5-2019

Fault Adaptive Workload Allocation for Complex Manufacturing Systems

Charlie B. DeStefano
University of Arkansas, Fayetteville

Fault Adaptive Workload Allocation for Complex Manufacturing Systems

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

by

Charlie B. DeStefano
University of Southern California
Bachelor of Science in Mechanical Engineering, 2010
University of Arkansas
Master of Science in Mechanical Engineering, 2014

May 2019
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

ABSTRACT

This research proposes novel fault adaptive workload allocation (FAWA) strategies for the health management of complex manufacturing systems. The primary goal of these strategies is to minimize maintenance costs and maximize production by strategically controlling when and where failures occur through condition-based workload allocation.

For complex systems that are capable of performing tasks a variety of different ways, such as an industrial robot arm that can move between locations using different joint angle configurations and path trajectories, each option, i.e. mission plan, will result in different degradation rates and life-expectancies. Consequently, this can make it difficult to predict when a machine will require maintenance, as it will depend not only on the type and quality of the machine, but the actual tasks and mission plans it is performing. Furthermore, effective maintenance planning becomes increasingly challenging when dealing with complex systems, such as manufacturing production lines, that have multiple machines all performing different tasks, as the different degradation rates of each task will likely cause sporadic failures, leading to excessive work stoppages and lost production.

In response, this work proposes novel strategies for optimizing maintenance schedules through fault adaptive workload allocation (FAWA). This work will show how we can alternate between multiple mission plans and task assignments to control degradation across multiple components, guiding failures to occur at optimal times and locations. We will present two unique strategies for degradation control. The first strategy attempts to synchronize maintenance by utilizing multiple mission plans and task assignments, such that the healthiest components do the most work, whenever possible, in order to compensate for the more degraded components.

This promotes balanced degradation and synchronized failures across all components, allowing the number of work stoppages to be minimized. The second strategy involves desynchronizing maintenance by alternating between mission plans and task assignments where the healthiest components do either the most work or the least work in order to maintain an optimal difference between component degradation rates, such that overlapping failures are minimized. In this work, FAWA is applied to several case studies involving two types of manufacturing systems: industrial robot arms and 3D printers.

ACKNOWLEDGEMENTS

I would like to acknowledge the guidance and support provided to me by my advisor, Dr. David Jensen, who has allowed me the opportunity to pursue my wildest ideas and contribute to many intellectually stimulating projects over these past years. I also would like to mention my appreciation to the members of my dissertation committee, Dr. Zhenghui Sha, Dr. Haitao Liao, Dr. Wenchao Zhou, and Dr. Harry Pierson. Thank you all for your assistance in making this research possible.

Last, but not least, I would like to acknowledge my sincere appreciation to my parents, Pete and Sue DeStefano, my siblings, Sam and Mary, and countless other friends and family. Thank you all for believing in me and providing constant support and encouragement throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PUBLISHED PAPERS

1. C. DeStefano, D. Jensen. “Adaptive Mission Planning and Analysis for Complex Systems.” 2016. Journal of Computing and Information Science in Engineering.

2. C. DeStefano, D. Jensen. “Fault Adaptive Mission Planning: Increasing Useful-Life and Reducing Downtime Through Condition-Based Decision-Making.” 2019. Journal of Computing and Information Science in Engineering. (in review)

3. C. DeStefano, D. Jensen. “Fault Adaptive Workload Allocation for Complex Manufacturing Systems.” 2019. (in progress)

CHAPTER 1

INTRODUCTION

The primary objective of this research is to explore new health management strategies for complex manufacturing systems with the goal of minimizing maintenance costs and maximizing production. Many current health management strategies are based on increasing system robustness and resilience, i.e. the ability to effectively withstand disturbances and the ability to return to a nominal state after a disturbance, respectively [1]. Unfortunately, these types of safeguards tend to require expensive or redundant components in concert with extensive design efforts. Moreover, while it is obviously desirable to protect against failures through improved designs, unfortunately, no matter how well a system is built, eventually, it is guaranteed to fail. Furthermore, when dealing with manufacturing systems, the primary issue is not machines breaking down, but rather machines breaking down at inopportune times. For most manufacturers, the main issue associated with downtime is not necessarily the costs of repairing a machine, but rather the revenue lost by not being able to maintain production. Further complicating the issue, for large manufacturing systems that are comprised of multiple machines performing a variety of different tasks, each task will likely have different workload requirements, putting different stresses on the system, resulting in a wide disparity in machine life-expectancies. Unfortunately, this means that maintenance will likely be required at irregular intervals based on which operations the different machines are tasked with. Therefore, while it is certainly important to increase the time between failures with robust and resilient machines, the goal of this research is to investigate new ways to minimize downtime through strategies capable of actively controlling when each machine fails, such that maintenance schedules can be optimized.

The strategies proposed in this work involve constant health evaluation and optimized workload allocation on both a part-to-part and machine-to-machine basis. Achieving optimal health management for complex systems can be incredibly challenging because most complex actions require unbalanced workload allocation, and therefore, degradation almost always occurs asymmetrically across a system's components. This degradation will typically be characterized as either a deterioration in performance, increased probability of failure, or a reduction in available resources. However, one of the defining qualities of complex systems is that they typically have multiple tasks that they can do with many potential mission plans for each, i.e. different ways to achieve each task, such as a robot arm being able to choose between any number of joint angle configurations and path trajectories to reach between two points. The challenge of predicting the life-expectancy for such systems comes from the fact that each of the potential tasks, along with each of their available mission plans, will put different stresses on the system's health. Therefore, not only can two identical systems have different lifespans based on performing different tasks, they could in fact have different lifespans while performing the same task, if they use different mission plans. When dealing with a single machine, this potential disparity may not be a major issue; whether a machine breaks down on a Tuesday or a Wednesday is probably of little concern. However, if you have multiple machines, such as in a manufacturing setting, and each machine's production depends on every other machine being operational, when a single machine breaks down, the whole system must be shut down, which can result in large amounts of lost production revenue over time. Therefore, knowing exactly when a machine will fail can be incredibly valuable, and being able to actively control when it fails, even more so.

Most manufacturing planning seems to involve identifying a machine's mission plan such that its task is completed while optimizing some metric, such as minimizing total cycle time, and then this plan will typically be implemented, unchanged, for the entirety of the machine's life. For example, an industrial robot arm may be tasked with moving an object between two points and the mission plan that is implemented is the trajectory that achieves this movement the fastest. Unfortunately, in a large manufacturing setting with multiple machines performing different tasks with different workload requirements, such as one robot arm performing a task with a small movement and another performing a large movement, each machine will experience different degradation rates per cycle. Therefore, having every machine maintain the same mission plan will almost certainly result in asymmetrical failure times from component-to-component and machine-to-machine leading to excessive downtime. In response to this problem, this work was motivated by the following question: is it possible to strategically alternate between different mission plans and tasks in order to control when and where failures occur? As we explain throughout this work, the simple answer is yes, and it can be achieved through fault adaptive workload allocation. Fault adaptive workload allocation (FAWA) is a novel process of artificially inducing desired degradation profiles by strategically alternating between different tasks and mission plans based on the current and projected health of each component in the system. However, many additional questions had to be answered before this process could be effectively deployed. For example, what type of failure information is required and how do we obtain it? How are the alternate mission plans chosen? How and when should alternate mission plans be initiated? What strategies should be used if you want to synchronize failures versus if you want to desynchronize failures? How do the strategies change when dealing with health-based compared to resource-based systems? How do the strategies change when dealing with constant versus increasing degradation rates? All of these questions are answered throughout this work, starting with a general overview of the solutions in Chapter 3, followed by detailed explanations and multiple case study examples in Chapter 4.

This research evaluates the effectiveness of FAWA on multiple manufacturing case studies involving industrial robot arms and 3D printers in a simulation environment. The first case study we examine involves using FAWA to synchronize health and resource-based failures for both single and multiple robot arm scenarios. For the single robot arm example, we look at alternating between different mission plans, i.e. joint angle configurations, in order to balance joint-to-joint degradation. Then, for the multiple robot arms example, we look at alternating between different tasks in order to balance machine-to-machine degradation. The second case study looks at resource-based failures involving 3D printers. In this study, multiple 3D printers with similar material capacities will be performing collaborative printing operations but each printer will be responsible for printing either different sized parts or a different number of parts. FAWA will be used to optimally plan printing assignments in order to optimize when each printer runs out of material. Before getting into these case studies, however, the following chapter will provide background information on current methods for performing failure analysis and health management, followed by a chapter covering the general theory and methodology of FAWA.

First, however, it should be noted that throughout this work, when we refer to a "component" we do not necessarily mean an individual part of a machine, but rather a "part" to the system's "whole", i.e. a component could be a joint and the system be a robot arm, or a component could be a robot arm and the system be the entire assembly line. Additionally, a "task" is a high-level objective of the system, and a "mission plan" is the specific sequence of actions taken by the components to accomplish said task.

1.1 Dissertation Outline

The outline of this research is highlighted as follows:

Chapter One gives an introduction of what the research is about, its goals, and lays out a general outline of the rest of the paper;

Chapter Two gives a general background on traditional failure analysis, fault adaption, and maintenance strategies;

Chapter Three gives general overview of fault adaptive workload allocation (FAWA) and breaks down the different strategies and when to use them;

Chapter Four presents multiple case studies for using fault adaptive workload allocation for optimizing maintenance schedules for industrial robot arms and 3D printers;

Chapter Five offers conclusions made about this research, the effectiveness of the proposed FAWA strategies, and makes recommendations about potential future work that can be done to expand on the efforts made in this work;

Chapter Six covers additional research done in the domain of machine learning and proposes a new type of ensemble classifier.

CHAPTER 2

BACKGROUND

2.1 Related Research

In order to optimize a system's lifespan and maintenance schedule the first step is to calculate the projected lifespan, however, for complex systems this can actually be quite difficult. It used to be believed that a system's lifespan depended solely on its duration of use or its age. However, for complex systems that have wide ranging functionality, capable of performing multiple operations, identical systems can have vastly different lifespans based on the different sequences of operations that they perform throughout their lives. Unfortunately, most existing failure analysis methods, such as FMEA [2,3,4,5], FMECA [6,7], FFDM [8,9], RED [10,11,12], and FFIP [13,14], can identify how a system might fail, what might lead to a failure, or how failures might propagate throughout the system, but none of them take into account how a real-world complex system is actually being used or how it is expected to be used in the future. Therefore, while these methods are beneficial during early stages of design to gain a general idea of how a system may react to various failures, they are not particularly useful in optimizing a system's lifespan once it is in use.

When approaching the task of optimizing the useful-lifespan of a system that is already in use, regular maintenance is currently the preferred approach. However, because a maintenance strategy and its schedule can lead to extensive costs, up to 60-80% of overall costs for some military projects [15], optimizing a system's maintenance has become essential. Maintenance costs continue to rise due to both increased costs of fixing equipment, due to the expensive nature of many new technologies, and also because as products become more valuable, the more revenue is lost every minute that production is halted. Therefore, in an attempt to improve maintenance, strategies such as Condition-Based Maintenance (CBM) have been developed [16,17]. Unlike traditional Preventive Maintenance techniques [18], which have strict schedule-driven maintenance practices that may lead to unnecessary inspections, CBM schedules its maintenance based on how a system is being used and how each component is actually degrading. This allows for fewer inspections by only performing them at critical junctures. Unfortunately, while CBM addresses the problem of how to optimize a system's maintenance schedule for a given operation, it does not address optimizing the system's mission plans and task assignments. This is despite the fact that the tasks and mission plans that are implemented are the driving factors behind how degradation will accumulate in each component, and therefore, ultimately define the system's maintenance needs. In response to this omission, FAWA aims to build on the CBM agenda of utilizing component health states to drive maintenance scheduling. However, where CBM simply optimizes maintenance scheduling for a specific mission plan, FAWA aims to utilize multiple mission plans and task assignments in order to induce optimal degradation across all components, controlling failures and allowing for a truly optimized maintenance schedule.

A degradation control technique with similar goals and strategies to FAWA is the technique of Fault Detection, Isolation and Recovery (FDIR). Similar to FAWA, FDIR aims to limit the effects of a fault on the overall system, primarily by limiting the use of faulty components and using the healthy components more, or in a different way, than they normally would during nominal operating conditions [19,20,21,22]. Unfortunately, FDIR tends to rely on redundant components or preprogrammed corrective behaviors. In contrast, FAWA determines adaptive behaviors using reliability and resource-based metrics at the time of fault detection or when a designated degradation threshold is exceeded in order to ensure it is the most optimal strategy at that time. Also, FDIR tends to aim for simply providing acceptable functionality after a failure occurs, whereas FAWA aims to accomplish its current task while simultaneously optimizing their future maintenance schedules. Furthermore, FDIR is only implemented after a total component failure, whereas FAWA is more proactive and performs re-optimization at various stages of degradation. Lastly, FDIR has primarily been used for aerospace or electronic applications, whereas, for the purposes of this research, FAWA is applied to manufacturing systems.

Fault-tolerant control (FTC) [23,24] and Automated Contingency Management (ACM) [23,25] are also similar approaches to FAWA as they both aid a system's ability to re-plan a mission or reconfigure system controls based on health diagnostic information in order to ensure mission success. The primary differences however, are that FTC and ACM, similar to FDIR, primarily focus on completing a single mission while minimizing some sort of mission parameter, such as time or fuel consumption, whereas FAWA focuses on maximizing the number of mission cycles the system can perform in its remaining lifespan, while also maintaining the primary goal of optimizing future maintenance needs. Additionally, like FDIR, FTC's and ACM's corrective actions only take place after a total component failure, whereas FAWA makes proactive corrective actions at multiple stages of degradation. Moreover, FAWA uniquely incorporates the use of fault-augmented physics models for physically dependent systems in order to provide a more accurate simulation of how the system will behave under various failure states for a particular mission plan.

Dynamic workload adjustment [26] is a fault adaptive strategy that most closely resembles FAWA's goal of maintenance optimization and it has been applied to manufacturing for controlling the residual life distribution of parallel systems. This method looks at how a manufacturing line with multiple parallel units performing the same operation can adapt its total workload distribution in order to avoid overlapping failures, such that a minimum number of units is always available to meet productivity requirements. However, this method does not look at workload adjustment within a single unit, rather it assumes that each unit will perform the same operation from birth to death. Furthermore, it tends to assume that all units are performing similar operations. In contrast, this proposal's work investigates workload adjustment on the component level for individual units, and also on the unit level between multiple units where each one is potentially performing different tasks.

Lastly, the FAWA methodology presented in this research, is an extension of previous works developed by the authors on Failure Identification for Complex Mission Analysis (FICMA) [27] and Adaptive Mission Planning and Analysis (AMPA) [28], as well as being a new application of Fault-Augmented Model Extension research (FAME) [16,29]. The primary purpose of FICMA and AMPA was to examine potential missions for complex systems in order to identify which missions would provide the most adaptability. In addition, they also provided a preliminary examination of how FAWA could be implemented on a robot arm. FAME, on the other hand, provides Modelica components with parameterizable fault mechanisms as inputs so that partial failure analysis can be done in a physics-based simulation environment. Therefore, to accomplish this research's goals of providing a simulation and model-based fault adaptive workload allocation framework, the efforts of FICMA, AMPA, and FAME were combined and expanded.

CHAPTER 3

FAULT ADAPTIVE WORKLOAD ALLOCATION

3.1 General Overview

Fault adaptive workload allocation (FAWA) is defined in this work as alternating between tasks and mission plans at strategic points throughout a system's life, such that the required workload of each component is optimally distributed in order to control when and where failures occur. The primary requirement for FAWA to be possible is that a system's tasks must be adaptable. Adaptable tasks are system-level mission objectives that can be completed multiple ways by using different sequences of component-level behaviors, i.e. tasks that have multiple mission plans. By having more than one mission plan capable of accomplishing a task, a system is able to react to internal and external failures in a more optimal fashion. Each unique mission plan will have different component workload requirements, resulting in different component stresses and thus, different degradation profiles. Therefore, by having multiple mission plans, a system will be more likely able to find an option that utilizes each component in an optimal way, based on their current and projected health. However, simply having adaptable tasks does not guarantee fault adaptiveness. Despite there being more than one way to accomplish a task, there may be one mission plan that remains optimal under all scenarios, and thus, the additional mission plans will provide no added benefit as they will never be implemented. Therefore, it is necessary to make the distinction that having adaptable tasks provides the potential of a system being fault adaptive but it does not guarantee it.

For an example of a fault adaptive system let us examine the case of a smartphone. Its component-level behaviors, e.g. making a phone call or sending an email, can only be accomplished one way; if the microphone breaks, there isn't another way to talk during a phone call, and if there is no internet, you cannot send an email. However, looking at the system-level mission objective of transmitting a message to another person, if the phone's microphone breaks and the internet is down, a text message can still be sent. This is a very simple example of how having an adaptable task, i.e. being able to transmit a message through multiple means, allows a complex system to become fault adaptive by using alternative mission plans in the face of component failures.

Similarly, let us examine a vehicle with a system-level mission objective of driving multiple cycles back and forth between point A and point B, where the component-level behaviors are the individual left and right turns taken on a given route; imagine driving to and from work on the same route day after day. As previously stated, for the vehicle to be fault adaptive it must have an adaptable task with multiple possible mission plans, which in this case would mean there must be multiple routes that can be taken between points A and B. Then, in the event of component degradation, such as the front right tire experiencing excessive tread wear, an alternate route could be taken that utilizes more right turns than lefts in order to balance the degradation across each tire (because outside wheels are required to travel more distance during turns than inside wheels, by taking the alternate route with more right turns, the left tires would see an increase in wear rate compared to the right). Then, after multiple cycles on this route, once the left tire's wear surpasses that of the right, the original route with more left turns could then be re-implemented in order to maintain balanced degradation. However, some factors, such as mission time or resource consumption, may rule out certain alternatives, limiting the system's fault adaptiveness. This is an example of an n-of-n system where n=4 in reference to the number of tires.

