DISSERTATION

A COMBINED CLASSIFICATION AND QUEUING SYSTEM OPTIMIZATION APPROACH FOR ENHANCED BATTERY SYSTEM MAINTAINABILITY

Submitted by

Badruddin (Rudy) Pirani

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: James Cale

Steven Simske
Erika Miller
Josh Keller

ABSTRACT

A COMBINED CLASSIFICATION AND QUEUING SYSTEM OPTIMIZATION APPROACH FOR ENHANCED BATTERY SYSTEM MAINTAINABILITY

Battery systems are used as critical power sources in a wide variety of advanced platforms (e.g., ships, submersibles, aircraft). These platforms undergo unique and extreme mission profiles that necessitate high reliability and maintainability. Battery system failures and non-optimal maintenance strategies have a significant impact on total fleet lifecycle costs and operational capability.

Previous research has applied various approaches to improve battery system reliability and maintainability. Machine learning methodologies have applied data-driven and physics-based approaches to model battery decay and predict battery state-of-health, estimation of battery state-of-charge, and prediction of future performance. Queuing theory has been used to optimize battery charging resources ensure service and minimize cost. However, these approaches do not focus on pre-acceptance reliability improvements or platform operational requirements.

This research introduces a two-faceted approach for enhancing the overall maintainability of platforms with battery systems as critical components. The first facet is the implementation of an advanced inspection and classification methodology for automating the acceptance/rejection decision for batteries prior to entering service. The purpose of this "pre-screening" step is to increase the reliability of batteries in service prior to deployment. The second facet of the proposed approach is the optimization of several critical maintenance plan design attributes for battery systems. Together, the approach seeks to simultaneously enhance both aspects of maintainability (inherent reliability and cost-effectiveness) for battery systems, with the goal of decreasing total lifecycle cost and increasing operational availability.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The United States military uses various types of large platforms (e.g., ships, submersibles, ground and air vehicles) to execute a wide range of operational missions throughout the world. These platforms are equipped with weapons, communication, surveillance and/or other critical systems used to carry out specific operational needs. Batteries are a key power source for many of these systems providing either primary or critical back up power that is essential for system execution and platform safety. These battery systems differ from those seen in commercial applications due to the unique operational and environmental requirements that they must meet. These requirements and the need to ensure high reliability and performance (often through short replacement cycles) provide a primary driver for maintenance and procurement planning, which can result in significant lifecycle costs.

Most batteries today are complex systems made up of cells, a battery management system (BMS), and packaging; the complexity of which is dependent on many factors. A cell is the simplest part within the battery (e.g,. AA commercial "battery"). Multiple cells are connected into series and/or parallel configurations to construct the battery. The BMS is used to monitor and support the management of the battery/cells. Depending on the battery chemistry (e.g., lead acid, lithium-ion, etc.) the cells and/or the battery is packaged in specific ways to support the desired performance and safety mitigation.

The Navy faces challenges with battery systems due to the growing need for more power, longer operating time, and higher reliability. As new capabilities are added to platforms or existing ones are advanced, platform power needs grow, increasing the importance in battery systems. For many platforms the battery system operates as a secondary source of power in order to meet operational capability and to support platform safety. In these situations battery failures can lead to loss of operational time or a possible loss of the platform and personnel. This makes the battery systems performance and maintainability critical.

In order to maintain the platform critical systems, the Navy must take a proactive and conservative approach in regards to system lifecycle management. This requires significant resources and funding at various support levels to maintain the battery systems. However, a large impact to the platform availability and program costs can occur when unscheduled maintenance actions are required due to unforeseen performance issues or lead to issues with the host system/platform. Premature battery failures can result in loss of operational time, unplanned costs, and risk to platform safety as well as impacts to other mission and program requirements.

Because maintenance costs are directly influenced by the reliability of subsystems and components, a path for reducing these costs and achieving overall higher system maintainability is considered. The research here proposes a dual focused approach where classification methodology is applied to battery pre-acceptance screening in order to improve system reliability and then a maintenance strategy optimization is analysis is applied to reduce lifecycle costs and increase platform operational readiness.

The proposed classification method is the identification of cells (components within the battery) which are likely to fail earlier than planned and to prevent their installation or continued use. The act of discriminating between cells which are more likely to fail from those which are not (in a probabilistic sense) is referred to herein as the cell *classification problem*. Because there are costs associated with repairing or replacing failed batteries, it is desirable to perform classification as part of a pre-deployment acceptance test and evaluation. For batteries already fielded, the approach can identify those platforms at risk and allow for corrective actions to be scheduled proactively to better support the platform within the identified maintenance cycles. Desirable attributes of the classification method are (*i*) the ability to classify parts based on easy to perform and minimally invasive battery cell measurements and (*ii*) use of a mathematically simple and computationally efficient algorithm.

Further lifecycle cost reductions and operational availability improvements are considered through *maintenance strategy* optimization. Here the maintenance strategy refers to the application of resources (e.g., organizational, intermediate, and depot maintenance facilities) to balance system maintenance and total program costs in support of the fielded systems. The approach will optimize several critical maintenance plan design attributes through modeling and simulation in order maximize platform operational availability ("uptime") and minimize lifecycle costs.

To demonstrate this approach, an example case study will be utilized where a battery system has been fielded and supported in a generic platform that has show performance issues. There are always several of these platforms in various operational or deployed states and batteries are procured periodically to meet the planned battery lifecycle replacements plan. Battery systems have experienced early life failures were a significant number of cells have declined in capacity prior to their expected end of life. This failure requires unscheduled maintenance actions that have a significant lifecycle cost and operational schedule impact. However, not all cells within the failing system exhibit this earlier in life performance decay. This suggests that issue is related to cell variations that may fall within classes, "good" and "failure prone."

