3-2004

Age Replacement and Service Rate Control of Stochastically Degrading Queues

Patrick S. Chapin

AGE REPLACEMENT AND SERVICE RATE CONTROL OF STOCHASTICALLY DEGRADING QUEUES

THESIS

Patrick S. Chapin, Second Lieutenant, USAF

AFIT/GOR/ENS/04-02

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, Department of Defense or the United States Government.

AFIT/GOR/ENS/04-02

AGE REPLACEMENT AND SERVICE RATE CONTROL OF STOCHASTICALLY DEGRADING QUEUES

THESIS

Presented to the Faculty

Department of Operational Sciences

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 in Operations Research

Patrick S. Chapin, B.S.

Second Lieutenant, USAF

March 2004

AFIT/GOR/ENS/04-02

AGE REPLACEMENT AND SERVICE RATE CONTROL OF STOCHASTICALLY DEGRADING QUEUES

Patrick S. Chapin, B.S.

Second Lieutenant, USAF

Approved:

Dr. Jeffrey P. Kharoufeh Thesis Advisor	Date
Dr. James W. Chrissis Committee Member	Date

AFIT/GOR/ENS/04-02

Abstract

This thesis considers the problem of optimally selecting a periodic replacement time for a multiserver queueing system in which each server is subject to degradation as a function of the mean service rate and a stochastic and dynamic environment. Also considered is the problem of optimal service rate selection for such a system. In both cases, the performance metric is the long-run average cost rate. Analytical expressions are obtained, in terms of Laplace transforms, for the nonlinear objective functions, necessitating the use of numerical Laplace transform inversion to evaluate candidate solutions in conjunction with standard numerical algorithms. Due to the convexity of the objective function, the optimal replacement time is computed using a hybrid bisection-secant method which yields globally optimal solutions. The optimal service rates are obtained via gradient search methods but are only guaranteed to provide locally optimal solutions. The analytical results are implemented on three notional examples that demonstrate the benefits of dynamically adjusting service rates under the described maintenance policy.

Acknowledgements

I would like to take this opportunity to thank all those who made this thesis possible. First, I would like to thank my advisor, Dr. Jeff Kharoufeh. Without his guidance and instruction, this thesis would never have happened. I am grateful for the many hours he spent assisting me in this effort.

Thanks are also due to my Committee Member, Dr. Jim Chrissis, for his insightful questions and careful editing of the document. This thesis is a better product because of his efforts. Furthermore, I would like to thank the rest of the outstanding AFIT faculty for giving me the knowledge and capability to address this thesis problem.

Most importantly, I would like to thank my wife for her love and understanding during this process. She stoically accepted my absences and mood swings, and still found it in her heart to love me. Thank you, love.

Patrick S. Chapin

Table of Contents

List of Figures

List of Tables

AGE REPLACEMENT AND SERVICE RATE CONTROL OF STOCHASTICALLY DEGRADING QUEUES

1. Introduction

1.1 Background

In this thesis, the optimal control of a stochastically degrading queueing system is considered. A stochastically degrading queueing system is one in which the server(s) of the system accumulate wear over time, and at some point cease to operate effectively. To optimally control a stochastically degrading system is to change or control various parameters of the system to achieve the most desirable result, depending on the particular attributes of the system being considered. The queueing system parameters that are assumed to be controllable in this thesis are: 1) the time at which servers are to be replaced, and 2) the mean service rate of each of the servers.

This type of system is common in the real world, especially in the manufacturing environment. For example, an outside diameter grinding wheel incurs wear as it grinds workpieces to a desired finish, and after a certain level of wear is reached, it can no longer effectively achieve engineering specifications. Another example of a degrading queueing system is a satellite telecommunications network. The satellites, which may be considered single server queueing stations, require power to operate. Typically, such power is provided by large solar panels. The performance of these panels degrades over time due to solar radiation, small particle strikes and other hazards found in the space environment, until the panels can no longer power the satellite, representing a failure of the server. Assume the operator of the satellite network may alter the rate at which the satellites transmit messages, perhaps allow-

ing those satellites to furl their solar panels in an effort to prevent damage. Both of these examples include servers that are subject to a myriad of factors external to the system, yet can have significant impact on the wear rate of the server(s). For example, the temperature of the surrounding environment could impact the grinding wheel by inducing heat expansion in the workpiece, causing it to require additional grinding, and hence causing the grinding surface to wear prematurely. Therefore, it may be possible for the grinding machine to wear faster or slower depending on the environment. Also, the rate at which damage is accrued by the solar panels of a satellite is completely dependent on the space environment. Both the space environment and the temperature of the surrounding environment of the grinding wheel are unpredictable, and hence, must somehow be characterized as they evolve in time. Usually, the objective of controlling these types of systems is to minimize some measure of cost, or achieve a desired measure of performance. The minimization of costs and maintenance of performance standards are of vital importance to organizations in a highly competitive global market. Additionally, such systems can be controlled to ensure worker safety.

Two methods of control are examined in this thesis. The first method of control is an optimal replacement strategy. A k-replacement policy implies that all the servers in a k-server queueing system will be replaced every T time units, and that no other repairs or replacements occur. The goal for the first portion of this thesis is to compute the value of T that minimizes the long-run expected cost rate of the system. The second method of control seeks to balance the tradeoff between the benefits of high service rates and the liabilities of an increased failure rate. Higher service rates lead to increased throughput, shorter queue lengths, and reduced waiting times for customers. On the other hand, high service rates also lead to a high failure rate which, in turn, leads to higher costs due to premature failures. Therefore, the objective is to compute the optimal mean service rates that balance this tradeoff and minimize the long-run expected cost rate.

1.2 Problem Definition and Methodology

At its core, this problem is a single-station queueing optimization problem where the server(s) accumulate wear and eventually fail. This queueing system is subject to an external environment that can be modelled as a finite-state stochastic process. That is, the random environment has a finite number of states between which it transitions randomly and the environment spends a random amount of time in each state. When the environment is in a given state, the server degrades according to a linear rate dependent on that state.

For the purposes of this thesis, a k-replacement policy for the system is assumed. That is, all k servers in the system are replaced every T time units, with repair or replacement occurring only at integer multiples of T. One of the consequences of this policy is that there may be times at which the number of operating servers is not sufficient to fulfill the stability condition of the queue. Therefore, it may be possible for the number of customers in the system to become unbounded. To ensure that this does not happen, it is assumed that an outside service provider is employed to pick up the workload of a failed server. That is, if a server has failed, then customers that would have been served by the failed server are routed to the outside entity. This external service provider, however, charges a relatively higher price than the in-house server.

This thesis has two main objectives. The first objective is to compute the optimal length of a replacement interval, T, given that the rate of wear on each server is governed by the state of the ambient operating environment. The second objective is to optimally compute the mean service rate for each state of the environment, given that the linear rate of wear on each server is a function of both the mean service rate and the state of the environment. The objective function for both optimization problems is the long-run expected cost rate.

To optimize the long-run cost rate, the cost function itself must be defined. The cost function examined in this thesis can be broken into four parts. The first component of the cost function is the replacement cost which is the cost required to replace the servers. The next component of the cost function is the work cost, or the cost of operating the servers at the specified service rate. The third cost component is a holding cost, which reflects cost of having a backlog of jobs to process. The last cost component is associated with the inconvenience generated by servers failing before they are replaced. This component takes into account the added cost of rerouting incoming customers to an external service provider that handles the workload of the failed server(s), as well as the payment due to this external service provider. It is assumed that the increases in holding costs are significantly higher than the cost of employing the outside service provider.

The minimization of this cost function is examined with respect to the replacement time and the environment-state-dependent mean service rates. Because the failure time distribution of the individual servers cannot be expressed analytically in the real domain, it is necessary to approximate the optimal solutions to both the replacement time problem and the service rate problem. Approximate solutions are obtained via the secant method in the case of the replacement time, or a simple gradient search procedure in the case of the service rate problem. Both depend on numerical inversion of Laplace transform expressions for the server failure time distribution function.

This thesis contributes to the current Operations Research literature by uniting techniques from failure modelling, optimal replacement theory, and the optimal control of queues to solve two difficult problems in the theory of deteriorating queues. Though these distinct areas have been well developed, relatively little research can be found in the current literature on the optimization of queues with failing servers. The results of this research may potentially save organizations valuable resources by

eliminating superfluous maintenance activities and mitigating the risk of catastrophic failures.

1.3 Thesis Outline

The next chapter of this thesis examines the current body of literature in the realms of optimal replacement strategies and the optimal control of queueing service rates. In addition, a method for determining the failure time distribution of the server is reviewed.

In chapter 3, the system under consideration is mathematically characterized, and the long-run expected cost rate is examined. The first subsection of that chapter proves the convexity of the long-run expected cost rate with respect to the replacement time. A simple solution procedure using the secant method is then discussed. The next subsection presents the necessary modifications to the model to examine the optimal service rate control problem, and a numerical gradient search method is discussed as a solution technique. Chapter 4 presents several numerical examples of both problems, and the results are discussed and compared with simulation data. Finally, chapter 5 provides concluding remarks and future research directions.

2. Literature Review

There are two main areas of literature relevant to this thesis: i) optimal replacement strategies for stochastically deteriorating systems, and ii) the optimal control of queueing service rates. As such, literature concerning elements of these two topics is presented.

2.1 Optimal Replacement Models

Survey papers by McCall [25], Pierskalla and Voelker [31], and Valdez-Flores and Feldman [37] review the optimal replacement literature from the early 1950s through the late 1980s. These papers provide a good summary of the optimal replacement literature and models during that period.

Valdez-Flores and Feldman [37] divide optimal replacement models into four broad categories, but underlying all four categories is the same basic premise; the objective is to minimize some given cost function, or maximize a profit function. A policy is said to be optimal if no other policy returns a lower cost or higher profit. The first category is inspection models where the amount of wear accumulated by the machine is unknown. Additionally, it may be unknown if the system has failed. The objective of an inspection model is to determine the optimal inspection and repair policy to minimize costs. The second category of replacement models is minimal repair models. A minimal repair model is usually used in complex systems with several components where, each time the system fails, a decision is made to either replace the entire system or repair the failed component. Valdez-Flores and Feldman [37] define a minimal repair to be one in which the repair of the failed component restores function to the system, but the failure rate of the overall system remains unchanged. The question then becomes,"When is it optimal to replace the entire system as opposed to performing minimal repairs?"

The third category of optimal replacement models is the replacement of systems that degrade due to random damage shocks, also called shock models. These models assume that the system is exposed to randomly occurring shocks, each shock inflicting a random amount of damage, with damage accumulating additively until the system is replaced or fails. These models are usually not age replacement models, in that most shock models give conditions for which there exists a control-limit policy. A control-limit policy is a rule in which replacement occurs when accumulated damage exceeds some threshold α or at failure. The final category of optimal replacement models reviewed in [37] is a miscellaneous category. Here Valdez-Flores and Feldmann [37] describe a wide variety of possible models including completely observable models, partially observable models, replacement with spares that are not on hand, and models for systems not in continuous operation. To elaborate, a completely observable model is one in which the state of degradation of the system is always known and failures are detected instantaneously. A partially observable model is one in which the true state of the system is not known but can be inferred from some other easily observable variable. This leads to the case in which the only available information is the probability of the system being in a particular state.

One example of an inspection model is provided by Jeang [16], who developed a linear tool-wear model based on linear wear of a tool with normally distributed linear coefficients. This model in turn allows the author to apply Bernstein's distribution [3] to represent the lifetime of the tool. This model assumes costs are incurred for replacement of the tool, sudden tool failure during processing, and for diminished quality due to imperfections in the workpiece caused by a worn tool. The author provides a method of calculating the optimal replacement time, as well as the optimal inspection period and initial offset of the tool to minimize the objective function. These results are relevant to the problem of this thesis in that the methodology is applicable to analyzing the system. While the underlying failure distribution of the

tool in [16] differs from that under consideration in this thesis, the results of [16] would suggest that a stationary optimal policy should exist.

There exists a large body of recent shock model literature which, at first glance, does not appear to be relevant to the system in this thesis. However, the general solution techniques are often independent of the method of wear accumulation. Using the results of Kharoufeh [19], one may compute the distribution of failure time for each tool and therefore have some information on the about the current state of the tool. This makes these papers, especially those that deal with partial or imperfect information, valid methodologies.

An interesting shock model was analyzed by Waldmann [38] who incorporates a random environment that affects the magnitude of the shocks received by the system. The author provided a proof of the optimality of a general state-dependent control-limit policy and derives bounds on the critical values. This paper is useful in that it gives conditions for which an optimal control-limit policy will exist in a random environment. These types of optimal policies are also considered by Perry [30], who considered a shock model with linear restoration between the arrivals of shocks. Nakagawa [26] developed a shock model for n units in parallel, where the optimal control policy was to replace the system if k of the n components had failed.

Hopp and Kuo [15] develop a shock model for the maintenance of aircraft engines while taking imperfect information into account. This differs from the traditional control-limit policies in that the underlying amount of wear on the system is unknown. In addition, the shocks are assumed to arrive according to a non-stationary Poisson process. The optimal inspection and replacement policies were found by discretizing the state space and solving the resulting Markov decision process. Another example of a shock model is due to Hopp and Nair [14] who account for technological change in the replacement model. This model relaxes the assumption that the only reason to replace a system is due to wear, and introduces technological advancement as another possible reason for replacement. The model assumes that there can be

exactly one technological improvement and that its arrival is unknown, but predicted via technological forecasts. The optimal policies generated by the author are still of the control-limit type; however they are not stationary because the technological forecast is assumed to be non-stationary.

Finally, Hopp and Wu [13] derive a multi-action model for a standard discrete time maintenance problem with imperfect information. The system is assumed to have n states where state 1 is the "best" state and state n is the worst. At each decision point the operator must choose from a set of actions, $A = \{0, 1, \ldots, n\}$ that deterministically move the state of the system to the level of the action taken (choosing action 0 indicates doing nothing). The authors examine two cases, state-dependent and state-independent maintenance. In state-dependent maintenance, if the operator chooses action a and the system is actually in state $i \leq a$, then the system remains in state i. Under this condition, there does not exist a control limit structure, but computation of an optimal policy is still possible utilizing a finite Markov decision process formulation. The state-independent results are far stronger, however, where they take on the familiar control-limit structure.

Rounding out the categories of replacement models is the catch-all "miscellaneous" category. In what follows, models which do not fit neatly into any of the previous categories are examined. Shirmohammadi, *et al.* [34] develop a model where the system degrades but is repaired in one of two ways. First, it is repaired on a fixed, time-based schedule. Second, the system undergoes emergency repair whenever the system fails prior to periodic maintenance. The authors define a policy where the system is repaired on a periodic basis, but if an emergency repair is required within a certain time before the renewal time, the scheduled periodic maintenance should be skipped. The authors then derive a method of solving for the time radius in which the periodic maintenance is skipped if an emergency failure is observed. This type of behavior could be useful in the context of the problem examined in this thesis, where a new replacement policy might be to fix each individual server as it failed,

then replace the system after some length of time. In this case, it may be possible to find a threshold number of servers s such that if s servers have failed within a certain radius of the replacement time, then that replacement should be skipped. This is an avenue of potential future research and is not studied here.

Another paper due to Zhou, *et al.* [40] presents a cutting tool model based on perfect information. This model assumes both the capacity of each new tool and the capacity consumption of each workpiece are, respectively, i.i.d. random variables. The authors model the system as a renewal process where the system is replaced whenever the remaining capacity of the current tool drops below the control-limit. The optimal control-limit is found by differentiating the long term expected cost rate and setting it equal to zero. This methodology is not unlike the proposed age replacement methodology used in this thesis.

Koyanagi and Kawai [21] examined an $M/G/1$ queue where the server degrades according to an underlying finite-state Markov chain: the server has $s+2$ states $S = \{0, 1, \ldots, s, s+1\}$ where state 0 indicates a "good as new" server and state $s+1$ indicates a failed state, with states 1 through s denoting successively increasing levels of wear. The authors assume that when a system is repaired, all customers in the system are forced out of the system incurring a cost per ejected customer, and all new arrivals are rejected, again incurring a cost per rejected customer. Using a semi-Markov decision process and dynamic programming, an optimal policy based on both the number of customers in the queue and the state of deterioration of the server is computed. This optimal policy has a two dimensional switch curve structure. The threshold property, the one dimensional analog to the switch curve structure, implies that if the state variable exceeds the threshold, a repair should be undertaken. The switch curve structure is similar in that it partitions the state space, which in [21] is a two-dimensional plane, into two partitions. A simple example of this structure is illustrated in Figure (2.1). The optimal policy is to do nothing if the state vector lies within Region A of the state space, and replace if the state vector is in Region B.

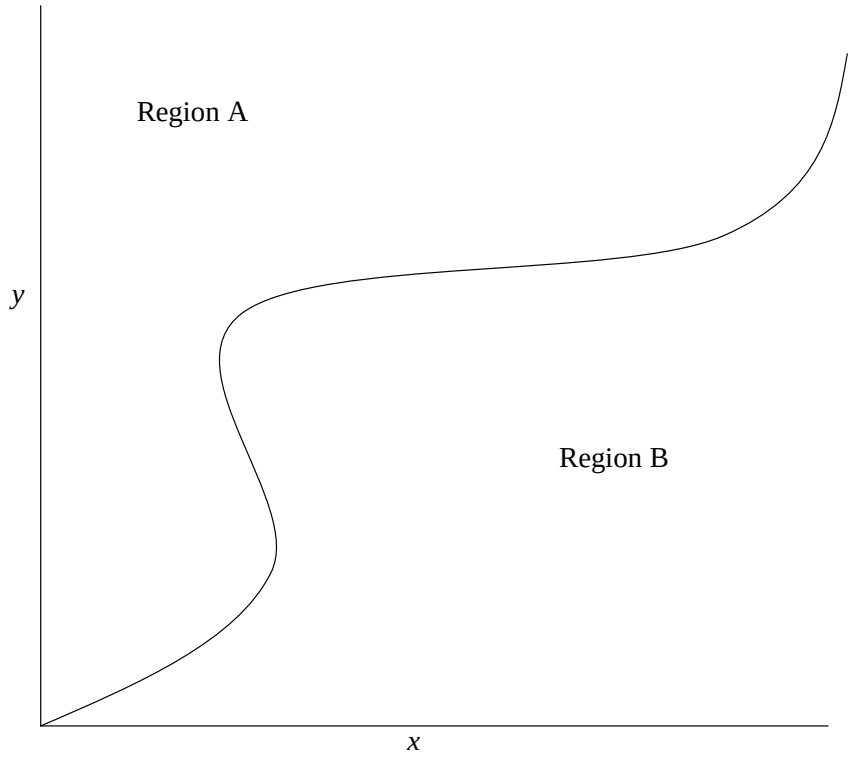

Figure 2.1 An example of a switch curve structure.

This methodology may also be valid for the analysis of the system in this thesis, even though there are significant differences between the respective objective functions.

2.2 Optimal Service Rate Control

This section shifts the focus from determining when the server should be replaced, to the problem of optimally controlling the service rate of the server(s). While none of the following papers examine a deteriorating server, they provide background and foundational methodologies that can be modified to optimally control a server that is subject to wear.

The optimal service rate problem describes scenarios wherein it is possible to adjust the service rate of the server in a queueing system with the purpose of minimizing some cost function. Many real-world systems have this ability, most

notably cutting and machining tools which can be operated at different speeds, but tend to induce increased cost at higher operational speeds. Crabill [6] provides the basis of this area of the literature, he derives a model where the state space of the underlying queue is infinite (i.e., the waiting area of the queue has infinite capacity), but that there is a finite number of allowable service rates. Crabill attacks the problem by first solving a finite-state problem, and then extending those results to the infinite case. In the finite case, Crabill shows the existence of an optimal policy such that the solution partitions the state space of the queue into K disjoint connected subsets with a single service rate used in each subset. Moreover, the policy is monotone, which implies that the service rate does not decrease as the number of customers in the system increases. In the infinite case, it is possible to lose that property, and so the assumption that the holding cost with i customers in the system approaches ∞ as $i \to \infty$ is applied in order to retain the monotonicity. These results are proven in the case where the number of allowable service rates $K = 2$, but Crabill notes that K can be increased through straightforward yet tedious calculation. Sabeti [32] examined the $M/M/1/K$ case under a slightly different cost structure, but found the same type of optimal policy.