In an n-of-n system all components must be operational at the same time and if one fails, the whole system must be shutdown, e.g. if you get one flat tire, you must pull the car over. When dealing with n-of-n systems, the goal is to synchronize maintenance by inducing a balanced final health state such that when one component fails all other components are also about to fail. For non-repairable systems, this is beneficial as it extends life-expectancy by reducing the likelihood of any one component experiencing a premature failure while other components still have useful-life remaining that would have been wasted. Moreover, for repairable systems, balanced degradation helps synchronize maintenance schedules as all components can be repaired or replaced simultaneously, effectively minimizing the number of work stoppages.

Alternatively, for k-of-n systems, where only k out of n components need to be functioning for the system to still operate, the goal is to desynchronize maintenance by inducing an optimally spaced, unbalanced final health state, such that failure overlaps, and therefore, work stoppages, are minimized. Remarkably, despite synchronizing and desynchronizing maintenance being completely opposite tasks, FAWA allows for both to be achieved with only minor variations to its strategy.

As we will expand on in the following sections, synchronizing maintenance involves always using the healthiest components the most, in order to compensate for the more degraded components, in an attempt to balance degradation. Then, contrastingly, desynchronizing maintenance involves alternating between using the healthiest components the most and using them the least, such that an optimal separation in degradation rates is maintained, minimizing overlapping failures. The overall flow chart for this methodology is depicted in Figure 3.1.

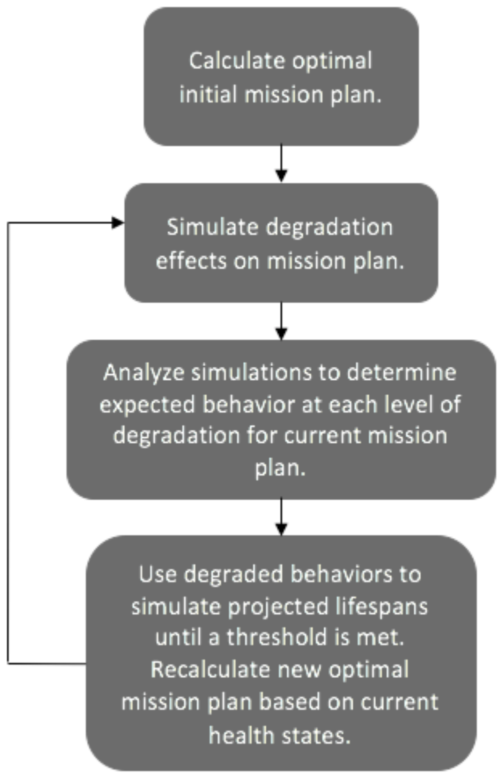

Calculate optimal initial mission plan.
Simulate degradation effects on mission plan.
Analyze simulations to determine expected behavior at each level of degradation for current mission plan.
Use degraded behaviors to simulate projected lifespans until a threshold is met. Recalculate new optimal mission plan based on current health states.

FIGURE 3.1 Flow chart for Fault Adaptive Workload Allocation (FAWA) methodology

3.2 FAWA Strategy for Synchronizing Maintenance

As previously mentioned, for n-of-n systems the ultimate goal is to achieve synchronized maintenance by inducing balanced degradation across each component in a system. Unfortunately, most tasks that a complex system will be asked to perform will not naturally have a balanced workload distribution. Therefore, except in rare cases, if a system uses the same mission plan to complete a task for the entirety of the system's life, it will almost always generate imbalanced degradation and therefore, will result in sporadic maintenance schedules. To counteract this, FAWA utilizes a rule-based planning algorithm that optimizes workload allocation based on the order of component health. Whenever possible, a new task or mission plan is implemented that gives the most degraded components the least demanding workload, based on current degradation rates, and the least degraded components are given the most demanding workload. By using the most extreme case, i.e. where the most degraded component does as little as possible, we maximize the life of the most degraded component while simultaneously allowing as much time as possible for the least degraded to catch up prior to a failure, effectively maximizing the likelihood of obtaining a balanced final health state. For constant degradation models, simulating this is quite simple as it does not matter when in the life cycle a mission plan is implemented as it will always have the same degradation rate. However, when switching between mission plans with progressive degradation rates things can become much more complicated.

For mission plans with progressive degradation rates, how they affect a component's projected degradation will depend on the component's health level at the time when they are initiated. Therefore, the specific time of when different mission plans are implemented can have a major impact on overall life-expectancy. For example, imagine running a race where you have three potential mission plans: you can either walk, jog, or sprint. The rate of exhaustion you experience per mile, and ultimately, your race time, completely changes based on the specific sequence you implement these running styles. If you sprint a mile, then jog a mile, then walk a mile, you will feel completely different and get a different time than if you walked a mile, jogged a mile, then sprinted a mile, despite the fact that you used each mission plan the exact same amount. This is because the difference in walking speed when you are fresh versus tired is much smaller than the difference in sprinting speed when you are fresh versus tired. Therefore, the degradation you experience from sprinting a mile at the start will be different than the degradation you experience from sprinting a mile at the end. Similarly, when alternating between mission plans with progressive degradation for FAWA, you cannot simply use one mission plan for a certain number of cycles and then switch to another mission plan and implement its degradation rate assuming the same number of cycles have passed. Instead, the entire life cycle for the alternative mission plan must be evaluated to find how many cycles it would have taken to get to its current health level, and then that component must be recalibrated to that number of cycles. For example, imagine an initial mission plan has a slow degradation rate for one component but a fast degradation rate for another, such that after 100 cycles component A is at a health level of 95% and component B is at 80%. Then, in order to balance degradation, we switch to an alternative mission plan where component A has a much faster degradation rate, such that if we had started with this new mission plan it would have reached 95% health after only 25 cycles, and component B has a much slower degradation rate, such that if we had started with this plan it would have reached 80% after 300 cycles. Therefore, component A must be recalibrated to behave as if it has only performed 25 cycles and component B must be recalibrated to behave as if it has performed 300 cycles in order to maintain the accurate degradation rates for the current health levels of each component. Ultimately, for systems with progressive degradation rates, this type of recalibration needs to be done for all components, every time an adaptive mission change takes place.

How frequently FAWA makes these adaptive mission changes is based on the standard deviation between the health states of each component. An upper limit threshold will then be set, such that anytime the standard deviation between component health states exceeds this threshold, an optimal workload allocation is calculated and necessary mission plan changes are made. Standard deviation is used because since every task and mission plan will have different degradation rates, if we were to use schedule-based intervals we would be forced to recalculate optimal intervals for every different task, whereas by using standard deviation, optimal intervals can automatically be determined in real-time by simply monitoring component health states. When determining the optimal standard deviation threshold, however, we must define what is optimal: that which maintains constant balance or that which results in the fewest number of adaptive changes. As has been mentioned numerous times at this point, a balanced final state is always preferred, and therefore, if the number of changes makes no difference, we would use a threshold equal to the maximum final state deviation we would be willing to accept, such as 1%. This would then make sure that degradation was balanced at all times. However, for certain cases, there may be a cost, whether in money or time, associated with changing mission plans, and so it may be better to simply end with a balanced health state, even if it means going through phases of imbalance. In this case, numerous simulations are ran to find the optimal threshold value that minimizes the deviation in final health states while also minimizing the number of changes needed.

Lastly, as we will discuss in greater detail in the next chapter, most system's lifespans are time-based, however, for FAWA we will want to convert all lifespans and degradation rates into cycle-based. This is due to the fact that each machine in a manufacturing setting will likely be performing different tasks that have different cycle times, meaning that even though the global time is the same for each machine, the effective usage time will be different, making failure prediction, and therefore, failure control, that much more difficult. For example, a machine may be said to have a mean time between failures of 10,000 hours, but depending on the task that the machine is performing, the actual number of cycles achievable can be vastly different. Therefore, we want to find out the actual number of achievable cycles for a given task in order to get a more meaningful prediction. We will do this by finding out the cycle times for different potential mission plans and then using them to identify their unique degradation per cycle rates based on the proportion of the given time-based lifespans that is used during each cycle.

3.3 FAWA Strategy for Desynchronizing Maintenance

Desynchronized maintenance is only desired when dealing with k-of-n systems. As previously stated, for n-of-n systems the primary goal is to synchronize maintenance to minimize the number of stoppages and be able to repair or replace as many components at the same time as possible. However, for k-of-n systems, the primary goal is to never have more than n-minus-k failures happen at the same time, or within the same time window that it takes to repair or replace a failed component. This is to say that the system can operate effectively as long as k out of the n units are operational, such that if k=3 and n=4, then there are 4 components and at least 3 of them must be operational at all times. These types of systems typically describe parallel systems that are likely performing similar operations, and it is simply a production volume problem where in order to meet a quota at least k machines must be producing at all time. Typically, when all components are performing the same operation, their workloads are naturally balanced. However, in systems where extra components can be introduced or certain components can be overloaded, such as one machine working double-time, FAWA is able to maximize uptime by having one machine rest, one machine work double time, and all the remaining machines work normally, artificially inducing an imbalance in degradation rates that can then be controlled such that overlapping failures are minimized.

In order to optimize the imbalance between degradation rates and induce desynchronized failures, FAWA uses two standard deviation thresholds, rather than just one when synchronizing failures. There is an upper threshold that is treated exactly like when we are synchronizing failures, which is to say that when it is exceeded, meaning the gap between health states for each component is growing, FAWA induces the shrinking of this gap by changing mission plans such that the most degraded component is used the least and the least degraded component is used the most. Additionally, however, there is a second, lower threshold that when it is exceeded, meaning the health states are getting too balanced, FAWA induces the separation of this gap by implementing the mission plan that has the most degraded component do the most and the least degraded do the least. By having an upper and lower threshold FAWA is able to maintain optimal separation between failure times. Similar to when synchronizing maintenance, when dealing with progressive degradation rates, recalibration of the cycles completed will have to be performed based on each component's health state after every adaptive change.

The upper and lower standard deviation thresholds are determined through iterative simulations that test all possible combinations and finds the one that results in the fewest overlapping failures and therefore, produces the most uptime.

CHAPTER 4

FAULT ADAPTIVE WORKLOAD ALLOCATION CASE STUDIES: OPTIMIZED MAINTENANCE SCHEDULING FOR INDUSTRIAL ROBOT ARMS AND 3D PRINTERS

The manufacturing systems of interest for this work are industrial robot arms and 3D printers, and multiple case studies will be explored in this chapter for each. With FAWA, we can answer two important questions that help optimize maintenance schedules: how should a given machine go about completing its designated task assignment, based on its components' current health states, and which machines should perform which tasks. For a reliability-based system, such as an industrial robot arm, the system's health is based on the physical degradation of its components. This degradation is characterized by an increase in probability of failure and typically results in a decline in performance. For example, in addition to an increase in probability of failure, degradation to a joint's motor may result in reduced torque capabilities, which in turn makes degradation per cycle progressively worse. A failure is then defined as when a probability of failure threshold is met. For these systems, each task and associated mission plan will demand different physical requirements from each component, and therefore, for the case studies involving robot arms, FAWA optimizes the degradation of the system by alternating between different joint angle configurations and path trajectories such that a component can do more or less depending on its current health state compared to all of the other components. Alternatively, for a resource-based system, such as the 3D printers, the health is based on how much of a needed resource remains accessible and a failure is defined as when the resource is depleted below an acceptable limit. Therefore, for the 3D printer case studies, FAWA will be used to optimize which printers should print which parts, and will be determined by the printer material available to each printer and the volume of material required for each printed part.

First, FAWA will be used to balance degradation of individual joints on a single robot arm. Subsequently, it will be used to balance degradation across multiple arms where each one is performing a different task. Next, FAWA will be applied to multiple 3D printers that are working together on collaborative printing jobs. Here, we will examine multiple mobile robots working together in a chunk-based printing strategy where each printer is responsible for a different number of chunks. Each of these aforementioned studies will involve n-of-n systems where the goal will be to balance degradation and synchronize maintenance. Then, additional studies will examine a k-of-n 3D printing system where the goal will be to desynchronize maintenance by inducing imbalanced degradation in order to limit the number of overlapping failures. First, we will look at a parallel system of printers, each printing the same part, and will use FAWA and variable printing speeds to desynchronize refills and maintain a constant production. Lastly, we will examine using FAWA with an extra, back-up printer that can take the place of any printer that is out of material in order to maximize uptime.

4.1 Overview of Using FAWA for Industrial Robot Arms

Industrial robot arms were chosen as the first system to demonstrate FAWA's capabilities because, as a result of the multitude of possible operations they can perform, they provide multiple challenges when it comes to accurate failure prediction and therefore, maintenance scheduling. The system was further complicated by introducing degrading performance, such that as the system's health degrades so does its performance, meaning that the degradation per cycle becomes magnified as the system ages. Furthermore, due to the complexity associated with the physics of an industrial robot arm, certain tasks may not have any alternative mission plans, and the ones that do, may have poor distributions of degradation rates, and thus, FAWA may not always be the preferred option. For example, in some cases every possible mission plan may require the shoulder joint of the arm to have the highest workload, and therefore, the plan that balances the workload best should always be used as any alternatives would simply make the shoulder joint perform extra work, resulting in faster degradation. Therefore, in this study multiple examples will be shown to demonstrate the types of situations where FAWA is recommended and those where it is not. Additionally, for comparisons to using FAWA, two alternative maintenance strategies will be examined: repairing a component only when it fails, and repairing all components, regardless of remaining health, whenever any component fails. These more traditional strategies will test the difference between minimizing the number of repairs versus minimizing the number of stoppages, respectively.

When implementing FAWA for a robot arm, the first step is to identify the desired mission task, i.e. the desired initial and final positions that the arm needs to move between. Once these positions are decided, possible mission plans, i.e. available joint angle configurations that result in the robot arm reaching the chosen positions, can be determined through kinematic equations. For the purposes of this research, it was assumed that there were no obstacles and that the arm was free to move in a direct path between any two points. Additionally, it was assumed that there were no requirements on end-effector orientation.

Next, possible paths are sorted by the maximum total distance traveled by each joint, and then sorted by the average distance traveled, with the resulting minimum being chosen as the initial path plan. This is done to minimize the total distance travelled by any individual joint while also limiting the total distance travelled by all the joints combined. Minimizing the total distance travelled by each joint first rather than minimizing average distance first is done because, assuming that each joint is able to rotate simultaneously, the cycle time of the arm is solely dependent on the slowest joint and is therefore dictated by the largest distance.

Subsequently, as previously mentioned, degradation thresholds based on the standard deviation of current health states are implemented, such that when the total standard deviation exceeds the threshold, a new optimal mission plan is calculated. For this work, a standard deviation threshold of 0.5 was used. Despite this low of a threshold requiring numerous adaptive changes over the course of the system's life, for this system, an adaptive change, i.e. alternating between different joint angle configurations, should be able to performed quickly and easily, and therefore any additional time added to the overall cycle should be negligible. For other system's, as we will see later, adaptive changes between mission plans may take extra time and therefore, the selection of their thresholds will need to take into consideration the number of changes. Finally, at each threshold, a new mission plan is chosen based on the rule of minimizing the required rotation of the most degraded joint, followed by the second most degraded, followed by the least degraded.

It must be mentioned, however, that the benefits of FAWA are entirely dependent on the chosen mission task. Some tasks will naturally have access to fairly balanced mission plans, i.e. plans where each joint is exposed to roughly the same amount of stress, and thus, will not see significant benefits from FAWA, while other missions will only have access to unbalanced mission plans, in which case FAWA can provide incredible improvement over traditional methods. This case study will be done entirely using fault simulations, however, future studies will need to investigate using sensor data from actual robot arms to more accurately identify the effects of varying degrees of degradation.

Before being able to deploy fault adaptive strategies, however, degradation rates must be determined. This in and of itself can be quite challenging for complex physical systems. Most lifespans of complex systems, due to the wide variety of possible use cases, are given in time scales, such as hours of use, with side notes stating that times may vary drastically depending on use. Unfortunately, this is not very useful when trying to predict maintenance schedules for a complex manufacturing setting, because keeping track of how much time each machine has been used for, when each machine will likely be performing tasks that each have different cycle times, can be quite cumbersome. Furthermore, because total production numbers are mainly what we care about, time between failures is much less important than produced parts, or cycles, between failures. Therefore, for effective FAWA, we need a method for changing from time-based to cycle-based life-expectancies. For resource-based degradation models where system health is defined by the amount of a given resource that is available, this is relatively easy, as the total amount of a resource required per cycle, such as material needed for 3D printing a part, is fairly straight forward to calculate as the information must be predetermined when designing the part. However, when dealing with health-based systems where system health is defined by the system's reliability, such as a robot arm, this can become quite complex due to the different physics involved with each alternative mission plan, and is further complicated by the introduction of progressive degradation rates. Therefore, we need to perform degradation analysis using physics-based failure models, and thus, fault augmented Modelica simulations are used in order to calculate each joint's total cycle time at each level of degradation. Then, the cycle times associated with each joint's current health state are used to calculate each joint's degradation per cycle, which can then be used to determine their health, i.e. reliability, after each successive cycle of a given mission plan. An example of how these fault augmented simulations are carried out will be expanded on in the following section.