This dissertation is organized as follows. In Chapter 2 a background on general cell and battery technology, machine learning approaches for cell and/or battery health prognostics as well as a review of the maintenance planning optimization is provided. Chapter 3 details the case study battery system, applied maintenance strategy, and example baseline maintenance strategy for the case study review. Chapter 4 outlines the classification techniques investigated for battery screening. Chapter 5 shows the implementation and comparison of results of the selected classifier techniques. Chapter 6 describes the impacts of critical parameters changes to a battery system maintenance strategy. The dissertation concludes with a summary conclusion and identified future research in Chapter 7.

Chapter 2

Background

2.1 Battery Technology Overview

In order to develop and asses the battery or cell classification approaches, an understanding of battery systems and related features is required. The following section describes the general characteristics, functions and important performance parameters associated with battery systems and their components.

A battery is an electrochemical energy storage device where energy is transformed between a stored chemical and electrical state [1]. A battery is the combination of cells electrically connected in a series and/or parallel configurations. The cell is the basic electrochemical unit that provides electrical energy to the system. Battery configurations vary depending on the voltage, current, and capacity requirement of the application as well as the specific chemistry (e.g., lead acid, lithium-ion, etc.) being used. Energy is contained in the active material of the positive (cathode) and negative (anode) electrodes and is released or stored through oxidation-reduction (redox) reactions. Typically, the terms "battery" and "cell" are interchangeable, but for this dissertation a cell will be considered as the lowest unit manufactured and provided by a vendor. For this review, the battery shall be considered to be multiple cells assembled electrically in a series parallel configuration once installed on the platform to be discussed. Cells are made up of four basic elements; the anode electrode, cathode electrode, separator, and electrolyte. Figure 2.1 provides a general diagram of the charge and discharge reactions occurring in a typical cell.

The material, production and design of the core components of the cell are selected to achieve specific performance goals [1–4]. Specific cell chemistries (e.g., lead acid, lithium ion, etc.) are selected based on the performance characteristics associated with the anode and cathode material which are assembled to make the cell. Cell chemistry is selected to

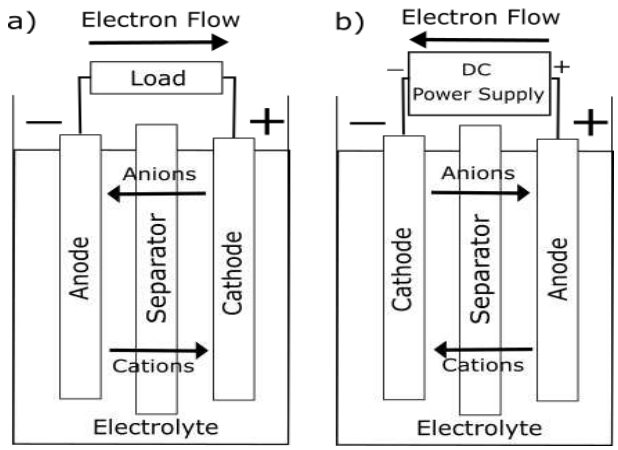

Figure 2.1: Basic Cell Diagram; a) discharging and b) charging.

meet key performance parameters (KPP) and support program requirements like lifecycle costs, system performance, and maintainability. Table 2.1 [1] provides a comparison of common rechargeable battery chemistries and their associated characteristics to highlight the variation in performance that can be found.

Table 2.1 highlights an important fact that cell performance, even within the same chemistry type, can be altered. This is typically done to produce cells that support specific applications or environments. An example is in the development of lead acid batteries for engine start applications as compared to uninterrupted power sources (UPS) [3]. In this example, a vendor may change parameters within the manufacturing process in order to modify the active material porosity in order to achieve higher rate discharge currents or improve overall energy densities. These manufacturing processes are part of establishing the design of the cell and thus influence performance. The understanding of this relationship is important as uncontrolled variation within the manufacturing process can lead to performance inconsistencies and premature failures within the same cell type.

The battery manufacturing process varies depending on the battery chemistry and vendor. Manufactures collect specific data during the manufacturing process that are used for in process quality control and for historical records in case there are performance issues once the cells/battery is fielded. This information is typically traced by assigned serial numbers

Table 2.1: Cell chemistry characteristics comparison

	Lead Acid		**Nickel Cadmium**		**Lithium-ion**
Chemistry	SLI	Portable	Vented Pocket Plate	Sealed	Cobalt Based
Cell Voltage	2.0	2.0	1.2	1.2	4.0
Operating Temperature	-40 to 55	-40 to 60	-20 to 45	-40 to 45	-20 to 50
Wh/Kg	35	30	20	35	150
Wh/L	70	90	40	100	400
Cycle Life	200 - 700	250 - 500	500 - 2000	300 - 700	1000+
Cost	Low	Low	Moderate	Moderate	High

or lot groups. Variation within these manufacturing process can lead to differences between individual cell performance. While variation cannot be eliminated, the goal is to minimize it to provide consistent performance. This is important as large performance variation between individual cells can have a negative impact on overall battery systems.

Pavlov [2] describes the impact of performance to valve regulated lead acid (VRLA) cells and batteries based on changes to manufacturing process and cell design. As an example, his research evaluated how the electrolyte soaking process (the time between the electrolyte fill and the start of cell formation) influenced the micro-structure of the active material as well as the gird (cathode and anode current collector) to active material interface. It also showed that the time allowed for the electrolyte to react to the cathode and anode material effected the amount of alpha and beta PbO_2 formed on the positive plate, which has a direct impact on cycle life. Pavlov and other researchers have conducted additional research on lead acid technology to show how changes in the manufacturing process and design (e.g. electrolyte concentration, electrolyte temperature, curing temperature) can have significant effect on a cell's performance [2–4].

