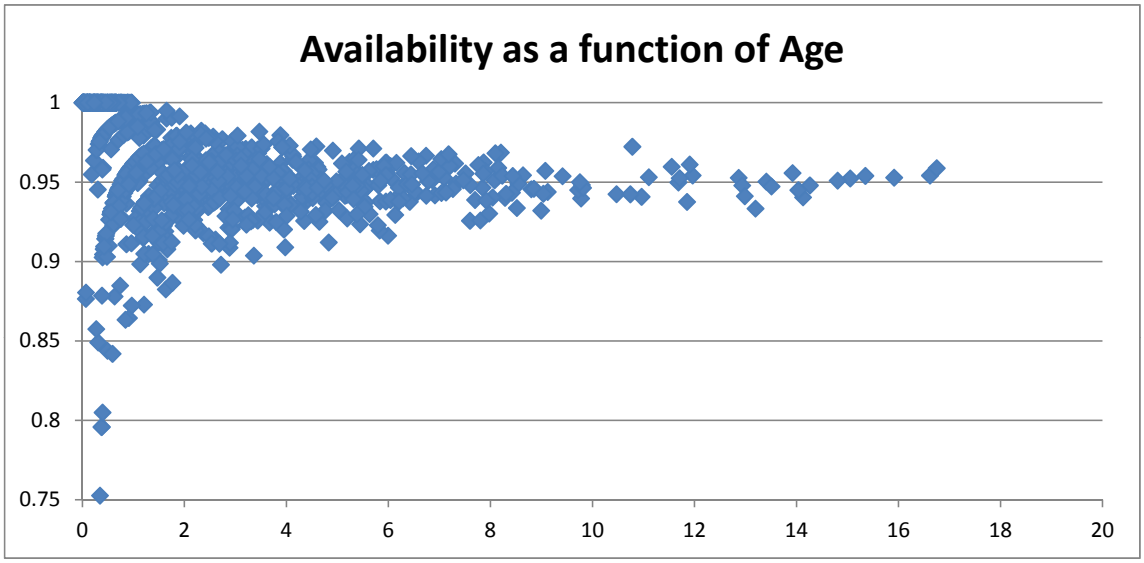


Figure 19. Availability vs Age for Machine 2.

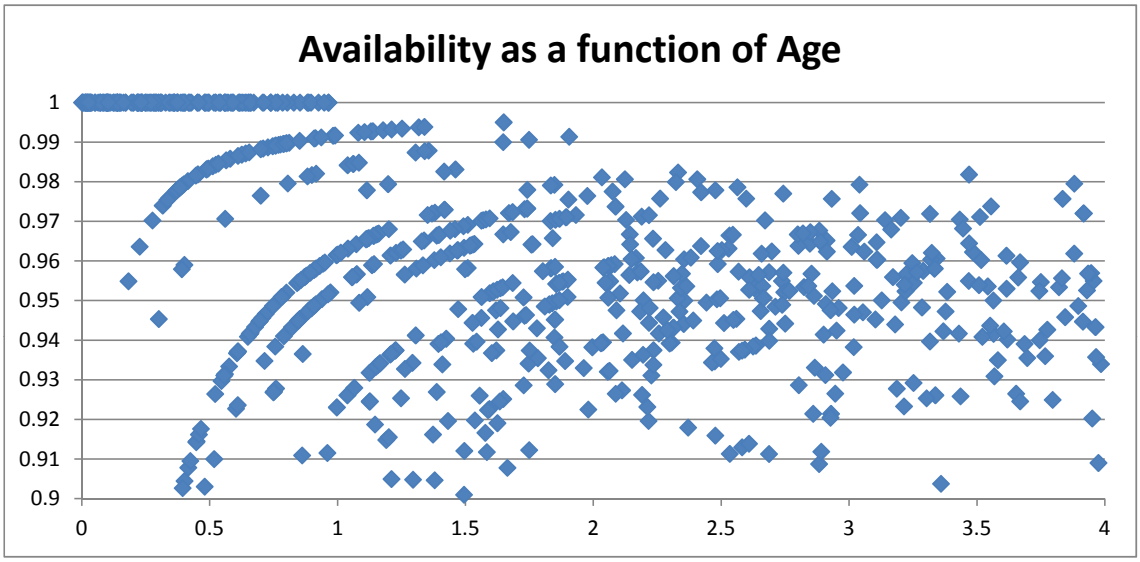


Figure 20. Enhanced Availability vs Age for Machine 2.