4.1.1 Degradation Analysis Using Fault Augmented Modelica Extensions (FAME)

As previously discussed, for many systems, degrading health does not simply increase the likelihood of failure but may in fact cause degradation in performance as well. Moreover, for physically dependent systems, cycle times cannot be assumed based solely on the components' individual behaviors. For example, a robot arm joint's cycle time and therefore, its degradation, cannot simply be associated with it's required degree of rotation, as it will also be based on the orientation with the physical world at any given time, i.e. a 30^o rotation up-and-down will not necessarily have the same cycle time, or degradation effects, as a 30^o rotation side-to-side. Furthermore, the positioning and movement of every joint in the arm is dependent on the others, such that the performance of the shoulder joint is directly tied to the joint configurations and movement of the elbow and wrist joints, due to the variations in moment arm and inertial forces. As such, the shoulder joint rotating 30^o will not necessarily have the same cycle time as the wrist joint rotating 30^o. Therefore, in order to accurately understand how such a system will behave under various operations and health states, it is crucial to have a physically dependent system model that can capture these interdependencies. As a response, this work examines using fault augmented Modelica extensions (FAME) [16,29] to model our robot arms, similar to the one depicted in Figure 4.1, performing various tasks and mission plans under varying degrees of degradation in a physics-based virtual environment. These models are then used to determine the expected degradation per cycle for each component for all possible mission plans, which will then be used for FAWA.

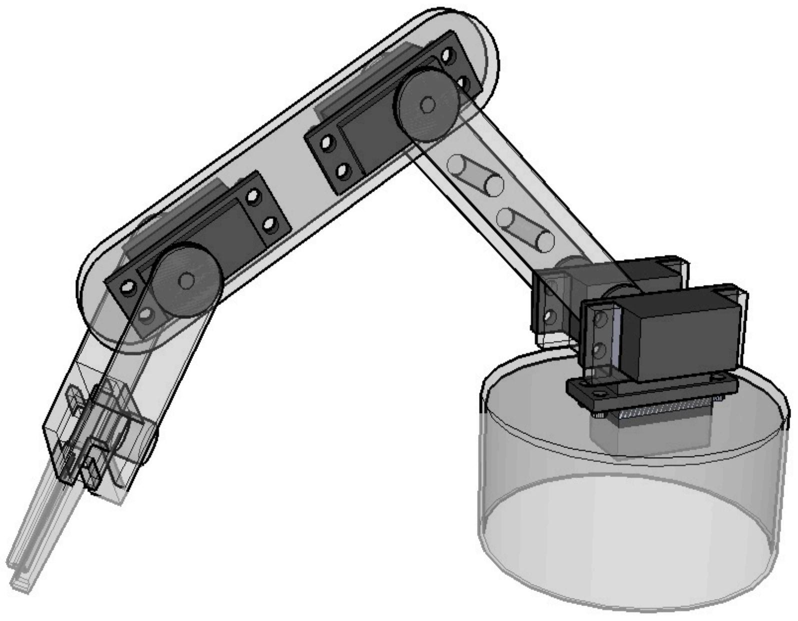


FIGURE 4.1 3-Linkage Robot Arm (Joint 1 is attached to the base, Joint 2 is the middle joint, and Joint3 is the farthest from the base)

For the robot arm system used here, the system-level mission task is to move a load between two positions and the component-level behaviors are the required rotations of the individual joints. The arm was modelled as having 3-linkages, 3-joints, and a rigid wrist, where Joint 1, i.e. the base joint, is capable of rotating from 0^o to $+180^o$, and Joints 2 and 3 are capable of rotating -170^o to $+170^o$, all in the XY-plane. This system was modelled in Dymola, a Modelica software, and was created using a modified version of an existing robot arm model found in the Modelica Standard Library, but with each component being replaced by a fault-augmented component. These models allow for varying degrees of degradation to be applied directly to specific components.

In this case study, degradation to the joints' motors, varying from 0-90% degraded with a 10% step size, were the progressive failure modes examined; 100% degradation was not modelled as it would simply result in a static arm and cycle times would go to infinite. The motors' performance was then modelled with a loss of rotational torque capabilities related to the percentage of degradation. This results in more time being required to both start and stop rotation as degradation worsens. This was simulated in Dymola by inducing degradation at each joint's motor output shaft, which is represented as the interface between an electromotive force (EMF), i.e. the torque source component, and a physical inertia (JMotor), i.e. the output shaft, and is highlighted in the model diagram shown in Figure 4.2.

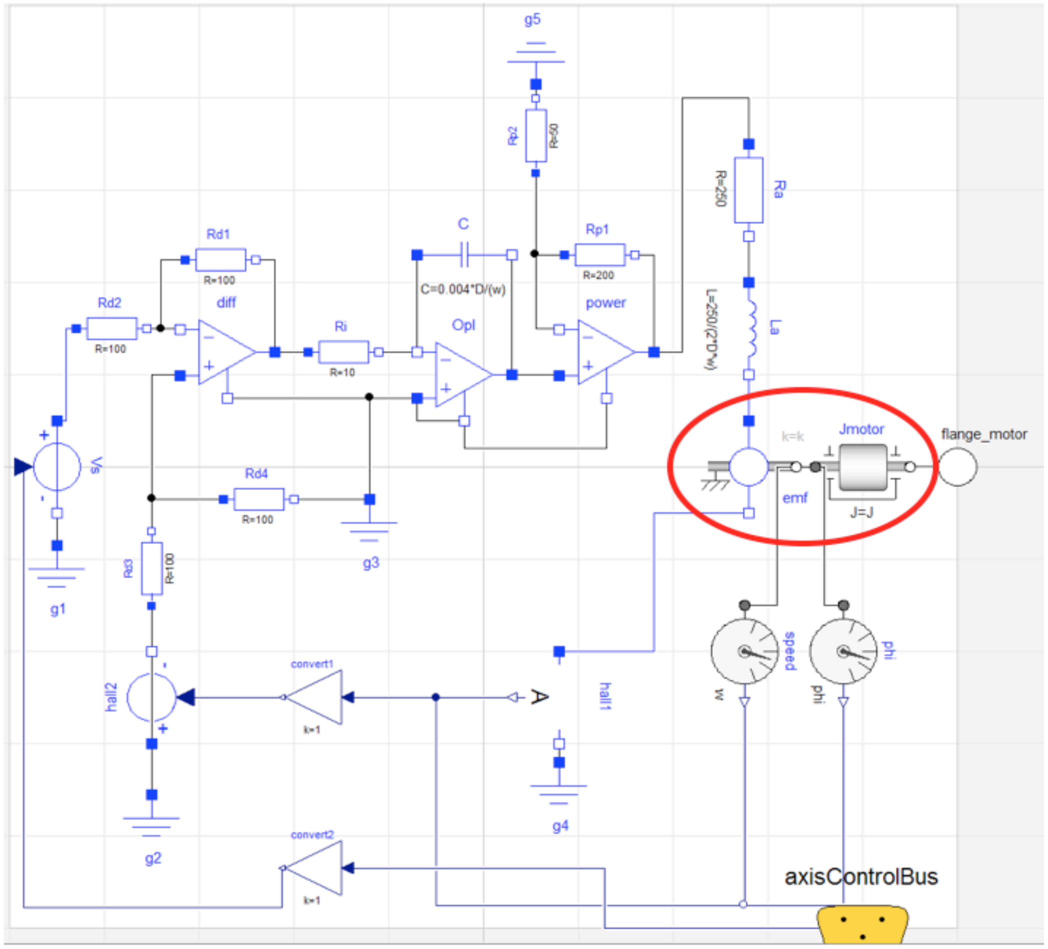


FIGURE 4.2 Dymola Motor Model (Circled area is the EMF-Jmotor connection, which represents the motor's output shaft where the FAME faults are implemented)

Next, mission plans including the starting and finishing angles of each joint, as well as their total angles travelled, are entered into the Dymola FAME model, producing plots of the expected behavior for each joint at each level of degradation, such as the example shown in Figure 4.3. This example is for a single joint attempting to rotate from 0^o to 75^o at 0-90% degradation with a 10% step size; the y-axis is the joint angle in degrees, the x-axis is time in seconds. Each line represents the different path trajectories at each degradation level where the more degraded the component becomes, the less steep the slope and the more pronounced the overshoot, as the joint cannot start or stop as quickly due its diminishing torque capabilities.

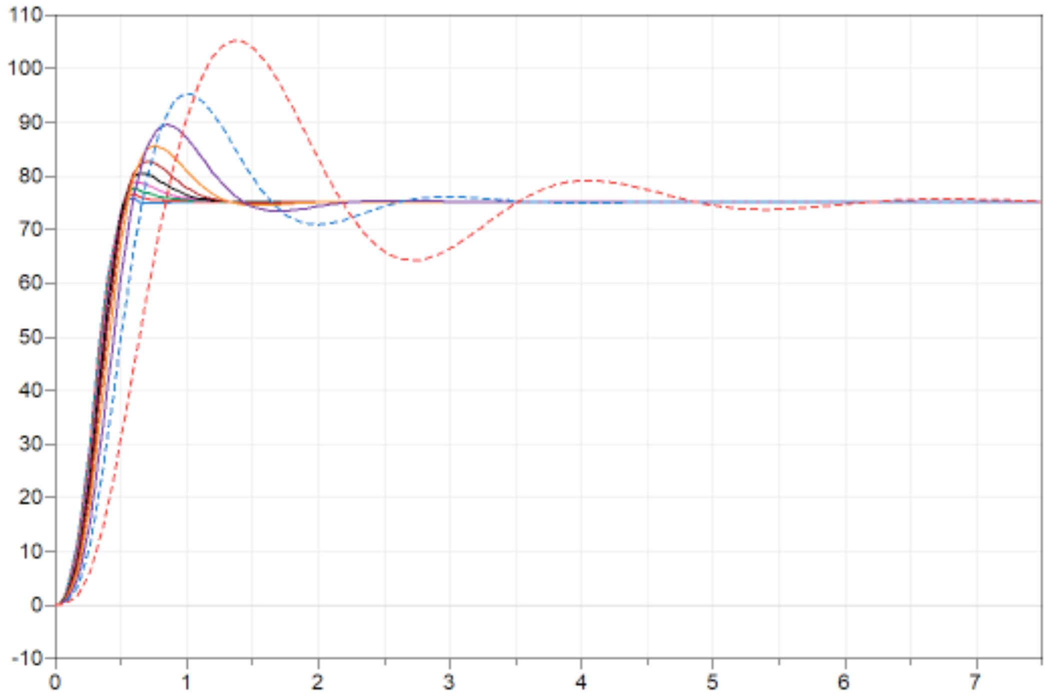


FIGURE 4.3 Example simulations for a joint rotating from 0^o to 75^o for each level of degradation. (Larger overshoot = larger percent degraded)
X-axis: Time (seconds), Y-axis: Angles (degrees)

From these models, total cycles times can be extracted at each 10% simulated step size. Then, through interpolation, expected cycle times can be determined for any health level, as depicted in Figure 4.4.

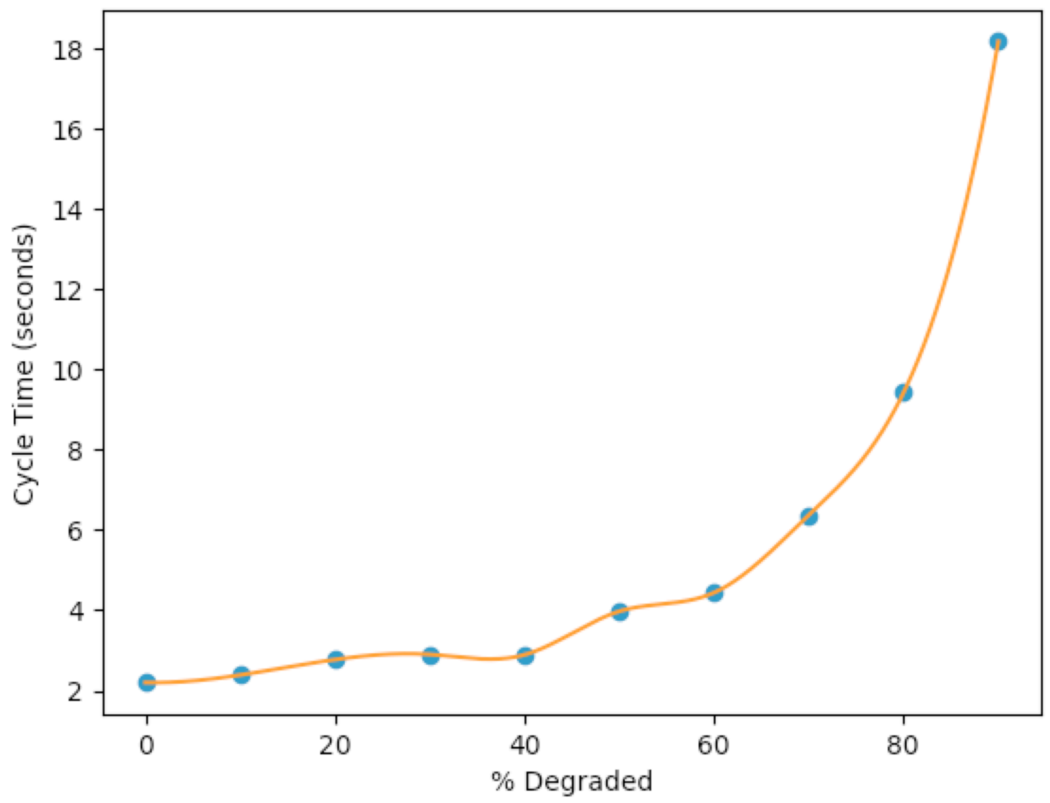


FIGURE 4.4 Example plot of the cycle times at different degrees of degradation for a joint rotating from 0° to 75° for each level of degradation

Lastly, these cycle times are used to predict degradation accumulation per cycle and ultimately, total expected lifespans. A Weibull distribution with a beta value equal to 2 will be used to simulate wear-out failures. Thus, the current health, i.e. the current reliability, for each joint is calculated after every cycle using the following equation:

$$Health_t = Health_{t-1} - \exp\left(-\left(\frac{totalTime}{projectedLifespan}\right)^{\beta}\right) \tag{4.1}$$

As previously mentioned, a complex system can have a wide range for its estimated lifespan and will greatly depend on how the system is used. For industrial robot arms, lifespans are given in "hours of use" and therefore, for the purposes of this case study, the degradation per cycle for each joint will be dependent on a projected lifespan of 80,000 hours [30]. Additionally, despite having the same cycle time, a joint moving 90^{o} at 90^{o} per second will obviously cause more degradation per cycle than moving 5^{o} at 5^{o} per second, therefore, a velocity factor is implemented based on the ratio of a joint's average rotational velocity to its optimal velocity, which for this case study was said to be 30^{o} per second. This corrective multiplier will need more research done on it in order to be truly accurate, however, due to the fact that it will also likely depend on the material composition of the joints, as well as other factors such as acceleration and force. Therefore, the total time since the last failure, which is used above in equation (4.1), is calculated using the following equation:

$$totalTime_t = totalTime_{t-1} + currentCycleTime * velocityFactor \tag{4.2}$$

Equation (4.2) is where we see the primary benefit of using FAME simulations. By using FAME models, we are able to capture how cycle times change for different operations as degradation accumulates, which then allows us to determine each component's life-expectancy in terms of cycles. Then, we can use these cycle-based lifespans to deploy FAWA, and ultimately, help better predict maintenance schedules and costs, as will be shown in the case studies investigated in the following sections. It should be noted, however, that we are not claiming that these degradation models used in the following case studies are perfectly accurate. For accurate predictions of degradation rates and total number of achievable cycles for a given mission task, detailed failure analysis will need to be done on an actual industrial robot arm system, where the impact of factors such as average and peak acceleration, force, temperature, as well as the size of the load and the material and quality of each joint, along with many other potential contributing factors to degradation, are investigated and verified with experimental data. However, for the purposes of this work, this is of little importance as we are not concerned with determining actual degradation models for the different tasks, but rather are concerned with the question of if you have known degradation models for a variety of alternative mission plans, what can you do with them? Do we have to sit back and passively wait for failures to happen, or can we do something to actively control when they happen? Thus, whether the degradation curves are accurate for an actual robot arm is not what we are trying to show, but rather, assuming that they are, can we use that information to artificially induce optimal degradation between all of the joints with our defined FAWA strategies?

4.1.2 Synchronizing Joint Maintenance Using Health-Based Thresholds

This first case study will examine using FAWA for the synchronization of joint maintenance for an industrial robot arm with a health-based threshold of 50%. This threshold means that maintenance takes place any time the system reliability drops below 50%; the real-time system reliability is calculated by taking the product of the individual joint reliabilities due to this being a series system. Three different tasks were analyzed, which included moving back and forth between the following sets of x-y coordinates: [2,1]←→[-1,2], [2,0]←→[0,1], and [1,0]←→[2,1].

First, the most balanced mission plan available for each task was found through the process described in section 4.1. Then, optimal alternative mission plans for each possible rank order of joint health states were identified, i.e. optimal mission plans were found for when Joint 1 is the most degraded, Joint 2 is the next most degraded, and Joint 3 is the least degraded, along with all other possible order combinations. Lastly, lifecycle simulations were carried out using equation (4.1) to identify how many cycles could be completed for each of the three designated tasks prior to a failure occurring. In order to compare the effectiveness of FAWA, we also carried out lifecycle simulations where we did not use FAWA and simply used the most balanced mission plan the entire time.