Several authors [1–5] discussed general impacts to battery performance, cell aging, and end of life failure modes. Cell or battery capacity decay is typically due to material aging but can be accelerated by several factors including how often it is cycled (discharged and charged), the environment (e.g., temperature), and being used outside normal operational

parameters [1–4]. The cell aging process can be classified into two categories: (*i*) aging that involves a gradual degradation over time that is possible to be monitored, and (*ii*) those which do not have any specific mode or observable sign until a major problem or rapid changes in performance occur (e.g., internal shorts). The sudden change in battery performance can be difficult to model as the ability to determine the exact failure mode or collect data from a failure is often difficult to obtain. The most significant factors to cell aging and performance decline include the following:

- Environment temperatures
- Discharging current rates
- Depth of Discharges (DoD)
- Charging rates (fast charging)
- Time intervals between full charge cycles

The most effective and simplest method of monitoring battery behavior that was discussed by these authors included observing voltage, current, temperature, and in some cases internal cell pressure. Some of these variables can be measured during battery operation without impacting the operations of the battery or the system it is supporting. This is refereed to as online measurement. These online measurements can suffer from signal noise, disturbances and poor quality, which can be a result of degraded sensors from harsh environments.

2.2 Valve Regulated Lead Acid Chemistry Overview

The case study used in this research employs data from a VRLA battery. A fundamental review of the chemistry and typical failure modes [1–4] was conducted to gain knowledge potentially helpful in the classification development efforts. The following section provides an overview of the VRLA battery chemistry.

The VRLA cell design utilizes a highly porous separator mat made from micro-glass fibers which are soaked in sulfuric acid (H_2SO_4). The VRLA battery utilizes a starved or limited electrolyte design and does not require routine water addition, unlike flooded lead acid batteries, due to limited water loss during charging. As in the basic lead acid battery design, the positive plate active material is lead oxide (PbO_2), while the negative plate active material is lead (Pb) [1]. During discharge, the sulfuric acid is consumed and both the positive and negative electrodes are converted to lead sulfate ($PbSO_4$) while water is produced. During charge, the reaction is reversed, converting the $PbSO_4$ back to the respective PbO_2, Pb and sulfuric acid. The overall chemical reaction is shown below.

$$PbO_2 + Pb + 2\,H_2SO_4 \rightleftharpoons 2\,PbSO_4 + 2\,H_2O \tag{2.1}$$

The starved electrolyte condition allows for uniform gas transfer between the plates which is required to support the recombination reactions which prevent electrolyte water loss. A pressure release vent is used to maintain an internal pressure, retaining gases long enough to allow for diffusion to take place. This allows for water to be retained in the cell, rather than being lost as in flooded lead acid batteries. However, excessive charging at high rates will generate gas buildup at such a rate that the recombination process will be inefficient and cause H_2 and O_2 to vent out of the cell. This will lead to the traditional failure mode of "cell dry-out".

Literature review [2, 4, 6] into VRLA failure modes was conducted in order to relate performance issues seen in the system used for this research to possible features identified within the manufacturing process and cell design. Research identified three major VRLA failures that resulted from what was termed premature capacity loss (PCL); (*i*) PCL-1: interface effects, (*ii*) PCL-2: active material effects, and (*iii*) PCL-3: negative plate effects.

The research [2, 4, 6] showed that PCL-1 was caused by the formation of non-conductive layers between the grid and active material. Non-conductive layers cause high resistance at the grid-to-active material interface. This creates heating during cycling and expansion of

the positive active material adjacent to the grid allowing localized discharge near the grid which limits capacity of the electrode. The cells exhibiting this issue showed a sudden loss of capacity during the first 10 to 50 cycles. This issue is often called the "antimony-free" effect because it was first seen on cycled lead calcium alloy cells.

PCL-2 was described [2, 4, 6] as being a deterioration of the connections between PbO_2 particles of the positive active material and is caused by the expansion of the positive active material (PAM) during cycling. Expansion causes an increase in the resistance in the active material as conductivity between particles is lost. Expansion leads to reduced capacity, material degradation, and an inability to convert the PbO_2 into usable material. This is made worse by high current rates and the amount of overcharge.

PCL-3 is due to an inability to recharge the negative plate [2, 4, 6]. It occurs late in life at about 200-250 cycles. Due to water consumption during cycling, the saturation of the separator decreases. With decreased saturation in the separator, there is an improved transfer of oxygen through the separator to the negative plate where it recombines and reduces the polarization potential of the negative electrode. Increased oxygen recombination requires higher amounts of overcharge.

Okada's PCL investigations [7] provided additional detail on the possible failure mode. It stated that PCL occurs when batteries were discharged frequently to a shallow depth of discharge (DoD). The failure mode occurred when the adhesion between grid and the PAM was poor, increasing the resistance at the interface. The paper suggested that this was due to high acid concentration around the grid-to-PAM interface, interacting with a corrosion layer that was mainly consisted of Beta-PbO_2.

2.3 Machine Learning Overview

With the advancements and reduction in cost of data acquisition capability and data storage devices, there exists new opportunities to collect significant amounts of data on the production and operational use of battery systems. Battery manufactures collect in process

production data in an effort to reduce variability and ensure product quality. Most batteries utilize a BMS to monitor KPPs (e.g., voltage, current, temperature, etc.) in order to ensure safety and optimize performance through automated actions or providing information to the users who in turn take action to manage the battery. Given the availability of this information, the opportunity exists to apply machine learning and classification methods to improve performance and lifecycle maintainability. These approaches could be utilized to evaluate manufacturing and performance features in an effort to identify issues, predict battery performance, and allow for preventative actions based on the recognition of patterns within the various data sets. The following section describes literature review on general machine learning information.