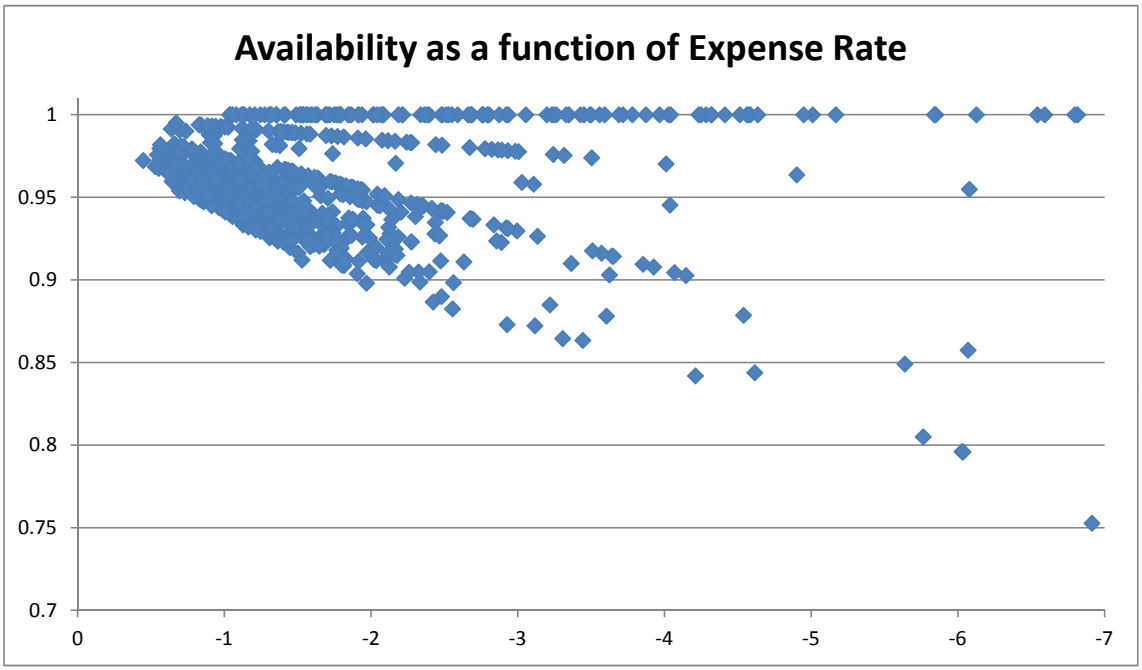


Figure 21. Availability vs Expense Rate for Machine 2.

V. Conclusions and Recommendations

5.1 Review

The research presents a method for constructing a maintenance process with periodic inspection intervals, then given a machine operating in that process, determines at which user defined failure events the machine should be repaired or replaced. Further, the method is also able to determine the best inspection process from among any group of alternatives provided by the machine operator. A semi Markov decision process is used to accomplish this, using policy iteration and the average reward model. Results are then simulated by constructing 4 hypothetical machines and 2 competing inspection processes. It is found in all four machines that the long inspection process is preferred to the short process and for all but the most sever failure type the repair action is preferred to the replacement action. Further, demonstrations are provided that show how machine and inspection parameters may be altered to affect the policy decision of the SMDP.

5.2 Insights

The process for specifying competing inspection processes is not a trivial undertaking, in this framework it results in an inherently under defined system, making parameter specification very flexible, but somewhat imprecise. Guidelines to aid precision and rigor were provided and used in constructing the 4 machines and their inspection processes. These machines were then evaluated by the SMDP to determine which failure events should result in a replace decision and which failure events should result in a repair decision. Simultaneously the SMDP also determined the best inspection process for that machine to use upon the specified machine. Policy iteration resulted in the same decisions at the same failure event types for all

four machines. This resulted in the insight that the algorithm was simply choosing between the cost of repair plus the opportunity cost of downtime or the cost of an immediate replacement; and for inspection decisions the SMDP was choosing the policy which had the lowest expected per cycle cost. This strongly implies that within this framework maintenance decisions and inspection process decisions are separately determined, as evidenced by the construction of the TTM, TPM, and TRM matrices.