The plots showing the different individual and system reliability profiles for each of the three tasks can be seen below in Figures 4.5, 4.7, and 4.8. As previously mentioned, however, the precise number of cycles being performed is not our concern, as we cannot guarantee the accuracy until further failure analysis is performed on actual robot arms and verified with experimental data. Instead, the proportional relationships between using FAWA and not using FAWA are what we are interested in.

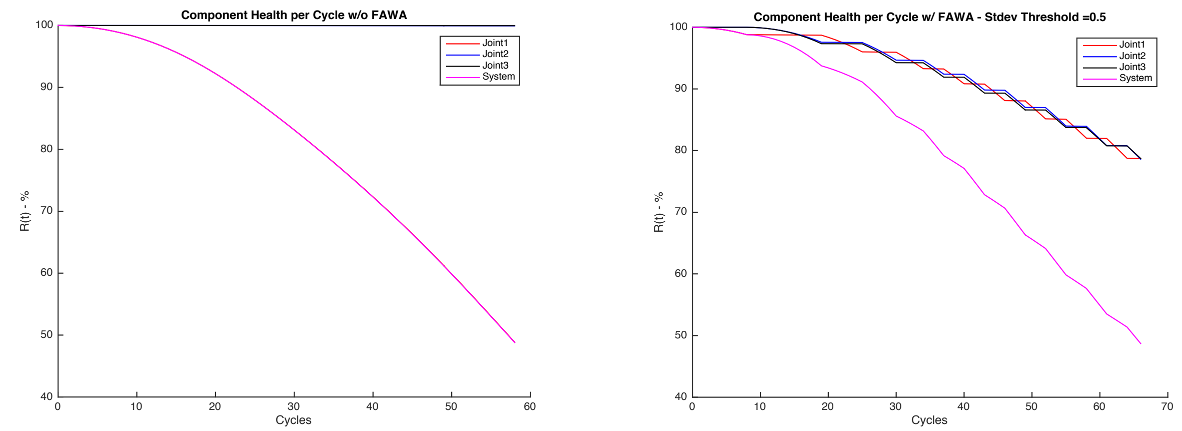


FIGURE 4.5 Robot Arm System Reliability Profiles with and without FAWA
Mission Task: [2,1] → [-1,2]. Cycles Done: 58 without FAWA vs 66 with FAWA
(Left: Reliability profiles for each joint and the total system without FAWA)
(Right: Reliability profiles for each joint and the total system with FAWA)

In Figure 4.5, we see that an additional 13% of cycles (66 vs 58) were able to be achieved before the system reliability threshold of 50% was surpassed, signally the first maintenance stoppage. The most balanced mission plan, and the one which the non-FAWA approach followed, required Joint 1 to rotate 88^o, Joint 2 to rotate 2^o, and Joint3 to rotate 2^o per half cycle. This resulted in Joint 1 being required to do nearly all of the work, resulting in its failure while Joints 2 and 3 remained practically brand new. Contrastingly, in the right portion of Figure 4.5, where FAWA was implemented, we see a clear alternating between different mission plans of different workloads, such that we are able to artificially induce a nearly perfectly balanced degradation profile, as well as generating an increase in cycles before the first failure. These mission plans that were alternated between using FAWA included joint configurations requiring the following total degrees to be rotated per half cycle, with the first value in each set representing Joint 1, the second representing Joint 2, and the third representing Joint 3: [88^o,2^o,2^o], [0^o,89^o,92^o], [0^o,91^o,88^o], and [90^o,0^o,0^o]. Examples of the first and second of these joint angle configurations, which are nearly identical to the third and fourth, are depicted in Figure 4.6. These alternate mission plans represent an ideal scenario where there is at least one mission plan where each joint does the most and one where each joint does the least. Having such a scenario guarantees that balance can be artificially induced when using a low standard deviation threshold.

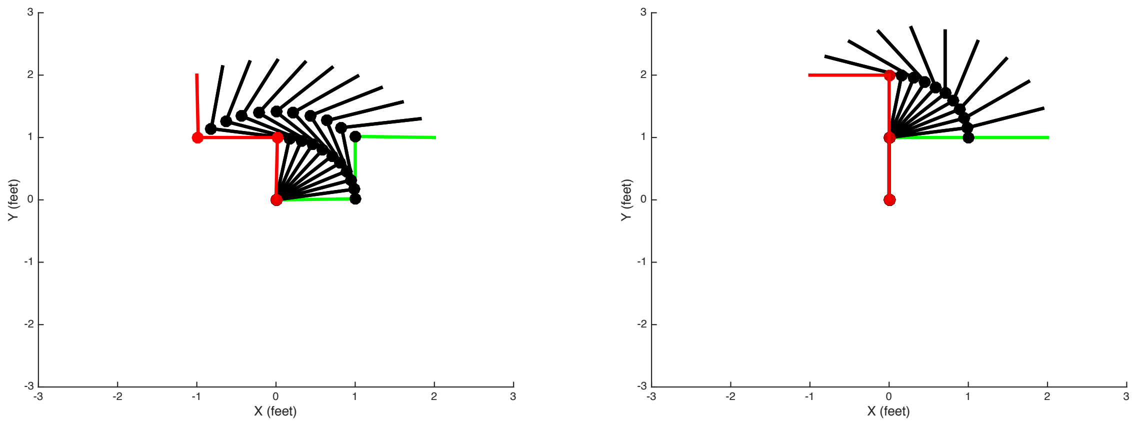


FIGURE 4.6 Examples of alternate joint configurations for moving from [2,1]→[-1,2]. (Green=Starting Position, Red=Final Position, Black=Intermediate Positions)

Unlike in Figure 4.5, Figures 4.7 and 4.8 are examples where the tasks do not have the necessary alternate mission plans in order to allow FAWA to generate much benefit. In these examples, shown below, the mission plans implemented included the following joint configurations: [75°,14°,-74°], [29°,122°,0°], [30°,0°,240°], [149°,-118°,0°], [0°,30°,210°], [89°,0°,-60°], and [0°,151°,-29°] for Figure 4.7, and [-43°,44°,45°], [0°,0°,90°], [0°,1°,88°], [1°,0°,89°], and [-88°,178°,0°] for Figure 4.8.

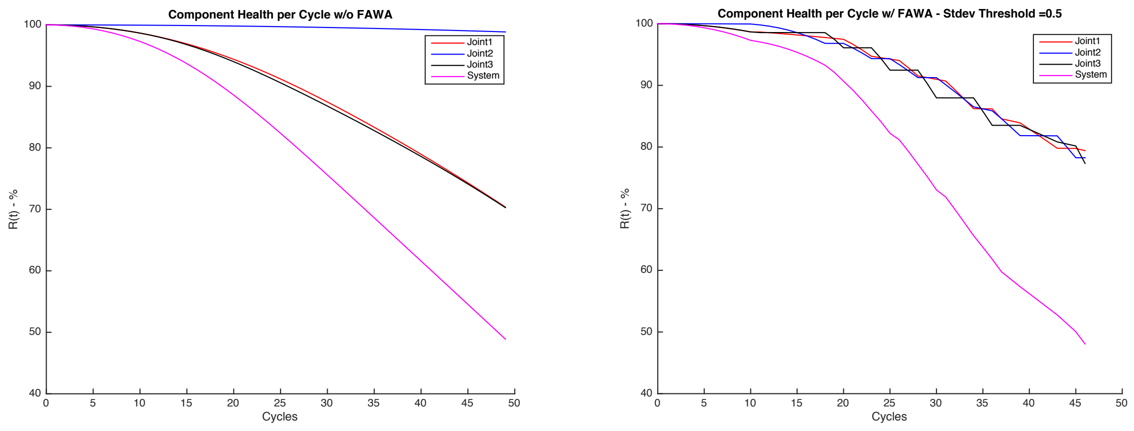


FIGURE 4.7 Robot Arm System Reliability Profiles with and without FAWA
Mission Task: [2,0] → [0,1]. Cycles Done: 49 without FAWA vs 46 with FAWA
(Left: Reliability profiles for each joint and the total system without FAWA)
(Right: Reliability profiles for each joint and the total system with FAWA)

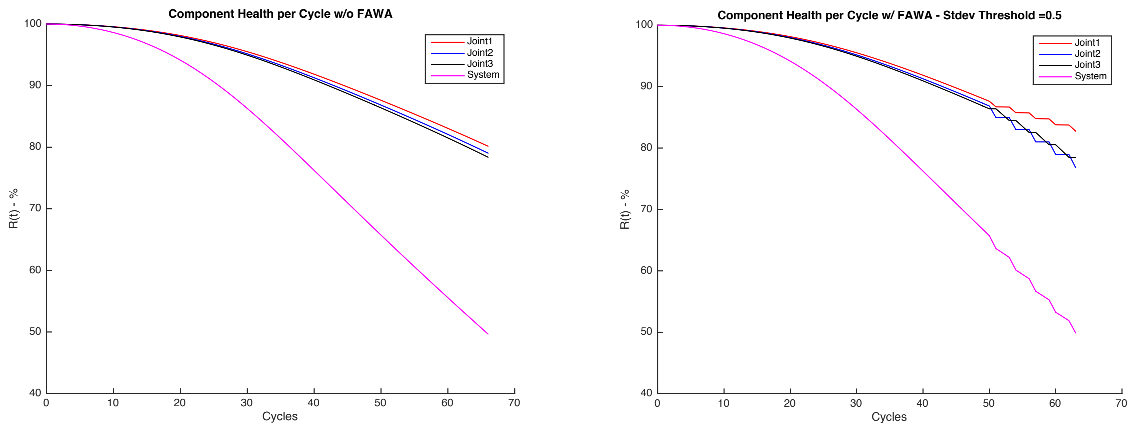


FIGURE 4.8 Robot Arm System Reliability Profiles with and without FAWA
Mission Task: [1,0] → [2,1]. Cycles Done: 66 without FAWA vs 63 with FAWA
(Left: Reliability profiles for each joint and the total system without FAWA)
(Right: Reliability profiles for each joint and the total system with FAWA)

Unfortunately, despite being able to create balanced final health states, the alternative mission plans for Figure 4.7 required excessive workloads and thus, caused increased degradation rates, resulting in fewer total cycles achievable. As previously mentioned, this wide disparity in degradation rates among alternatives is unavoidable for certain tasks, however, despite the fewer total cycles being achievable, by generating balance we are still able to extract value as it still enables us to implement FAWA on a system-wide scale, as we will explain later. On the other hand, in Figure 4.8, there was not a single available mission plan where Joint 1 had the highest workload, and thus, balance could not be generated using FAWA, as seen in the right portion of the figure where alternating between mission plans drove the degradation profiles further apart. Therefore, for this task, FAWA would not be implemented and the original mission plan requiring half cycle rotations of [-43^o,44^o,45^o] for the three joints, repectively, would be used for the entirety of the arms life, as it provided for adequate balance to be achieved naturally.

As was the case in Figure 4.7, sometimes balancing degradation will reduce the time between failures for a single arm, however, we still gain a benefit by using FAWA because by creating a balanced degradation from joint-to-joint, all three joints can effectively be treated as one component with a single degradation rate, allowing for system-wide scalability. Then, with system-wide scalability we can make up any lost value that we may have on the individual arm-level, by being able to synchronize maintenance on the system-level. As we will show in the following sections, by balancing the individual joints of a robot arm, we can treat a whole arm as having one degradation profile, and thus, can balance its degradation with other robot arms that are performing other tasks, similarly to how we just did with three individual joints. Therefore, as long as there are tasks that allow for balanced degradation joint-to-joint, we can continously scale up, theoretically using FAWA to balance degradation of all joints across an unlimited number of arms.

4.1.3 Synchronizing Multi-Robot Arm Maintenance Using Health-Based Thresholds

Once the degradation of the individual joints of a robot arm are synchronized, as was demonstrated in the previous example, the whole arm can effectively be replaced with a single degradation profile and thus, can then be synchronized with the degradation of any number of other arms in the system. For this example, each of the three tasks examined in the previous section will be treated as sequential tasks on an automotive plant's assembly line. Taking the three best fit lines, found using polynomial regression for the balanced mission plans found in Figures 4.5, 4.7, and 4.8, each of the robot arms can be treated as if they have a single degradation profile, which are shown in Figures 4.9, 4.10, and 4.11. Then, the arms performing each of the three different tasks can be balanced by using FAWA for task re-assignment in the exact same process as when balancing the three different joints of a single arm. Therefore, ultimately, as will be shown below, by using FAWA on both the component-level, i.e. joint-to-joint, and the system-level, i.e. arm-to-arm, the maintenance of all nine joints across the three arms can be synchronized.

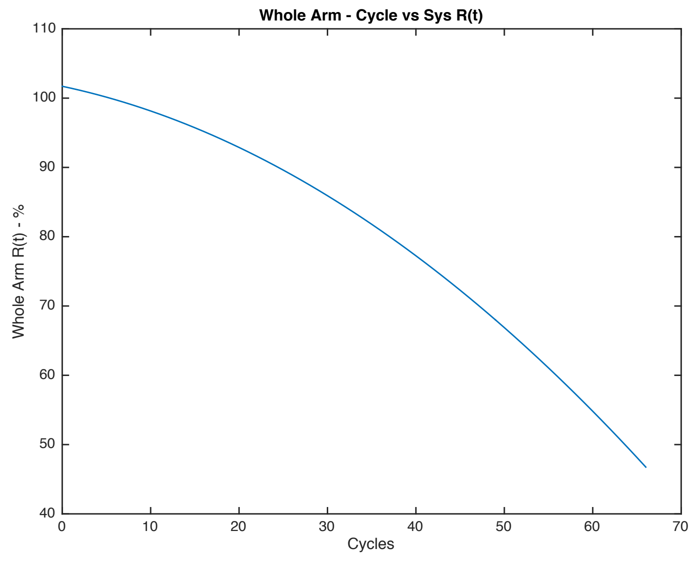


FIGURE 4.9 Balanced Robot Arm System Reliability Profiles
Mission Task: [2,1] → [-1,2]

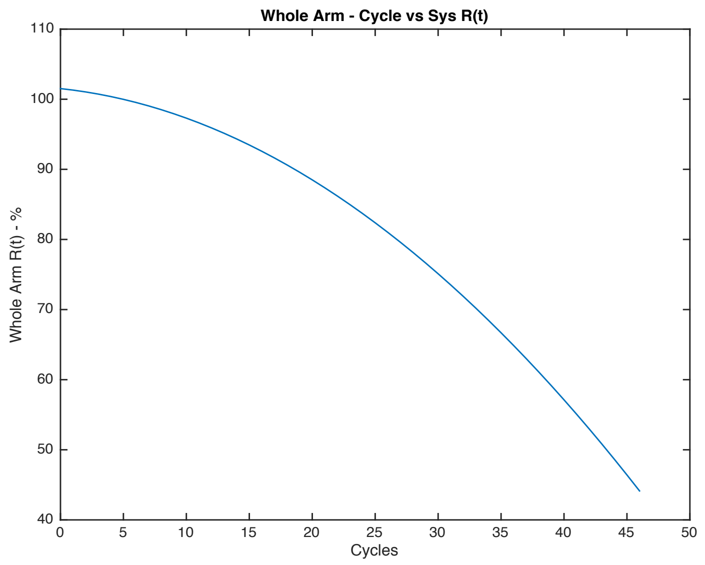


FIGURE 4.10 Balanced Robot Arm System Reliability Profiles
Mission Task: [2,0] → [0,1]

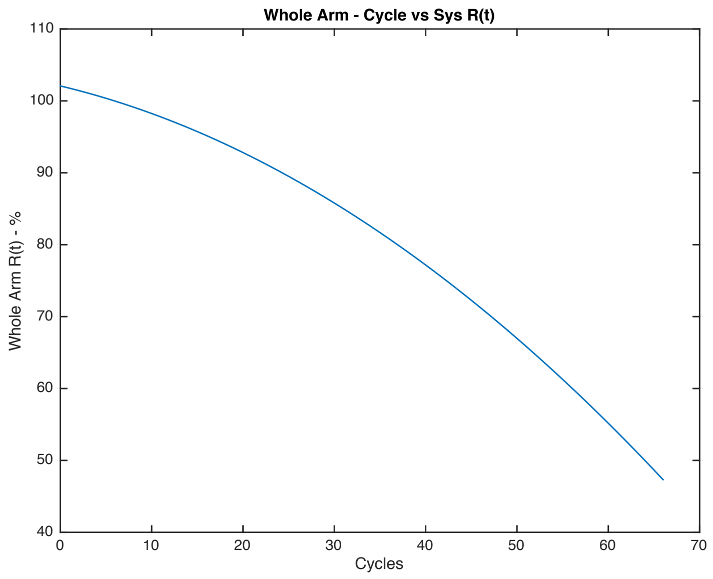


FIGURE 4.11 Balanced Robot Arm System Reliability Profile
Mission Task: [1,0] $\rightarrow$ [2,1]

As mentioned above, this example is meant to represent three different operations taking place in an automotive manufacturing plant. To model costs in this study, multiple factors needed to be determine. These factors included the number of cars needing to be produced per day, the cost to repair a machine, the cost of the car being produced, and the required downtime associated with a repair stoppage. For the number of cars being produced per day, the production values of the top 15 American car plants were averaged, giving a value of roughly 780 cars per day [31]. Next, according to one report, the average annual repair cost of an industrial robot arm is roughly $10,000 [32]. Therefore, after analyzing initial simulation results, shown below in Table 4.1, it was found that using traditional methods, i.e. not using FAWA, for this example there were an estimated 6 repairs every 100k cars, which at 780 cars per day comes out to roughly 17 repairs a year across all three arms, or roughly 5.7 repairs per arm per year. Therefore, we said that an estimated total cost per repair would be roughly $1,750. Next, according to Kelley Blue Book, the average new car cost is roughly $37,000 [33]. As such, at

780 cars per day, valued at $37,000 each, the average cost of downtime for this cases study comes out to be roughly $20,000 per minute, which is relatively close to the documented average downtime cost of $22,000 per minute for the auto industry [34]. Next, for this example, the total repair time was said to be 35 minutes, which includes 30 minutes for repair and 5 additional minutes for any kind of additional stoppage time, such as the time it takes a repair technician to make his way to the machine. Next, the change time required for any FAWA adaptive mission changes, i.e. task re-assignments, was said to be 20 seconds. This value is entirely dependent on the layout of the assembly line and the types of tasks being performed. Obviously, this change time may be much larger in practice if there are obstacles between machines, or if end effectors must be switched out for different tooling tips, or any number of other issues that may require additional time. However, based on the possible savings achievable by using FAWA, as will be shown below, this work hopes to at least start the conversation of potentially changing how manufacturing plant layouts are designed in order to allow for more adaptability, such that machines could theoretically be positioned on tracks that could re-position them quickly between different task assignments. Lastly, the system reliability threshold used for this example was once again set at 50%. With these factors all inputted, simulations were ran for the three different tasks from earlier, depicted in Figures 4.9, 4.10, and 4.11, for three different maintenance strategies: using FAWA, not using FAWA and only repairing an arm when it fails, and not using FAWA and repairing all arms together any time a single one fails. Plots depicting degradation profiles and maintenance schedules for each strategy can be seen below in Figures 4.12, 4.13, and 4.14, respectively.