Literature review [8–12] in machine learning and classification identified several common high level pattern recognition approaches. A statistical model is one of the simplest approaches available. It is based on statistics and probabilities where the focus is in identifying possible feature sets which are chosen in such a way that different patterns occupy non-overlapping feature spaces. The effectiveness of the feature set is determined by how well patterns from different classes can be separated. After performing the analysis of the probability distribution of a pattern belonging to a certain class, a decision boundary is determined. Here the patterns are projected to some pre-processing operations to make them suitable for training purposes. The system learns from the training patterns and adapts itself to recognize or classify the unknown test patterns. The distance between the patterns is determined in the statistical space and then these feature values are presented to learned system and in this way classification is performed.

Structure or syntactic models [8, 9] utilizes the relation between features and can take into account more complex relationships between features then the numerical feature sets used in statistical models. The patterns to be recognized are called primitives and the complex patterns are represented by the inter-relationship formed between these primitives and the defined rules associated with this relationship(s). This model is used in pattern

recognition to provide a description of how the primitives are interconnected based on the overall hierarchical structure. However, the implementation of a structure model approach may lead to issues due to noise within the data sets and the conclusions derived from the training data. Thus, very large training sets may be required.

Template matching models [8–10] are used in image processing to determine the similarity between two samples, pixels or curves to localize and identify shapes within an image. In this approach a template of the pattern of interest is stored and is compared to the image under review. This is done with each pixel while also taking into account all possible position in the input image. In image processing the template is compared to the image of interest by maximizing the spatial cross-correlation or by minimizing a distance. After calculating the matching rate for every possibility, the largest one which exceeds a predefined threshold is selected. This process does not work efficiently in the presence of distorted patterns. Template matching is used mainly for the identifying the location of objects within an image, tracking of moving objects, registration of images between different spectra or different photography time.

Neural networks models [8–10] are a series of algorithms trained to resolve complex problems based on available information. The model is loosely based on the human brain and is designed to recognize patterns within the data set. The nodes of the network are formed from artificial neurons where the directed edges (with weights) are connections between neuron outputs and neuron inputs. A training process is used to identify the importance of the contribution of the preceding neuron and thus develop weighting factors for the various nodes. Neural networks have the ability to learn complex non-linear relationships, use sequential training procedures, and adapt themselves to the data. The number of neuron layers can improve the performance of a system, but trade-offs must be maintained between the size of network and the resulting complexity. The layers must be large enough to adequately represent the problem domain and small enough to permit the variation within the training data. Neural Networks were reported to have been successfully applied to data mining, document

classification, financial forecasting, the field of industrial product testing as well as many other applications.

A maximum entropy classifier [9, 13] is a logic regression model that is used to predict the probabilities of more than two possible outcomes. The approach assumes that a linear combination of the observed features and some problem-specific parameters can be used to determine the probability of each particular outcome. Training data is used to develop the best possible values of the parameters for a given problem. The training data used to describe the distribution is typically a set of real-valued variables or "features." The features values are expected to match the average value for a set of sample points taken from the target distribution. The derived entropy from the data set (i.e., that is most spread out or closest to uniform) is used to determine the probability and classify the information.

The k-nearest neighbor algorithm (k-NN) [9,12] is a method for classifying objects based on closest training examples in the feature space. The k-NN approach classifies objects based on an example-based learning approach where the function is only approximated locally. The object is assigned to the class that is most common to its surrounding k-nearest neighbors. The model is developed by utilizing training data that has been pre-classified to points within an n-dimensional Euclidean space. New data sets needing a class label are placed into the space and classification is conducted at this time. Classification of the new data is done by assigning the label which is most frequent among the k-training samples nearest its location. The selection for k is dependent on the available data set. The k determination value can be based on an analysis of best fit, where multiple k value can be evaluated to provide the best sum squared error (SSE) of differences.

Support vector machines (SVM) [9,10,12] classify objects by constructing a N-dimensional hyper-plane that optimally separates the data into two categories. SVM attempts to create decision boundaries that maximize the distances between the two classes. SVM is an efficient method of finding an optimal hyper-plane for separating non-linear data. SVM training algorithm builds a model based on provided training data where the examples have been

correctly classified within the categories. The standard SVM is a two-class SVM which takes a set of input data and predicts the possible class, for each input, among the two possible classes. However, SVM can be combined with ensemble approaches (e.g., decision tree, voting) in order perform higher level classifications. It was noted that SVM can be sensitive to noise in small and medium data sets due to the data set being used to set the boundaries for the system.

Principal Component Analysis (PCA) [14] was identified as an important tool in supporting various efforts including data visualization, pattern recognition, risk management and multiple statistical based efforts. It is a methodology to support dimensional reduction where data transformation is done in order to identify a lower dimensional space. PCA utilizes the variation within a data set to establish a new axis, beginning with the direction of the largest variation. It centers the data by subtracting off the mean and then chooses the direction with the largest variation. It finds another axis orthogonal to the first by reviewing the remaining variation. This is continued until all the needed axis are defined. The last few axes are typically found with very little variation and thus can be removed without affecting the variability in the data leading to a reduction in dimensional space. The new axis are called the principal components (PC). Standard PCA approaches are sensitive to outliers or noise data which can have an effect on the solutions.

Linear Discriminant Analysis (LDA) [9, 15] utilizes a linear combination of features to classify two or more groups or events. It is a dimensionality reduction technique similar to PCA were features in high dimensional spaces are placed in lower dimensional space. LDA attempts to find an optimized decision boundary for data classification by utilizing the covariance and probability of the classes within the data set. It attempts to define the scatter of data within each class and then maximize the distance between the two classes while minimizing the within class space. This is used to define the classes and can reduce the number of dimensions.