5.3 Potential Future Research

Future potential for this method lies in constructing estimates from real world data, or in constructing estimates to synthesize data. This approach is appealing because of its potential for expansion and flexibility. However, the flexibility it permits can also be daunting because it requires a great deal of attention to the specification of machine and inspection process parameters.

Expanding the number of operational states beyond the simple one-state paradigm used in this study could be pursued. In this way it would be possible to simulate processes whose lifetime distributions are not subject to a constant failure rate throughout their life, or machines which have distinct operation regimes. This is akin to using different degradation states, so care should be taken in assigning parameter values to those additional operational states. Again, the Kharoufeh method [7] could aid in the specification of the degradation states if a clear way to define the states did not already exist. Along these same lines, permitting more than one type of failure to be discovered during an inspection might also be pursued. Both of these evolutions only require expanding the state space.

The problem of event-driven upgrading could be examined by this method. Using several linked decision subspaces it would be possible to simulate a machine which migrates from one group of states to another in the optimal manner. The different

groups of states would possess different operational parameters, and different failure states to represent the new realities of the machine after it had received its mid-life upgrade. The policy resulting from this would choose the optimal machine events at which to upgrade the system. In pursuing this process a good deal of attention should be paid to making sure the upgrade actions are properly specified to ensure the machine has an incentive to upgrade. Of course, if an upgrade is not economical, it will not be chosen; however, an MDP/SMDP will never choose an action that in the long run would require higher costs without greater productivity, or similar productivity with lower operational costs.

The machine comparison method outlined in Chapter III is also a method which warrants further examination. While the method ensures the rate of failure and failure blend across individual inspection cycles are consistent, it does not consider failure accumulation across many inspection cycles. That is to say, a 3% monthly risk is not equivalent to a 36% annual risk; but, through the time normalization of inspection cycles, that is what is done. Further work should investigate the implications of this, and examine the best way to handle this effect while maintaining machine identity. It might also be furtive to investigate the tradeoff between inspection costs, cycle lengths, and failure discovery efficiency to determine if any relationship can be determined that would indicate an intrinsic superiority to a given type of inspection process.

Finally, considering this problem in the structure of renewal theory would prove to be an interesting pursuit. The approach of this problem would be similar to an MDP/SMDP problem, however, instead of defining a TTM, TPM, and TRM for each action, only a kernel matrix and a TRM would be needed. The construction of a comparable process involving the kernel matrix is not a trivial pursuit. The derivation of the kernel matrix alone can be very difficult, and the construction of a process that

could handle generally stated problems is more difficult. Perhaps the best way to approach such a problem could be exhaustive enumeration of the possible policies, but this becomes prohibitive very quickly for larger problems. Indeed, removing the comfort of well defined transitions is a very challenging problem.

5.4 Conclusion

Though MDPs only consider stationary policy, there is another interpretation that can be drawn observing the decisions in a time-dependent manner. If one considers each decision the process makes at the point in time of that decision, the prescribed policies interpretation is to not repair anything that could be replaced for less at that time. This is saying, if it is assumed that the relative cost of repairs and replacement is not static in time, the repair/replace recommendation can be interpreted to mean replace at anytime the repair cost is greater than the cost of an equal replacement. In a sense this violates the static policy requirement of MDPs; nevertheless, if a system evolves with time so must its MDP. This, of course, makes intuitive sense and is another benefit that we can realize from these results. The two main results are equally intuitive; that the formulation of this method allowed the SMDP to consider the inspection decision and repair/replace decision separately, and that because of this separation, the prescribed policy from the SMDP was simply a policy of choosing the lowest cost.