A seminal work by Stidham and Weber [35] extended these results by rigorously proving that there exists a monotonic, stationary policy that is average-cost-optimal for an $M/M/1$ queue under moderate conditions. Specifically, the assumption of finite allowable service rates is relaxed so that the allowable service rates need only be bounded above. Stidham and Weber's work [35] also contains other results, including the existence of optimal, monotonic, stationary policies for queues under various other constraints such as no arrivals, or alternate queueing disciplines. This paper is critical to this thesis effort in that it proves the existence of a stationary, optimal policy for at least the $M/M/1$ case. Jo [17] also studied the $M/M/1$ queue, and derived sufficient conditions for optimal policies in both the finite and infinite

planning horizons. The optimal policies were shown to have a monotonic structure in both cases, though Stidham and Weber's [35] proof was more elegant.

George and Harrison [9] further extend the results of Stidham and Weber [35] by deriving the optimal service rate of a $M/M/1$ queue in which the allowable action space is uncountably infinite and unbounded, $A = [0, \infty)$, as well as dropping assumptions on holding costs. George and Harrison note that under these assumptions there is no general result that guarantees an optimal policy. To circumvent this problem, the authors truncate the holding cost such that after some point N, there is no increase in the holding cost for an additional customer in the queue. George and Harrison then rigorously prove that an optimal policy must exist for the altered problem, and then show that these altered problems converge to the optimal policy of the original problem as N gets large. [9] is useful to this research effort in that it allows more latitude in defining the holding costs. Although the relaxation of the upper bound requirement for the service rates is useful, most real world systems posses a meaningful upper bound on the service rate due to physical considerations.

Grassmann, *et al.* [10] approach the problem of optimal service rates from a completely different direction. The authors derive one optimal service rate for an $M/G/1$ queue in steady state via differentiation of the cost function. Using a general cost structure, the authors are forced to derive the number of customers in the system and the throughput of the system. They do this using Bayes' rule to relate the probability of having i customers before an arrival, which is commonly known to be equal to the probability of having i customers after a departure from the system since Poisson arrivals see time averages (PASTA). Once this relation is known, the throughput can be derived in terms of only the effective arrival rate and the probability of having i customers after a departure. This in turn allows them to derive the expected number of customers in the queue. Then, taking the derivative of the overall cost function, which is difficult, the optimal solution is computed. These results are useful in that they apply to queues with generalized service rates,

which implies that some solution should exist in the $M/G/1$ case of the deteriorating system under consideration in this research.

Doshi [8] examined the continuous-time control of an $M/G/1$ queue by examining the residual workload of the system at any time t, where continuous-time control implies that the "operator" of the system is able to alter the service rates at any time, not only at arrival or service completion times. The author derives sufficient conditions for a stationary policy to be optimal. Also, assuming the service and holding costs are non-decreasing and convex, the author shows that these sufficient conditions are met by a monotonic policy.

2.3 Queues Subject to Failure

The next section of this literature review considers one partial bridge between the two areas discussed above, i.e. those of failure models and queueing optimization. This section considers queueing models in which the server is subject to breakdowns, but there is still little on the optimization of these systems.

There exist several papers in the literature involving queues subject to breakdowns. One such paper is by Thiruverngadam [36], in which a $M/G/1$ queue subject to server breakdowns is examined. Thiruverngadam examines three variations of this theme. The first case models the situation where multiple overlapping breakdowns are possible, and customers will be served only when all breakdowns have been cleared. The second case assumes that one breakdown causes the entire system to cease functioning, and hence there can be only one breakdown at any time. The last case does not allow for idle failures. That is, a breakdown cannot occur when the queue is empty. Each case is modelled as a multi-class queueing system under the "preemptive resume priority." For all cases, Thiruverngadam derives the expected number of breakdowns, the expected number of customers in the queue, the stability condition and the steady state probabilities of the system. Thiruverngadam uses

partial differential equations with boundary conditions to derive these results. The solution is in the form of Laplace transforms. This methodology, while powerful, does not lend itself to the problem in this thesis due to the assumption that breakdowns occur according to a Poisson process. It does however provide a background for examining queues with breakdowns, and may be an excellent technique to use when optimizing queues with exponential failure times.

Avi-Itzhak and Naor [4] tackled a similar problem as Thiruverngadam [36], examining three additional cases. Their first model describes a situation where the repair process does not begin unless a customer is present to discover the breakdown. The next model presents the case where repair actions begin at the request of a customer who wishes to increase the standard of service. The final model considers the case where failures can only occur when no customer is being serviced. The authors derive the expected waiting time, and via Little's Law, the expected number of customers in the queue. This analysis is done from a probabilistic viewpoint, hence no partial differential equations are utilized, and all results are in terms of the expected values of the service and repair time distributions.

Mitrany and Avi-Itzhak [24] extend the work of Thiruvengadam, Avi-Itzahk and Naor by allowing for more than one server. Mitrany and Avi-Itzhak's model is roughly equivalent to Thiruvengadam's [36] model 2 and Avi-Itzhak and Naor's [4] model 2, where a breakdown causes the server to cease functioning, and hence cannot have another breakdown until it is repaired. The authors note that this can be viewed as a preemptive priority system with two classes of customers where there are N high priority customers. The authors derive the generating function of the long-run number of customers in the system, and from this the long-run expected number of customers in the system. The authors calculate two specific cases. In the first case, N is assumed to be 1, and these results are shown to agree with [4]. The second case, where $N = 2$, yields a complex generating function that cannot be easily computed, and hence, the authors use numerical methods to find the expected

number in the system for various parameter values. This paper is more applicable to this thesis than the previous two, since the model under examination here also assumes multiple servers.

A final extension of the previous series of articles is due to Neuts and Lucantoni [29]. The authors examine a model similar to the one of Mitrany and Avi-Itzhak [24], except that there are a limited number of repair crews (Mitrany and Avi-Itzhak in effect examined the case where the number of repair crews was equal to the number of servers). The authors develop numerical algorithms to compute the steady-state probabilities and waiting time distribution of the system. Numerical computation is again required because, as in [24], these problems are not analytically tractable in general.

Another way to handle server failures in queueing is via retrial queues. Kulkarni and Choi [23] examine one such retrial queue that is subject to breakdowns and repairs. In this retrial queue, customers are assumed to arrive according to a Poisson process and require i.i.d. service times. If the server is empty, the arriving customer enters service. If the server is busy, the customer conducts a retrial, or attempts to join service again after a random, exponentially distributed amount of time. The sever is assumed to have an exponential lifetime, and repairs are assumed to have a general distribution. The authors examine two models. In the first model, if a customer's service is interrupted by a breakdown, then that customer can either leave the system or join the retrial queue. In the second model, the customer is allowed to remain at the service station until the server again becomes available. Kulkarni and Choi tackle this problem by examining an embedded Markov chain within the non-Markovian stochastic process and derive the limiting probabilities of the system from this Markov chain. Both the existence and the limiting behavior of the system are rigorously proven. Additionally, Wang, *et al.* [39] derived several reliability measures for a similar system. These measures include availability, failure frequency, and the reliability function.

2.4 Server Failure Distributions

The last section of this literature review considers articles that are useful and necessary for the problem presented in this thesis. The first series of articles addresses the failure distribution that is used to model the failure times of the server in this thesis. Other papers examine previous work regarding queueing systems in random environments.

In the previous subsections of this review, there have been many models for the failure times of the object under consideration, from the shock models of subsection 2.1 to the assumed exponential failure time in all of subsection 2.3. Without quantifying these failure distributions, there is no way to make meaningful statements about the behavior of any system subject to failure.

The failure distribution used in this thesis was derived by Kharoufeh [18]. In this paper, the author derives explicit results for the lifetime distribution and the moments of a system that accumulates damage linearly, but with rates that are dependent on an external random environment. These results are in the form of multi-dimensional Laplace-Stieltjes transforms. Kharoufeh and Sipe [19] extended the results of [18] by reducing the multi-dimensional transforms to one-dimensional transforms.

Because this failure distribution is subject to a random environment, it is worthwhile to examine other articles concerning queueing systems in random environments. The first article in this series is the seminal work by Neuts [27]. In that paper the author examined a $M/M/1$ queue subject to an environment defined by a m-state continuous-time Markov chain. Specifically, when the environment is in state j, the mean arrival rate is assumed to be λ_j and the mean service rate is assumed to be μ_j. The author develops methodologies to compute the length of a busy period, the steady-state probabilities, and the waiting time distributions. Once the steady state probabilities have been computed, it is a simple matter to compute all

the moments, marginal and conditional densities of the queue length. These results are critical to the research presented in this thesis in that the steady state probabilities of the complicated queueing system under examination can be computed. In [28], Neuts extended the results of [27] to the $M/M/c$ queue.

In this thesis the objective is to bring together the research on optimal replacement models and queue optimization to minimize the long-run expected cost rate of a queue subject to server breakdowns. The majority of previous research in these areas assumed single server systems with exponential failure or no server failure at all. This thesis assumes a multiple-server queueing system where each server is subject to continuous wear dependent on a random environment that dictates the service distribution of the servers through the mean service rates and the associated wear rate of the servers. This process is very similar to the model examined by Kharoufeh [18] and Kharoufeh and Sipe [19]. When failures occur, an external service provider will be utilized to process the workload of the failed servers. The k-server system will be maintained according to a k-replacement policy, where all servers are replaced every T time units. The goals of this thesis are two-fold. The first goal is to compute the optimal replacement time T, where the mean service rates are constants. The second goal is to compute the mean service rate in each environmental state such that the long-run expected cost rate is minimized. This chapter has given an overview of the literature relevant to the study of this system, as well as fundamental literature on queues subject to breakdowns. In the next chapter, both the model and solution methodology are formally described.

3. Mathematical Model Description

The optimal time to replace k servers and the optimal mean service rates of a queueing system subject to the influences of a random environment are examined in this chapter. The queueing system under consideration is a variation of a $GI/G/k$ queueing system, where the mean service rate is dependent on the state of the random environment. The mean service rate when the random environment is in state j is denoted by μ_j. The mean arrival rate is assumed to be equivalent for all states of the environment and is denoted by λ.

The random environment is modelled as a continuous-time stochastic process $\{Z(t) : t \geq 0\}$ on a finite sample space $S = \{1, 2, \ldots, \ell\}$. It is assumed that $\{Z(t) : t \geq 0\}$ is an ergodic process and thus possesses a steady-state probability distribution given by

$$q_j = \lim_{t \to \infty} P\{Z(t) = j\}, \quad j \in S. \tag{3.1}$$

Define the stochastic process that describes the number of jobs in the system by $\{X(t) : t \geq 0\}$ where the state space of this process is $E \subseteq \{0, 1, 2, \ldots\}$.

It is assumed that all k servers are identical. Because the performance metric of interest is the long-run expected cost rate, the steady-state behavior of this queue is examined, and hence the stability condition is assumed to be satisfied. Explicitly, it is assumed that

$$\frac{\lambda}{k\bar{\mu}} < 1 \tag{3.2}$$

where

$$\bar{\mu} = \sum_{j=1}^{\ell} q_j \mu_j \tag{3.3}$$

represents the mean service rate of the k servers. Denote the accompanying steady-state probability distribution of the long-run number of jobs in the system as

$$p_j = \lim_{t \to \infty} P\{X(t) = j\}, \quad j \in E. \tag{3.4}$$

Each server accumulates wear over time as a function of its mean service rate. An example of this behavior is an outside-diameter grinding wheel which wears faster as the processing speed increases. It is assumed that the mean service rates depend explicitly on the state of a governing random environment. That is, if $Z(t) = j$ with $j \in S$, then the mean service rate of servers is μ_j. This implies that the rate of wear accumulation must also depend on the governing random environment. Let $r(j)$ denote the rate of wear accumulation when $Z(t) = j$. Let $\{Y_m(t) : t \geq 0\}$, $m = 1, 2, \ldots, k$, be the stochastic process which represents the cumulative wear on server m up to time t. Define x to be the maximum allowable wear on each individual server, such that once the wear on any given server exceeds x, that server has failed. Then the state space of $Y_m(t)$ is a subset of $\mathbb{R}_+ \equiv \{x : x \in \mathbb{R},\ x \geq 0\}$. Finally, for each individual server define

$$T_x^m = \inf\{t : Y_m(t) \geq x\} \tag{3.5}$$

as the instant of time that the mth server fails. It is also assumed that the distribution of T_x^m is stationary, and that it has cumulative distribution function (CDF) denoted by $F_x^m(t)$. Note that because all servers are assumed to be identical, the failure times T_x^m are identically distributed and, hence, $F_x(t) = F_x^m(t)$ for all m.

It is further assumed that there is no preference in choosing servers. That is, a waiting customer chooses the first available server. If faced with N available servers, the customer at the head of the waiting line chooses any one of the available servers with probability $1/N$. Whenever a server fails, it is instantaneously replaced by an external server working at the same rate as the failed server, and hence, there is no interruption in the service of customers. The objective is two fold. First, determine the optimal time to replace servers using a k-replacement policy. Second, compute the values of μ_j for each state in the random environment such that the long-run expected cost rate is minimized.

The sequence of server replacement times constitutes a renewal process. Because this thesis assumes an age replacement model, the system renews once every T time units, and hence, the inter-renewal time is always T time units in length, where T is a deterministic value. From rudimentary renewal theory, a stochastic process $\{N(t) : t \geq 0\}$ whose state space is a subset of $\mathbb{Z}_+ \equiv \{z : z \in \mathbb{Z}, z \geq 0\}$ is said to be a renewal process if it is a random walk that transitions only in the positive direction, i.e. for $t_2 \geq t_1$, $N(t_1) = j \Rightarrow N(t_2) \geq j$, and that inter-renewal times are non-negative, independent, identically distributed (i.i.d) random variables $\{\tau_n\}$ with common CDF given by $G(x) := P\{\tau_n \leq x\}$, for all n. With respect to the k-server queueing system, define the stochastic process $\{N(t) : t \geq 0\}$ as the number of times all k servers have been replaced up to time $t > 0$. Then it is clear that the state space of this process is a subset of $\mathbb{Z}_+$ and that this process transitions by unity in the positive direction. Moreover, the inter-replacement times are independent and identically distributed since $\tau_n = T$ for all n. Therefore, this system constitutes a renewal process, where the nth cycle is defined to be the nth inter-renewal time, or the time between the $(n-1)$st and the nth renewal.

Figure 3.1 depicts a sample path for one of the k servers. In the first cycle, note that this server is replaced at time T. However, in the second cycle the cumulative degradation, $Y_m(t)$, reaches the threshold x before time $2T$ and the server has failed. This occurs at the time labeled T_x^m. The server remains in a failed state until the next group replacement, which occurs at time $2T$.

The main benefit of the renewal structure is that it allows the utilization of the renewal reward theorem. Let $C(t)$ denote the cost incurred by the system up to time t, and let R_n denote the cost incurred by the system during the nth cycle. Define $r = E[R_n]$ and $\tau = E[\tau_n]$ where $0 \leq E[R_n] < \infty$ and $0 < E[\tau_n] < \infty$. Then the renewal reward theorem states (see [22])

Theorem 3.1 *As* $t \longrightarrow \infty$,

$$\frac{E[C(t)]}{t} \longrightarrow \frac{r}{\tau}. \tag{3.6}$$

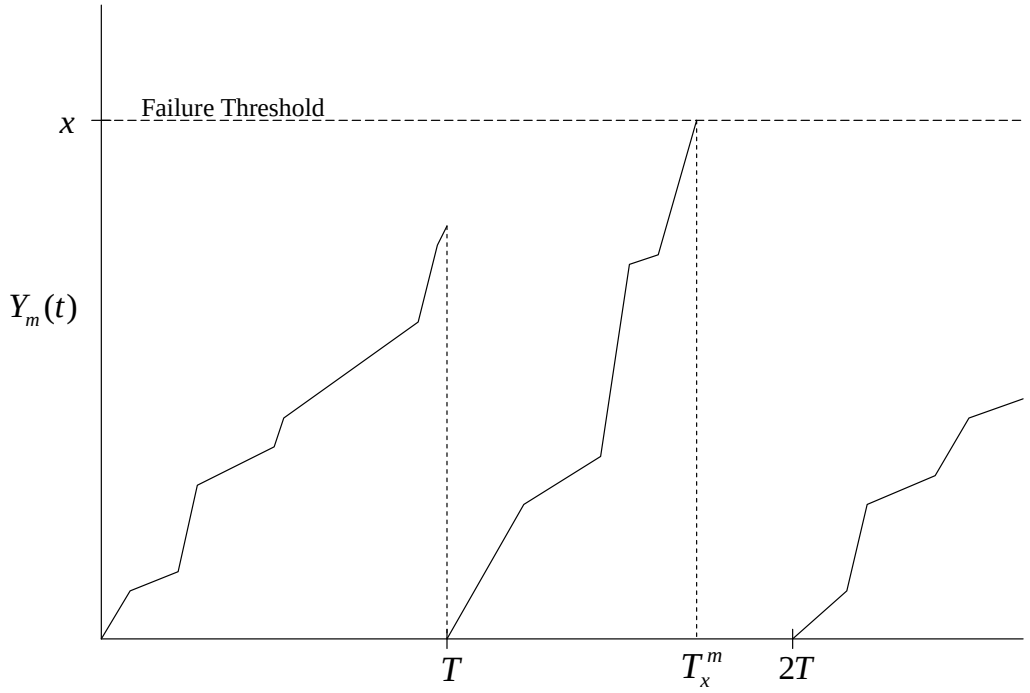

Figure 3.1 A sample path of the wear level of one server.

Application of Theorem 3.1 requires explicit specification of the expected cost equation of this generalized queueing system. The overall objective is to minimize the long-run expected cost rate of the queue. The cost function is assumed to have four main components, each of which is defined in terms of the cost per cycle.

First, C_N is defined as the cost of replacing the servers during a cycle. Define the constant c_N as the fixed cost per unit replaced when a server is replaced. As an example, if there are k servers in the system and they are all replaced every T time units, then $C_N = kc_N$. Further, because this value is deterministic, it is obvious that

$$E[C_N] = kc_N. \tag{3.7}$$

Next, there is a cost associated with the time a job spends in the system; this cost per cycle is denoted C_H. Define c_H as the holding cost per job per unit time. Obviously, at any instant of time t, the holding cost rate (cost per unit time) is simply the number of jobs in the system multiplied by c_H. Then the expected value

of the holding cost rate is c_H multiplied by the long-run expected number of jobs in the system, which is commonly denoted as L. It follows that the expected holding cost over one cycle is the expected cost rate multiplied by the expected cycle length, which in this case is T. Therefore,

$$E[C_H] = c_H LT. \tag{3.8}$$

The third component of the cost function represents the cost incurred by simply operating the system denoted per cycle as C_W. It is assumed that this cost is a function of the mean service rate, which is in turn a function of the random environment. Let the constant $c_W(\mu_j)$ denote the cost of processing one customer when the mean service rate is μ_j, $j \in S$. By the work conservation property of this system, the expected number of customers coming into the system must equal the expected number exiting the system. Therefore the expected number of customers entering the system over a cycle of length T is simply λT and the expected work cost per unit time is the expected number of customers served over the cycle, multiplied by the expected cost per customer. This is

$$E[C_W] = \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j). \tag{3.9}$$

Equation (3.9) assumes that all customers are processed by servers that have not failed; however, in this research the servers may fail. Hence, there must be some additional cost incurred by the system due to these failures, the final component of the cost function.