PCA and LDA have been utilized together in [16] to support facial recognition efforts to improve general accuracy, especially when the sample size is small. Here the authors used the PCA approach to project the facial image to a new subspace in order to resolve the identified generalization problem. By applying the PCA algorithm to the initial data set, the authors are able to reduce the within class variance for classification. With the initial image (x) transformed into the subspace (y), the LDA can then perform classification into a new space (z). Using a large data set of images for training and testing, the authors were able to show significantly higher accuracy with LDA using PCA inputs over LDA alone.

Other research [17] combined PCA and LDA in an effort to improve image recognition accuracy. The authors had identified that LDA alone had issues in the classification of images where there were high-dimensions and small sample sizes. To resolve this issue an approach referred to by the authors as "PCA plus LDA" was utilized. The authors' goal for the paper was to show the theoretical foundation on why this approach was successful. The authors performed PCA analysis using the training data set and chose to use all the generated PCs's. The were used to transform the space into R^m where $m = rank S_t$. Here m represents the "rank of the total scatter matrix" and S_t represents the "total scatter matrix." Once the space has been transformed, the authors derived the optimal discriminant vectors to act as projection axes to form the new feature extractor.

Utilizing a small number of training samples, the authors compared the accuracy of the proposed approach to other more established methods (e.g., fisherface, enhanced fisher linear discriminant model, optimal fisher linear discriminant). Using an available database of 40 distinct subjects each with 10 different images for experimentation, the authors were able to show higher classification accuracy with the proposed approach with training sample sizes of 3, 4, and 5 images.

An approach titled the "normal" method for classification [12, 18], which here will be referred to in this dissertation as the simple generalized classifier (SGC), is a statistical based classifier that assigns samples to classes based on their distance from the expected value of

each class. The "normal" or SGC classifier utilizes the relationship of the data associated with a feature or metric by developing a critical point (x_c) in order to determine class selection. The process allows for multiple features or metrics to be used and then allows for assignment of relative weights to each of the individual feature/metric classifications. This allows for improved accuracy of the classification. While this approach utilizes a simple procedure, it is able to distinguish between more than two classes of data, allows the creation of non-correlated data features from a smaller set of features, and can include rules based on expert system knowledge.

The type and use of training data is essential in the development of any of the discussed approaches. Several authors [9,11,12] shared that there were three main approaches types for the utilization of training data; (i) supervised learning, (ii) unsupervised learning, and (iii) semi-supervised learning. Supervised learning is described as utilizing training data that has been interpreted or labeled with the correct predefined outputs by someone with available knowledge. The data is utilized to generate models that are verified with the training data as well as used to accept and correctly assign new data. In contrast, unsupervised learning assumes the training data has not been organized or labeled and attempts to identify inherent patterns within the data set that can be used to develop a model that can determine the correct output of new data. Finally, a semi-supervised approach combines the supervised and unsupervised approaches by using a combination of labeled and unlabeled data (i.e typically a small set of labeled data combined with a large amount of unlabeled data).

2.4 Battery Health Prognostics

Previous literature has applied modeling and machine learning techniques to monitor, assess, and predict the cell/battery performance in an effort to extended operational life, identify issues, and improve reliability. Battery performance issues can lead to system or platform operational loss, mission interruptions, and/or full system malfunctions that could lead to disastrous consequences for the host system or platform. Given this, it is important

to be able to determine the batteries current and future performance. The following section describes information collected through literature review related to the use of machine learning techniques for battery health prognostics.

Several authors [19–21] provided an overview of stat of the art approaches used to determine the battery system state of health (SOH). The first general approach discussed was the direct measurement approach where specific tests are conducted on the cells or battery to gauge specific performance characteristics. One of the most common approaches is to conduct a full discharge at a specific constant current rate to determine capacity as wells collect data on performance features like individual cell voltage. This approach provides only an understanding of the current system's SOH and alone can not project the remaining useful life (RUL) or forecast future issue. Another direct measurement test discussed was monitoring internal impedance changes over time in order to predict SOH. However, internal impedance can be effected by various environmental conditions (e.g temperature) and thus it is difficult to provide a dependable estimate of the SOH. This is especially true for specific battery chemistries like lithium-ion. Electrochemical impedance spectroscopy (EIS) has also been used as a noninvasive method used in laboratory testing to observe the performance degradation of cells and batteries. The test can reflect internal cell impedance and electrochemical reactions which change as the battery ages or if there are internal defects. While these SOH tests described here are straight forward in laboratory environments, they can be very difficult to implement in fielded systems.

Another general methodology discussed [5, 19, 20] was model based approaches. Here the performance data from the battery is compared to a developed model. There are several model types that have been used to support cell and battery health prognostics. Experienced based models correlate expert knowledge and experience with observed situations to determine the RUL from historical data. These models rely heavily on the experts to specify the system rules and the ability to create the fuzzy data sets to define the system characteristics.

Data driven models [5,19,20] rely on previously observed data to predict the projection of a system's state or to match the derived pattern to a historical one in an effort to determine the RUL. The results obtained from this type of model are generally better than those from an experienced based model. However, these results are not easily explained or related to physical meanings within the cell as they are developed as "black box" models. These models generally require significant amounts of data to describe the full range of scenarios that the system can be subjected too. A critical issue with data driven techniques is that when the data availability is not satisfied, or the data is biased, the results can be imprecise or even incorrect entirely.

Physics based models [5,19] rely on an understanding of the system's failure mechanisms to build mathematical description of the performance degradation to predict RUL. The models are derived directly from first principles and an understanding of the physical mechanisms occurring within the cells. In order to determine the model parameters for the degradation, specific experimentation and analysis is typically required. However, if the failure modes are not clearly understood or the behavior of the system over the full range of operating conditions is not known, then this model may be difficult to construct. Model parameter identification also requires extensive experimentation. A physics-based model is often built case by case meaning that different cell chemistries or even cells built from different vendors will require tailored modifications to the model.