Appendix A. SMDP Policy Iteration Procedure

The following summary provides a description of the policy iteration algorithm. The full code for the algorithm used can be found in Gosavi's book [6]. There you can also find full examples to aid in understanding and error checking.

In policy iteration, for both average reward and discounted reward algorithms, the goal is to step to a better policy at each iteration. This is accomplished by solving a linear system of equations at each iteration. An outline of steps for policy iteration using the average reward criterion with discussion follows:

1. Begin by choosing an arbitrary policy for your system. While the policy can be made truly arbitrary, an operator with some idea of the intrinsic behavior of the system may desire to set the initializing policy near what they suspect might be optimal, though this is not necessary.

2. Next, the system of equations for the first iteration needs to be established. The definitions of variable follows, then the equation is given:

    - The h term is the value function. Its argument is the state of the system, its subscript is the iteration number. These are the unknowns, there are as many of these as there are states.
    - The $\bar{r}$ term is the average reward. The arguments of it should be interpreted as the average reward from state i, under the policy(μ) used in the k iteration.
    - μ is the policy. If it is accompanied by a specific argument, it is referencing the policy for that state. μ used without an argument represents the policy vector.
    - ρ_k is the average reward of the k iteration. This is one additional unknown.

- j is the next state the system may transition to.
- S is the state space of the system.
- $p(I, \mu_k(i), j)$ is the probability of going from state i to state j under policy μ_k.
- a is the specific action, from $A(i)$, the policy set for the system at state i. This is used in the next step.

$$h_k(i) = \bar{r}(i, \mu_k(i)) - \rho_k + \sum_{j=1}^{|S|} p(i, \mu_k(i), j) h_k(j) \tag{1.1}$$

This equation is essentially saying that the value function of state i is equal to the average reward expected in state i under the current policy, minus the average reward yielded by that policy in that iteration, plus the average reward anticipated from the next transition. Given that this system has one more unknown than the number of equations, in order to solve the system one of the h terms should be set to zero so that the system is not underdetermined, ρ should not be set to zero.

3. After solving this system of equations, the policy improvement step discovers the next policy the MDP will iterate over. This is done by selecting, for each state, the new action which results in the greatest expected reward for that state. The collection of these new action decisions will be the new policy. The functional equation is stated:

$$\mu_{k+1} = \arg\max_{a \in A(i)} \left[\bar{r}(i, a) + \sum_{j=1}^{|S|} p(i, a, j) h_k(j) \right] \tag{1.2}$$

A special consideration worth mentioning here: where possible the policy should be kept that same. That is to say, if two actions return the same reward for

state i, and one of those actions is in the current policy, it should remain the same.

4. If the new policy just discovered by the previous step is identical to the old policy, then the policy is optimal; otherwise, increment your index by 1 and return to solving the system of equations for the value functions.

When the optimal policy is finally discovered the Bellman equations are expressed as follows, where $\widehat{\mu}$ denotes the optimal policy:

$$h_{\widehat{\mu}}(i) = \bar{r}(i, \mu(i)) - \rho_{\widehat{\mu}} + \sum_{j=1}^{|S|} p(i, \mu(i), j) h_{\widehat{\mu}}(j) \tag{1.3}$$