The final component of the cost function is associated with a premature server failure. As noted previously, when a server fails before it is replaced, the jobs that would have been processed by the failed server are instead processed by an external service provider at increased cost. Let C_F denote the additional cost incurred per

cycle because of failed servers. Further, let the constant c_F denote the total additional cost of serving one customer using the outside service facility. The unit cost c_F encompasses not only the higher price charged by the external service provider, but may also include inconvenience costs due to rerouting jobs to the external server or the cost of customer ill-will generated because of the transfer. The expected value of C_F is the expected number of customers served by the outside service facility multiplied by c_F.

To compute the expected number of customers served by the external facility, it is necessary to condition on the random number of failed servers in one cycle. First define $M(t)$ as the (random) number of failed servers up to time t with $M(0) \equiv 0$. Since the probability that a server has failed by time t is known via the CDF $F_x(t)$, it is clear that $M(t)$ has a binomial distribution with parameters k and $F_x(t)$. Stated more precisely, for $m = 1, 2, \ldots, k$,

$$P\{M(t) = m\} = \frac{k!}{m!(k-m)!}(F_x(t))^m(1 - F_x(t))^{k-m} \tag{3.10}$$

and

$$E[M(t)] = kF_x(t). \tag{3.11}$$

Next, let T_F^m be the length of time between the mth server's failure and the next replacement time. Then

$$T_F^m = \max\{T - T_x^m, 0\} = (T - T_x^m)^+. \tag{3.12}$$

Since all servers are assumed to be identical, it is clear that $E[T_F^i] = E[T_F^j] = E[(T - T_x)^+]$ for all $i, j = 1, 2, \ldots, k$. This value may be computed via conditioning

on the random failure time, T_x, as follows:

$$\begin{aligned}
E\left[(T-T_x)^+\right] &= E\left[E\left[(T-T_x)^+ | T_x\right]\right] \\
&= \int_0^\infty E\left[(T-T_x)^+ | T_x = t\right] dF_x(t) \\
&= \int_0^T E\left[(T-T_x) | T_x = t\right] dF_x(t) \\
&= \int_0^T (T-t)\, dF_x(t) \\
&= T\int_0^T dF_x(t) - \int_0^T t dF_x(t) \\
&= TF_x(T) - \int_0^T t dF_x(t) \\
&= TF_x(T) - \left(tF_x(t)\Big|_0^T - \int_0^T F_x(t)\, dt\right) \\
&= TF_x(T) - TF_x(T) + \int_0^T F_x(t)\, dt \\
&= \int_0^T F_x(t)\, dt. \qquad (3.13)
\end{aligned}$$

Finally, observe that because there is no preference on which server is assigned a particular job, in the long run each server processes an equal share of the customers that exit the system. Now recall that when the queue is in equilibrium, the expected number of customers arriving to the system during a time period must equal the expected number of customers exiting the system over the same period. Define N_F as the random number of customers that undergo service during the time between a server failure and the end of a cycle (i.e., the time between failure and replacement). Then

$$E[N_F] = \lambda E[(T-T_x)^+] = \lambda \int_0^T F_x(t)\, dt. \qquad (3.14)$$

Since $E[N_F]$ is the expected number of customers that undergo service, either from the internal servers or the external processing facility, and each server processes an equal share of the exiting customers in the long run, the expected number of

customers that undergo service with a failed server is the proportion of failed servers multiplied by $E[N_F]$. Therefore, the expected number of customers that undergo service with a failed server during one cycle can be computed via conditioning on the number of servers that failed during the cycle as follows:

$$\begin{aligned} E[C_F] = E[E[C_F|M(T)]] &= c_F E[N_F] \sum_{m=0}^{k} \frac{m}{k} \frac{k!}{m!(k-m)!} (F_x(T))^m (1 - F_x(T))^{k-m} \\ &= c_F E[N_F] \left(\frac{E[M(T)]}{k} \right) \\ &= c_F \left(\lambda \int_0^T F_x(t)\, dt \right) \left(\frac{k F_x(T)}{k} \right) \\ &= c_F \lambda F_x(T) \int_0^T F_x(t)\, dt. \end{aligned} \tag{3.15}$$

At this point, it is important to note that the Laplace-Stieltjes transform (LST) of the distribution $F_x(t)$ has been derived by Kharoufeh and Sipe [19], and those results are used here. In [19], α is defined to be the initial distribution of the random environment $\{Z(t) : t \geq 0\}$. However, since this thesis assumes steady-state conditions for the random environment, set α to be the vector of steady-state probabilities of $\{Z(t) : t \geq 0\}$, previously defined as $\{q_n\}$. The next two definitions are the same as in [19]; define $\mathbf{R}_D \equiv diag(r(1), r(2), \ldots, r(\ell))$ and $\mathbf{Q}$ to be the infinitesimal generator matrix of the random environment process $\{Z(t) : t \geq 0\}$. Since $F_x(t)$ is the cdf of T_x^m, the LST of $F_x(t)$ is given by (cf. Kharoufeh and Sipe [19])

$$\tilde{F}_x(s) = \alpha \exp\left[\mathbf{R}_D^{-1} (\mathbf{Q} - sI)\, x\right] \mathbf{1}, \tag{3.16}$$

where s is the transform variable associated with time and $Re(s) > 0$, and $\mathbf{1}$ is a column vector of ones. Note that the Laplace transform (LT) of a function is equal to the LST of a function multiplied by s^{-1}. Then the LT of the CDF is

$$F_x^*(s) = s^{-1} \alpha \exp\left[\mathbf{R}_D^{-1} (\mathbf{Q} - sI)\, x\right] \mathbf{1}. \tag{3.17}$$

Hence the value of $F_x(t)$ can be found via the inverse Laplace transform of Equation (3.17) evaluated at the desired time t. For the sake of brevity, let

$$\Psi(s) \equiv \alpha \exp(\mathbf{R}_D^{-1}(\mathbf{Q} - sI)x)\mathbf{1}. \tag{3.18}$$

In Equation (3.15) it is clear that it is also necessary to know something of the integral of the CDF. Fortunately, this is relatively easy in the transform space. Applying fundamental results of Laplace transform theory, it is well known (see [22]) that

$$\mathcal{L}\left(\int_0^T F_x(t)dt\right) = \frac{F_x^*(s)}{s} = s^{-2}\alpha \exp(\mathbf{R}_D^{-1}(\mathbf{Q} - sI)x)\mathbf{1}. \tag{3.19}$$

Therefore, when this transform is inverted, it is evaluated at $t = T$.

The expected value of each component over one cycle has been examined, with the goal of finding an expression for the total expected cost per cycle. This cost for any cycle is given by

$$\begin{aligned} E[R_n] &= E[C_N] + E[C_W] + E[C_H] + E[C_F] \\ &= kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda F_x(T) \int_0^T F_x(t)\, dt \\ &= kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda \mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\} \mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\} \end{aligned} \tag{3.20}$$

so that this expected value is a function of both the replacement time T and the vector of service rates $\vec{\mu} \equiv \left[\begin{array}{cccc} \mu_1 & \mu_2 & \ldots & \mu_\ell \end{array}\right]$. Therefore, the objective function for

the minimization of the long-run expected cost rate is

$$\begin{aligned} g(T,\vec{\mu}) &= \lim_{n\to\infty} \frac{E[R_n]}{T} \\ &= \frac{kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda F_x(T) \int_0^T F_x(t)dt}{T} \\ &= \frac{kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda \mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\} \mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\}}{T} \end{aligned} \tag{3.21}$$

The following two sections address minimizing this objective function by controlling T and $\vec{\mu}$ respectively. In both cases $g(T,\vec{\mu})$ is defined in terms of complex inverse Laplace transforms that require numerical inversion techniques.

3.1 Optimal Replacement Time

The goal in this section is to compute an optimal value, T^* such that the long-run expected cost rate is minimized. To this end, assume that the service rate for each environmental state is given. Therefore in this section $\vec{\mu}$ is considered to be a constant vector that cannot be controlled, and hence, the wear rates, $r(j),\ j = 1, 2, \ldots, \ell$, are also considered to be constant.

The overall objective is to balance the benefits of frequent replacements with the increased cost of these replacements. Obviously, as the replacement time becomes shorter, the probability of failure decreases, and therefore, fewer customers will be processed by an outside service facility. However, replacing k servers can be a costly proposition, and hence, lengthening the replacement interval may decrease costs.

The first task accomplished in this section is the formulation of the mathematical program. Next, the convexity of the objective function with respect to T is proven. Finally, a numerical technique to find T^* is discussed. It is necessary

to accept approximate solutions due to the complex numerical Laplace transform inversions.

Let decision variable, T, represent the length of the replacement interval for the system. The mathematical program is thus

$$\min g(T) = \frac{kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda \mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\} \mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\}}{T}$$

subject to

$$0 < T < \infty \tag{3.22}$$

The objective is to find T^* such that Equation (3.22) is minimized. Proposition 3.1 confirms the existence of an optimal replacement time T^*.

Proposition 3.1 *The long-run expected cost rate of Equation (3.22) is convex in* T.

Proof. Let $u, v \in \mathbb{R}_+$ and $\gamma \in (0, 1)$. It will be proven that

$$g[\gamma u + (1-\gamma)v] \leq \gamma g(u) + (1-\gamma) g(v). \tag{3.23}$$

by showing

$$\gamma g(u) + (1-\gamma) g(v) - g[\gamma u + (1-\gamma)v] \geq 0. \tag{3.24}$$

Using Equation (3.22) the objective function is

$$g(T) = \frac{kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda F_x(T) \int_0^T F_x(t)dt}{T}. \tag{3.25}$$

Now for simplicity let $A = kc_N$, $B = c_H L + \lambda \sum_{j=1}^{\ell} q_j c_W(\mu_j)$ and $H(T) = c_F \lambda F_x(T) \int_0^T F_x(t)dt$. Therefore Equation (3.25) is simply

$$g(T) = \frac{A + B \times T + H(T)}{T}. \tag{3.26}$$

Therefore,

$$\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] = \gamma\left(\frac{A + Bu + H(u)}{u}\right) + (1-\gamma)\left(\frac{A + Bv + H(v)}{v}\right) - \left(\frac{A + B[\gamma u + (1-\gamma)v] + H[\gamma u + (1-\gamma)v]}{\gamma u + (1-\gamma)v}\right). \tag{3.27}$$

Using common denominators to combine the fractions yields

$$\begin{aligned}\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] = \\ \gamma v[\gamma u + (1-\gamma)v]\left(\frac{A + Bu + H(u)}{uv[\gamma u + (1-\gamma)v]}\right) \\ + (1-\gamma)u[\gamma u + (1-\gamma)v]\left(\frac{A + Bv + H(v)}{uv[\gamma u + (1-\gamma)v]}\right) \\ - uv\left(\frac{A + B[\gamma u + (1-\gamma)v] + H[\gamma u + (1-\gamma)v]}{uv[\gamma u + (1-\gamma)v]}\right).\end{aligned} \tag{3.28}$$

By arranging like terms simplifies Equation (3.28) to

$$\begin{aligned}\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] = \\ \frac{A\left[\gamma^2 uv + \gamma(1-\gamma)v^2 + \gamma(1-\gamma)u^2 + (1-\gamma)^2 uv - uv\right]}{uv[\gamma u + (1-\gamma)v]} \\ + \frac{B\left\{\gamma uv[\gamma u + (1-\gamma)v] + (1-\gamma)uv[\gamma u + (1-\gamma)v] - uv[\gamma u + (1-\gamma)v]\right\}}{uv[\gamma u + (1-\gamma)v]} \\ + \frac{\left\{[\gamma^2 uv + \gamma(1-\gamma)v^2]H(u) + [\gamma(1-\gamma)u^2 + (1-\gamma)^2 uv]H(v) + uvH[\gamma u + (1-\gamma)v]\right\}}{uv[\gamma u + (1-\gamma)v]}.\end{aligned} \tag{3.29}$$

Since $H(T) = \lambda F_x(T)\int_0^T F_x(t)dt$, it is necessary to realize that because $F_x(t)$ is a CDF, $F_x(t) \geq 0$ for all $t \in \mathbb{R}_+$. Therefore, zero is a lower bound for $F_x(t)$ and also

for $H(T)$. Noting that the B coefficients cancel out and setting $H(T) = 0$ yields

$$\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] \geq \frac{A\left\{[\gamma^2 + (1-\gamma)^2 - 1]uv + \gamma(1-\gamma)(u^2+v^2)\right\}}{uv[\gamma u + (1-\gamma)v]}. \quad (3.30)$$

Expanding the squared terms yields

$$\begin{aligned}\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] &\geq \frac{A\left\{[\gamma^2 + 1 - 2\gamma + \gamma^2 - 1]uv + (\gamma - \gamma^2)(u^2+v^2)\right\}}{uv[\gamma u + (1-\gamma)v]} \\ &= \frac{A\left\{2[\gamma^2 - \gamma]uv + (\gamma - \gamma^2)(u^2+v^2)\right\}}{uv[\gamma u + (1-\gamma)v]}. \quad (3.31)\end{aligned}$$

Now note that the denominator is a function of positive numbers, and hence can never be less than zero. Also, A is a constant cost, and is assumed to be greater than zero. Therefore Equation (3.31) less than zero only if the coefficient of A is less than zero. Thus, the coefficient of A must be examined. Note that $\gamma - \gamma^2 \geq 0$ for all $\gamma \in (0,1)$, and that $(\gamma - \gamma^2) = -(\gamma^2 - \gamma)$. Then

$$\begin{aligned}2[\gamma^2 - \gamma]uv + (\gamma - \gamma^2)(u^2+v^2) &= (\gamma - \gamma^2)[(u^2+v^2) - 2uv] \\ &= (\gamma - \gamma^2)(u-v)^2 \quad (3.32)\end{aligned}$$

and this coefficient must be greater than zero, then

$$\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] \geq \frac{A\left\{(\gamma - \gamma^2)(u-v)^2\right\}}{uv[\gamma u + (1-\gamma)v]} \geq 0 \quad (3.33)$$

and

$$\gamma g(u) + (1-\gamma)g(v) - g[\gamma u + (1-\gamma)v] \geq 0 \quad (3.34)$$

and the result is proved. ■

Since $g(T)$ is convex in T, it has at most one stationary point, and the global minimum is either at that stationary point, or the boundary points of the feasible set of T, i.e. $T = 0$ or $T = \infty$. Therefore, to find the stationary point, it is necessary to differentiate Equation (3.22) with respect to T, set the result to zero and solve for T. Applying the quotient rule to take the derivative of Equation (3.26) yields

$$\frac{dg(T)}{dT} = \frac{T\left(B + H'(T)\right) - \left(A + BT + H(T)\right)}{T^2}. \tag{3.35}$$

To find the stationary point, find T such that $g'(T) = 0$:

$$\begin{aligned}
0 &= \frac{T\left(B + H'(T)\right) - \left(A + BT + H(T)\right)}{T^2} \\
&= T\left(B + H'(T)\right) - \left(A + BT + H(T)\right) \\
&= T\left(B + H'(T) - B\right) - H(T) - A \\
&= TH'(T) - H(T) - A. \\
&= T\frac{d}{dT}\left[c_F\lambda F_x(T)\int_0^T F_x(t)dt\right] - c_F\lambda F_x(T)\int_0^T F_x(t)dt - A \\
&= c_F\lambda T\frac{d}{dT}\left[F_x(T)\right]\int_0^T F_x(t)dt + c_F\lambda T F_x(T)\frac{d}{dT}\left[\int_0^T F_x(t)dt\right] \\
&\quad -c_F\lambda F_x(T)\int_0^T F_x(t)dt - A \\
&= c_F\lambda T\frac{d}{dT}\left[F_x(T)\right]\int_0^T F_x(t)dt + c_F\lambda T[F_x(T)]^2 \\
&\quad -c_F\lambda F_x(T)\int_0^T F_x(t)dt - A
\end{aligned} \tag{3.36}$$

Once again, exploiting the fact that the Laplace transform of $F_x(t)$ is known, this equation can be expressed in terms of inverse Laplace transforms. First, by fundamental results of Laplace transform theory, it is well known (see [22]) that, for a continuous function $f(t)$,

$$\mathcal{L}\left\{\frac{df(t)}{dt}\right\} = sf^*(s) - f(0). \tag{3.37}$$

By Equation (3.17) it is clear that

$$F^*(s) = s^{-1}\Psi(s) \tag{3.38}$$

and hence

$$\mathcal{L}\left\{\frac{d}{dT}\left[F_x(T)\right]\right\} = s \times s^{-1}\Psi(s) - F_x(0) = \Psi(s). \tag{3.39}$$

Using the identity in Equation (3.37), Equation (3.36) can be rewritten as

$$\begin{aligned} 0 &= c_F\lambda T\mathcal{L}^{-1}\left\{\Psi(s)\right\}\mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\} + c_F\lambda T\left(\mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\}\right)^2 \\ &- c_F\lambda\mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\}\mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\} - A. \end{aligned} \tag{3.40}$$

Note that Equation (3.36) is a complex integral formula, which implies that it is impossible to isolate T even if $F_x(T)$ is analytical. Therefore, some numerical search procedure is required, regardless of the server failure distribution. Since the left-hand side of Equation (3.40) is zero, any numerical root finding technique can be used to find the optimal value T^*. This thesis employs a hybrid bisection and secant method (see [5]), which is reviewed further in chapter 4.

In the next section, it is assumed that the replacement time, T, is a constant, and the aim is to compute an approximate optimal service rate for each environmental state using a numerical search algorithm.

3.2 Optimal Service Rate Selection

The objective of this section is to compute an optimal vector of service rates $\vec{\mu}$ that minimizes the the long-run expected cost rate. In contrast to section 4.1 in which T^* was computed and $\vec{\mu}$ was held constant, in this section T is now fixed, and an "approximately optimal" value of $\vec{\mu}$ is computed.

Since the service rates vary in this section, the rate of wear in each environmental state must also vary. To this end, define $r_j(\mu_j)$ as the rate of wear while the environment is in state j and the service rate for that state is μ_j. It is assumed that $r_j(\mu_j)$ is non-decreasing in μ_j. Because $[r(j)]$ has changed, it is necessary to redefine the matrix $\mathbf{R}_D$ as well. Let

$$\mathbf{R}_D \equiv diag[r_1(\mu_1), r_2(\mu_2), \ldots, r_\ell(\mu_\ell)]. \tag{3.41}$$

The goal is to balance the benefits of faster service rates against the negative costs of higher failure rates. Higher service rates lead to cost savings in reduced holding costs, but increase the cost of operating the machines, and increase the failure rates. This increase in the failure rate, of course, increases the expected number of customers served at external service facilities.

Before defining the mathematical program, it is necessary to discuss constraints. First, in order to ensure the stability of the queueing system, it is necessary to ensure that all service rates are greater than the arrival rate, as per [12]. Further, in real world systems it is reasonable to assume the existence of some upper bound on the service rate, simply because it is impossible to work infinitely fast. In the context of this problem, it is easy to choose the constant cost rates to induce infinite optimal service rates. To guard against the trivial solution of $\mu_j = \infty$, $j \in S$, it is necessary to assign a real-valued upper bound, say ω, to each service rate.

Therefore, the nonlinear program (NLP) is of the form

$$\min g(\vec{\mu}) \quad = \quad \frac{kc_N + c_H LT + \lambda T \sum_{j=1}^{\ell} q_j c_W(\mu_j) + c_F \lambda \mathcal{L}^{-1}\left\{s^{-1}\Psi(s)\right\} \mathcal{L}^{-1}\left\{s^{-2}\Psi(s)\right\}}{T}$$

subject to

$$\mu_i > \lambda, \quad i = 1, 2, \ldots, \ell$$
$$\mu_i \leq \omega, \quad i = 1, 2, \ldots, \ell. \tag{3.42}$$

Because of the complex structure of the objective function, numerical analysis is the only possible solution technique available. Therefore, a gradient search algorithm is used to find an approximate optimal solution. This algorithm is slightly modified in that the gradient is estimated using central differences, as per [2]. These methods are examined more closely in chapter 4.