One paper [19] highlighted that the optimal approach would be to utilize hybrid models that would combine these model types together to best describe the battery system and the impacts from its environment. Their efforts assessed each of the potential hybrid types to provide an overview of the related benefits and risks.

A combined experience-based and data-driven model is one where the domain knowledge used to define the system‘s fault states while the data driven models are used to refine the rules created by the expert knowledge and to estimate the RUL. This approach can provide flexibility of integrating domain knowledge into data driven models for system state

or health estimation that can be used to derive RUL. However, expert knowledge may not be able to describe the system performance in all states which could lead to issues in the predictive capability of the reasoning system. However, this issue may not be significant from the practical point of view because the ultimate goal is to estimate RUL using long-term prediction instead of predicting intermediate life residual time.

A combined experience and physics based Model [19] utilizes both models in which the output of the experience based model is often used to enhance the physics-based model. The experience based model can also be found to estimate the system health state based on which RUL can be predicted. Expert knowledge is mainly used to support the determination of system failure while the physics based models are used to perform the actual RUL prediction.

A combined multiple data driven models [19] can take the form of two possible approaches. The first is where a data driven model can be used to estimate the internal system state when it is not directly measurable from the sensor signals. The estimated system state can then be used to extrapolate the future system state to predict RUL using another data driven model. The other approached outlined in the paper is where different competing data driven models can be developed for RUL prediction purposes. The results of different models can be aggregated to improve the prediction performance by a carefully designed fusion mechanism. Combining the results of multiple models can yield marked reduction in the prediction error as discussed in the above examples. However, it is critical to carefully design a fusion mechanism to ensure that the information is correctly combined. In addition, building multiple models is time-consuming and computationally intensive, which can be limiting in certain applications.

Finally, the paper [19] discussed the potential use of combining data driven model and physics based model, which could take several shapes. The authors discussed the several ways that the individual models may be combined to provide battery health prognostic information. For example, a data driven model could be used to establish a correlation between the online measurement to the internal cell state, which makes it possible to use a

mathematically sound physics based model to predict the system's internal health. Another approach discussed utilizing a data driven model to predict future measurements that are then used as inputs to a physics based model to predict RUL. This hybrid approach further addresses the issue of data availability when updating a physics based prediction model during long-term operations using a data driven model to predict future measurements. If the future measurement can be accurately predicted by the data driven model, it indeed can correct the physics based model in long-term operations, especially when the performance degradation does not quite follow the fault growth model. The trade-off is that if the data driven prediction performs poorly on future measurement prediction, the prediction result can significantly derail, and it could be worse than the physics based model.

The authors also discussed the potential of combining all three model types; experience based, data driven, and physics based models. It was stated that it can be extremely difficult and impractical to implement this approach due to the difficulty that might be encountered by each model type. However, it is potentially beneficial to leverage the strengths of all types of models and fuse all types of information (e.g., domain knowledge, maintenance feedback, and condition data and physics). Challenges remain in how to aggregate results from different competing models, how to design an appropriate fusing mechanism to integrate heterogeneous information, and how to utilize data driven models to reduce the prediction uncertainty. The authors demonstrated that the use of a hybrid approach for determining the RUL of a lithium-ion cell by combining the physics based and data driven models. This approach showed improved accuracy and confidence over a standard particle filter used for prediction performance.

These methodologies have been used in various forms in an effort to determine battery SOH, RUL, and other important performance parameters. Research [20] was conducted on the use of machine learning approaches in developing a SOH estimator for predicting battery performance in electric vehicles. The authors identified health indicators based on the battery performance data to define signatures related to capacity degradation. The authors utilized

the change in internal cell ohmic and polarized resistances as key indicators for battery health. An extreme learning machine (ELM) was used to identify the correlation between the health indicators and capacity decline to improve the speed and accuracy of the estimation of SOH. The identified relationships between cell capacity and internal cell impedance was significant based on the results, but will not be the same for all battery technologies nor correlate to all the potential end of life failure modes. There may be scenarios were internal impedance is not changing in proportion to the capacity degradation due to unique failure conditions. In addition, environmental conditions like cold temperature could have a significant impact on the estimator. Lower temperatures slow the internal chemical reactions and thus impact the ability of some cells to respond to discharges or charges. This would impact the internal resistance parameters and thus affect the results of the estimator.

Authors Barsali and Ceraolo [22] provided details on the development of lead acid cell models and a review of which models may fit specific purposes. Their model was represented as an equivalent electrical circuit combined with dynamic equations representing internal cell parameters. The cell‘s resistance and capacitance were defined as a function of the cell's state of charge (SOC) and electrolyte temperature. One approach discussed to determining the parameters of the model was by starting from a set of lab tests of a real cell or battery. Testing would consist of a several specific discharges and charges at different constant currents and environmental temperatures, along with a digital simulator of the model.

However, given the number of parameters that may affect cell performance, the authors felt that it was very difficult to develop reasonable results based on testing alone. The authors suggested that parameter identification should be broken into smaller tasks, where elements of the equivalent electrical circuit would be defined independently and then combined later. The testing and parameter development were divided into the following [22]:

- *Parameters referring to the battery capacity.* Battery capacity was defined by a specific equation that accounted for electrolyte temperature, current, and internal cell parameters. The parameters are defined by cycling the battery at various temperatures at

defined capacities. The currents and temperatures chosen to perform the tests are representative of the possible operating conditions foreseen for the battery.

- *Parameters referring to the main branch of the electric equivalent circuit.* These parameters are related to state of charge and the cell's resistances associated with discharging the cell. These parameters can be derived from a series of constant current discharges for a set period of time and then documenting the subsequent transient response up to the complete stabilization of voltage, so that the stabilized voltage can be equated to the battery electromotive force.