Appendix B. Java Simulation Code

The following is the java simulation code used to generate the data for the SMP. This code was used for machine 1; each machine used slightly different code due to parameter differences.

```
package machine1sim;
import java.util.*;
/**
 * @author Teddy Shiveley
 */
public class Machine1Sim {

    public static void main(String[] args) {

        Machine generic=new Machine();
        generic.perform_loop();

    }
}

class Machine {

    double Age, Expense, replacement_cost, production_rate,
           op_failure_a, op_failure_b, op_failure_c, op_period,
           ins_failure_a, ins_failure_b, ins_failure_c, inspection_time,
           ins_cost, Disc_reward_a, Disc_reward_b, Disc_reward_c,
           fail_a_cost, fail_b_cost, fail_c_cost,
```

```
        fail_a_time, fail_b_time, fail_c_time, downtime,
        Cum_op_fails, Cum_ins_fails, lambda;

 int    count_a, count_b, count_c, replace_signal;
        //the above only need to be ints,
        //but if using Arraytostring, they have to be doubles

public Machine () {    //constructor function which makes the machine
                       //these never have a return, so no return is necessary
    Age= 0;            //initializing the machine object...
    Expense= 0;        //other properties defined as follows
    count_a=0;         //general note of caution:
    count_b=0;         //since these variables are of class double
    count_c=0;         //Java will not understand fractions that would
    downtime=0;        //result in irrational numbers
    replacement_cost=-1;
    production_rate=10;
    op_period=.9167;        //this scheme is about a on year cycle with
    inspection_time=.0833;  //11 months up and 1 month inspection


    op_failure_a=0.05;         //probability of experiencing
    op_failure_b=0.1;          //operational failures
    op_failure_c=0.01;
    Cum_op_fails=op_failure_a+op_failure_b+op_failure_c;
```

```
ins_failure_a=0.1;        //probability of discovering a failure
ins_failure_b=0.2;        //during an inspection
ins_failure_c=0;
Cum_ins_fails=ins_failure_a+ins_failure_b+ins_failure_c;

fail_a_time=.005479;   //time penalties imposed for having a failure
fail_b_time=.03836;    //this is roughly about a 2 week repair time
fail_c_time=.1667;     //2 month downtime

fail_a_cost=-.1-fail_a_time*production_rate;     //routine
fail_b_cost=-.3333-fail_b_time*production_rate;  //serious
fail_c_cost=-2-fail_c_time*production_rate;      //catastrophic
             //note the downtime costs in addition to the base costs

ins_cost=-.05-inspection_time*production_rate;
//this means the cost of an inspection is rougly 5% of replacement cost...

Disc_reward_a=fail_a_cost/2;    //this is the nominal benefit to
Disc_reward_b=fail_b_cost/2;    //discovering a failure from
Disc_reward_c=fail_c_cost/2;    //inspection instead of operation
```

```
    //implicit lambda; this will be used when failure occur from operation
    lambda=-Math.log(1-Cum_op_fails)/op_period;
    replace_signal=0; //Binary value that is used to exit  the loop
}

public void perform_loop(){
     int reps=10000; //repititions for loop

     double[] Ages= new double[reps];    //initializing data collection arrays
     double[] Costs= new double[reps];
     double[] Type_a_fails= new double[reps];
     double[] Type_b_fails= new double[reps];
     double[] Type_c_fails= new double[reps];
     double[] Downtimes= new double[reps];

     double[] Availability= new double[reps];
     double[] Expense_rate= new double[reps];

     //Machine generic=new Machine(); turns out this isn't needed here

     for(int i = 0; i < reps; ++i){

     operational(); //the simulation

     Ages[i]=this.Age; //populating array with results
     Costs[i]=this.Expense;
```

```
        Type_a_fails[i]=this.count_a;
        Type_b_fails[i]=this.count_b;
        Type_c_fails[i]=this.count_c;
        Downtimes[i]=this.downtime;
        Availability[i]=(this.Age-this.downtime)/this.Age;
        Expense_rate[i]=this.Expense/this.Age;

        refresh(); //begin sim again
        }

        //these commands are more efficient at printing output (no loop)

//       System.out.println("Ages=\n" + Arrays.toString(Ages));
//       System.out.println("Costs=\n" +Arrays.toString(Costs));