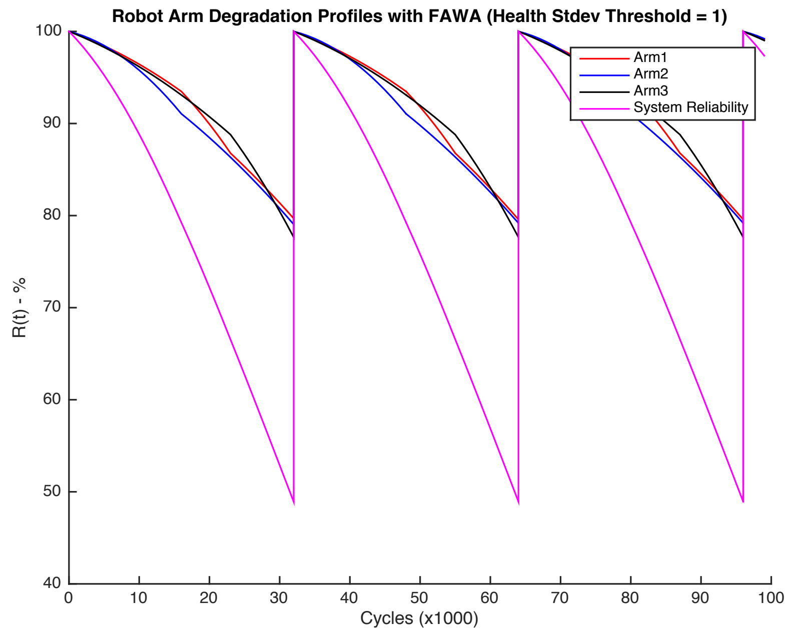


FIGURE 4.12 Using FAWA to balance joint degradation across 3 different robot arms performing 3 different tasks. In order to minimize the number of adaptive changes, a health standard deviation threshold of 1 was used, requiring 2 task re-assignments. Maintenance takes place on all components when the system reliability threshold of 50% is exceeded. (Cycles vs Reliability)

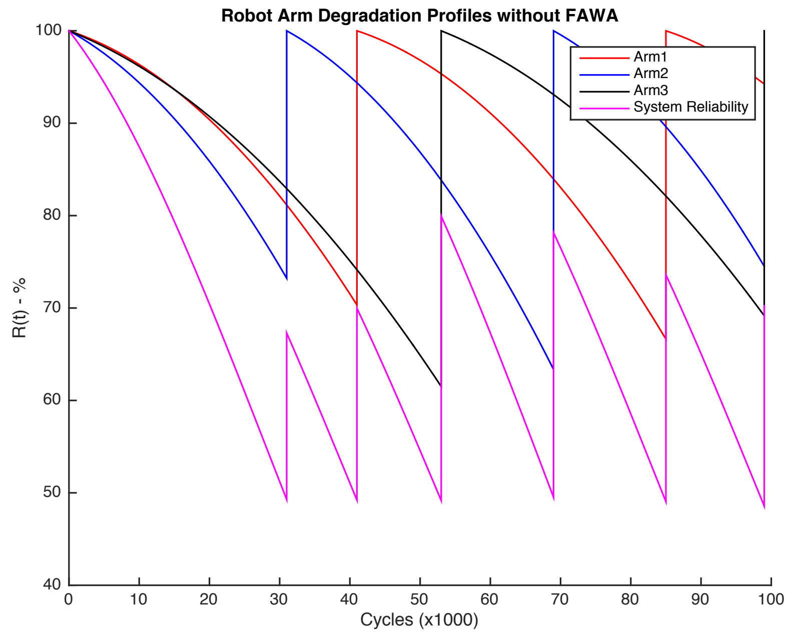


FIGURE 4.13 Not using FAWA. Maintenance takes place only on the most degraded component when the system reliability threshold of 50% is exceeded. (Cycles vs Reliability)

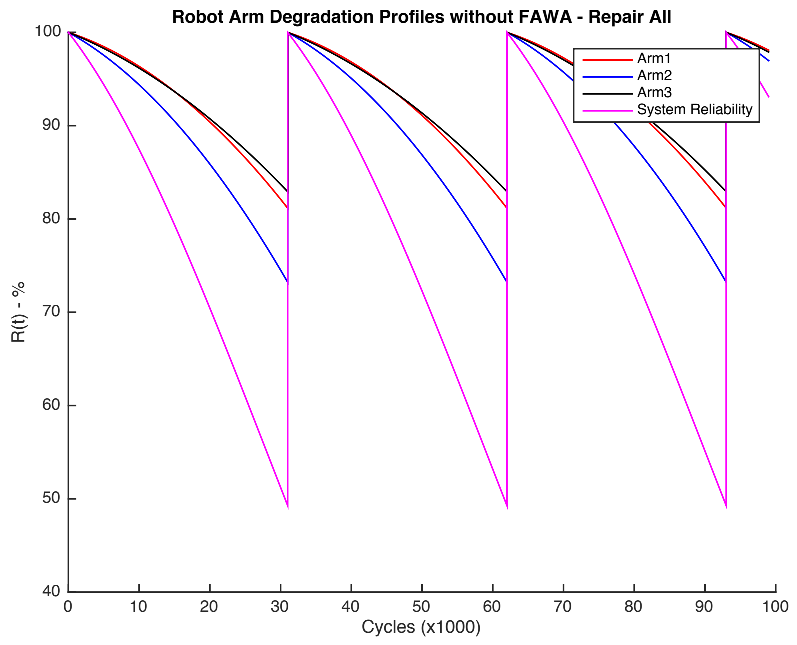


FIGURE 4.14 Not using FAWA. Maintenance takes place on all components when the system reliability threshold of 50% is exceeded.
(Cycles vs Reliability)

From these maintenance simulations, a number of statistics can be examined for each maintenance strategy including the number of cycles before the first stoppage, the number of repairs required, the number of stoppages required, the average cycles between stoppages, the estimated value lost per year, and the estimated downtime per year. However, since the simulation was only ran for 100k cycles, the results per year were based on the fact that 780 cycles are completed per day. Therefore, the simulation results were simply multiplied by 2.847 (780*365/100000) to obtain the results per year. The cost associated with each maintenance strategy, i.e. the value lost per year, was calculated using equation (4.3). This value essentially represents the cost due to actual repairs plus the lost revenue as a result of downtime.

$$valueLost = \left(Stoppages{*}\left(stoppageCost{*}(additionalStoppageTime + changeTime{*}changesPerStoppage + repairProcess{*}repairTime)\right) + Repairs{*}repairCost\right){*}\left(\frac{dailyOutput{*}365}{100000}\right) \tag{4.3}$$

Similarly, the estimated downtime per year was calculated using equation (4.4):

$$Downtime = Stoppages{*}(additionalStoppageTime + changeTime{*}changesPerStoppage + repairProcess{*}repairTime){*}\left(\frac{dailyOutput{*}365}{100000}\right) \tag{4.4}$$

In equations (4.3) and (4.4), Stoppages refer the number of stoppages per 100k cycles; stoppageCost refers to the amount of revenue lost per minute and is calculated using equation (4.5); additionalStoppageTime refers to the additional minutes allotted to account for the maintenance technician to arrive at the failed component; changeTime and changesPerStoppage are only applicable for FAWA and are 0 otherwise; repairTime is the time to repair a single unit, while repairProcess depends on the number of available maintenance technicians and refers to whether it is a simultaneous repair, in which case it is equal to 1, or sequential repair, in which case it is equal to the ratio of Repairs to Stoppages; Repairs is the total number of repairs per 100k cycles and repairCost is how much a single unit costs to repair; and lastly, dailyOutput refers to how many cycles need to be completed per day.

$$stoppageCost = \frac{productValue{*}dailyOutput}{24 * 60} \tag{4.5}$$

The resulting maintenance statistics of each strategy depicted in Figures 4.12, 4.13, and 4.14, can be seen below in Table 4.1.

TABLE 4.1 Maintenance statistics for 3 robot arms with simultaneous repairs. Examining effects of using FAWA, not using FAWA and repairing a component only after it fails, and not using FAWA and repairing all components any time a single component fails.

	FAWA	No FAWA	No FAWA (Repair All)
Cycles Before First Stoppage (x1000)	32	31	31
Repairs per 100k Cycles	9.36	6.75	9.69
Stoppages per 100k Cycles	3.12	6.00	3.23
Avg. Cycles Between Stoppages (x1000)	32	16.66	31
Est. Value Lost per Year	$6,405,179	$12,019,884	$6,490,318
Est. Value Saved per Year w/ FAWA	---------	$5,614,705	$85,139
% Value Saved per Year w/ FAWA	---------	46.7%	1.3%
Est. Downtime per Year	317.3 min	598.1 min	321.4 min
Est. Downtime Saved per Year w/ FAWA	---------	280.8 min	4.1 min
% Downtime Saved per Year w/ FAWA	---------	46.9%	1.3%

Table 4.1 shows that using FAWA saves roughly 46.7% of the costs associated with not using FAWA. However, when not using FAWA and repairing all components together, there is much less improvement; relatively speaking, of course, as despite only being a savings of 1.3% for this example, $85,139 would still be a significant amount. Furthermore, in regards to the time needed to implement an adaptive change, i.e. task re-assignment, discussed in the previous paragraph, based on the estimated savings of 4.1 minutes of downtime, given that there are 3.12 stoppages every 100k cycles (8.9 stoppages per year), with 2 changes per stop (17.8 changes per year), an additional 13 seconds could be afforded to implementing an adaptive change and still receive some benefit. Therefore, as long as a task re-assignment could be completed in under 33

seconds, FAWA would be the best strategy for this example system. In the event that this time requirement could not be met, then the best strategy would be to perform repairs on all components any time a single one fails, as it would result in an estimated $5,529,566 saved compared to repairing a component only when it fails. Furthermore, the reason why the "% Value Saved" and "% Downtime Saved" are roughly equal is because of the disparity in product value and repair costs for this example. With downtime costing $20,000 per minute and stoppages causing 35 minutes of downtime, the average stoppage costs $700,000 worth of lost production value, whereas the cost of actually repairing an arm is only $1,750. As such, if the "% Downtime Saved" value is close to the "% Value Saved" this implies that the product value and daily output are driving maintenance costs, whereas if "% Downtime Saved" is significantly larger than the "% Value Saved" that would imply that the cost of repairing a machine is the primary driver behind maintenance costs and by increasing either the product value or the daily output, you could significantly increase the "% Value Saved". Ultimately, the "% Downtime Saved" value serves as an upper limit of potential "% Value Saved" for the given set of degradation rates. As an additional side note that should be mentioned, the reason why there are slightly more repairs than stoppages when not using FAWA, despite only repairing components when they fail individually, is because there are always going to be a certain number of overlapping failures at some point, which will naturally result in partial synchronization, and thus, these intermittent overlaps must be incorporated into our estimations.

Unfortunately, while FAWA was able to provide significant benefits in the scenario above, it is because it was assumed that all repairs can happen simultaneously, which would require that there are at least an equal number of maintenance technicians to the number of failed components in the system. Under these circumstances, the driving factor of cost is the number of stoppages, which is minimized by using FAWA. However, if there was only one maintenance technician, such that repairs had to be performed one at a time, then the driving factor of costs becomes the number of repairs, which is minimized by not using FAWA and only repairing a component when it is the cause of the system reliability threshold being exceeded. This scenario is depicted in Table 4.2 below, where not using FAWA and only repairing a component when it fails results in the far superior strategy over the other options. Further investigation will need to be done to find the appropriate number of maintenance technicians that should be employed, based on the average salary of a technician and the value saved by having an adequate number of them. Furthermore, by having the more structured maintenance schedule with fewer stoppages, maintenance technicians would not be needed as much or as often and therefore, by not having to have them on-call at all times to be able to take care of the sporadic failures caused by not using FAWA, costs could be reduced by only having a full staff of technicians on days when synchronized maintenance is expected.

TABLE 4.2 Maintenance statistics for 3 robot arms with sequential repairs. Examining effects of using FAWA, not using FAWA and repairing a component only after it fails, and not using FAWA and repairing all components any time a single component fails.

	FAWA	No FAWA	No FAWA (Repair All)
Cycles Before First Stoppage (x1000)	32	31	31
Repairs per 100k Cycles	9.36	6.75	9.69
Stoppages per 100k Cycles	3.12	6.00	3.23
Avg. Cycles Between Stoppages (x1000)	32	16.66	31
Est. Value Lost per Year	$17,103,671	$13,293,514	$17,533,923
Est. Value Saved per Year w/ FAWA	---------	-$3,810,157	$430,252
% Value Saved per Year w/ FAWA	---------	-28.7%	2.5%
Est. Downtime per Year	851.1 min	661.6 min	872.5 min
Est. Downtime Saved per Year w/ FAWA	---------	-189.5 min	21.4 min
% Downtime Saved per Year w/ FAWA	---------	-28.7%	2.5%

These different cases have highlighted how the benefits of FAWA, as well as the benefits of performing repairs individually or all at once when not using FAWA, can drastically change based on factors such as the number of maintenance technicians, the product value, the daily output, or the amount of time it takes to perform a task re-assignment. Therefore, while FAWA may not always be the best choice, this study has shown that what starts as the best maintenance strategy can easily become suboptimal with massive cost effects if costs or production output change or even something as basic as being low on maintenance workers due to an unforeseen sick day. Therefore, whether it is deciding whether to use FAWA or simply deciding whether to repair all components at the same time, factors such as repair costs, stoppage costs, task re-assignment time, and the size of the maintenance crew must all be considered.

4.1.4 Synchronizing Multi-Robot Arm Maintenance Using Resource-Based Thresholds

For this next case study, we will be analyzing a different type of scenario where rather than using system reliability thresholds to guide maintenance, individual thresholds for each arm are used. Typically, system reliability would always be used when dealing with physical failures, however, let us examine a resource-based scenario where rather than basing health on the probability of a physical failure, we base each unit's health on a discrete level of some remaining resource, such as the remaining charge for a battery-powered robot. For example, imagine a set of robot arms working in tandem somewhere where they were required to operate off of battery power, such as on Mars. Then, rather than maintenance involving 30 minutes to repair the arm, we say it requires 30 minutes to recharge the battery. Assuming this type of scenario, maintenance schedules will be based on the individual degradation profiles rather than the system as a whole. We will use the same tasks and degradation models as before, but will assume the degradation rates refer to battery charge dissipation as a result of each mission plan rather than physical joint degradation, and as such the thresholds will be set at 0%, such that the battery can be drained completely before needing to recharge. Otherwise, the same types of simulations as in the previous case studies will be done, and we will also claim that whatever is being built on Mars has the same value as the average vehicle, i.e. $37,000 per unit at 780 units produced per day. However, the $1,750 repair costs will be discarded as we will assume that recharging the battery is free via solar power. Additionally, for this study we will assume that each robot can recharge its battery simultaneously. Lastly, rather than running the simulation for 100k cycles as before, because the thresholds are 0%, rather than 50%, simultations were ran for

300k cycles to allow enough time to plot multiple recharge cycles. The degradation profiles for each of the three maintenance strategies can be seen below in Figures 4.15, 4.16, and 4.17, with the corresponding maintenance statistics found in Table 4.3.