- *Parameters referring to the parasitic reaction branch of the electric equivalent circuit.* This refers to the internal resistance that causes self discharge. The parasitic reaction current can be determined by completely charging the battery to a set voltage and finding the stabilized current.

- *Parameters referring to the battery thermal model* The proposed battery thermal model is a simple capacitance-thermal resistance model. These two parameters can be derived experimentally or obtained from the manufacturer. Approximate estimates of the parameters can be obtained by means of the usual techniques for heat transfer problems, based on the battery mass, shape, and case material.

The other approach the authors discussed used manufacture's specification sheets to aid in defining the cell's parameters. Battery manufactures will provide average or expected performance data on their products. This will typically include rated capacities at different currents and end-of-discharge voltages that could be used in place of deriving the values from testing. The manufacturer may also supply additional information on the impact of temperature on the cell's capacity, often in terms of a "temperature coefficient" values. Other parameters such as internal resistances may also be available by the vendor if the vendor are willing to or able to share the data. The parameters are then derived based on the vendor provided information.

Within the same paper [22], the authors provide an example of an approach on the development of an equivalent electrical circuit model including how to derive the needed parameters. The model parameters did include the impact of temperature and current draw, which are important to accurately simulate cell performance. However, the described approach did not account for the impact of performance based on aging or of the various possible failure modes. The model would only support simulation of a new, good performing cell that would never degrade.

Chen and Rincon-Mora [23] conducted a review of several battery model approaches as well. They began with a discussion of electrochemical models which are mainly used to optimize the physical design aspects of batteries, characterize the fundamental mechanisms of power generation and relate cell operational parameters such as voltage and current with design parameters such as electrolyte concentration. While informative, these model types are complex and time-consuming because they involve a system of coupled time variant partial differential equations. The authors felt that these mathematical models were too abstract to be used to support fielded battery systems.

The authors [23] also discussed electrical equivalent models that utilize a combination of voltage sources, resistors, and capacitors for design and simulation. There have been many electrical models of batteries, but most fall within the following categories: (*i*) Thevenin, (*ii*) impedance, and (*iii*) run time based models.

A Thevenin based model utilizes resisters in series combined with resistor-capacitor (RC) parallel network circuits to predict battery response to transient load at a particular SOC by assuming a constant open circuit voltage (OCV). The assumption of a constant OCV is an issue as it prevents the model from capturing steady state battery voltage variations and continual operating information.

The impedance based models utilize EIS to obtain an AC equivalent impedance model within a specified frequency domain and then use a complicated equivalent circuit to fit the recorded impedance spectrum. The fitting process is difficult, complex, and non-intuitive.

Impedance based models only work for a fixed SOC and temperature setting, and therefore they cannot predict DC response or battery RUL.

A run time based model uses a complex circuit network to simulate battery run time and DC voltage response for a constant discharge current. They can predict neither runtime nor voltage response for varying load currents accurately.

A comparative study [24] was done on various prediction techniques in order to define their strengths and weaknesses as well as reviewing model accuracy against various trade-offs, like complexity and computational burden. The authors' study focused on showing how regression, classification and state estimation algorithms can help in battery health management. In order to compare the different approaches the authors began with the evaluation of lithium-ion cells to collect performance data for training and validation.

For the effort data was used from lithium-ion cells that were cycle life tested at 60% SOC at temperatures corresponding to 25°C and 45°C. EIS measurements were made periodically to determine impedance changes in the electrode-electrolyte interface as a function of cell life. EIS measurements were conducted by discharging the cells from a fully charged state to the specified OCV corresponding to the target SOC. Changes in the internal parameters of the battery were observed as shifts in EIS data plots and battery capacity degradation. Following an eight to twelve-hour rest at the defined OCV, which allowed the cells to reach electrochemical equilibrium, the impedance was measured using a four-terminal connection over a frequency range of 10 kHz to 0.01 Hz, with a minimum of eight points per decade of frequency. This test was performed on all cells at 60% SOC. Features are extracted from the collected data and were used to create a circuit equivalent model. The parameters of interest were the double layer capacitance C_{DL}, the charge transfer resistance R_CT, the Warburg impedance R_W, and the electrolyte resistance R_E. The values of these internal parameters change with various aging and fault processes like plate sulfation, passivation, and grid corrosion.

The data collected from this testing was used to support multiple prediction techniques with the following observations:

- *Statistical based baseline model.* The initial approach used was a simple data driven routine to establish a baseline for battery health prediction performance and uncertainty assessment. This was based on the calculated linear relationship between $R_E + R_C T$ and the capacity decline over time at baseline temperature, 25°C. However, through validation experimentation it was shown that the predictive RUL was off by five (5) week.

- *Probabilistic regression model.* The authors evaluated a Gaussian process regression (GPR) method to estimate the RUL. GPR is a technique for non-linear regression that computes posterior degradation estimates by constraining the prior distribution to fit the available training data. The relationship between the internal parameters $R_E + R_{CT}$ and the battery capacity was again learned from experimental data at 25°C. Analysis showed that the amount of training data was important to support good predictions. Predictions only using data points from early in life training data failed to accurately predicted the end of life. However, as additional training data was added to show continued aging of the cell, the data accuracy increased significantly.

- *Particle filter model.* The authors decided to evaluate the use of particle filters (PF) as they identified the need to track cell performance trends as they changed over time and then modify the predictions to conform to established degradation models. PF uses both the information collected from the process measurements but also incorporate any models available for the process. The authors combined PF with relevance vector machines (RVMs) which uses Bayesian inference to obtain sparse solutions for regression and classification. Utilizing training data, the RVM regression was performed to find representative aging curves which were used to develop relevant decay parameters.

These decay parameters then influence the model's internal cell parameters like R_{CT} or R_E.