//       System.out.println("Downtimes=\n" + Arrays.toString(Downtimes));
//       System.out.println("Expense_rate=\n" +Arrays.toString(Expense_rate));
//       System.out.println("Count_a=\n" + Arrays.toString(Type_a_fails));
//       System.out.println("Count_b=\n" + Arrays.toString(Type_b_fails));
//       System.out.println("Count_c=\n" + Arrays.toString(Type_c_fails));

        //these commands create output that is easily exported to excel

        System.out.println("Ages, Costs, Downtimes, Expense_rate, "
                           + "Count_a, Count_b, Count_c, Availability");
        for(int i = 0; i < reps; ++i){
          System.out.println(Ages[i]+","+ Costs[i]+","+Downtimes[i]
```

```
                        +","+Expense_rate[i]+","+Type_a_fails[i]
                        +","+Type_b_fails[i]+","+Type_c_fails[i]
                        +","+Availability[i]);
     }
}

public void operational(){
    //this code is the machine in the operational state
    //it can either sucessfully survive to an inspection or
    //experience a failure

    //provided a replacement has not been triggered...
    if(this.replace_signal==0){

        double seed;     //random number to determine if failure
        seed=Math.random();



        if(seed>this.Cum_op_fails){
           //machine survives operation to an inspection
           this.Age=this.Age+this.op_period;
           inspection();
        }else{
            double reseed;
            double exponum;
            reseed=Math.random();
```

```
            //we need a new random here because otherwise types of failures
            //would only occur in the times of there probability bins
            exponum=-Math.log(1-this.Cum_op_fails*reseed)/this.lambda;
            this.Age=this.Age+exponum;
            //add appropriate amnt to age and send to failure
             if(seed<=this.op_failure_a){
                 fail_a();
             }
             if(seed>(this.op_failure_a)
                     && seed<=(this.op_failure_a+this.op_failure_b)){
                 fail_b();
             }
             if(seed>(this.op_failure_a+this.op_failure_b)
                     && seed<=(this.Cum_op_fails)){
                 fail_c();
             }
         }
}else{
 //do nothing if replace signal is triggered and exit the loop
     }
 }


 public void inspection(){
  //this code is the machine under inspection
  //either a failure can be discovered,
```

```
//or it can return to the operational state

//increment all counters  at the begininng since
//an entity remains under inspection for the entire state
this.Age=this.Age+this.inspection_time;
this.downtime=this.downtime+this.inspection_time;
this.Expense=this.Expense+this.ins_cost;

double seed;
seed=Math.random();
if(seed>this.Cum_ins_fails){
    //return the machine to service
    operational();
}else{
    if(seed<=this.ins_failure_a){
        this.Expense=this.Expense+this.Disc_reward_a;
        fail_a();
    }
    if(seed>(this.ins_failure_a)
            && seed<=(this.ins_failure_a+this.ins_failure_b)){
        this.Expense=this.Expense+this.Disc_reward_b;
        fail_b();
    }
    if(seed>(this.ins_failure_a+this.ins_failure_b)
            && seed<=(this.Cum_ins_fails)){
        //this.Expense=this.Expense+this.Disc_reward_c;
```

```
            fail_c();
         }
    }
 }


    public void fail_a (){
        this.Age=this.Age+this.fail_a_time;
        this.Expense=this.Expense+this.fail_a_cost;
        this.downtime=this.downtime+this.fail_a_time;
        this.count_a=this.count_a+1;
        operational();
    }
public void fail_b (){
        this.Age=this.Age+this.fail_b_time;
        this.Expense=this.Expense+this.fail_b_cost;
        this.downtime=this.downtime+this.fail_b_time;
        this.count_b=this.count_b+1;
        operational();

}

public void fail_c (){

        //this is replacement, so now the accounting will be a little different
        this.Expense=this.Expense+this.replacement_cost;
```

```
        this.count_c=this.count_c+1; //should only ever be 1
        //instantaneous replacement, so time penalties not incurred
        this.replace_signal=1;
        operational();
}

public void refresh (){
  this.Age=0;
  this.Expense=0;
  this.count_a=0;
  this.count_b=0;
  this.count_c=0;
  this.downtime=0;
  this.replace_signal=0;
  }
}
```