4. Numerical Experiments

In this chapter, the results of chapter 3 are illustrated on three notional example problems. Each of the examples are examined with respect to both age replacement and service rate selection. Moreover, the replacement time results are compared to simulated data to verify the accuracy of the results. When selecting the optimal replacement time, the objective is to find the value of T such that Equation (3.40) is equal to zero. This is accomplished via a combination of the bisection and secant methods [5], both of which are described. When selecting service rates, the object is to minimize the objective function of Equation (3.42) using the gradient search method.

For each example problem, numerical inversion of Laplace transforms is employed. This inversion (in one dimension) will be accomplished by the algorithm of Abate and Whitt [1]. Moreover, Monte Carlo simulation is utilized to verify the accuracy of the Laplace transform inversions. Specifically, a simulated CDF is used in the same solution procedure outlined in the previous paragraph instead of the numerical Laplace transform inversions. Each simulation uses 500,000 simulated failure times to evaluate the CDF every 0.0001 time units from the minimum simulated failure time to the maximum simulated failure time. The number of CDF evaluations is large because the methods presented here require numerical integration of the CDF via the trapezoidal method. All methods and algorithms were implemented in the MATLAB® environment, version 6.5.0 Release 13 and executed on a Pentium® III, 650 mhz notebook computer with 512 MB of RAM.

4.1 Numerical Analysis Techniques

This section briefly discusses the numerical techniques employed in the experiments. Two categories of techniques are used, root-finding algorithms and gradient search algorithms.

The objective of a root-finding algorithm is to find a root for some function f. The Bisection method is examined first. This method calls for the repeated halving of subintervals known to contain a root of the function. Figure 4.1 shows an example of the bisection method using an arbitrary function $f(x)$. The bisection method requires an interval $[a, b]$ such that $f(a) \times f(b) < 0$. In the example of Figure 4.1, points p_0 and p_1 are inputs to the algorithm, with $a = p_0$ and $b = p_1$. The next point p_2 is generated by taking the midpoint of a and b and replacing one of the endpoints of the interval. If $f(a) \times f(p_1) > 0$ then p_1 replaces the lower bound,

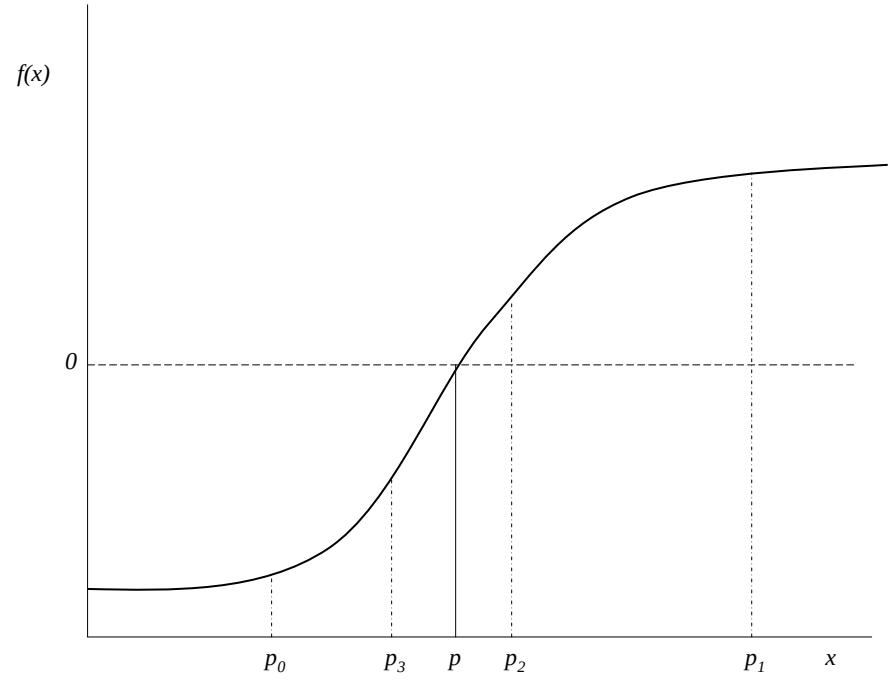

Figure 4.1 An example of the bisection method.

a; otherwise p_1 replaces the upper bound b. The midpoint of p_0 and p_1 is p_2, and because $f(p_0) \times f(p_2) < 0$, p_2 replaces the upper bound of the interval. The process is repeated, where p_3 is the midpoint of p_0 and p_2, which then becomes the lower bound of the interval because $f(p_0) \times f(p_1) > 0$. The main disadvantage of the bisection method is that it is slow to converge. However, it is not susceptible to the instability caused by little to no change in the function value as the secant method which is next described.

The secant method uses the x-intercept of the line connecting the previous two points in the iteration as the next point. Figure 4.2 shows one example of the secant method. Again, p_0 and p_1 are inputs to the system, but it is not required

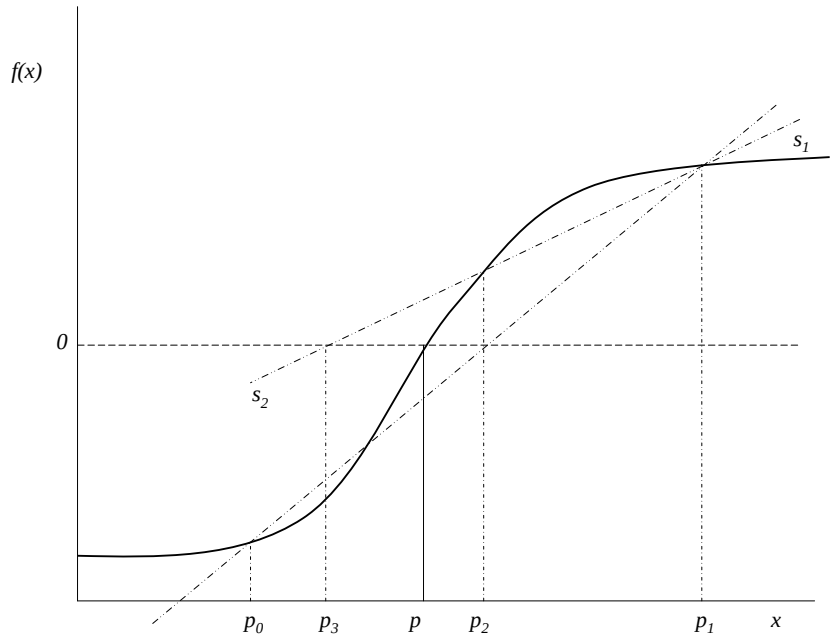

Figure 4.2 An example of the secant method.

that $f(a) \times f(b) < 0$. The next point, p_2, is the x-intercept of the line connecting p_0 and p_1, and p_3 is the x-intercept of the line connecting p_1 and p_2. The primary advantage of the secant method is increased convergence speed. However, in certain circumstances the secant method is unstable. If the function under consideration has a region where there is no change in the function, the secant line between two consecutive points may not actually intercept the x-axis. One example of this is shown in Figure 4.3. Because $f(p_1) - f(p_0)$ is essentially zero, the secant line never intersects the x-axis. The bisection method is used to combat this potential error. The bisection method is employed until $[|f(x_{k+1} - f(x_k)|]/f(x_k) < 0.1$ or 10,000 iterations is reached. The final two points of the bisection method are used as the starting points of secant method, which terminates when $p_{k+1} - p_k < \epsilon = 10^{-12}$ or when 10,000 iterations is reached. The purpose of using the bisection method

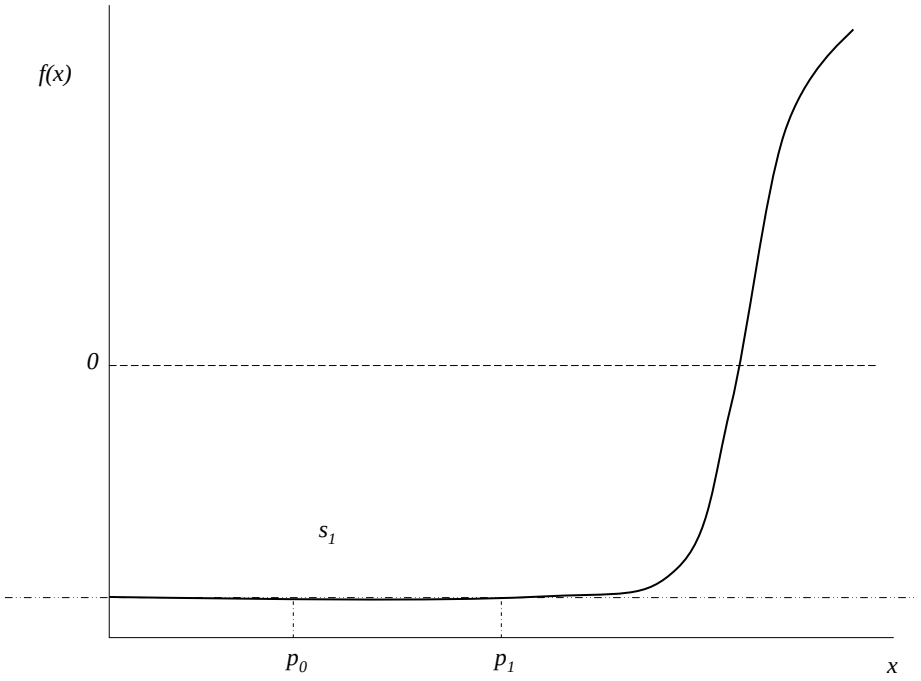

Figure 4.3 An example of the failure of the secant method.

is to have starting points close enough to the solution so that the secant method converges.

The gradient search algorithm, also commonly known as the steepest descent method, is the final numerical analysis technique used in this thesis. This method finds local minima for an arbitrary function $f : \mathbb{R}^n \to \mathbb{R}$. Essentially, this method has three steps. First, the function is evaluated at an initial approximation. Next, a direction from the initial approximation that results in a decrease in the value of the function evaluation is determined. Third, move an appropriate amount in this direction and call this point the next point in the iteration. The algorithm terminates when one of four conditions is met. First, the algorithm is stopped at 10,000 iterations, as an unsuccessful trial. Second, the algorithm is stopped if the norm of the gradient is close to zero, and this is considered a successful trial. Thirdly, if the step size in the direction of the gradient is less than $\epsilon = 10^{-12}$ the algorithm is

halted as a successful trial. Lastly, if $|f(x_{k+1}) - f(x_k)| < \epsilon$, the solution is stopped as a successful trial.

For the purposes of this thesis, the gradient is approximated using central differences as per [2]. The central difference approximating the gradient in the x_1 direction of the arbitrary function f is defined as

$$\frac{\partial f(x_1, x_2, \ldots, x_k)}{\partial x_1} = \frac{f(x_1 + c, x_2, \ldots, x_k) - f(x_1 - c, x_2, \ldots, x_k)}{2c} \tag{4.1}$$

where c is a small constant. For the purposes of this thesis, $c = 0.005$. Also note that the derivative of the simulated CDF (required in Equation (3.40)) will be approximated using central differences.

4.2 Example I

Consider a single outer diameter grinding wheel that grinds metallic workpieces to specific dimensions. Suppose that workpieces fall into one of two categories based on the material hardness and that workpieces in the second category have a higher hardness rating due to fluctuations in the manufacturing process. These components are assumed to induce more wear on the grinding wheel, but do not change the probability distribution of the time required to process the workpiece. Assume that the queueing system is $M/M/1$ with service rate parameter $\mu = 1.1$ and arrival rate parameter $\lambda = 1$.

The type of workpiece being processed at time t is denoted by $Z(t)$, (i.e. $Z(t) \in S = \{1, 2\}$ for all t) in this example. Assume that $\{Z(t) : t \geq 0\}$ can be appropriately modelled as a CTMC with infinitesimal generator matrix

$$\mathbf{Q} = \begin{bmatrix} -0.7 & 0.7 \\ 1.9 & -1.9 \end{bmatrix}. \tag{4.2}$$

Next, assume that the rate of wear incurred on the grinding wheel is equal to 1/10th the service rate, and that workpieces in the second category induce twice the rate of wear for a given service rate as category one workpieces. Then the matrix of wear rates is

$$\mathbf{R}_D = \begin{bmatrix} 0.11 & 0 \\ 0 & 0.22 \end{bmatrix}. \tag{4.3}$$

The maximum allowable damage is assumed to be $x = 1.0$ unit. The cost coefficients associated with this example are defined as in Table 4.1.

Table 4.1 Cost coefficients; Example I.

Cost Coefficients	Value
c_N	18.0
c_H	15.0
$c_W(\mu_j)$	$5\mu_j, \quad j = 1, 2$
c_F	6.0

Several sample points of the numerical CDF of the server lifetime are provided in Table 4.2, where the comparison is made between the Monte-Carlo simulation and the numeric Laplace inversion using the Abate and Whitt [1] algorithm. The usual measure of performance when comparing a simulated CDF to an analytical CDF is the maximum absolute deviation (MAD) in probability. For this example, the MAD is 0.0013.

Table 4.2 Simulated versus analytical CDF; Example I.

t	Simulated CDF	Laplace Inversion	Absolute Deviation
4.4	0.000000	0.000000	0.000000
4.6	0.000118	0.000097	0.000021
4.8	0.000652	0.000649	0.000003
5.0	0.002142	0.002156	0.000014
5.2	0.005554	0.005540	0.000014
5.4	0.012094	0.012141	0.000047
5.6	0.023630	0.023681	0.000051
5.8	0.042074	0.042232	0.000158
6.0	0.069344	0.069989	0.000645
6.2	0.108360	0.108933	0.000573
6.4	0.159880	0.160488	0.000608
6.6	0.224936	0.225148	0.000212
6.8	0.302652	0.302158	0.000494
7.0	0.389568	0.389359	0.000209
7.2	0.483436	0.483265	0.000171
7.4	0.579064	0.579386	0.000322
7.6	0.672416	0.672739	0.000323
7.8	0.758264	0.758515	0.000251
8.0	0.832620	0.832753	0.000133
8.2	0.892892	0.892837	0.000055
8.4	0.937330	0.937837	0.000507
8.6	0.968256	0.968584	0.000328
8.8	0.986762	0.987234	0.000472
9.0	0.996412	0.996649	0.000237
9.2	1.000000	0.999982	0.000018

Table 4.3 summarizes the results of the combined bisection and secant root-finding algorithm using both the analytical and simulated CDF. A comparison of these results reveals that the techniques provide similar solutions to the replacement time problem, with very little difference in the computed optimal replacement times. The small error between simulated results and Laplace inversion results is likely caused by error introduced by trapezoidal integration and the use of central differences to approximate the derivative of the CDF.

Table 4.3 Results for optimal replacement time; Example I.

Method	T^*	$g(T^*)$
Laplace Inversion CDF	7.272270	158.125345
Simulated CDF	7.281900	158.125240

For the service rate selection problem, assume the replacement time T is fixed, with $T = 7.272270$, and $\vec{\mu} = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix}$ is the vector of decision variables. An upper and lower bound is assumed to exist for all mean service rates. In the context of this problem, assume that $1 < \mu_j \leq 200$ for all states j in the random environment. These upper and lower bounds on the service rates make searching the entire allowable space of mean service rates computationally feasible. To accomplish this, multiple gradient searches were implemented, starting from each corner point of the feasible space of mean service rates (i.e., the gradient search is performed four times with starting points $(1, 1)$, $(1, 200)$, $(200, 1)$, and $(200, 200)$). The results of these gradient searches are summarized in Table 4.4. Examination of these results shows that each gradient search computed approximately the same local minimum, making it likely that $\vec{\mu} = (3.176616, 1.000000)$ is the global minimum. Moreover, the optimal service rates resulted in the reduction of the long-run expected cost rate by 130.16 units, which implies an expected savings of course of the cycle is 946.56 units. This is calculated by multiplying the long-run expected cost rate by the cycle length.

The service rate selection problem is more complex because the decision variables impact the distribution of the failure times; every time $\vec{\mu}$ changes, the resulting

Table 4.4 Service rate selection problem; Example I.

Starting Point	$\vec{\mu}^*$	$g(\vec{\mu}^*)$
(1,1)	(3.176616,1.000000)	27.965521
(200,1)	(3.176616,1.000000)	27.965521
(1,200)	(3.176616,1.000000)	27.965521
(200,200)	(3.176616,1.000000)	27.965521

CDF of the failure time of the server changes. Therefore, every time the objective function is evaluated, one must generate a simulated CDF. This is computationally infeasible given the large number of objective function evaluations required by the gradient search algorithm.

4.3 Example II

Now assume the grinding wheel processes workpieces that are identical. At any point in time, the grinding wheel is operated by one of a pool of five workers of varying skill levels. That is, $Z(t)$ denotes the worker operating the machine at time t so that $Z(t) \in S = \{1, 2, 3, 4, 5\}$ for all t. Each worker processes workpieces with the same mean service rate, but each worker wears the wheel at a different rate due to varying worker skill levels.

The random process defining which worker is operating the machine at any given time is modelled as a CTMC with infinitesimal generator matrix

$$\mathbf{Q} = \begin{bmatrix} -76.4653 & 19.1376 & 28.2982 & 16.5118 & 12.5177 \\ 25.6793 & -81.1850 & 28.9809 & 14.8722 & 11.6526 \\ 15.2226 & 29.5272 & -87.6932 & 29.3102 & 13.6332 \\ 28.4067 & 18.8163 & 11.3262 & -79.1669 & 20.6177 \\ 14.1677 & 10.1135 & 19.5026 & 25.4786 & -69.2624 \end{bmatrix}. \tag{4.4}$$

The rate of wear incurred by the wheel due to worker j's operation of the machine is

$$r(j) = \begin{cases} \mu_j & j = 1 \\ (\mu_j)^2 & j = 2 \\ \exp(\mu_j) & j = 3 \\ \ln(\mu_j) & j = 4 \\ (\mu_j)^3 & j = 5 \end{cases} . \tag{4.5}$$

Also, the cost coefficients for this example are listed in Table 4.5.

Table 4.5 Cost coefficients; Example II.

Cost Coefficients	Value
c_N	10.0
c_H	5.0
$c_W(\mu_j)$	$\mu_j^2, \quad j = 1, 2, 3, 4, 5$
c_F	20.0

Several sample points of the numerical CDF of the server lifetime are provided in Table 4.6. For this example, the MAD is 0.0012.

To solve the optimal replacement time problem, assume that each worker processes workpieces with a mean rate $\mu_j = 2$ for all j. The object is to compute the optimal time to replace the grinding machine. These results are found in Table 4.7. It is clear that all methods yield similar results.

For the service rate selection problem, assume that management has chosen to implement the optimal replacement solution, i.e., $T = 2.198010$. Now the objective is to adjust the mean service rate of each worker to minimize the long-run cost rate. Then $\vec{\mu} = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 & \mu_4 & \mu_5 \end{bmatrix}$ is the vector of decision variables. Again, assume that, due to safety concerns and physical limitations, no worker can process workpieces with a mean service rate greater than fifteen and no lower than 1, i.e. $1 < \mu_j \leq 15$ for all j. The results of the gradient search algorithm at the 32 corner points of the feasible region are summarized in Table 4.8. Again, benchmarking via simulation is computationally infeasible. As seen in Table 4.8,

Table 4.6 Simulated versus analytical CDF; Example II.

t	Simulated CDF	Laplace Inversion	Absolute Deviation
1.20	0.000000	0.000000	0.000000
1.30	0.000354	0.000364	0.000010
1.40	0.002822	0.002819	0.000003
1.50	0.010024	0.009960	0.000064
1.60	0.025154	0.025184	0.000030
1.70	0.051938	0.051992	0.000054
1.80	0.092950	0.092951	0.000001
1.90	0.148630	0.149125	0.000495
2.00	0.219174	0.219585	0.000411
2.10	0.300984	0.301537	0.000553
2.20	0.389780	0.390850	0.001070
2.30	0.482266	0.482720	0.000454
2.40	0.572396	0.572438	0.000042
2.50	0.655334	0.656026	0.000692
2.60	0.730060	0.730609	0.000549
2.70	0.794480	0.794546	0.000066
2.80	0.847350	0.847347	0.000003
2.90	0.889446	0.889448	0.000002
3.00	0.922282	0.921924	0.000358
3.25	0.970608	0.970581	0.000027
3.50	0.990382	0.990424	0.000042
4.00	0.999376	0.999329	0.000047
4.50	0.999982	0.999972	0.000010
5.00	1.000000	0.999999	0.000001

Table 4.7 Results for optimal replacement time; Example II.