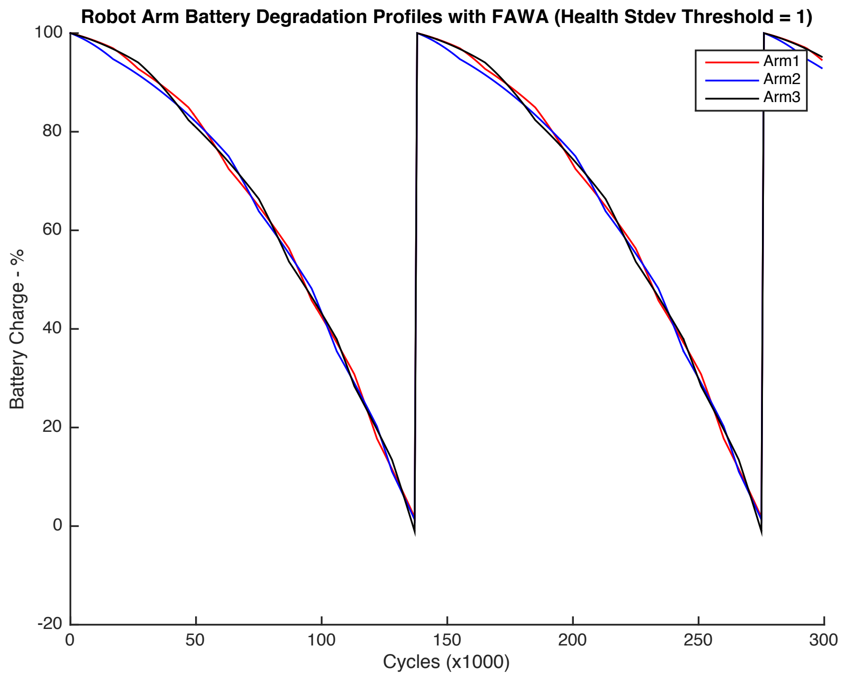


FIGURE 4.15 Using FAWA to balance battery degradation across 3 different robot arms performing 3 different tasks. In order to minimize the number of adaptive changes, a health standard deviation threshold of 1 was used, requiring 12 task re-assignments. All batteries are recharged whenever any battery reaches 0% charge.

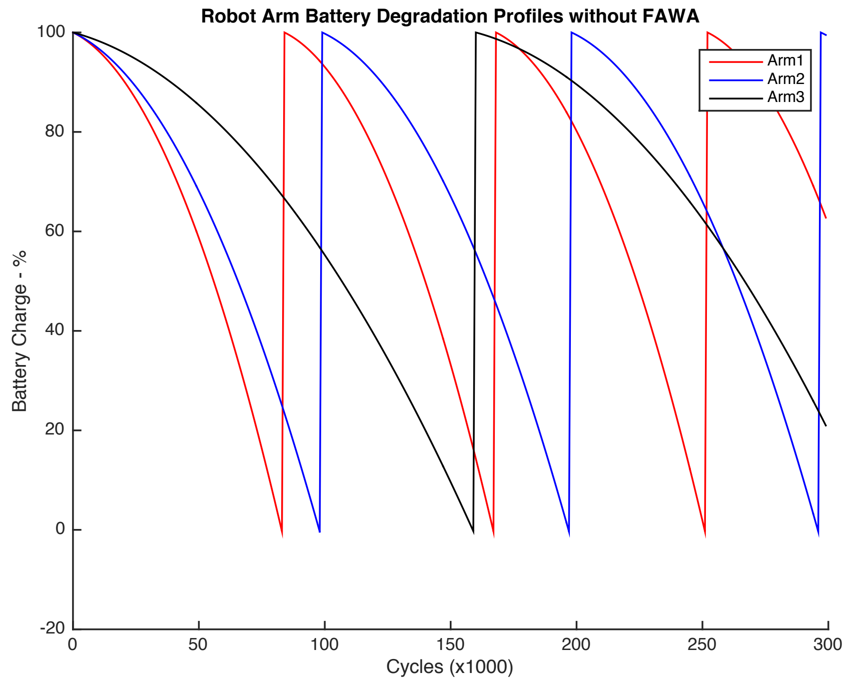


FIGURE 4.16 Not using FAWA. Batteries are recharged only when it reaches 0% charge.

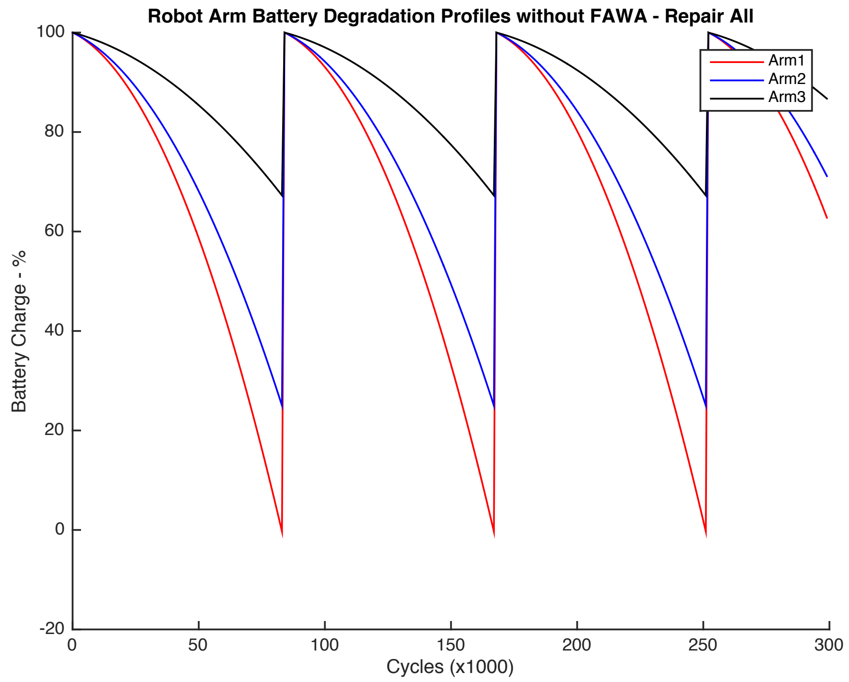


FIGURE 4.17 Not using FAWA. All batteries are recharged whenever any individual battery reaches 0% charge.

TABLE 4.3 Maintenance statistics for 3 robot arms with simultaneous recharges. Examining effects of using FAWA, not using FAWA and recharging a battery only after it fails, and not using FAWA and recharging all batteries any time a single one fails.

	FAWA	No FAWA	No FAWA (Recharge All)
Cycles Before First Stoppage (x1000)	137	83	83
Recharges per 300k Cycles	6.57	8.48	10.84
Stoppages per 300k Cycles	2.19	8.26	3.61
Avg. Cycles Between Stoppages (x1000)	137	36.33	83
Est. Value Lost per Year	$1,414,388	$4,711,659	$2,062,360
Est. Value Saved per Year w/ FAWA	---------	$3,297,271	$647,972
% Value Saved per Year w/ FAWA	---------	70.0%	31.4%
Est. Downtime per Year	70.6 min	235.1 min	102.9 min
Est. Downtime Saved per Year w/ FAWA	---------	164.5 min	32.3 min
% Downtime Saved per Year w/ FAWA	---------	70.0%	31.4%

As shown by the maintenance statistics in Table 4.3, FAWA once again produced tremendous improvement in value saved; 70% improvement compared to when not using FAWA and recharging batteries only when they fail, and over 31% improvement compared to when not using FAWA and recharging all batteries any time a single one fails. Also, based on the 32.3 minutes of downtime saved when task re-assignments take 20 seconds, as long as a re-assignment takes less than roughly 97.5 seconds, FAWA will remain the best strategy. This increase in improvement compared to the last study is primarily due to the fact that the longer a system is able to run, the more imbalanced the health states would normally become, and ultimately, the larger the disparity between final health states, the more benefit FAWA provides. In the previous examples, when dealing with system reliability thresholds, the individual reliability is not allowed to get very low, due to the fact that the system reliability for series systems is the product of each of the individual reliabilities. This results in frequent maintenance and a lot of unused life of the individual components. This case study helps highlight the fact that the more disparate the individual degradation rates of a system, the more beneficial FAWA becomes.

4.1.5 Conclusions of FAWA for Industrial Robot Arms

These case studies involving industrial robot arms explored many challenges associated with maintenance planning. First, we addressed the challenge of how to capture the effects of progressively degrading performance as a system ages. This is a significant problem when dealing with systems, such as industrial robot arms, where there are a variety of different conceivable mission plans, i.e. joint angle configurations and path trajectories, and each one will be affected differently by system degradation. To handle this problem, FAME models were created in order to show that it is possible to identify how degradation affects the system's performance for any task, by capturing the actual physics of the system and performing different simulations for each mission plan at a multitude of degradation levels. Using FAME, we were able to see that different robot arm mission plans, even if they required the same degree of rotation, would experience different degradation effects based on the overall positioning and orientation of the whole arm. Unfortunately, this means there is no easy way to calculate degradation effects on an unseen mission plan without running such a simulation. However, if failure data was able to be collected for a number of real world mission plans, the FAME models could be made to match these degrading behaviors and then could conceivably be used to predict the expected degradation effects on any potential mission plan without needing to collect new failure datasets for each one.

Next, this case study showed that, while FAWA can provide great benefits for some mission plans, there are still limitations based on the availability of adequate alternative plans. However, when balanced degradation is possible, FAWA allowed for complete scalability, providing degradation control joint-to-joint and arm-to-arm. In this study, we also investigated the impact of two different maintenance strategies in addition to FAWA: repairing components individually at the time of their failure and repairing all components any time a single failure occurs. We showed the impact that several factors can have on the effectiveness of each of these strategies, including whether or not all maintenance needs can be handled simultaneously or if they must be done sequentially, as well as other factors such as product value, required daily production, task re-assignment times when using FAWA, and whether we use system reliability thresholds or independent thresholds for each component. By including such factors, we showed that this is not a simple problem to solve and that the various potential combinations of these decision variables, can result in vast differences in cost and downtime for each strategy. When not using FAWA, if repairs must be done sequentially, repairing components at the time of their individual failures is typically the best strategy, however if repairs can be done simultaneously, all components should usually be repaired during every stoppage. However, this is only the case when dealing in high volume production where downtime incurs higher costs than repairs. This is due to the fact that when repairing components individually, we are minimizing the number of repairs and their associated costs, whereas when we repair all components together, we are minimizing the number of stoppages and their associated costs. Furthermore, for most of the examples we explored, FAWA was found to be the best maintenance strategy, and the scenario where it was not the best was due to the excessive repair time from being forced to perform repairs sequentially, reducing the benefit of minimizing stoppages, paired with incredibly expensive downtime as a result of the high product value and daily production output. Therefore, if the repair time was improved, the daily output or product value was reduced, or more maintenance technicians were available, such that repairs could take place simultaneously, then FAWA would once again become the best option.

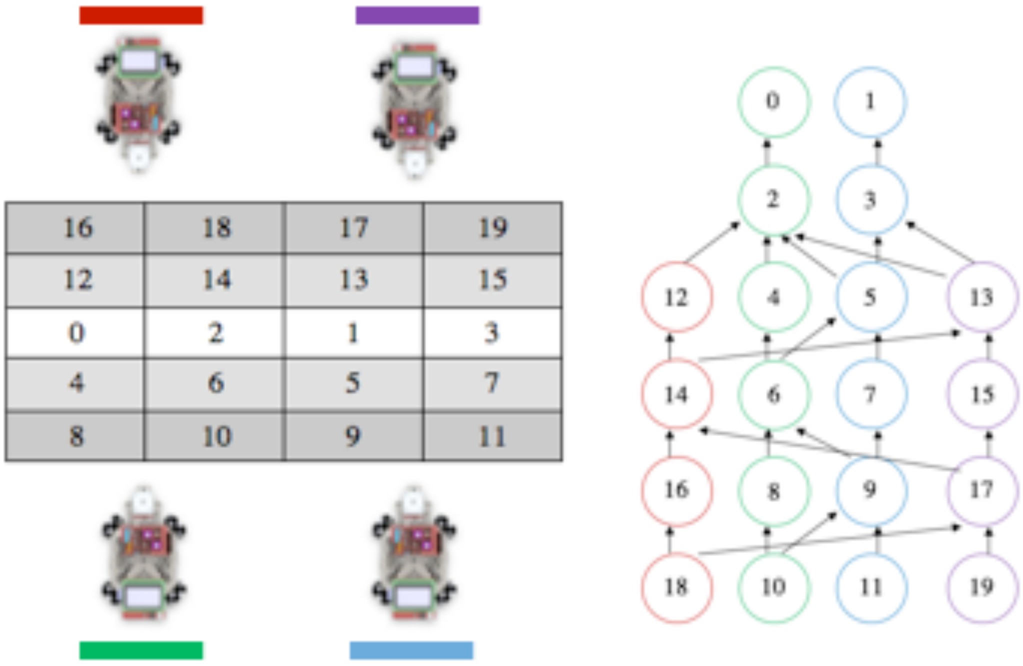
4.2 Overview of Using FAWA for 3D Printers

For the next set of case studies, we will explore using FAWA for several resource-based 3D printing examples, as opposed to the predominantly reliability-based industrial robot arm examples in the previous sections. Similar to the final battery-powered robot arm example, rather than modeling physical behaviors of the system and physical degradations, the system's health will instead be based entirely on how much printer material remains available and thus, how many parts can be printed before a refill is needed.

For this section, we wanted expand to a different type of manufacturing system in order to show that FAWA can be applied to a variety of systems. Once again, as long as there are adaptable tasks that can be carried out many different ways, FAWA can be used to optimize maintenance. Therefore, each of the examples that follow will pertain to multiple printers working in large, collaborative printing schemes. In each example, there will be printers that are either required to print different sized parts or a different number of parts. Therefore, for this study, degradation rates are simply based on how much material is required to print based on the size or quantity of a required print job. As such, FAWA will be used to optimize refill schedules by controlling which printers should print which parts, based on each printer's remaining material levels and the required material demands for each part.

4.2.1 Synchronizing Refills Across Multiple 3D Printers

This first case study will explore using FAWA with mobile robots used for collaborative 3D printing [35,36,37]. The idea behind this type of printing is that a large part can be broken down into smaller chunks that can then be printed in a specific sequence by multiple mobile printers, such that parts of any size can be manufactured and not be relegated the size of a traditional, stationary printer's limited workspace. The first example we will address involves 4 printers collaboratively printing a part comprised of 20 chunks, as depicted in Figure 4.18. The numbers associated with each chunk represent the sequence in which each chunk needs to be printed and the arrows represent the physical dependencies of which chunks must be printed before the others. In the right-hand portion of the figure, each column represents a printer's workload, such that for this example, 2 printers will be printing 4 chunks and 2 printers will be printing 6 chunks. Furthermore, because each chunk must be printed in a specific sequence, if one printer runs out of material, all other printers must stop as well while it is refilled. Therefore, this type of collaborative printing will be treated as an n-of-n system, and so, the goal will be to use FAWA to synchronize the refill schedules of each printer by optimally alternating between which printers print which chunks.



16	18	17	19
12	14	13	15
0	2	1	3
4	6	5	7
8	10	9	11

FIGURE 4.18 Collaborative chunk-based 3D printing using 4 mobile printers. Each value represents the specific order in which that chunk must be printed, each color represents the different robots' assignments, and the arrows represent the dependencies of which chunk must be printed before the other. Figure redrawn based on [38].

For this example, based on an arbitrarily chosen chunk size, we said that each printer spool contains enough material for 720 chunks. Therefore, the two printers printing 4 chunks per part would have enough material for 180 parts, and the two printers printing 6 chunks per part would have enough material for 120 parts. Furthermore, we said that it takes 5 minutes to refill a printer plus 10 additional minutes of stoppage time required for the maintenance technician to walk to the machine along with any necessary boot-up time. Based on material used by Ultimaker printers [39], a refill is said to cost $50, and we are claiming that the value of the parts being printed are $25 and that 100 parts are being printed per day. As such, the degradation profiles i.e. material usage profiles, create by the same three maintenance strategies used in the robot arm examples can be seen in Figures 4.19, 4.20, and 4.21, and their associated maintenance statistics can be found in Table 4.4. Equations (4.3), (4.4) and (4.5) were once again used to calculate the estimated profit and downtime associated with each strategy.

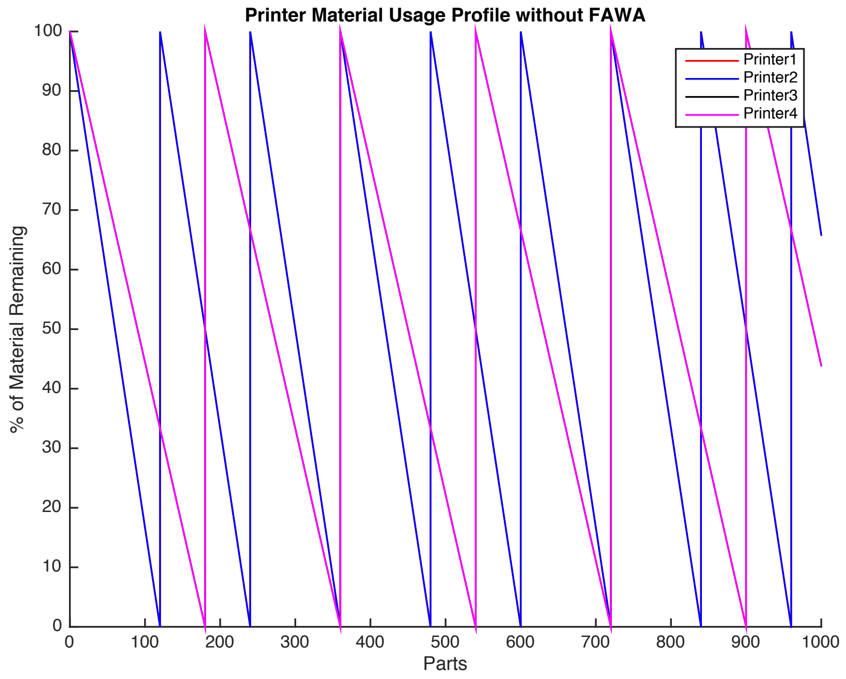


FIGURE 4.19 Usage rates and maintenance schedules when not using FAWA and only refilling a printer when it is empty.