Bibliography

[1] Bellman, Richard. *Dynamic Programming*. Mineola, NY: Dover Publications, 2003.

[2] Bertsekas, Dimitri P. *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2005.

[3] Crabhill, Thomas B. "Optimal Control of a Maintenance System with Variable service rates", *Operations Research*, 22:736–745, 1974.

[4] Denardo, Eric V. *Dynamic Programming Models and Applicatoins*. Mineola, NY: Dover Publications, 2003.

[5] Gebraeel, Nagi and Jing Pan. "Prognostic Degradation Models for Computing and Updating Residual Life Distributions in Time-Varying Environments", *IEEE Transactions on Reliability*, 57:539–550, 2008.

[6] Gosavi, Abhijit. *Simulation-Based Optimization*. Norwell, MA: Kluwer Academic Publishers, 2003.

[7] Kharoufeh, Jeffrey P. and Steven M. Cox. "Stochastic Models for Degradation-Based Reliability", *IIE Transactions*, 37:533–542, 2005.

[8] Kharoufeh, Jeffrey P., Daniel E. Finkelstein, and Dustin G. Mixon. "Availability of Periodically Inspected Systems with Markovian Wear and Shocks", *IEEE Transactions*, 37:203–217, 2006.

[9] Kharoufeh, Jeffrey P., Christopher J. Solo, and M. Yasin Ulukus. "Semi-Markov Models for Degredation-Based Reliability", *IIE Transactions*, 42:599–612, 2010.

[10] Khintchine, A. Y. *Mathematical Methods in the Theory of Queues*. London: Charles Griffin and Company Limited, 1960.

[11] Kulkarni, Vidyadhar G. *Modeling and Analysis of Stochastic Systems*. Boca Raton, FL: Chapman & Hall/CRC, 2010.

[12] Meeker, William Q. and Luis A. Escobar. *Statistical Methods for Reliability Data*. Hoboken , NJ: John Wiley & Sons, 1998.

[13] Puterman, Martin L. *Markov Decision Processes*. New York, NY: John Wiley and sons, 1994.

[14] Ravindran, A., Don T. Phillips, and James J. Solberg. *Operations Research Principles and Practice*. New York, Ny: John Wiley and Sons, 1987.

[15] Ross, Sheldon M. *Introduction to Probability Models*. Boston, MA: Academic Press, 2007.

[16] Si, Xiao-Sheng, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. "Remaining Useful Life Estimation- A Review on the Statistical Data Driven Approaches", *European Journal of Operations Research*, 213:1–14, 2011.

[17] Sobel, Matthew J. "Myopic solutions of Markov Decision Processes and Stochastic Games", *Operations Research*, 29:995–1009, 1981.

[18] Tao, Zongwei, Ross B. Corotis, and J. Hugh Ellis. "Reliability-Based Structural Design with Markov Decision Processes", *Journal of Structural Engineering*, 121:971–980, 1995.

[19] Tomasevicz, Curtis L. and Sohrab Asgarpoor. "Optimum Maintenance Policy Using Semi-Markov Decision Processes", *Electric Power Systems Research*, 79:1286–1291, 2009.

[20] White, D. J. "Optimal Revision Periods", *Journal of Mathematical Analysis and Applications*, 4:353–365, 1962.

[21] White, D. J. "A Survey of Applications of Markov Decision Processes", *The Journal of the Operational Research Society*, 44:1073–1096, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21–03–2013	Master's Thesis	Aug 2011 – Mar 2013

4. TITLE AND SUBTITLE

Determining Optimal Machine Replacement Events with Periodic Inspection Intervals

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Shiveley, Theodore C., Captain USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENC-13-M-17

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Mr. David L. Merrill
Director, AMC/A9
1 Soldier Way
Scott AFB, IL 62225
dave.merrill.5@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AMC A9

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research will examine the optimal maintenance and replacement policies for a generic machine with periodic inspection intervals. The considered reliability models consist of a single machine that can fail during operation or else may be found to be inoperative during regularly-scheduled maintenance inspections. A distinction will be made between spontaneously-occurring failures during operation and those that are discovered during inspections. Since the elapsed time between inspections is constant, the resulting stochastic reliability process becomes non-Markovian, and thus a Semi-Markov Decision Process (SMDP) framework must be employed in order to determine the cost-optimal stationary policy consisting of repair and replace decisions and inspection intervals. Using the methodology developed here, a system controller will be able to readily develop an inspection-based strategy to optimize the overall costs of maintaining systems with a variety of failure characteristics over a finite time horizon.

15. SUBJECT TERMS

Markov Decision Processes, Policy Iteration, Optimal Maintenance, Inspection Processes, Lifetime Estimation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj James D. Cordeiro (ENC)
U	U	U	UU	97	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636 x4398; James.Cordeiro@afit.edu