Method	T^*	$g(T^*)$
Laplace Inversion CDF	2.198010	13.915432
Simulated CDF	2.203700	13.913700

there is little difference between any of the results; and hence it is conjectured that $\vec{\mu}^* = (2.240, 2.086, 1.853, 2.257, 1.755)$. It should be noted that the long-run cost rate of 13.564971 is slightly lower than the long-run expected cost rate of 13.915432 under the assumptions of the optimal replacement strategy. This shows that for this particular value of T, it is beneficial to allow the workers to operate at different speeds

dependent on their skill level. This leads to a reduction of the long-run expected cost rate of 0.35 units, with an expected cost savings of 0.77 units over one cycle.

Table 4.8 Service rate selection problem; Example II.

Starting Point	$\vec{\mu}^*$	$g(\vec{\mu}^*)$
(1,1,1,1,1)	(2.240335,2.085599,1.853378,2.257171,1.754528)	13.564971
(200,1,1,1,1)	(2.240237,2.085548,1.853392,2.257296,1.754530)	13.564971
(1,200,1,1,1)	(2.240338,2.085697,1.853351,2.257106,1.754516)	13.564971
(1,1,200,1,1)	(2.240335,2.085599,1.853378,2.257171,1.754528)	13.564971
(1,1,1,200,1)	(2.240155,2.085707,1.853354,2.257283,1.754519)	13.564971
(1,1,1,1,200)	(2.240233,2.085642,1.853368,2.257251,1.754529)	13.564971
(200,200,1,1,1)	(2.240327,2.085703,1.853350,2.257114,1.754516)	13.564971
(200,1,200,1,1)	(2.240338,2.085686,1.853354,2.257113,1.754518)	13.564971
(200,1,1,200,1)	(2.240229,2.085565,1.853390,2.257294,1.754530)	13.564971
(200,1,1,1,200)	(2.240242,2.085624,1.853377,2.257248,1.754528)	13.564971
(1,200,200,1,1)	(2.240237,2.085675,1.853353,2.257216,1.754512)	13.564971
(1,200,1,200,1)	(2.240182,2.085702,1.853356,2.257259,1.754520)	13.564971
(1,200,1,1,200)	(2.240236,2.085596,1.853384,2.257271,1.754530)	13.564971
(1,1,200,200,1)	(2.240171,2.085692,1.853359,2.257276,1.754521)	13.564971
(1,1,200,1,200)	(2.240217,2.085712,1.853361,2.257212,1.754513)	13.564971
(1,1,1,200,200)	(2.240236,2.085671,1.853351,2.257217,1.754510)	13.564971
(200,200,200,1,1)	(2.240373,2.085696,1.853345,2.257069,1.754511)	13.564971
(200,200,1,200,1)	(2.240288,2.085666,1.853358,2.257182,1.754527)	13.564971
(200,200,1,1,200)	(2.240208,2.085672,1.853369,2.257247,1.754521)	13.564971
(200,1,200,200,1)	(2.240106,2.085708,1.853350,2.257326,1.754512)	13.564971
(200,1,200,1,200)	(2.240222,2.085695,1.853358,2.257222,1.754517)	13.564971
(200,1,1,200,200)	(2.240240,2.085639,1.853373,2.257241,1.754527)	13.564971
(1,200,200,200,1)	(2.240239,2.085697,1.853324,2.257225,1.754536)	13.564971
(1,200,200,1,200)	(2.240232,2.085585,1.853386,2.257281,1.754531)	13.564971
(1,200,1,200,200)	(2.240201,2.085749,1.853290,2.257245,1.754549)	13.564971
(1,1,200,200,200)	(2.240236,2.085571,1.853390,2.257284,1.754530)	13.564971
(200,200,200,200,1)	(2.240132,2.085790,1.853326,2.257248,1.754503)	13.564971
(200,200,200,1,200)	(2.240227,2.085640,1.853375,2.257251,1.754525)	13.564971
(200,200,1,200,200)	(2.240224,2.085668,1.853367,2.257237,1.754522)	13.564971
(200,1,200,200,200)	(2.240229,2.085629,1.853376,2.257258,1.754528)	13.564971
(1,200,200,200,200)	(2.240236,2.085665,1.853349,2.257219,1.754508)	13.564971
(200,200,200,200,200)	(2.240236,2.085571,1.853390,2.257284,1.754530)	13.564971

4.4 Example III

This section examines a telecommunications satellite constellation comprised of four satellites. Suppose this constellation is modelled as a queueing system, where each satellite represents one server in a multiserver queueing system. For the sake of convenience, assume that this constellation can be appropriately modelled as a $M/M/4$ queueing system, with arrival rate $\lambda = 1$. Assume that the mean service rate of the satellites is adjustable, but increasing the service rate requires more power to increase the available bandwidth. This power must be provided by a set of large, yet fragile, solar panels which can be damaged by the space environment. It is assumed the solar panels can be furled to minimize damage when the power requirements of the satellite are low.

Now assume that the space environment can be adequately modelled as a 10-state CTMC where the infinitesimal generator matrix is randomly generated. The rate at which the CTMC transitions between each pair of states is continuously and uniformly distributed between 100 and 600. Suppose the solar panels, and hence the satellites, incur damage linearly with the rate of damage dependent on the state of the environment. For the purposes of this example, assume that the rate of wear for state j is

$$r(j) = \left(\frac{j\mu_j}{2}\right)^2. \tag{4.6}$$

Also assume that a satellite's solar panels are non-functional when the cumulative damage reaches 1.0. The cost coefficients used in this example are found in Table 4.9.

Table 4.9 Cost coefficients; Example III.

Cost Coefficients	Value
c_N	5.0
c_H	20.0
$c_W(\mu_j)$	$\mu_j^2, \quad j = 1, 2, \ldots, 10$
c_F	10.0

Table 4.10 Simulated versus analytical CDF; Example III.

t	Simulation CDF	Laplace Inversion	Absolute Deviation
1.30	0.000000	0.000001	0.000001
1.40	0.000012	0.000017	0.000005
1.50	0.000114	0.000125	0.000011
1.60	0.000650	0.000658	0.000008
1.70	0.002762	0.002631	0.000131
1.80	0.008386	0.008389	0.000003
1.90	0.022122	0.022150	0.000028
2.00	0.049408	0.049784	0.000376
2.10	0.097208	0.097403	0.000195
2.20	0.168744	0.168981	0.000237
2.25	0.213422	0.213835	0.000413
2.30	0.263740	0.264110	0.000370
2.35	0.318896	0.318930	0.000034
2.40	0.377044	0.377153	0.000109
2.45	0.437342	0.437449	0.000107
2.50	0.498400	0.498400	0.000000
2.55	0.558750	0.558597	0.000153
2.60	0.616586	0.616736	0.000150
2.65	0.671882	0.671691	0.000191
2.70	0.722562	0.722569	0.000007
2.75	0.768520	0.768737	0.000217
2.80	0.809424	0.809830	0.000406
2.90	0.875516	0.876519	0.001003
3.00	0.923458	0.923986	0.000528
3.10	0.955104	0.955584	0.000480
3.25	0.981922	0.981954	0.000032
3.50	0.996828	0.996836	0.000008
4.00	0.999954	0.999957	0.000003
4.25	0.999998	0.999998	0.000000
4.50	1.000000	0.999999	0.000001

Several sample points of the numerical CDF of the server lifetime are provided in Table 4.10, where the comparison is made between the Monte-Carlo simulation and the numeric Laplace inversion using the Abate and Whitt [1] algorithm. In this example problem the MAD is 0.0012.

Table 4.11 summarizes the results of both the secant root finding algorithm and a gradient search for T^*, where the service rate is assumed to be $\mu_j = 2$ for all j.

A comparison of these results reveals that the techniques provide similar solutions to the replacement time problem.

Table 4.11 Results for optimal replacement time; Example III.

Method	T^*	$g(T^*)$
Laplace Inversion CDF	2.728415	22.073548
Simulated CDF	2.734100	22.073067

The service rate selection problem is much more complex than in the previous sections. Since there are now ten states in the random environment, there are ten decision variables. Further, the feasible region in this example is a ten-dimensional hypercube. This implies that there are a total of 1024 corner points to the feasible space. Again, suppose management has decided to implement the results of the optimal replacement problem, and that $T = 2.734100$. The objective is to adjust the service rates, and through the service rates, the amount of power required by the satellite to minimize the long-run cost rate. Again, for the sake of simplicity, an upper bound is assumed to exist for all mean service rates, and hence $\mu_j \leq 20$ for all j. Similarly, assume a lower bound of $1/4 \leq \mu_j$ to ensure stability. Tables 4.12 and 4.13 present the results of the gradient search algorithm using only a small subset of the available corner points. These tables clearly show that the gradient search converged to approximately the same point for all the starting points listed in the tables. Furthermore, the long-run expected cost rate of 21.376584 is less than the long-run expected cost rate of 22.073067 implying that, in this case, adjusting the service rate is beneficial. This results in a 0.70 reduction in the long-run expected cost rate and an expected savings of 1.90 units over the course one cycle.

These three examples have illustrated the techniques to compute the globally optimal replacement time as well as the locally optimal mean service rates. While the mean service rates are only guaranteed to be local minima, the fact that many iterations of the gradient search algorithm converged to the same point provides strong evidence that the local minimum may be globally optimal.

Table 4.12 Service rate selection problem; Example III.

Starting Point	$\vec{\mu}^*$	$g(\vec{\mu}^*)$
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(2.383157,2.360698,2.325098,2.259523,2.171928, 2.067354,1.955693,1.842288,1.688630,1.565746)	21.376584
(20, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(2.383180,2.360704,2.325067,2.259522,2.171924, 2.067379,1.955699,1.842294,1.688622,1.565742)	21.376584
(0, 20, 0, 0, 0, 0, 0, 0, 0, 0)	(2.383162,2.360674,2.325057,2.259536,2.171952, 2.067361,1.955706,1.842292,1.688621,1.565740)	21.376584
(0, 0, 20, 0, 0, 0, 0, 0, 0, 0)	(2.383134,2.360696,2.325058,2.259534,2.171948, 2.067368,1.955704,1.842291,1.688620,1.565739)	21.376584
(0, 0, 0, 20, 0, 0, 0, 0, 0, 0)	(2.383117,2.360691,2.325075,2.259535,2.171953, 2.067356,1.955706,1.842292,1.688619,1.565738)	21.376584
(0, 0, 0, 0, 20, 0, 0, 0, 0, 0)	(2.383190,2.360665,2.325055,2.259533,2.171950, 2.067357,1.955706,1.842293,1.688622,1.565741)	21.376584
(0, 0, 0, 0, 0, 20, 0, 0, 0, 0)	(2.383153,2.360681,2.325074,2.259535,2.171951, 2.067328,1.955709,1.842297,1.688623,1.565741)	21.376584
(0, 0, 0, 0, 0, 0, 20, 0, 0, 0)	(2.383273,2.360640,2.325045,2.259517,2.171941, 2.067361,1.955702,1.842290,1.688618,1.565735)	21.376584
(0, 0, 0, 0, 0, 0, 0, 20, 0, 0)	(2.383140,2.360677,2.325062,2.259541,2.171955, 2.067357,1.955706,1.842292,1.688620,1.565739)	21.376584
(0, 0, 0, 0, 0, 0, 0, 0, 20, 0)	(2.383219,2.360665,2.325081,2.259506,2.171928, 2.067397,1.955699,1.842291,1.688622,1.565742)	21.376584
(0, 0, 0, 0, 0, 0, 0, 0, 0, 20)	(2.383215,2.360663,2.325079,2.259509,2.171930, 2.067396,1.955700,1.842291,1.688621,1.565741)	21.376584

Table 4.13 Service rate selection problem; Example III.

Starting Point	$\vec{\mu}^*$	$g(\vec{\mu}^*)$
(0, 20, 20, 20, 20, 20, 20, 20, 20, 20)	(2.383164,2.360668,2.325072,2.259535,2.171952, 2.067350,1.955707,1.842292,1.688622,1.565741)	21.376584
(20, 0, 20, 20, 20, 20, 20, 20, 20, 20)	(2.383331,2.360637,2.325029,2.259508,2.171930, 2.067387,1.955695,1.842296,1.688624,1.565745)	21.376584
(20, 20, 0, 20, 20, 20, 20, 20, 20, 20)	(2.383371,2.360620,2.325008,2.259515,2.171940, 2.067350,1.955706,1.842296,1.688626,1.565746)	21.376584
(20, 20, 20, 0, 20, 20, 20, 20, 20, 20)	(2.383362,2.360623,2.324999,2.259526,2.171948, 2.067321,1.955710,1.842299,1.688627,1.565746)	21.376584
(20, 20, 20, 20, 0, 20, 20, 20, 20, 20)	(2.383379,2.360613,2.325012,2.259513,2.171939, 2.067353,1.955705,1.842295,1.688626,1.565746)	21.376584
(20, 20, 20, 20, 20, 0, 20, 20, 20, 20)	(2.383419,2.360607,2.324997,2.259509,2.171937, 2.067345,1.955705,1.842296,1.688626,1.565746)	21.376584
(20, 20, 20, 20, 20, 20, 0, 20, 20, 20)	(2.383396,2.360613,2.324988,2.259525,2.171947, 2.067317,1.955710,1.842299,1.688627,1.565746)	21.376584
(20, 20, 20, 20, 20, 20, 20, 0, 20, 20)	(2.383345,2.360624,2.325017,2.259517,2.171941, 2.067352,1.955706,1.842296,1.688626,1.565746)	21.376584
(20, 20, 20, 20, 20, 20, 20, 20, 0, 20)	(2.383344,2.360636,2.324999,2.259526,2.171947, 2.067352,1.955706,1.842296,1.688626,1.565746)	21.376584
(20, 20, 20, 20, 20, 20, 20, 20, 20, 0)	(2.383344,2.360636,2.324999,2.259526,2.171947, 2.067323,1.955710,1.842299,1.688627,1.565747)	21.376584
(20, 20, 20, 20, 20, 20, 20, 20, 20, 20)	(2.383412,2.360609,2.324998,2.259512,2.171938, 2.067342,1.955705,1.842296,1.688627,1.565746)	21.376584

5. Conclusions and Future Research

This thesis unites techniques for optimal tool replacement and the optimal control of queueing systems to optimally control a k-server queueing system in which the servers are subject to failures due to a stochastic and dynamic environment. Specifically, the queueing system is controlled through two distinct methods. First, assuming a k-replacement policy, the optimal replacement time was analytically computed. Second, the locally optimal mean service rate for each environmental state was computed. Both the optimal replacement time and the mean service were computed using numerical search techniques, due to the complex structure of the objective function as well as the existence of Laplace transforms of matrix exponentials. Additionally, notional examples of these types of systems were examined, not only to show how these methodologies may be applied, but also to illustrate the effectiveness of the numerical Laplace transform inversion.

The assumed k-replacement policy of the system was modelled explicitly as a renewal process, which in turn allowed the application of the renewal-reward theorem. The renewal-reward theorem provides an expression for the long-run expected cost rate, which was the measure of performance used in this thesis. The objective function was derived with respect to the cost per renewal cycle. Of the four cost components, three were relatively straightforward. The fourth component, the cost incurred by outside processing, was defined as a fixed cost paid to process each customer served by an external service provider. This expected cost per cycle requires the evaluation of the expected number of customers served by the outside service provider. An analytical expression for this quantity was obtained using a conditional expectation argument.

Using the long-run average cost criterion as the primary metric of interest, the next step was to compute the optimal replacement time. Prior to this derivation, Proposition (3.1) proved the convexity of the objective function with respect to

the replacement time. This proposition is critical in that it guarantees an optimal solution for the replacement time problem in one of two places: either at a stationary point or at infinity. Thus, simple root-finding techniques, such as the bisection and secant methods, were used to compute the stationary point. When such a point is found, it is known to be the optimal replacement time. When these root-finding methods fail, the optimal replacement time is infinite, and hence, the best policy is simply to never replace the servers and rely exclusively on the outside service provider.

After the optimal replacement time was computed, the problem of optimal service rate selection for each environmental state was considered. Unfortunately, the objective function cannot be shown analytically to be convex with respect to the service rates, and hence there may be many stationary points, each of which may be a minimum, a maximum or a saddle-point. The gradient search method was used to compute the optimal service rates. However, this method finds local minima and not a global minimum. To circumvent this, multiple gradient searches were used with different starting points. While this technique does not guarantee a globally optimal solution, it will at least find several local minima which have lower objective function values than the initial starting points.

The next step in the thesis process was to illustrate these methods through numerical examples. A series of three notional scenarios were used to illustrate the utility of the techniques presented. Once each scenario was modelled as a queueing system with the appropriate failure mechanisms, Equation (3.40) was used to compute the optimal replacement time by employing the bisection and secant root-finding techniques in conjunction with numerical Laplace transform inversion. To benchmark these results, a simulation model was used to construct an empirical distribution of the server failure times. These two CDFs (analytical and simulated) were then used in Equation (3.36) to calculate competing T^* values. The optimal

solutions were found to be nearly identical in each example. Because the objective function is convex with respect to T, these stationary points must be global minima.

The optimal service rate selection problem was more difficult. In each scenario the optimal replacement time was used as the fixed replacement time. The gradient search algorithm was then employed to control the mean service rates to further reduce the long-run expected cost rate. These results were not benchmarked utilizing simulation because each change in a service rate alters the distribution of the server's failure time, and hence requires a new simulated CDF, an extremely computationally intensive process. Because the gradient search method only finds local minima, multiple initial conditions were used in an effort to adequately search the feasible space and identify the global minimum. In all the examples examined in this thesis, the result of the gradient search was approximately identical for each starting point, which provides a reasonable measure of evidence that a global minimum has been reached. In each case, the objective function value was decreased by employing the gradient search algorithm.

This work has contributed two methodologies for controlling a queueing system in which the servers are subject to break downs. The queueing system is interesting in that it models the effects of the surrounding environment on the degradation of the server. The first methodology computes the optimal replacement time of the queueing system. This methodology is guaranteed to converge to an approximate optimal replacement time because it was shown that the objective function is convex with respect to the replacement time. Furthermore, this methodology has great potential to save organizations valuable resources by avoiding premature failures and unnecessary replacements. The second methodology allows the controller of the queueing system to adapt to the random environment by adjusting the service rate whenever the status of the environment changes. While this technique is not guaranteed to find a globally optimal long-run expected cost rate, it will at least find a local minimum, whose objective function value must be equal to the initial

starting point or less. Therefore, by using this technique, it is possible to adjust the service rates to further reduce long-run average operating costs.

The most obvious avenue of further research lies in the relaxation of the assumptions of the system dynamics. For example, there exist many replacement policies other than the k-replacement policy examined in this thesis, and optimizing the queueing system under any one of them would further extend the current state of the art. Also, there are many techniques in the tool replacement literature that could be used in conjunction with this thesis. One example is an inspect-and-replace policy, where at fixed intervals of time, the operator inspects the system and takes some action based on that inspection. Perhaps the true level of wear is not easily inspected, a scenario that is common in the real world, in which it would be interesting to examine the best course of action to take.