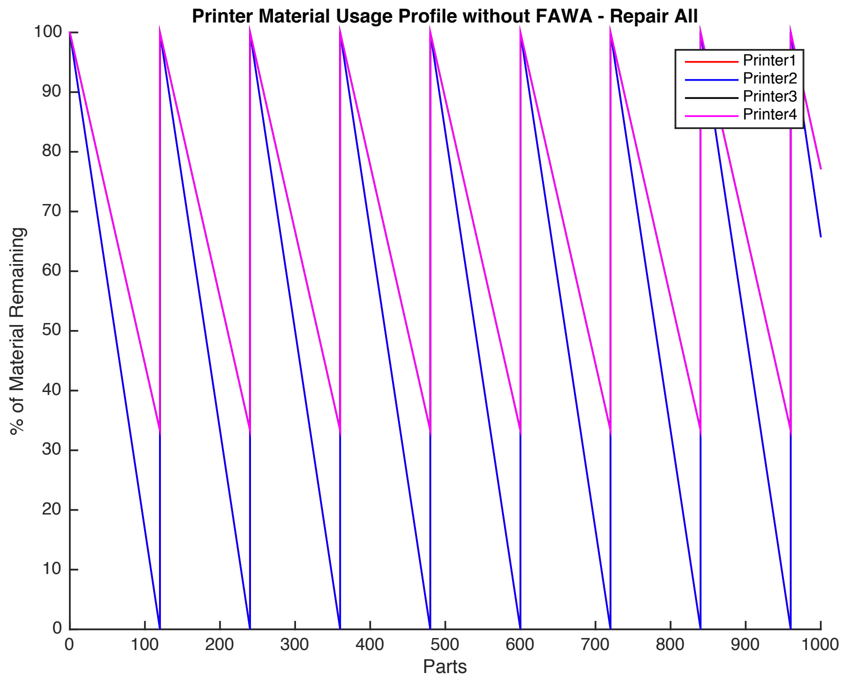


FIGURE 4.20 Usage rates and maintenance schedules when not using FAWA but refilling all printers any time any of them become empty.

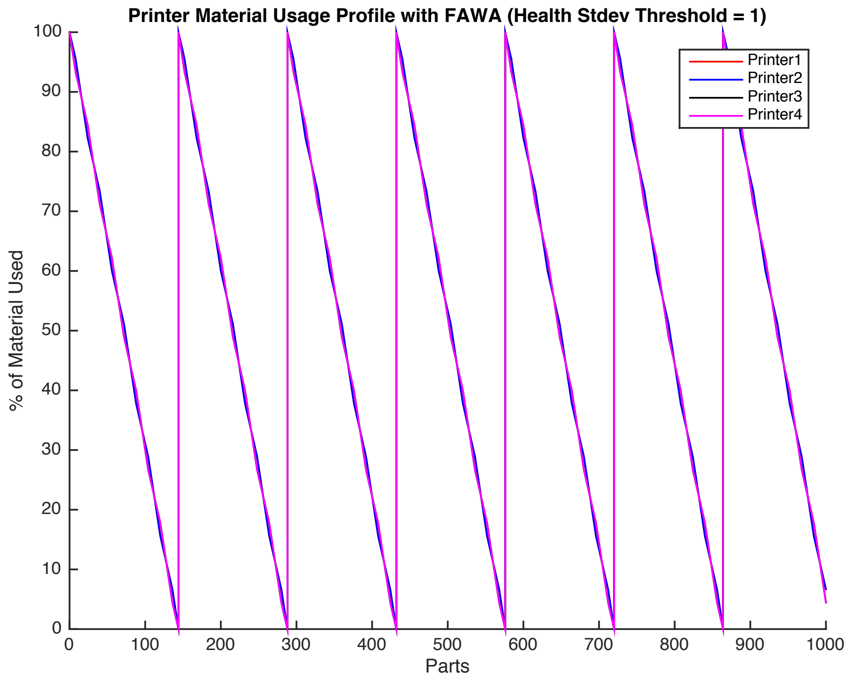


FIGURE 4.21 Usage rates and maintenance schedules when using FAWA.

TABLE 4.4 Maintenance statistics for 4 collaborative 3D printers with simultaneous refills. Examining effects of using FAWA, not using FAWA and refilling a printer only after it runs out, and not using FAWA and refilling all printers any time a single one runs out.

	FAWA	No FAWA	No FAWA (Refill All)
Parts Printed Before First Stoppage	144	120	120
Refills per 1000 Parts	27.8	27.8	33.3
Stoppages per 1000 Parts	6.9	11.1	8.3
Avg. Parts Printed Between Stoppages	144	90	120
Est. Profit per Year	\$861,345	\$861,072	\$851,114
Est. Value Lost per Year	\$51,155	\$51,429	\$61,386
Est. Value Saved per Year w/ FAWA	---------	\$274	\$10,231
% Value Saved per Year w/ FAWA	---------	0.5%	16.7%
Est. Downtime per Year	264.6 min	423.4 min	317.6 min
Est. Downtime Saved per Year w/ FAWA	---------	158.8 min	53.0 min
% Downtime Saved per Year w/ FAWA	---------	37.5%	16.7%

As seen in the Figure 4.21 and Table 4.4, FAWA clearly the best strategy for this example as it is able to effectively synchronize all refills, minimizing the number of stoppages, maximizing the number of parts printed between stoppages, and providing savings in both value and downtime. While the “% Value Saved” is incredibly small compared to the overall profit, this is due to the long print time, fast refill time, and low product value, which make the cost of downtime fairly insignificant. However, in the future, when 3D printers become more advanced and are able to print higher valued products at higher speeds, the cost of downtime will increase and FAWA will be able to produce much more significant benefits. This is shown by the large

"% Downtime Saved" value. It just so happens that for this example, downtime is incredibly cheap, but, as just mentioned, if downtime became expensive due to increased product value, increased production capabilities, or elongated refill times, then FAWA would be able to provide up to 37.5% savings for these degradation rates.

4.2.2 Desynchronizing Refills using Variable Print Speeds to Maximize Uptime

The next case study we examined was for a set of five 3D printers working in parallel, each printing the same type of part, with the goal of maintaining a constant rate of production. However, because each printer is printing the same part, their degradation rates will be the same as well, and therefore, each printer will naturally run out of material at the same time, stopping production. Consequently, we can treat this as a k-of-n system, where we do not necessarily care how many printers are working, all that matters is that there are enough printing at all times to maintain a minimum production level. Therefore, we will investigate having different printers print at different speeds in order to artificially induce different degradation rates. Then, we will implement FAWA's desynchronization strategy in order to generate optimally spaced failure times and minimize overlapping failures, such that we can maximize uptime and maintain a minimum production level.

We assumed that a printer can print at variable speeds, however to maintain print quality and prevent too fast of printing, we said that the possible print speeds are only 0x, 1x, and 2x. For this example, we assumed a minimum production of 120 total parts per day was required, meaning each printer needs to print an average of 24 parts a day. Furthermore, each printer is said to have enough material for 24 parts, and therefore, a refill would generally be required every 24 hours. As such, when operating at 1x speed, each printer would use roughly 4.2% of its material capacity per hour. Next, similar to the previous case study, the cost of a refill was said to be $50, and the refill time was again said to require 15 total minutes. This time, however, each printed part was valued at $100.

First, the expected production if FAWA was not used and each printer printed their respective 24 parts per day with synchronized refills was simulated, as seen in Figure 4.22. Next, in order to utilize FAWA's desynchronization strategy, we created an imbalance in the system's material usage rates by taking advantage of the variable speed printing; one printer printed at 2x, one printed at 0x, and the other three printed at 1x speed. As such, when working at 1x speed, each printer used roughly 4.2% of its material capacity per hour, whereas working at 2x speed it used roughly 8.4% of its material per hour. Then, by monitoring the standard deviation between each printer's remaining material and instituting the strategy described in section 3.3, we were able to force desynchronized failures with the 2x speed and then swap out the printer that had been at 0x to take over as the empty printer was refilled, which then generated the new optimally spaced maintenance schedule, depicted in Figure 4.23. The improved production statistics achieved by implementing FAWA can be found in Table 4.5. The total profit per year was calculated using equation (4.6).

$$Profit = \%Uptime * dailyMinPrints * 365 * productValue - Refills * refillCost \tag{4.6}$$

"%Uptime" refers to the percentage of time where the minimum number of daily prints are achieved. "dailyMinPrints" refers to the number parts that must be printed each day (120 for this case), and 365 refers to days in the year to get the number of parts printed per year. "Refills" refers to the total number of refills in a year and "productValue" and "refillCost" refer to the cost of individual parts and refills, respectively.

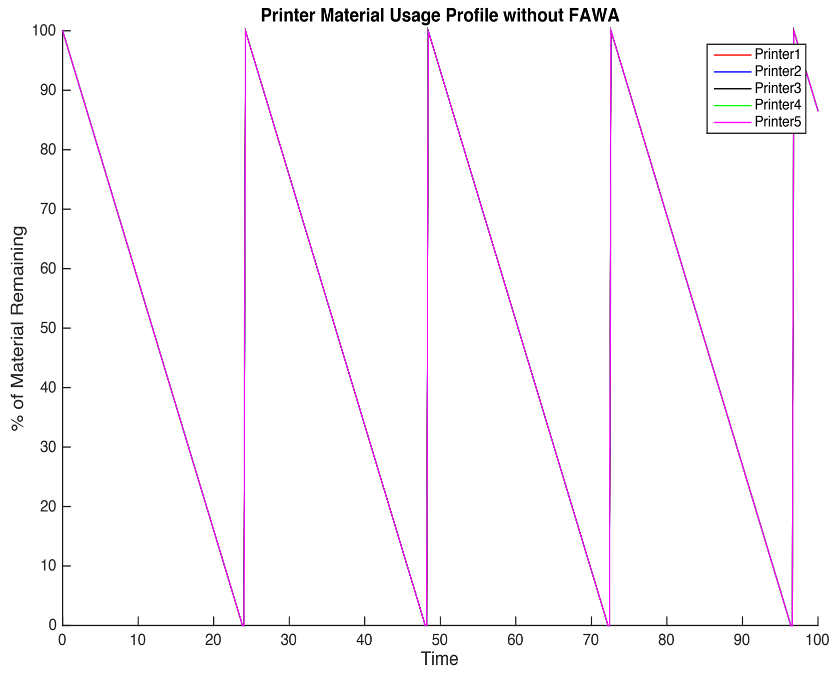


FIGURE 4.22 Usage rates and maintenance schedule when not using FAWA and maintaining 1x printing speed for all printers.

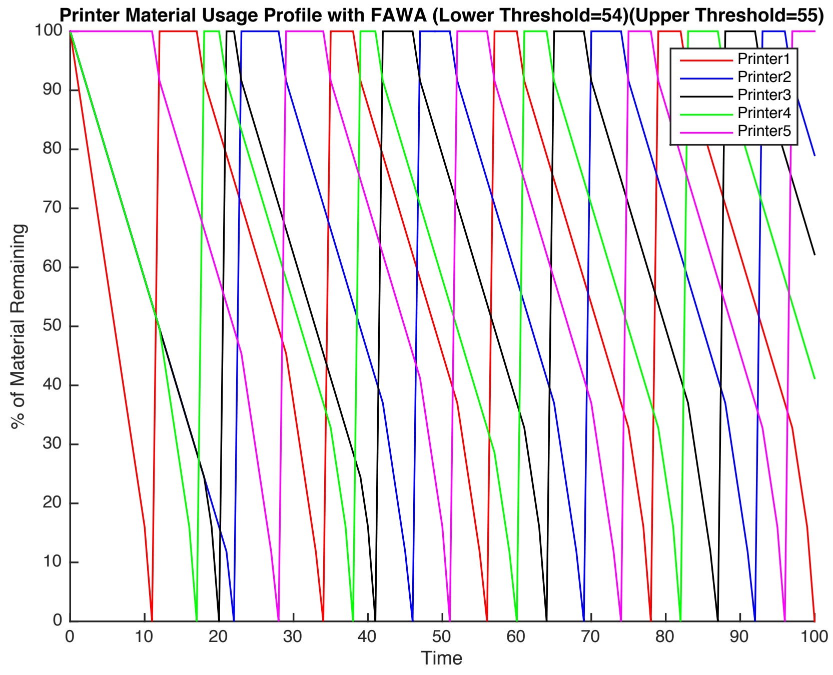


FIGURE 4.23 Usage rates and maintenance schedule when using FAWA and incorporating one printer at 2x, one printer at 0x, and the remaining three at 1x printing speeds.

TABLE 4.5 Maintenance statistics for 5 collaborative 3D printers with simultaneous refills. Examining effects of using FAWA and not using FAWA.

	FAWA	No FAWA
% Uptime Maintained	100.0%	99.0%
Est. Profit per Year	$4,292,859	$4,244,220
Est. Extra Profit Earned per Year w/ FAWA	----------	$48,639
Est. % Profit Increase per Year w/ FAWA	----------	1.1%

As we can see in Table 4.5, by desynchronizing failures with the use of variable speed printing, we were able to achieve 100% uptime. However, because repair time is so small compared to print time, even without FAWA we still achieve 99% uptime. Unfortunately, because refilling a printer is so quick and easy, unless daily production levels or the value of the printed parts are excessively high, the benefits of FAWA for increased production profit will be limited. Therefore, as previously mentioned, until 3D printing technologies advance to be able to print at much faster speeds or with much more valuable materials, FAWA will only provide limited improvements in terms of "% Profit Increase".

4.2.3 Desynchronizing Refills using an Extra Printer to Maximize Uptime

Lastly, for this final case study, we once again want to maximize uptime in order to achieve a constant level of production. However, this time we will investigate printing for assembly, where we have multiple printers each printing different sized pieces that will then be assembled together at a later stage. This means that equal numbers of each part need to be produced. We assume that each part requires a different percentage of a printer's material capacity but it is also assumed that each part is printed in the same amount of time. There is one part that requires 8% of a printer’s capacity per part, one that requires 2%, one that requires 1%, and one that requires 3%. Additionally, we say that all pieces of the total part can be printed in 1 hour, resulting in a desired level of production of 24 assembly kits being printed per day. Moreover, similar to the previous examples, each refill is said to take a total of 15 minutes, and each part is again valued at $100. Lastly, the price of the extra printer is said to be $6,000. While this initially appears to be an n-of-n system, due to the fact that the printers have to stay synchronized in terms of the total number of parts printed, we do not want to synchronize failures, but rather, as previously stated, we want to maintain a constant level of production. Therefore, we introduce an extra printer to the system, such that we are able to treat this as a k-of-n system and thus, again use FAWA’s desynchronization strategy to maintain uptime.

First, as seen in Figure 4.24, we simply ran the simulation without FAWA and no extra printer. Next, we ran simulations without FAWA but with an extra printer, and then with FAWA and an extra printer, and these maintenance schedules can be seen in Figures 4.25 and 4.26, respectively. The maintenance statistics for each of these cases can then be found in Table 4.6.

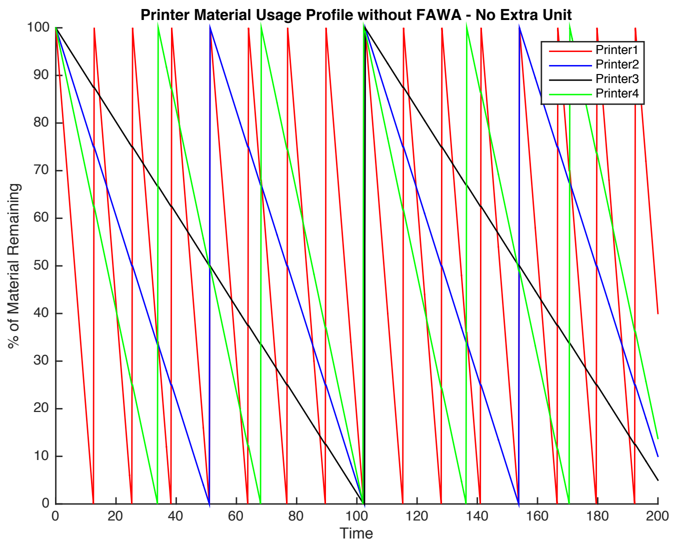


FIGURE 4.24 Usage rates and maintenance schedule when not using FAWA and no extra printer.

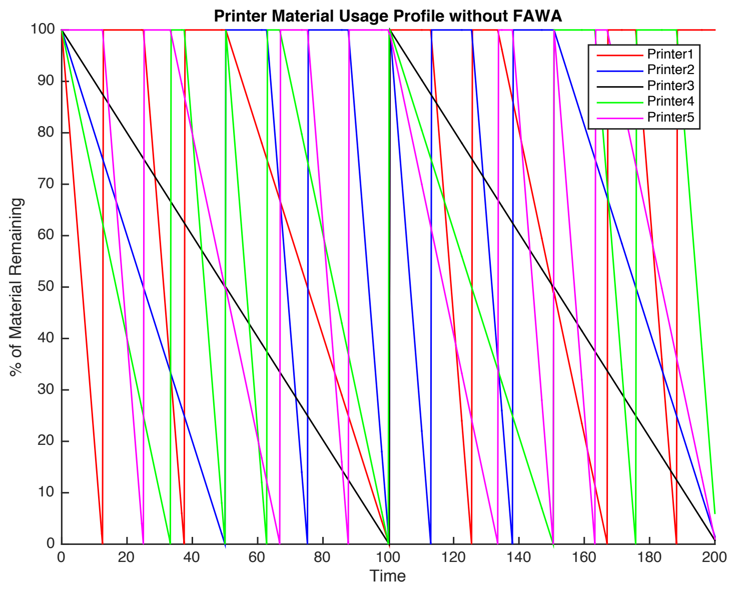


FIGURE 4.25 Usage rates and maintenance schedule when not using FAWA but incorporating an extra printer.