Another assumption that could be relaxed is that of nondiscriminatory server selection. This assumption states that even though there may be an idle, operative server, it is possible to send a customer to the outside service provider at a higher cost. If a more logical server-preference system were introduced, it would alter the expected number of customers that undergo service with the outside service provider, and hence alter the results of this thesis.

Further research could also be conducted into alternative numerical techniques. There are many (derivative-free) algorithms to minimize functions other than the gradient search method, and one of these techniques may be able to provide a globally optimal solution to the service rate selection problem. In addition, further research could be aimed at simultaneously controlling both the age replacement time and service rates.

Finally, another interesting avenue of future research is an explicit performance analysis of the queueing system described in this thesis. The assumption of external service providers ensures that server failures do not effect the long-run performance of the queue. Relaxation of this assumption would require the derivation of traditional

queueing performance measures, such as the long-run expected number of customers in the system and the expected waiting time in the system. Such an analysis may more realistically capture the dynamics of the system as queue instability may result due to server failures.

Bibliography

1. Abate, J. and W. Whitt (1995). Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, **7**, 36-43.

2. Andradóttir, S. (1998). Chapter 9: Simulation optimization. *Handbook of Simulation*. John Wiley & Sons, Inc., New York.

3. Ahmad, M. and A. K. Sheikh (1984). Bernstein reliability model: derivation and estimation of parameters. *Reliability Engineering*, **8**, 131-148.

4. Avi-Itzhak, B. and P. Naor (1963). Some queuing problems with the service station subject to breakdown. *Operations Research*, **11**, 303-320.

5. Burden, R. L. and J. D. Faires (2001). *Numerical Analysis*. 7th ed., Brooks/Cole, Pacific Grove CA.

6. Crabill, T. B. (1972). Optimal control of a service facility with variable exponential service times and constant arrival rate. *Management Science*, **18**, 560-566.

7. Daigle, J. N. and M. N. Magalhães (1991). Transient behavior of $M/M^{ij}/1$ queues. *Queueing Systems*, **8** 357-378.

8. Doshi, B. T. (1978). Optimal control of the service rate in an M/G/1 queueing system. *Advances in Applied Probability*, **10**, 682-701.

9. George, J. M. and J. M. Harrison (2001). Dynamic control of a queue with adjustable service rate. *Operations Research*, **49**, (5), 720-731.

10. Grassmann, W. K., Chen X. and B. R. K. Kashyap (2001). Optimal service rates for the state-dependent M/G/1 queues in steady state. *Operations Research Letters*, **29**, 57-63.

11. Gross, D. and C. M. Harris (1998). *Fundamentals of Queueing Theory*. 3rd ed., Wiley & Sons Inc, New York.

12. Harris, C. M. (1967). Queues with state dependent stochastic service rates. Operations Research. **15**, 117-130

13. Hopp, W. J. and S. Wu (1988). Multiaction maintenance under markovian deterioration and incomplete state information. *Naval Research Logistics*, **35** 447-462.

14. Hopp, W. J. and S. K. Nair (1994). Markovian deterioration and technological change. *IIE Transactions*, **26** (6), 74-82.

15. Hopp, W. J. and Y. Kuo (1998). An optimal structured policy for maintenance of partially observable aircraft engine components. *Naval Research Logistics*, **45**, 335-352.

16. Jeang, A. (1999). Tool Replacement policy for probabilistic tool life and random wear procss. *Quality and Reliability Engineering International*, **15**, 205-212.

17. Jo, K. Y. (1983). Optimal service rate control of exponential queueing systems. *Journal of the Operations Research Society of Japan*, **26**, 147-165.

18. Kharoufeh, J. P. (2003). Explicit results for wear processes in a Markovian environment. *Operations Research Letters*, **31**, 237-244.

19. Kharoufeh, J. P. and J. A. Sipe (2004). Evaluating failure time probabilites for a Markovian wear process. Forthcoming in *Computers and Operations Research*.

20. Kodera, T. and M. Miyazawa (2002). An M/G/1 queue with Markov-dependent exceptional service times. *Operations Research Letters*, **30**, 231-244.

21. Koyanagi, J. and H. Kawai (1995). An optimal maintenance policy for a deteriorating server of an $M/G/1$ queueing system. *Proceedings of Stochastic Modelling in Innovative Manufacuring*, Cambridge, UK, July 21 - 22, 215-224.

22. Kulkarni, V. G. (1995). *Modeling and Analysis of Stocahstic Systems*. 1st ed., Chapman & Hall, London.

23. Kulkarni, V. G. and B. D. Choi (1990). Retrial queues with server subject to breakdowns and repairs. *Queueing Systems*, **7**, 191-208.

24. Mitrany, I. L. and B. Avi-Itzhak (1968). A many-server queue with service interruptions. *Operations Research*, **16**, 628-638.

25. McCall, J. J. (1965). Maintenace policies for stochastically failing equipment: a survey. *Management Science*, **11**, 493-521.

26. Nakagawa, T. (1979). Replacement problem of a parrallel system in random environment. *Journal of Applied Probability*, **16**, 203-205.

27. Neuts, M. F. (1978). The M/M/1 queue with randomly varying arrival and service rates. *Opsearch*, **15** (4), 139-157.

28. Neuts, M. F. (1978). Further results on the M/M/1 qeueue with randomly varying rates. *Opsearch*, **15** (4), 158-168.

29. Neuts, M. F. and D. M. Lucantoni (1979). A Markovian queue with N servers subject to breakdowns and repairs. *Management Science*, **25**, 849-861.

30. Perry, D. (2000). Control limit policies in a replacement model with additive phase-type distributed damage and linear restoration. *Operations Research Letters*, **27**, 127-134.

31. Pierskalla, W. P. and J. A. Voelker (1976). A survey of maintenance models: the control and surveillance of deterioriating systems. *Naval Research Logistics Quarterly*, **23**, 353-288.

32. Sabeti, H. (1973). Optimal selection of service rates in queueing with different cost. *Journal of the Operations Research Society of Japan*, **16**, 15-35.

33. Sennott, L. I. (1989). Average cost optimal stationary policies in infinite state Markov decision processes with unbounded costs. *Operations Research*, **37** (4), 626-633.

34. Shirmohammadi, A. H., Love, C. E., and Z. G. Zhang (2003). An optimal maintenance policy for skipping imminent preventive maintenance for systems experiencing random failures. *Journal of the Operational Research Society*, **54**, 40-47.

35. Stidham, S. Jr. and R. R. Weber (1989). Monotonic and insensitive optimal policies for control of queues with undiscounted costs. *Operations Research*, **37** (4), 611-625.

36. Thiruvengadam, K (1963). Queuing with breakdowns. *Operations Research*, **11**, 62-71.

37. Valdez-Flores, C. and R. M. Feldman (1989). A survey of preventive maintenance models for stochastically deteriorating single unit systems. *Naval Research Logistics*, **36**, 419-446.

38. Waldmann, K. H. (1983). Optimal replacement under additive damage in randomly varying environments. *Naval Research Logistics Quarterly*, **30**, 377-386.

39. Wang, J., J. Cao and Q. Li (2001). Reliability analysis of the retrial queue with server breakdowns and repairs. *Queueing Systems*, **38**, 363-380.

40. Zhou, C., Chandra, J. and Richard Wysk (1990). Optimal cutting tool replacement based on tool wear status. *International Journal of Production Research*, **28** (7), 1357-1367.

Appendix A. Simulation Code

```
%*************************************************************************
% PROGRAM MARKOV
%
%      The purpose of this code is to simulate the failure distribution of a server
%      undergoing environmentally driven wear.  The environment is modeled by a
%      finite state CTMC which dictates the rate of wear on the server.
%
%      Author:  Patrick S Chapin, altered from original source code by
%               J.P. Kharoufeh, Ph.D.
%      Date:  1 Feb 2004
%      Last Revised:  7 Mar 2004
%*************************************************************************

% VARIABLE DEFINTIONS
**********************************************************************
%
% L  = The maximum amount of wear for one server
% k  = an index variable
%
%*************************************************************************

% VECTOR/FUNCTION DEFINITIONS*************************************************
%
% T = Vector of simulated failure times
% B = Vector for the initial probability distribution of the Markov process
% R = Vector of transition rates
% P = Probability transition matrix
% V = Vector of wear rates (i.e., V(i)= wear rate when environment is in state i).
% Z = Vector of environment states
% Q = Infitesimial Generator Matrix of the random environment.
%*************************************************************************

% The program assumes a state space of the form S={1,2,...K}
function Output = markov(soe)

TimeStart = clock;
%markov_input;                          % Obtain intialization parameters
L = 1.0; %Max wear
N = 2; %Size of Environment
T = 1:25000; %Number of Tries
B = [1 zeros(1,N-1)]; %Initial Condition
V = []; %Vector of R_D

%establish Q.
if soe == 2
   a = 19;
   b = 7;
   Q = [-b b;a -a];

   Q = Q*(1/10);
   %define the service rate
   mu = [1.1,1.1];
   V = [1*mu(1),2*mu(2)];
```

```
   V = V*(1/10);
elseif soe == 5
   Q=[-76.4653   19.1376 28.2982 16.5118 12.5177
      25.6793    -81.185  28.9809 14.8722 11.6526
      15.2226    29.5272 -87.693229.3102 13.6332
      28.4067    18.8163 11.3262 -79.166920.6177
      14.1677    10.1135 19.5026 25.4786 -69.2624];
   Q=Q./10;
   %define the service rate
   mu = [2,2,2,2,2];
   V(1) = mu(1);
   V(2) = (mu(2))^2;
   V(3) = exp(mu(3));
   V(4) = log(mu(4)+1);
   V(5) = (mu(5))^3;
   V= V*(1/10);
elseif soe == 10
   Q = [  -3472.932 573.5848  518.6157  247.7199  222.8372  282.9065  276.7697  486.86    377.6327  486.0055
     528.9639       -3464.2659          346.6875  268.6725  494.174   390.0185  421.1792  561.8889  120.426   332.2554
     215.1789       379.774   -2638.3644          445.3362  471.2741  265.6516  244.8543  334.6254  134.4901  147.1798
     110.7218       506.8413  170.6219  -2467.6896          142.3301  107.2723  533.6251  270.8044  410.6428  214.8299
     142.7507       411.9563  491.1781  101.2544  -3160.385 140.8118  596.2472  501.9074  517.8909  256.3882
     499.5321       392.8141  165.9629  104.3732  586.9225  -3679.0386          479.72    591.5364  423.7168  434.4606
     257.7228       585.8839  100.8181  323.5683  358.0541  564.9279  -3297.6269          230.9467  326.5551  549.15
     563.2924       295.443   340.6363  232.606   471.9588  585.0955  238.3625  -3438.1854          289.4849  421.306
     192.9471       252.2573  157.9861  374.6915  508.0261  531.4705  259.1157  250.9286  -3058.4467          531.0238
     316.1591       119.4228  205.5623  578.4263  399.5693  167.2138  333.538   177.418   582.0245  -2879.3341
   ];
   Q=Q./ 100;
   mu=[2,2,2,2,2,2,2,2,2,2];
   for k = 1:soe
      V(k) = (k*mu(k)/2)^2;
   end
   V= V*(1/100);
else
   return
end

%Solve for the steady state probabilites of the CTMC environment,
%and make those stead state probs the initial condition.
Qnew11 = [Q';ones(size(Q,1),1)'];
RHS11 = [zeros(size(Q,1),1);1];
q11=Qnew11\RHS11; %11s added to ensure I'm not overwriting anything later on in the program
B = q11';


R = 1:N;
for i = 1:N
  R(i)=-Q(i,i);
end

for i = 1:N
  for j = 1:N
    P(i,j)=Q(i,j)/(sum(Q(i,:))-Q(i,i));
  end
  P(i,i)=0.0;
end
```

```
V(1) = mu(1);
V(2) = mu(2)*2;
V = V/10;

Z = [];                                 % Initialize Z.

for k = 1:length(T)
  Z = [];
  Z(1)=rando(B);                        % Initial state of the environment at time 0
  T(k) = exprv(R(Z(1)));                % Time spent in initial state
  D = 0.0;                              % At time 0, amount of wear is 0
  D = D + T(k)*V(Z(1));                 % Add accumulated wear to cumulative wear
  i=1;
  while D < L                           % Do while cumulative wear is less than L
    Z(i+1) = rando(P(Z(i),:));          % Use the matrix P to determine next state
    new_time = exprv(R(Z(i+1)));        % Obtain a new exponential time via exprv().
    D = D + new_time*V(Z(i+1));         % Update cumulative wear
    T(k) = T(k) + new_time;             % Update total time
    i=i+1;
  end
  T(k)=T(k)-(1/V(Z(i)))*(D-L);
end

%Left over code from Kharoufeh's original source code.
%Y=[];
% Y = zeros(size(T));
% m=0;
% for i = 1:length(T)
%     if T(i)>mean(T)
%         Y(i)=T(i);
%         m = m+1;
%     end
% end
%
% res_avg = sum(Y)/m;
% disp(res_avg);

%T2 = T.^2;
% disp('E(T)=')
% disp(mean(T));
% disp('E(T2)=')
% disp(mean(T2));
% disp('Var(T)=')
% disp((std(T))^2);
%get_cdf;
%disp(F');

%Getting the CDF

t=floor(min(T))-.01:0.0001:ceil(max(T))+.01;

F = 1:length(t);  %This ensures that t and F are vectors of equal length
%U = 1:length(t);

% Compute the cdf value at the point t0
```

```
for i = 1:length(t)
   F(i) = 0.0;
   for j = 1:length(T)
      if T(j)<= t(i)
         F(i) = F(i) + 1/length(T);
      else
         F(i) = F(i);
      end
   end
end

Output = [t' F'];

string = strcat('simulationresultsfor',int2str(soe),'state.txt');

fid = fopen(string,'w');
for k=1:length(t)
  fprintf(fid, '%4.6f\t%4.6f\n',t(k),F(k));
end

TimeToRun = clock - TimeStart;
fprintf(fid, '\n%4.6f',TimeToRun);
fclose(fid)


%*********************************************************************************
%  rando.m generates a discrete random variable in S={1,2,...,n} given a distribution
%  vector p = [p1 p2 ... pn].

function [index] = rando(p)
u = rand;
i = 1;
s = p(1);

while ((u > s) & (i < length(p))),
  i=i+1;
  s=s+p(i);
end

index=i;

%*********************************************************************************
%exprv generates an exponential random variable using the rand function.
function eq = exprv(lambda)

eq = -(1/lambda)*log(1-rand(1));
```

Appendix B. Root-Finding Code

```
%*****************************************************************************
% PROGRAM Hybrid_Root_Finding
%
% The purpose of this MATLAB program is to find the
% root of the give objective function
%
%       Author:  Patrick S. Chapin, AFIT, GOR-04M
%       Date:  21 Oct 03
%       Last Revised:  4 Feb 04
%       References: Burden, R. L. and J. D. Faires (2001). NUMERICAL ANALYSIS, 7th
%                edition, Brooks/Cole, Pacific Grove, CA.
%       Inputs: sizeofenvironment, the size of the random environment.
%       Note that this program is set up accomplish 3 examples, a 2 state,
%          a 5 state and a 10 state problem.
%
%
%*****************************************************************************

% VARIABLE DEFINTIONS*********************************************************
% C_N = Replacement cost for 1 server
% C_H = Holding cost for 1 customer per unit time
% C_F = Additional cost of serving 1 customer at an outside facility.
% kservers = the number of servers
% lambda = arrival rate of customers to the system.
% mu = the service rate at each of the environmental states
% soe = Size of the random environment.
% Q = infitesimal generator matrix of the random environment.
% qswitch = a counter so that the the steady state solutions of the random environment
%          only calculated once.
% R_D = matrix of the rate of wear during each environmental state.  if the
%          environment is in state j and the wear rate is a, then R_D(j,j)
%          = a.
% Type1 - Type4 = Counters to differentiate between different laplace inversion
%          functions.  It is easier to tell what is going on if Type1 is
%          passed as opposed to just 1.
%*****************************************************************************


function Final_Cost_Rate = Chapin_Hybrid_Root_Finding(sizeofenvironment)

   %fid = fopen(fid2,'w');

   maxiterations = 10000;
   StepCounter = 1;


   global Tolerance
   global soe
   global lambda
   global upperbound
   global qswitch
   global C_N

   lambda = 1;
   soe = sizeofenvironment;
          upperbound = 5000;
```

```
        qswitch = 1;
Tolerance = 1*10^-12;

if soe == 2
   mu = [1.1;1.1];
elseif soe == 5
   mu = [2;2;2;2;2];
elseif soe == 10
   mu = [2;2;2;2;2;2;2;2;2;2];
else
   return
end

Tzero = .001;
Tone = 1;

tempcounter = 1;

%Find an upper and lower bound of the root
while Chapin_Objective_Function(Tzero,mu) * Chapin_Objective_Function(Tone,mu) > 0
   %Tzero = Tzero*.5;
   Tone = Tone * 2;
   if tempcounter > 500
      fprintf('Upper bound is greater than 2^500.  Hence no singularity found');
      return
   end
   tempcounter = tempcounter +1;
end

%now do the bisection method

tempcounter = 1;

while abs(Chapin_Objective_Function(Tzero,mu) -
hapin_Objective_Function(Tone,mu))/Chapin_Objective_Function(Tzero,mu) > .1
   Ttemp = (Tzero + Tone)/2;
   if Chapin_Objective_Function(Ttemp,mu) < 0
      Tzero = Ttemp;
   else
      Tone = Ttemp;
   end

   if tempcounter > 500
      break
   end

   tempcounter = tempcounter + 1;
end

if Chapin_Objective_Function(Tzero,mu) * Chapin_Objective_Function(Tone,mu) > 0
   return
end

qzero = Chapin_Objective_Function(Tzero,mu);
qone = Chapin_Objective_Function(Tone,mu);
```

```
%Implement the secant method, see Burden
while StepCounter < maxiterations

   T = Tone-qone*(Tone-Tzero)/(qone-qzero);

   if mod(StepCounter,10) == 0
      fprintf('step');
   end

    if abs(T-Tone) < Tolerance
      %fprintf(fid,' \n Successful proceedure, less than tolerance change between steps');
      %fprintf(fid,'\n The Solution is: %4.9f',T);
      Final_Cost_Rate = T;
      %fprintf(fid,'\n It took %d steps',StepCounter);
      %fprintf(fid,'\n \n');
      %fclose(fid);
      fprintf('\nComplete. T* = %f, function eval is %f',T,Chapin_Objective_Function(T,mu));
      return
   end

   StepCounter=StepCounter+1;

   Tzero = Tone;
   qzero = qone;
   Tone = T;
   qone = Chapin_Objective_Function(T,mu);


end

fprintf(fid,' \n Maximum Iterations Reached, proceedure unsuccessful \n');

%fclose(fid);

%*************************************************************************
% PROGRAM Chapin_Obective_Function
%
% The purpose of this MATLAB program is to evaluate the objective
% function of the queueing system under consideration
%
%       Author:  Patrick S. Chapin, AFIT, GOR-04M
%       Date:  21 Oct 03
%       Last Revised:  4 Feb 04
%       References: None
%       Inputs:  T, the replacement time of the system
%                mu, the vector of service rates
%
%*************************************************************************

% VARIABLE DEFINTIONS ***************************************************
% C_N = Replacement cost for 1 server
% C_H = Holding cost for 1 customer per unit time
% C_F = Additional cost of serving 1 customer at an outside facility.
% kservers = the number of servers
% lambda = arrival rate of customers to the system.
% soe = Size of the random environment.
```

```
% Q = infitesimal generator matrix of the random environment.
% qswitch = a counter so that the the steady state solutions of the random environment
%         only calculated once.
% R_D = matrix of the rate of wear during each environmental state.  if the
%         environment is in state j and the wear rate is a, then R_D(j,j)
%         = a.
% Type1 - Type4 = Counters to differentiate between different laplace inversion
%         functions.  It is easier to tell what is going on if Type1 is
%         passed as opposed to just 1.
%*************************************************************************

% VECTOR/FUNCTION DEFINITIONS*********************************************
%
%*************************************************************************

function output = Chapin_Objective_Function(T,mu)

MaximumWear = 1.0;



global soe
global lambda
global qswitch
global q
global R_D
global C_N

kservers =1;
C_N = 15;

if soe == 2
   C_F = 2;
   C_N = 18;
   C_H = 15;
elseif soe ==5
   C_F = 20;
   C_N = 10;
   C_H = 5;
elseif soe == 10
   C_F = 10;
   C_N = 5;
   C_H = 20;
   kservers = 4;
else
   return
end




%Variables to denote which function I am inverse Laplacing.
Type1 = 1;
Type2 = 2;
Type3 = 3;
```

```
Type4 = 4;

%Assume M/M/1 Queue, 5 state random enivornment
if soe == 2
   a = 19;
   b = 7;
   Q = [-b b;a -a];

   Q = Q*(1/10);
elseif soe == 5
Q=[-76.4653       19.1376 28.2982 16.5118 12.5177
25.6793 -81.185  28.9809 14.8722 11.6526
15.2226 29.5272 -87.693229.3102 13.6332
28.4067 18.8163 11.3262 -79.166920.6177
14.1677 10.1135 19.5026 25.4786 -69.2624];

Q=Q./10;
elseif soe == 10
Q = [  -3472.932    573.5848   518.6157   247.7199   222.8372   282.9065   276.7697   486.86     377.6327   486.0055
528.9639   -3464.2659             346.6875   268.6725   494.174    390.0185   421.1792   561.8889   120.426    332.2554
215.1789   379.774    -2638.3644             445.3362   471.2741   265.6516   244.8543   334.6254   134.4901   147.1798
110.7218   506.8413   170.6219   -2467.6896             142.3301   107.2723   533.6251   270.8044   410.6428   214.8299
142.7507   411.9563   491.1781   101.2544   -3160.385   140.8118   596.2472   501.9074   517.8909   256.3882
499.5321   392.8141   165.9629   104.3732   586.9225   -3679.0386             479.72     591.5364   423.7168   434.4606
257.7228   585.8839   100.8181   323.5683   358.0541   564.9279   -3297.6269             230.9467   326.5551   549.15
563.2924   295.443    340.6363   232.606    471.9588   585.0955   238.3625   -3438.1854             289.4849   421.306
192.9471   252.2573   157.9861   374.6915   508.0261   531.4705   259.1157   250.9286   -3058.4467             531.0238
316.1591   119.4228   205.5623   578.4263   399.5693   167.2138   333.538    177.418    582.0245   -2879.3341
 ];


      Q=Q./ 100;
else
   return
end
%For now, assume r(j) = mu_j.
%R_D = diag(mu);
%solve for steady state probabilities of the random environment
%Solve qQ = 0 and q'*m = 1
if qswitch ==1
   qswitch = 2;
   Qnew = [Q';ones(size(Q,1),1)'];
   RHS = [zeros(size(Q,1),1);1];
   q=Qnew\RHS;
end
%the purpose of the next statement is to check to ensure the given policy
%matches the size of the random environment.  If not, this statement should
%cause an error and abort the routine.
mu - q;

for k=1:soe
      if soe == 10
         R_D(k,k) = (k*mu(k)/2)^2;
      elseif soe == 5
         R_D(1,1) = mu(1);
         R_D(2,2) = mu(2)^2;
         R_D(3,3) = exp(mu(3));
         R_D(4,4) = log(mu(4)+1);
```

```
        R_D(5,5) = mu(5)^3;
      elseif soe == 2
        R_D(k,k) = mu(k)*k;
      end
end
if soe == 2
   R_D = R_D * (1/10);
elseif soe == 5
   R_D = R_D * (1/10);
elseif soe == 10
   R_D = R_D * (1/100);
end


inverselap1 = Chapinmod_invt_lap(T,mu,Type1,Q,R_D,MaximumWear,q);
inverselap2 = Chapinmod_invt_lap(T,mu,Type2,Q,R_D,MaximumWear,q);
inverselap3 = Chapinmod_invt_lap(T,mu,Type3,Q,R_D,MaximumWear,q);
inverselap4 = Chapinmod_invt_lap(T,mu,Type4,Q,R_D,MaximumWear,q);



output =  C_F*lambda*(T*inverselap3*inverselap2 + T*inverselap1*inverselap1 -
inverselap1*inverselap2) - kservers*C_N;



%=====================================================================
%The Invert Laplace function -- This code is from Dr. J. P. Kharoufeh

function f1 = Chapinmod_invt_lap(t,mu,type,Q,R_D,maxwear,q)
rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;

for k=0:m
  d=nchoosek(m,k);
  c=[c d];
end
for t = t; %50.0; %t=0.5:0.5:20.0
  tx = t; %[tx t];
  ntr=15;
  u=exp(A/2)/t;
  x=A/(2*t);
  h=pi/t;
  su=zeros(m+2);
  sm=chapin_evaluate_fct(x,0,mu,type,Q,R_D,maxwear,q)/2;
  for k=1:ntr
    y=k*h;
    sm=sm+((-1)^k)*chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q);
  end
  su(1)=sm;
  for k=1:12
    n=ntr+k;
    y=n*h;
    su(k+1)=su(k)+((-1)^n)*chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q);
  end
  av1=0; av2=0;
```

```
  for k=1:12
    av1=av1+c(k)*su(k);
    av2=av2+c(k)*su(k+1);
  end
  f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end

%=====================================================================
%The functional Evaluation Required in the invert Laplace function

function eq=chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q)

s=x+y*i;
% Input the form of your Laplace transform here.  For example, if you have
% the LT of the pdf of an Exp(lambda) r.v., then the LT is:
%lambda = 5.0;
%z = (5/(5+s));
%eq = real(z);
m=ones(size(Q),1);
I=eye(size(Q));

%=================================================
% Type 1
if type == 1
   z = (1/s) * q' * expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);

%End Part 1
%=================================================
%Type 2
elseif type ==2
   z = (1/s)^2 * q' * expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);
elseif type ==3
%     z = q' * inv(R_D) * maxwear * expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   z = q' *  expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);
elseif type ==4
   z = (1/s)* q' * inv(R_D) * maxwear * expm(inv(R_D)*(Q-(s*I))*maxwear) * m + (1/s)^2 * q' *
          expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);
end


%*************************************************************************
% PROGRAM Chapin_Obective_Function
%
% The purpose of this MATLAB program is to evaluate the objective
% function of the queueing system under consideration
%
% *******NOTE:  This function uses the empirical CDF generated using the
% *******             Simulation Data to evaluate the objective function
%
%       Author:  Patrick S. Chapin, AFIT, GOR-04M
%       Date:  21 Oct 03
%       Last Revised:  4 Feb 04
```

```
%       References: None
%       Inputs:  T, the replacement time of the system
%                mu, the vector of service rates
%
%*************************************************************************

% VARIABLE DEFINTIONS ***************************************************
% C_N = Replacement cost for 1 server
% C_H = Holding cost for 1 customer per unit time
% C_F = Additional cost of serving 1 customer at an outside facility.
% kservers = the number of servers
% lambda = arrival rate of customers to the system.
% soe = Size of the random environment.
% Q = infitesimal generator matrix of the random environment.
% qswitch = a counter so that the the steady state solutions of the random environment
%         only calculated once.
% R_D = matrix of the rate of wear during each environmental state.  if the
%         environment is in state j and the wear rate is a, then R_D(j,j)
%         = a.
% Type1 - Type4 = Counters to differentiate between different laplace inversion
%         functions.  It is easier to tell what is going on if Type1 is
%         passed as opposed to just 1.
%*************************************************************************

% VECTOR/FUNCTION DEFINITIONS*********************************************
%
%*************************************************************************

function output = Chapin_Objective_Function(T,mu)

MaximumWear = 1.0;

C_F = 10;
C_H = 8;

global soe
global lambda
global qswitch
global Tformat %This is the matrix of the cdf.


kservers =1;
C_N = 10;

if soe == 2
   C_F = 6;
   C_N = 18;
   C_H = 15;
elseif soe ==5
   C_F = 20;
   C_N = 10;
   C_H = 5;
elseif soe == 10
   C_F = 10;
   C_N = 5;
   C_H = 20;
```

```
   kservers = 4;
else
   return
end



if qswitch == 1
   qswitch = 2;
   if soe == 2
      load Tformat2.mat
   elseif soe == 5
      load Tformat5.mat
   elseif soe ==10
      load Tformat10.mat
   else
      return
   end
end

CDFEVAL = GetCDF(T);
INTEVAL = integratecdf(T);
DERIVEVAL = derivofcdf(T);

output =  C_F*lambda*(T*DERIVEVAL*INTEVAL+T*(CDFEVAL)^2-CDFEVAL*INTEVAL) -
kservers*C_N;

function output = GetCDF(T)

   global Tformat
   %objective is to return the value of the cdf at a point.
   Times = Tformat(:,1);
   Values = Tformat(:,2);
   if T >= max(Times)
      output = 1;
   elseif T <= min(Times)
      output = 0;
   else
      k=1;
      while Times(k) < T
         k = k+1;
      end
      if T == Times(k)
         output = Values(k);
      else
         if k == 1
            fprintf('problem');
            T
            Times(k)
         end
         j = k-1;
         while Times(j) > T
            j=j-1;
         end
         %interpolate between the two points
         Temp1 = Values(k);
```

```
        Temp2 = Values(j);
        Temp3 = Times(k);
        Temp4 = Times(j);
        Temp5 = (T-Times(j));
        output = (Values(k)-Values(j))/(Times(k)-Times(j))*(T-Times(j))+Values(j);

    end
end

function output = integratecdf(T)

    global Tformat
    Times = Tformat(:,1);
    Values = Tformat(:,2);

    if T >= max(Times)
        output = trapz(Times,Values) + T-max(Times);
    elseif T <= min(Times)
        output = 0;
    else
        k=1;
        while Times(k) < T
            k = k+1;
        end
        if T == Times(k)
            newtimes = Tformat(1:k,1);
            newvalues = Tformat(1:k,2);
            output = trapz(newtimes,newvalues);
        else
            j=k-1;
            while Times(j) > T
                j=j-1;
                fprintf('problem');
            end
            interp = (Values(j)-Values(k))/(Times(j)-Times(k))*(T-Times(k))+Values(j);
            newvalues = Tformat(1:j,2);
            newtimes = Tformat(1:j,1);
            newvalues = [newvalues;interp];
            newtimes = [newtimes;T];
            output = trapz(newtimes,newvalues);
        end
    end

function output = derivofcdf(T)
    global Tformat
    Times = Tformat(:,1);
    Values = Tformat(:,2);

    if T >= max(Times)
        output = 0;
    elseif T <= min(Times)
        output = 0;
    else
        k=1;
        while Times(k) < T
            k = k+1;
```

```
    Temp1 = Times(k);
  end
  if T == Times(k)
    %We know T = Times(k), now find the derivative
    if k+50 > size(Times)
      k = size(Times);
    elseif k-50 < 1
      k = 51;
    end
    output = (Values(k+5)-Values(k-5))/(Times(k+5)-Times(k-5));
  else
    j=k-1;
    while Times(j) > T
      j=j-1;
      fprintf('problem');
      return
    end
    if k+50 > size(Times)
      k = size(Times);
    elseif k-50 < 1
      k = 1;
    end
    output = (Values(k+5)-Values(j-5))/(Times(k+5)-Times(j-5));

  end
end
```