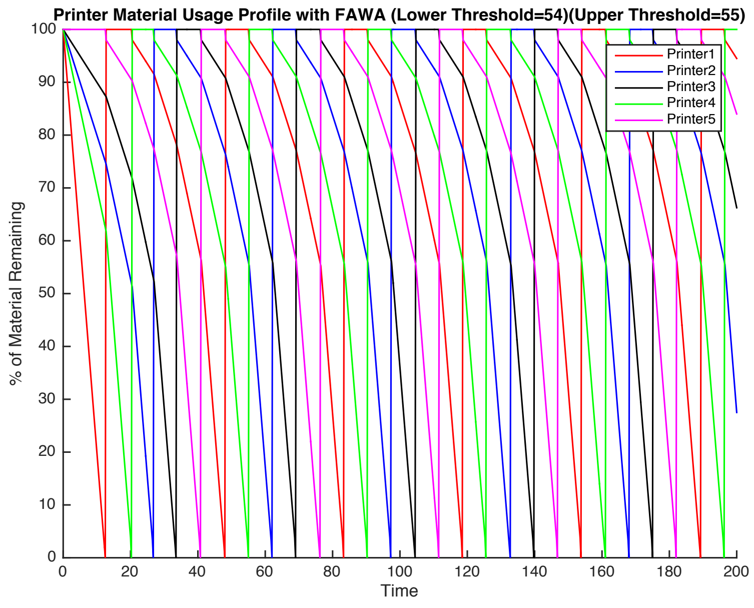


FIGURE 4.26 Usage rates and maintenance schedule when using FAWA and incorporating an extra printer.

TABLE 4.6 Maintenance statistics for 4 collaborative 3D printers with simultaneous refills. Examining effects of using FAWA with an extra printer, not using FAWA with an extra printer, and not using FAWA without an extra printer.

	FAWA	No FAWA	No FAWA (No Extra Printer)
% Uptime	100.0%	99.5%	97.5%
Total Profit per Year	$815,740	$811,395	$801,540
Total Extra Profit Earned per Year w/ FAWA	----------	$4,345	$14,200
% Profit Increase per Year	----------	0.5%	1.8%
Extra Unit Paid Off	----------	1.38 years	0.42 years

Once again, FAWA produces the best results, being the only strategy to achieve 100% uptime, however, it was only able to produce minor increases in profit earn per year. As has been discussed at length, what dictates how beneficial FAWA can be is the cost of downtime, which is dependent on the value of the part, the print speed, and refill time. Therefore, while the extra profit earned per year is currently minor, increases in any of the listed factors will only add to these gains. Additionally, in this example, as in all of the others, we predict that every refill will happen right on time and within 15 minutes. However, this could very well be disrupted by the fairly sporadic nature of the failures depicted in Figures 4.24 and 4.25. In contrast, using FAWA produces a very structured and balanced maintenance schedule, as shown in Figure 4.26, that would help technicians know exactly when they need to be available, increasing the likelihood that refills would always be taken care of immediately.

4.2.4 Conclusions of FAWA for 3D Printers

Based on the scenarios explored in this section, we were able to show that FAWA is an effective strategy for synchronizing maintenance and optimizing refills for printers in an n-of-n type system, such as in the mobile robot, chunk-based printing case study. Additionally, we showed that FAWA's desynchronization strategy can be used when there is an extra printer available or the printers are capable of printing at different speeds in order to maximize uptime and maintain constant production. While FAWA is predominantly the best strategy for optimizing refill schedules for collaborative printing, current 3D printing technology puts a significant limit to how beneficial it can be. As we have discussed throughout this work, the main benefit of deploying FAWA is that it optimizes downtime, however, if it takes 10 hours to print a $100 part, 1 minute of downtime is only worth about $0.17. Therefore, because current printing technology is so slow and refilling a machine is relatively fast, there is very little cost to downtime and therefore, optimizing downtime does not add significant value. However, assuming the technology continues to grow, producing more valuable products in much faster times, then, based on the "% Downtime Saved" values demonstrated in the examples above, FAWA could potentially add tremendous value in the future.

CHAPTER 5

CONCLUSIONS

When dealing with a complex system that is capable of performing multiple tasks and can accomplish each one a multitude of ways, it must be acknowledged that how the system fails depends on how it is being used. Therefore, when planning how a complex system should be used you need to understand the differences in how each option is likely to fail. You must be aware that the optimal mission plan at the beginning of life will not necessarily be the optimal plan at the end of life. Unfortunately, current failure analysis and mission planning techniques do not do this. Thus, in order to optimize a system's life, and therefore, its maintenance schedules, a new approach for combining failure analysis and mission planning was needed, and that is exactly what this research has tried to create with its proposed methodology of fault adaptive workload allocation (FAWA). The primary goal of this research was to show that if you know how a system will degrade, you do not have to simply watch it happen, but in fact you can implement rule-based workload allocation strategies that alternate between multiple available mission plans to optimally control degradation, ultimately creating more optimized maintenance schedules. This work showed that what may initially appear as multiple suboptimal mission plans, based on the chosen performance metric, can in fact be combined into a meta-plan that, over the course of a system's total life, is more optimal than any individual plan.

As we have mentioned several times over the course of this work, the primary goal of FAWA is to reduce the amount of downtime through controlled degradation, such that work stoppages are minimized. However, the significance of minimizing stoppages is entirely dependent on the actual value of downtime. If a 3D printer is so slow that it is only able to print

2 parts a day and the parts are only valued at $20, then saving 5 minutes of downtime is essentially worthless and FAWA will not provide much benefit. However, it is still our assumption that reducing the number of stoppages required to fully repair a system will likely always add at least some value. For example, being able to take your car in to the mechanic and have them repair all of your filters, change your oil, and rotate your tires, all at the same time, should always be preferred over taking it in one day to change the filters, the next day for the oil change, and then the next day for the tire rotation. Similarly, in a manufacturing setting, unless there is only one maintenance technician who happens to be standing right next to a machine when it fails, and there is no cool-down or warm-up time required for the machines, stopping production once to repair all parts, will likely always be preferred to stopping production multiple times to make the same number of repairs. For example, if you have 5 parts, it is very unlikely that 1 stop for 5 repairs would not be an improvement over 5 stops for 5 repairs. Furthermore, in addition to showing how it is possible to minimize the number of stoppages using FAWA, this research also helped show the difference between two other traditional, non-FAWA condition-based maintenance strategies, i.e. repairing components only when they fail and repairing all components any time a single component fails. Lastly, this research showed how the combination of many different factors, such as product value, daily production output, repair time, repair costs, and additional stoppage time, all interact with each other and how they can each drastically change the value of each potential maintenance strategy.

This research started with one unique question: is it possible to strategically alternate between multiple mission plans and tasks in order to control when and where failures occur in complex systems? To answer this, we first had to obtain the necessary failure information for our systems of interest, which involved developing unique physics-based, fault augmented system models that could be used to determine each component's degradation rates for multiple different mission plans based on the workload allocation of each plan. Next, we developed unique fault adaptive strategies that could identify optimal alternative mission plans in real-time, based on which components were healthiest and which were most degraded, such that, whenever possible, the healthiest components were given more or less of the required workload compared to the most degraded components. These strategies were then shown to provide the unique ability to either synchronize or desynchronize failure times depending on whether we were dealing with n-of-n or k-of-n systems, respectively, by alternating between different mission plans at various health-based, or resource-based, standard deviation thresholds. Ultimately, this work successfully showed that by taking advantage of the increased functionality of complex systems, we can actively control when and where failures occur by utilizing the unique contributions of FAWA.

Lastly, some might argue that degradation rates could be balanced simply by using higher quality materials or having more resource reserves for the components that are used the most. However, for complex systems that can perform a number of mission objectives, a number of different ways, each component may have the largest workload depending on which mission plan is chosen. For example, the exact same type of industrial robot arm may be used at multiple stations on an assembly line, each performing different operations; one task's optimal mission plan might use the shoulder joint the most and the other's might use the elbow joint the most. Therefore, when designing the robot, it would be impossible to know which component needs to be the most durable, and if certain components were made stronger and conditions were placed on how the system should be used to exploit this increased durability, that would defeat the entire purpose of having a multifunctional robot. Therefore, the most sensible way, as of now, to control degradation for complex systems is through fault adaptive workload allocation.

5.1 Future Work Recommendations

Currently, the degradation models are solely dependent on expected cycle time and average velocity for each joint. However, future models will need to account for other degradation factors, such as acceleration, force, and material, in order to determine more accurate degradation rates. To find these different degradation factors, actual failure data will need to be collected for numerous different tasks and mission plans. Then, some form of data analysis, likely with the help of machine learning algorithms, will need to be performed in order to classify the fundamental phenomena that cause degradation. Then, the FAME models will need to be updated to match these findings, which in turn could then, theoretically, be used for determining detailed degradation profiles and life-expectancies in terms of total number of cycles possible for any conceivable task and mission plan.

Additionally, FAWA will need to be applied to other, non-manufacturing, types of complex systems with a wide variety of mission objectives and component types in order to test its broad scale applicability. One idea would be to use it for passenger pick-up and route planning for ride-sharing platforms, such as Uber. Based on the number of available cars, remaining gas in each car, distance of requested rides, number of total requests, and the needed occupancy of each ride, among a number of other factors, could be used with FAWA to optimize which car should pick up which request, such that wait times are minimized. Uber surely already has sophisticated algorithms doing such things, however, it would be interesting to see if by using FAWA's strategies, we could find an optimal, desynchronized allocation, such that a minimum number of cars ever need gas at the same time, maximizing the number of cars on the road, or synchronize rides such that a maximum number of drop-offs occur just before projected rushes, maximizing the number of cars available during peak times.

Lastly, the effectiveness of the least-to-most degraded decision metric needs to be evaluated for its scalability to systems with vastly more components. Additionally, because the degradation per cycle calculations are probabilistic estimations, every cycle will not take exactly the same amount of time as our model predicts, and so, there will need to be some level of uncertainty added to the model in order to make it behave more realistically.

REFERENCES

[1] S. Bankers. “Robustness, Adaptivity, and Resiliency Analysis.” In *AAAI Fall Symposium.* 2010.

[2] N. Tague. “Failure Mode Effects Analysis (FMEA).” Excerpted from *The Quality Toolbox.* Second Edition, ASQ Quality Press. Pgs. 236–240. 2004. http://asq.org/learn-about-quality/process- analysistools/overview/fmea.html

[3] K. Stillings. “Advanced Failure Mode Effects Analysis.” 2010. http://www.asq501.org/images/Failure%20Mode% 20Effects%20Analysis_Advanced.pdf

[4] A. Nannikar, D. Raut, R. Chanmanwar, S. Kamble, D. Patil. “FMEA for Manufacturing and Assembly Process.” In *International Conference on Technology and Business Management.* 2012.

[5] “Performing a Failure Mode and Effects Analysis.” *Flight Assurance Procedure no. P-302-720.* www.everyspec.com.

[6] “Failure Modes, Effects and Criticality Analysis (FMECA) for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities.” *Department of the Army.* Technical Manual: TM 5-698-4. 2006.

[7] “Failure Mode, Effects and Criticality Analysis (FMECA).” *Reliability Analysis Center*. 1993.

[8] R. Stone, I. Tumer, M. Van Wie. “The Function-Failure Design Method.” In *ASME Journal of Mechanical Design*. Vol. 127. Pg. 397-407, 2005.

[9] R. Stone, I. Tumer, M. Stock. “Linking product functionality to historical failures to improve failure analysis in design.” *Research in Engineering Design,* Vol. 16(2) Pg. 96-108, 2006.

[10] K. Grantham-Lough, R. Stone, I. Tumer. “The risk in early design method.” *Journal of Engineering Design*, Vol. 20(2) Pg 144-173, 2009.

[11] I. Tumer, R. Stone. “Mapping function to failure mode during component development.”

[12] K. Grantham-Lough, R. Stone, I. Tumer. “Implementation Procedures for the Risk in Early Design (RED) Method.” *Journal of Industrial and Systems Engineering,* Vol. 2(2) Pg 126-143, 2008.

[13] D. Jensen, I. Tumer, T. Kurtoglu, C. Hoyle. 2012. “Application and Analysis of Complex Systems Using the Function Failure Identification and Propagation Framework.”

[14] T. Kurtoglu, I. Y. Tumer. “A graph-based fault identification and propagation framework for functional design of complex systems.” *Journal of Mechanical Design*, Vol. 130(5), 2008.

[15] P. Dallosta, T. Simcik. “Designing for Supportability: Driving Reliability, Availability, and Maintainability.” In *Defense AT&L: Product Support Issue*, 34-38. March-April 2012.

[16] B. Saha, T. Honda, I. Matei, E. Saund, J. de Kleer, T. Kurtoglu, Z. Lattmann. “A Model-Based Approach for an Optimal Maintenance Strategy.” In *European Conference of the Prognostics and Health Management Society*. 2014.

[17] A. Jardine, D. Banjevic. “A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance.” *Mechanical Systems and Signal Processing*, Vol. 20(7) Pg. 1483-1510. 2006

[18] R. Barlow, L. Hunter. “Optimum Preventive Maintenance Policies.” *Operations Research*, Vol. 8(1), Pg. 90-100. 1960.

[19] J. Rushby, J. Crow. “Evaluation of an Expert System for Fault Detection, Isolation, and Recovery in the Manned Maneuvering Unit.” *NASA Contractor Report 187466.* Dec. 1990.

[20] J. Carnes, A. Misra. “Model-Integrated Toolset for Fault Detection, Isolation and Recovery (FDIR).” *IEEE*. Pg. 356-363. 1996.

[21] I. Hwang, S. Kim, Y. Kim, C. Eng Seah. “A Survey of Fault Detection, Isolation, and Reconfiguration Methods.” *IEEE Transactions on Control Systems Technology*. Vol 18(3). May 2010.

[22] “Fault-Detection, Fault-Isolation and Recovery (FDIR) Techniques.” *NASA Technique DFE-7.*

[23] L. Tang, G. Kacprzynski, K. Goebel, A. Saxena, B. Saha, G. Vachtsevanos. “Prognostics-enhanced Automated Contingency Management for Advanced Autonomous Systems.”

[24] R. Maxion, D. Siewiorek, S. Elkind. “Techniques and Architectures for Fault-Tolerant Computing.” In *Annual Review of Computer Science*. Vol. 2 Pg. 469-520. 1987.

[25] J. Jiang, G. Niu. “PHM Technology Development towards Vehicle Automated Contingency Management.” In *3rd International Conference on Materials and Reliability.* Nov. 23-25. 2015.

[26] L.Hao, K. Liu. “Controlling the Residual Life Distribution of Parallel Unit Systems Through Workload Adjustment.” In *IEEE Transactions on Automation Science and Engineering.* Vol. 14(2) April 2017.

[27] C. DeStefano, D. Jensen. "Failure Identification for Mission Analysis for Complex Systems." In *Proceedings for the ASME Design Engineering Technical Conferences; International Computers & Information in Engineering Conference.* 2015.

[28] C. DeStefano, D. Jensen. "Adaptive Mission Planning and Analysis for Complex Systems." In *Journal of Computing and Information Science in Engineering.* 17(4). Oct 2016.

[29] R. Minhas, J. de Kleer, I. Matei, B. Saha, B. Janssen, D. Bobrow, T. Kurtoglu. "Using Fault Augmented Modelica Models for Diagnositics." In *Proceedings of the 10th International Modelica Conference*. 2014.

[30] Motion Controls Robotics. FAQ. https://motioncontrolsrobotics.com/2-of-5-most-commonly-asked-questions-from-new-customers-what-happens-if-the-cell-fails-after-you-the-supplier-have-left-our-site/

[31] J. Holmes. "The Top-Producing American Car Plants." 2012. https://www.automobilemag.com/news/the-15-top-producing-american-car-plants-151801/

[32] D. Conway. "Robots Will Save Manufacturing Billions." 2014. https://ark-invest.com/research/robots-will-save-manufacturing-billions

[33] "Average New-Car Prices Up More Than 1 Percent Year-Over-Year for December 2018, Closing the Strongest Year of Growth Since 2013, According to Kelley Blue Book." 2019. https://finance.yahoo.com/news/average-car-prices-more-1-110000010.html

[34] P. Daisyme. "Understanding the Financial Cost of Downtime in Manufacturing." 2018. https://due.com/blog/understanding-the-financial-cost-of-downtime-in-manufacturing/

[35] L. Poudel, Z. Sha, W. Zhou. "Mechanical strength of chunk-based printed parts for cooperative 3D printing." 2018. Elsevier. Procedia Manufacturing.

[36] L. Poudel, W. Zhou, Z. Sha. "Computational Design of Scheduling Strategies for Multi-Robot Cooperative 3D Printing." 2019. In *Proceedings of the ASME 2019 IDETC/CIE.*

[37] L. Marques, R. Williams, W. Zhou. "A Mobile 3D Printer for Cooperative 3D Printing. 2017. In *Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium*

[38] L. Poudel, C. Blair, J. McPherson, Z. Sha, W. Zhou. "A Heuristic Scaling Strategy for Multi-Robot Cooperative 3D Printing." Rapid Prototyping Journal. (Under Review)

[39] Ultimaker S5. https://sourcegraphics.com/3d/printers/ultimaker/ultimaker-s5/?gclid=Cj0KCQjwpsLkBRDpARIsAKoYI8yqnPf0Q8SV8V5w5x5Rdfro_eUMNHgduc4aHTcM04xoLGQDJjAdtfwaAg1nEALw_wcB