Appendix C. Gradient Search Code

```
%*****************************************************************************
% PROGRAM Chapin_Steepest_Descent
%
% The purpose of this MATLAB program is to use the method of steepest
% descent to minimize the long run expected cost rate of the system under
% consideration in Chapin's masters thesis.
%
%      Author:  Patrick S. Chapin, AFIT, GOR-04M
%      Date:  21 Oct 03, 2003
%      Last Revised:  21 Oct 03, 2003
%
%      References:  Burden, Richard L. and J. Douglas Faires.
%                   _Numerical_Analysis_, 7th ed.  Brooks/Cole, Pacific Grove CA.
%
%*****************************************************************************

% VARIABLE DEFINTIONS*************************************************************
% C_N = Replacement cost for 1 server
% C_H = Holding cost for 1 customer per unit time
% C_F = Additional cost of serving 1 customer at an outside facility.
% kservers = the number of servers
% lambda = arrival rate of customers to the system.
% mu = the service rate at each of the environmental states
% soe = Size of the random environment.
% Q = infitesimal generator matrix of the random environment.
% qswitch = a counter so that the the steady state solutions of the random environment
%         only calculated once.
% R_D = matrix of the rate of wear during each environmental state.  if the
%         environment is in state j and the wear rate is a, then R_D(j,j)
%         = a.
% Type1 - Type4 = Counters to differentiate between different laplace inversion
%         functions.  It is easier to tell what is going on if Type1 is
%         passed as opposed to just 1.
%*****************************************************************************

% VECTOR/FUNCTION DEFINITIONS*****************************************************
%
%*****************************************************************************

function output = Chapin_Steepest_Descent(initialguess,sizeofenvironment,fid)

maxiterations = 200;
   %Step 1
   StepCounter = 1;

   gradientg = ones(sizeofenvironment,1);
   ematrix = eye(sizeofenvironment);
   c=.005;

   global Tolerance
   global soe
   global lambda
   global upperbound
   global qswitch
   global kservers
```

```
if soe == 10
   kservers = 4;
   T= 2.734100;
elseif soe == 5
   T = 2.198010;
   kservers = 1;
elseif soe == 2
   T = 7.272270;
   kservers = 1;

end

lambda = 1;
soe = sizeofenvironment;
        upperbound = 300;
        qswitch = 1;
Tolerance = 1*10^-12;

        %Check for valid mu
        lastmu = muchecker(initialguess);


        %Step 2
        while StepCounter < maxiterations

    if mod(StepCounter,10)==0
       fprintf(' %d \n',StepCounter);
    end

    %Step 3
    g1 = Chapin_Objective_Function(T,muchecker(lastmu));

    for k = 1:sizeofenvironment
       %Build the gradient approximation using central differences
       gradientg(k) =  (Chapin_Objective_Function(T,muchecker(lastmu + c * ematrix(:,k)))-
hapin_Objective_Function(T,muchecker(lastmu - c * ematrix(:,k))))/(2*c);
    end

    grad0 = sqrt(dot(gradientg,gradientg));

    %Step 4
    if all(grad0 == 0)
       fprintf(fid,' \n finished, gradient equal to zero');
       if soe == 2
          fprintf(fid,'\n The Solution is: [%f %f]',lastmu);
       elseif soe == 5
          fprintf(fid,'\n The Solution is: [%f %f %f %f %f]',lastmu);
       elseif soe == 10
          fprintf(fid,'\n The Solution is: [%f %f %f %f %f %f %f %f %f %f]',lastmu);
       end

       Final_Cost_Rate = Chapin_Objective_Function(T,muchecker(lastmu));
       fprintf(fid,'\n The Final Cost Rate is: %f',Final_Cost_Rate);
       fprintf(fid,'\n It took %d steps',StepCounter);
       fprintf(fid,'\n \n');
       output = lstmu;
```

```
   return
end

gradientg = gradientg / grad0;


%Step 5
alpha1 = 0;
alpha3 = 8;
g3 = Chapin_Objective_Function(T,muchecker(lastmu - alpha3 * gradientg));

%Step 6
while (g3 >= g1)
   %Step 7
   alpha3 = alpha3 / 2;
   g3 = Chapin_Objective_Function(T,muchecker(lastmu - alpha3 * gradientg));

   %Step 8
   if alpha3 < Tolerance / 2
      fprintf(fid,' \n Very Small Step size, likely no improvement');
      if soe == 2
         fprintf(fid,'\n The Solution is: [%f %f]',lastmu);
      elseif soe == 5
         fprintf(fid,'\n The Solution is: [%f %f %f %f %f]',lastmu);
      elseif soe == 10
         fprintf(fid,'\n The Solution is: [%f %f %f %f %f %f %f %f %f %f]',lastmu);
      end

      Final_Cost_Rate = Chapin_Objective_Function(T,muchecker(lastmu));
      fprintf(fid,'\n The Final Cost Rate is: %f',Final_Cost_Rate);
      fprintf(fid,'\n It took %d steps',StepCounter);
      fprintf(fid,'\n \n');
      output = lastmu;
      return
   end
end

%Step 9
alpha2 = alpha3 /2;
g2 = Chapin_Objective_Function(T,muchecker(lastmu - alpha2 * gradientg));

%Step 10
h1 = (g2-g1)/alpha2;
h2 = (g3 - g2)/(alpha3 - alpha2);
h3 = (h2- h1)/alpha3;

%Step 11
alpha0 = .5*(alpha2 - h1/h3);
g0 = Chapin_Objective_Function(T,muchecker(lastmu - alpha0 * gradientg));

%Step 12
if g0 < g3
   alphanext = alpha0;
else
   alphanext = alpha3;
end
```

```
%Step 13
%update
oldmu = lastmu;
lastmu = muchecker(lastmu - alphanext * gradientg);

temp1 = abs(Chapin_Objective_Function(T,lastmu) - g1);
temp2 = Chapin_Objective_Function(T,lastmu);
%Step14, check for finish
if abs(Chapin_Objective_Function(T,lastmu) - g1) < Tolerance
   fprintf(fid,' \n Successful proceedure, less than tolerance change in objective function');
   if soe == 2
      fprintf(fid,'\n The Solution is: [%f %f]',lastmu);
   elseif soe == 5
      fprintf(fid,'\n The Solution is: [%f %f %f %f %f]',lastmu);
   elseif soe == 10
      fprintf(fid,'\n The Solution is: [%f %f %f %f %f %f %f %f %f %f]',lastmu);
   end

   Final_Cost_Rate = Chapin_Objective_Function(T,muchecker(lastmu));
   fprintf(fid,'\n The Final Cost Rate is: %f',Final_Cost_Rate);
   fprintf(fid,'\n It took %d steps',StepCounter);
   fprintf(fid,'\n \n');
   output = lastmu;
   return
end

%Step 16
StepCounter = StepCounter +1;

    end

    if StepCounter >= maxiterations-1
fprintf(fid,' \n Maximum Iterations Reached, proceedure unsuccessful \n');
   if soe == 2
      fprintf(fid,'\n The Solution is: [%f %f]',lastmu);
   elseif soe == 5
      fprintf(fid,'\n The Solution is: [%f %f %f %f %f]',lastmu);
   elseif soe == 10
      fprintf(fid,'\n The Solution is: [%f %f %f %f %f %f %f %f %f %f]',lastmu);
   end

   Final_Cost_Rate = Chapin_Objective_Function(T,muchecker(lastmu));
   fprintf(fid,'\n The Final Cost Rate is: %f',Final_Cost_Rate);
   fprintf(fid,'\n It took %d steps',StepCounter);
   fprintf(fid,'\n \n');
   output = lastmu;
    end


%=================================================================
%=================================================================
%Define a function to check the mus

function out = muchecker(mu)
```

```
global soe
global lambda
global upperbound
global Tolerance
global kservers



for k=1:soe
   %if mu(k) <= lambda(k) + 10^-8
   %    mu(k) = lambda(k) + 10^-8;
   %end
   if mu(k) <= lambda/kservers + 10^-8
      mu(k) = (lambda)/kservers + 10^-8;
   end
   if mu(k) > upperbound
      mu(k) = upperbound;
   end

       end

out = mu;


%*************************************************************************************
% PROGRAM Chapin_Obective_Function
%
% The purpose of this MATLAB program is to simply evaluate the objective
% function of the queueing system under consideration
%
%      Author:  Patrick S. Chapin, AFIT, GOR-04M
%      Date:  21 Oct 03
%      Last Revised:  4 Feb 04
%      References: None
%      Inputs:  T, the replacement time of the system
%               mu, the vector of service rates
%
%*************************************************************************************

% VARIABLE DEFINTIONS
**********************************************************************
% C_N = Replacement cost for 1 server
% C_H = Holding cost for 1 customer per unit time
% C_F = Additional cost of serving 1 customer at an outside facility.
% kservers = the number of servers
% lambda = arrival rate of customers to the system.
% soe = Size of the random environment.
% Q = infitesimal generator matrix of the random environment.
% qswitch = a counter so that the the steady state solutions of the random environment
%         only calculated once.
% R_D = matrix of the rate of wear during each environmental state.  if the
%         environment is in state j and the wear rate is a, then R_D(j,j)
%         = a.
% Type1 - Type4 = Counters to differentiate between different laplace inversion
%         functions.  It is easier to tell what is going on if Type1 is
%         passed as opposed to just 1.
```

```
%*****************************************************************************

% VECTOR/FUNCTION DEFINITIONS*************************************************
%
%*****************************************************************************

function output = Chapin_Objective_Function(T,mu)

MaximumWear = 1.0;
C_N = 10;
C_F = 10;
C_H = 8;

global soe
global lambda
global upperbound
global qswitch
global q
global kservers

if soe == 2
   C_F = 6;
   C_N = 18;
   C_H = 15;
   kservers = 1;
elseif soe ==5
   C_F = 20;
   C_N = 10;
   C_H = 5;
   kservers = 1;
elseif soe == 10
   C_F = 10;
   C_N = 5;
   C_H = 20;
   kservers =4;
else
   return
end



%Variables to denote which function I am inverse Laplacing.
Type1 = 1;
Type2 = 2;
Type3 = 3;
Type4 = 4;

%Assume M/M/1 Queue, 5 state random enivornment
if soe == 2
   a = 19;
   b = 7;
   Q = [-b b;a -a];

   Q = Q*(1/10);
elseif soe == 5
```

```
Q=[-76.4653       19.1376 28.2982 16.5118 12.5177
25.6793 -81.185  28.9809 14.8722 11.6526
15.2226 29.5272 -87.693229.3102 13.6332
28.4067 18.8163 11.3262 -79.166920.6177
14.1677 10.1135 19.5026 25.4786 -69.2624];

Q=Q./10;
elseif soe == 10
Q = [   -3472.932   573.5848   518.6157   247.7199   222.8372   282.9065   276.7697   486.86     377.6327   486.0055
528.9639  -3464.2659            346.6875   268.6725   494.174    390.0185   421.1792   561.8889   120.426    332.2554
215.1789  379.774    -2638.3644            445.3362   471.2741   265.6516   244.8543   334.6254   134.4901   147.1798
110.7218  506.8413   170.6219   -2467.6896            142.3301   107.2723   533.6251   270.8044   410.6428   214.8299
142.7507  411.9563   491.1781   101.2544   -3160.385  140.8118   596.2472   501.9074   517.8909   256.3882
499.5321  392.8141   165.9629   104.3732   586.9225   -3679.0386            479.72     591.5364   423.7168   434.4606
257.7228  585.8839   100.8181   323.5683   358.0541   564.9279   -3297.6269            230.9467   326.5551   549.15
563.2924  295.443    340.6363   232.606    471.9588   585.0955   238.3625   -3438.1854            289.4849   421.306
192.9471  252.2573   157.9861   374.6915   508.0261   531.4705   259.1157   250.9286   -3058.4467            531.0238
316.1591  119.4228   205.5623   578.4263   399.5693   167.2138   333.538    177.418    582.0245   -2879.3341
 ];


      Q=Q./ 100;
else
   return
end
%For now, assume r(j) = mu_j.
%R_D = diag(mu);
%solve for steady state probabilities of the random environment
%Solve qQ = 0 and q'*m = 1
if qswitch ==1
   qswitch = 2;
   Qnew = [Q';ones(size(Q,1),1)'];
   RHS = [zeros(size(Q,1),1);1];
   q=Qnew\RHS;
end
%the purpose of the next statement is to check to ensure the given policy
%matches the size of the random environment.  If not, this statement should
%cause an error and abort the routine.
mu - q;

for k=1:soe
      if soe == 10
         R_D(k,k) = (k*mu(k)/2)^2;
      elseif soe == 5
         R_D(1,1) = mu(1);
         R_D(2,2) = mu(2)^2;
         R_D(3,3) = exp(mu(3));
         R_D(4,4) = log(mu(4)+1);
         R_D(5,5) = mu(5)^3;
      elseif soe == 2
         R_D(k,k) = mu(k)*k;
      end
end
if soe == 2
   R_D = R_D* (1/10);
elseif soe == 5
   R_D = R_D * (1/10);
elseif soe == 10
```

```
   R_D = R_D * (1/100);
end


inverselap1 = Chapinmod_invt_lap(T,mu,1,Q,R_D,MaximumWear,q);
inverselap2 = Chapinmod_invt_lap(T,mu,2,Q,R_D,MaximumWear,q);
ReplacementCost = kservers * C_N;
HoldingCost = C_H * T * LFCT(mu,lambda,q);
WorkCost = lambda * T * WorkCostfct(mu,q);
FailureCost = C_F* lambda * inverselap1 * inverselap2;

output =  (ReplacementCost + HoldingCost + WorkCost + FailureCost)/T;


%=========================================================================
%The work cost function
function out = WorkCostfct(mu,q)
global soe
cost = 0;
for k = 1:soe


   if soe == 10
      cost = cost + q(k) *  (mu(k)^2);   % Here is were we put the work cost
   elseif soe ==5
      cost = cost + q(k) *  (mu(k))^2;
   elseif soe ==2
       cost = cost + q(k) * 5 * mu(k);
   end


end

out = cost;

%=========================================================================
%The waiting function
function out = LFCT(mu,lambda,q)
global soe
global kservers


mubar = mu' * q;
if soe == 10
   temp1 = 0;
   rho = lambda / (kservers * mubar);
   for k = 0:(kservers-1)
      temp1 = temp1 + 1/factorial(k) * (lambda /mubar)^k;
   end
   temp2 = temp1 + (1/kservers)*(lambda / mubar)^kservers*(1/(1-lambda/(kservers*mubar)));
   pnot = temp2^(-1);

   out = (pnot*((lambda/mubar)^kservers) * lambda)/(factorial(kservers)*kservers*(1-
         lambda/(kservers*mubar))^2) + lambda / mubar;

else
```

```
    out =  lambda / (mubar - lambda);
end

%=========================================================================
%The Invert Laplace function -- This code is from Dr. J. P. Kharoufeh

function f1 = Chapinmod_invt_lap(t,mu,type,Q,R_D,maxwear,q)
rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;

for k=0:m
  d=nchoosek(m,k);
  c=[c d];
end
for t = t; %50.0; %t=0.5:0.5:20.0
  tx = t; %[tx t];
  ntr=15;
  u=exp(A/2)/t;
  x=A/(2*t);
  h=pi/t;
  su=zeros(m+2);
  sm=chapin_evaluate_fct(x,0,mu,type,Q,R_D,maxwear,q)/2;
  for k=1:ntr
    y=k*h;
    sm=sm+((-1)^k)*chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q);
  end
  su(1)=sm;
  for k=1:12
    n=ntr+k;
    y=n*h;
    su(k+1)=su(k)+((-1)^n)*chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q);
  end
  av1=0; av2=0;
  for k=1:12
    av1=av1+c(k)*su(k);
    av2=av2+c(k)*su(k+1);
  end
  f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end

%=========================================================================
%The functional Evaluation Required in the invert Laplace function

function eq=chapin_evaluate_fct(x,y,mu,type,Q,R_D,maxwear,q)

s=x+y*i;
% Input the form of your Laplace transform here.  For example, if you have
% the LT of the pdf of an Exp(lambda) r.v., then the LT is:
%lambda = 5.0;
%z = (5/(5+s));
%eq = real(z);
m=ones(size(Q),1),1);
I=eye(size(Q));

%================================================
% Part 1, the expected value
if type == 1
```

```
   z = (1/s) * q' * expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);

%End Part 1
%=============================================
%Part 2, The integration of the cumulative distribution function
else
   z = (1/s)^2 * q' * expm(inv(R_D)*(Q-(s*I))*maxwear) * m;
   eq = real(z);
end

%*****************************************************************************************
% PROGRAM multrd
%
%   The purpose of this matlab code is simply to run all the corner point
%   initial solutions to the three example problems used in this thesis.
%
%       Author:  Patrick S. Chapin, AFIT, GOR-04M
%         Date:  21 Oct 03
% Last Revised:  4 Feb 04
%   References: None
%       Inputs: None
%
%
%*****************************************************************************************

function output = multrd

T1 = clock;

fprintf('\nRunning Multiple Runs');
fprintf('\nStarting the 2 State Problem');

fid = fopen('2stateresultswithalteredC_H.txt','w');

fprintf('\nStarting First Run');
Chapin_Steepest_Descent([0;0],2,fid);
fprintf('\nStarting Second Run');
Chapin_Steepest_Descent([50;50],2,fid);
fprintf('\nStarting Third Run');
Chapin_Steepest_Descent([50;0],2,fid);
fprintf('\nStarting Fourth Run');
Chapin_Steepest_Descent([0;50],2,fid);

fprintf('\n2 State Problem Complete, Beginning 5 state problem');
fclose(fid);



fid = fopen('5stateresults.txt','w');
fprintf('\nstarting first run, 5 state');
Chapin_Steepest_Descent([0;0;0;0;0],5,fid);
fprintf('\nstarting second run, 5 state');
Chapin_Steepest_Descent([15;15;15;15;15],5,fid);
fprintf('\nstarting 3 run, 5 state');
Chapin_Steepest_Descent([15;0;0;0;0],5,fid);
```

```
fprintf('\nstarting 4 run, 5 state');
Chapin_Steepest_Descent([0;15;0;0;0],5,fid);
fprintf('\nstarting 5 run, 5 state');
Chapin_Steepest_Descent([0;0;15;0;0],5,fid);
fprintf('\nstarting 6 run, 5 state');
Chapin_Steepest_Descent([0;0;0;15;0],5,fid);
fprintf('\nstarting 7 run, 5 state');
Chapin_Steepest_Descent([0;0;0;0;15],5,fid);
fprintf('\nstarting 8 run, 5 state');
Chapin_Steepest_Descent([15;15;0;0;0],5,fid);
fprintf('\nstarting 9 run, 5 state');
Chapin_Steepest_Descent([15;0;15;0;0],5,fid);
fprintf('\nstarting 10 run, 5 state');
Chapin_Steepest_Descent([15;0;0;15;0],5,fid);
fprintf('\nstarting 11 run, 5 state');
Chapin_Steepest_Descent([15;0;0;0;15],5,fid);
fprintf('\nstarting 12 run, 5 state');
Chapin_Steepest_Descent([0;15;15;0;0],5,fid);
fprintf('\nstarting 13 run, 5 state');
Chapin_Steepest_Descent([0;15;0;15;0],5,fid);
fprintf('\nstarting 14 run, 5 state');
Chapin_Steepest_Descent([0;15;0;0;15],5,fid);
fprintf('\nstarting 15 run, 5 state');
Chapin_Steepest_Descent([0;0;15;15;0],5,fid);
fprintf('\nstarting 16 run, 5 state');
Chapin_Steepest_Descent([0;0;15;0;15],5,fid);
fprintf('\nstarting 17 run, 5 state');
Chapin_Steepest_Descent([0;0;0;15;15],5,fid);
fprintf('\nstarting 18 run, 5 state');
Chapin_Steepest_Descent([15;15;15;0;0],5,fid);
fprintf('\nstarting 19 run, 5 state');
Chapin_Steepest_Descent([15;15;0;15;0],5,fid);
fprintf('\nstarting 15 run, 5 state');
Chapin_Steepest_Descent([15;0;15;15;0],5,fid);
fprintf('\nstarting 21 run, 5 state');
Chapin_Steepest_Descent([0;15;15;15;0],5,fid);
fprintf('\nstarting 22 run, 5 state');
Chapin_Steepest_Descent([15;15;0;0;15],5,fid);
fprintf('\nstarting 23 run, 5 state');
Chapin_Steepest_Descent([15;0;15;0;15],5,fid);
fprintf('\nstarting 24 run, 5 state');
Chapin_Steepest_Descent([0;15;15;0;15],5,fid);
fprintf('\nstarting 25 run, 5 state');
Chapin_Steepest_Descent([15;0;0;15;15],5,fid);
fprintf('\nstarting 26 run, 5 state');
Chapin_Steepest_Descent([0;15;0;15;15],5,fid);
fprintf('\nstarting 27 run, 5 state');
Chapin_Steepest_Descent([0;0;15;15;15],5,fid);
fprintf('\nstarting 28 run, 5 state');
Chapin_Steepest_Descent([15;15;15;15;0],5,fid);
fprintf('\nstarting 29 run, 5 state');
Chapin_Steepest_Descent([15;15;15;0;15],5,fid);
fprintf('\nstarting 30 run, 5 state');
Chapin_Steepest_Descent([15;15;0;15;15],5,fid);
fprintf('\nstarting 31 run, 5 state');
Chapin_Steepest_Descent([15;0;15;15;15],5,fid);
```

```
fprintf('\nstarting 32 run, 5 state');
Chapin_Steepest_Descent([0;15;15;15;15],5,fid);

fprintf('\n5 state problem finished');
fclose(fid);

% T2 = clock;
%
% TimeRequired = (T2-T1)'

fprintf('\nbegining 10 state problem');
fid = fopen('10stateresults.txt','w');
runs = 2;

fprintf('\nstarting 1 run, 10 state');
fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',[0;0;0;0;0;0;0;0;0;0]);
Chapin_Steepest_Descent([0;0;0;0;0;0;0;0;0;0],10,fid);

% fprintf('\nstarting 1 run, 10 state');
% fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',[0;0;0;0;0;0;0;0;0;0]);
% Chapin_Steepest_Descent([0;0;0;0;0;0;0;0;0;20],10,fid);

% fclose(fid);
%1 20

for k = 1:10
   tempmu = zeros(10,1);
   tempmu(k) = 20;
   tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
   fprintf(tempstring);
   fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
   Chapin_Steepest_Descent(tempmu,10,fid);
   runs = runs+1;
end

%2 20s
for k = 1:10
   %k = position of 1st 50
   for j = k:10
      if j ~= k
         tempmu = zeros(10,1);
         tempmu(k) = 20;
         tempmu(j) = 20;
         tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
         fprintf(tempstring);
         fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
         Chapin_Steepest_Descent(tempmu,10,fid);
         runs = runs+1;
      end
   end
end

%3 20s
for k = 1:10
   %k = position of 1st 50
   for j = k:10
```

```
      for l = j:10
         if j ~= k && l ~= j && l ~= k
            tempmu = zeros(10,1);
            tempmu(k) = 20;
            tempmu(j) = 20;
            tempmu(l) = 20;
            tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
            fprintf(tempstring);
            fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
            Chapin_Steepest_Descent(tempmu,10,fid);
            runs = runs+1;
         end
      end
   end
end

%4 20s
for k = 1:10
   %k = position of 1st 50
   for j = k:10
      for l = j:10
         for m = l:10
            if j ~= k && j~=l && j~=m && k~=l && l~=m && l ~= k
               tempmu = zeros(10,1);
               tempmu(k) = 20;
               tempmu(j) = 20;
               tempmu(l) = 20;
               tempmu(m) = 20;
               tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
               fprintf(tempstring);
               fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
               Chapin_Steepest_Descent(tempmu,10,fid);
               runs = runs+1;
            end
         end
      end
   end
end

%5 20s
for k = 1:10
   %k = position of 1st 50
   for j = k:10
      for l = j:10
         for m = l:10
            for n = m:10
               if j ~= k && j ~= l && j ~= m && j ~= n && k ~=l && k ~= m && k ~=n && l ~= m && l
~=n && m ~=n
                  tempmu = zeros(10,1);
                  tempmu(k) = 20;
                  tempmu(j) = 20;
                  tempmu(l) = 20;
                  tempmu(m) = 20;
                  tempmu(n) = 20;
                  tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
                  fprintf(tempstring);
```

```
                fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
                Chapin_Steepest_Descent(tempmu,10,fid);
                runs = runs+1;
            end
        end
    end
  end
end

%4 0s or 6 20s
for k = 1:10
  %k = position of 1st 0
  for j = k:10
    for l = j:10
      for m = l:10
        if j ~= k && j ~= l && m ~= l
          tempmu = ones(10,1)*20;
          tempmu(k) = 0;
          tempmu(j) = 0;
          tempmu(l) = 0;
          tempmu(m)=0;
          tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
          fprintf(tempstring);
          fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
          Chapin_Steepest_Descent(tempmu,10,fid);
          runs = runs+1;
        end
      end
    end
  end
end

%3 0s or 7 20s
for k = 1:10
  %k = position of 1st 0
  for j = k:10
    for l = j:10
      if j ~= k && j ~= l
        tempmu = ones(10,1)*20;
        tempmu(k) = 0;
        tempmu(j) = 0;
        tempmu(l) = 0;
        tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
        fprintf(tempstring);
        fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
        Chapin_Steepest_Descent(tempmu,10,fid);
        runs = runs+1;
      end
    end
  end
end

%2 0s or 8 20s
for k = 1:10
  %k = position of 1st 0
```

```
    for j = k:10
        if j ~= k
            tempmu = ones(10,1)*20;
            tempmu(k) = 0;
            tempmu(j) = 0;
            tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
            fprintf(tempstring);
            fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
             Chapin_Steepest_Descent(tempmu,10,fid);
            runs = runs+1;
        end
    end
end

for k = 1:10
    tempmu = ones(10,1)*20;
    tempmu(k) = 0;
    tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
    fprintf(tempstring);
    fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',tempmu);
     Chapin_Steepest_Descent(tempmu,10,fid);
    runs = runs+1;
end

tempstring = strcat('\nstarting ',int2str(runs),' run, 10 state');
fprintf(tempstring);
fprintf(fid,'\nStarting Point is: [%f %f %f %f %f %f %f %f %f %f]\n',[20;20;20;20;20;20;20;20;20;20]);
Chapin_Steepest_Descent([20;20;20;20;20;20;20;20;20;20],10,fid);


fclose(fid);

T2 = clock;
TimeRequired = (T2-T1)'
```

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
01-09-2004	**Master's Thesis**	March 2003 – Mar 2004

4. TITLE AND SUBTITLE

AGE REPLACEMENT AND SERVICE RATE CONTROL OF STOCHASTICALLY DEGRADING QUEUES

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Chapin, Patrick S., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/04-02

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

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis considers the problem of optimally selecting a periodic replacement time for a multiserver queueing system in which each server is subject to degradation as a function of the mean service rate and a stochastic and dynamic environment. Also considered is the problem of optimal service rate selection for such a system. In both cases, the performance metric is the long-run average cost rate. Analytical expressions are obtained, in terms of Laplace transforms, for the nonlinear objective functions, necessitating the use of numerical Laplace transform inversion to evaluate candidate solutions in conjunction with standard numerical algorithms. Due to the convexity of the objective function, the optimal replacement time is computed using a hybrid bisection-secant method which yields globally optimal solutions. The optimal service rates are obtained via gradient search methods but are only guaranteed to provide locally optimal solutions. The analytical results are implemented on three notional examples that demonstrate the benefits of dynamically adjusting service rates under the described maintenance policy.

15. SUBJECT TERMS
Queueing Theory; Control of Queueing Systems; Degrading Queueing Systems; Environment Driven Wear

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Jeffrey P. Kharoufeh, AFIT/ENS
U	U	U	UU	101	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4336; e-mail: jeffrey.kharoufeh@afit.edu