- *Sequential Monte Carlo (SMC) method.* Another PF technique implemented a recursive Bayesian filter using Monte Carlo (MC) simulations and is known as a SMC method. The state and measurement equations that describe the battery model include the decay parameters, the internal battery parameters (derived from measured data), and the potential system noise. This approach accommodates for the sources of uncertainty in feature extraction, regression modeling, and measurement. The PF framework is triggered by a diagnostic routine. The algorithm incorporates the model parameter as an additional component of the state vector and performs parameter identification in parallel with state estimation. Predicted values of the internal battery model parameters were used to calculate expected battery capacities. The current capacity estimate was used to compute the SOC while the future predictions are compared against end of life thresholds to derive RUL estimates. In the authors review the error in RUL improved as additional online (or real time) data was taken, showing an end of life error of 5.9 weeks compared to actual at the week 32 point and an end of life error of 2.6 weeks compared to actual at week 48.

The approaches outlined by the authors provided potential solutions to predicted battery end of life. Since aging can be effected by several outside influences (i.e., temperature, operational profile, etc.) a robust set of training data seems to be required. The training and testing/validation data used in this effort may have been too similar to accurately assess the approaches and it would be recommended that additional independent test data be used for additional validation. It should also be noted that performing periodic EIS evaluations on fielded systems is also difficult due to the type of equipment and test profiles required.

Chen and Rincon-Mora [23] also showed the development of an electrical equivalent model capable of predicting both the operational run time and the cell's current and voltage characteristics. A resistor and capacitor, RC, network simulates the current voltage response

based on the applied load. In order to correlate the SOC or available capacity to the OCV of the cell, a voltage-controlled voltage source was used.

While the authors were able to develop a workable model and show positive results as compared to an actual cell, the validation efforts did not include some of the critical parameters that are needed for battery modeling. The validation effort failed to evaluate the impacts of self discharge and cycle life to cell performance. It is unclear what the model's accuracy would have been after comparing the results to long term cycling. While the model does take into account dynamic loading, it did not account for the impact of environment on accelerating battery aging and thus impacting performance.

Authors [25] interested in determining battery SOH and RUL for electrical systems also utilized discharge curves to correlate identified indicators to the cell's expected capacity. In their effort they modeled the discharge curves of a lead acid cell, defining how the voltage decayed when a constant current load was applied. The voltage decay was then modeled by an equation that represented the two main sections of the voltage curve, a linear voltage interval followed by a non-linear voltage period. This analysis was conducted multiple times as the cells were aged, allowing the authors to evaluate and correlate the change in the discharge curves, capacity loss, and cell age.

The authors accumulated 51 cycles on a cell and evaluated 11 discharge curves within that time frame. As the cell aged, the performance (cell capacity) declined and was reflected as a change within the linear and non-linear portions of the discharge curve. This data was used to define a linear relationship to the change in the parameters of the model to the cycle age. Based on this relationship, cycle age could be projected and the model could then be used to predict the future voltage curve given a specific cycle. Capacity is then calculated based on the length (or time) of the curve as it decays to a set voltage cutoff. This approach may have some issues for implementation as it did not take into account any impacts on temperature or cycling variation in predicting cell capacity, both of which can effect the cells age and performance.

While these model approaches have been used to simulate and predict battery performance, other techniques utilizing various machine learning methods have also been used for battery health prognostics. Several authors have utilized entropy as the bases for forecasting battery health. One paper [26] utilized sample entropy (SampEn) and the estimated SOH to train SVM and RVM in order to determine a cell/battery decay rates. SampEn was used to monitor the variation within battery performance since an aging or degrading battery will have a higher variation in cell voltages at the end of the discharge, causing increased entropy values. The SOH is calculated by dividing the nominal capacity at present time by the nominal capacity at some initial time. The SampEn was used as the data input while the SOH is used as the target vector. The SampEn and the SOH were used to train the Support SVM and RVM, whose results were then compared. In the paper RVM outperformed SVM in predicting SOH in the experimentation.

The research conducted in this paper [26] was limited in several ways. First, discharge curves will vary depending on several factors such as battery chemistry, discharge current rate, depth of discharge, and environment (i.e., temperature) which would alter the SampEn value and skew results. Also, battery monitoring accuracy in the field may not provide precise enough data to calculate the SampEn value to the degree required and thus there may be a need for specific discharge routines to compare the information correctly. While additional work may be required, the SampEn approach could be used to identify other battery markers like shorted cells or undercharge conditions that can be found in battery systems.

SampEn was also reviewed by other researchers [27] in an effort to develop a reliable way of determining an accurate SOC and SOH in electric vehicle applications. The SOH and SOC for electric vehicles is important to ensure proper utilization and safety due to the use of lithium-ion batteries. Determining this information through direct testing like a capacity test is often time-consuming and sometimes impossible to conduct depending on the platform or application. The additional cycling to achieve the information can also reduce

the battery life. Therefore the authors felt that it was important to develop an accurate and robust capacity estimator for a rapid and reliable battery health management.

The authors proposed an enhanced sample entropy based capacity estimator for a lithium-ion cells. They defined sample entropy as a process to quantify the regularity and complexity of a time series data set. The proposed approach determines the sample entropy based on the measured cell voltage while being subjected to a developed discharge profile described as a hybrid pulse power characterization (HPPC). This calculated entropy is used as the input to the estimator. The sample entropy and the capacity of a reference cell at three different temperatures are adopted to train the estimator by non-linear least-squares optimization.

The capacity estimator is first offline calibrated based on the representative aging data sets of a reference battery. The reference battery is tested at different temperatures so that the associated sample entropy would include the variation in capacity. The correspondence between the capacity and sample entropy of the reference cell at each temperature is modeled using a third-degree polynomial function optimized by the non-linear least-squares numerical optimization algorithm. When defined by the representative battery, the estimator is deployed to other systems.

The approach attempts to account for the variations in temperature, but the approach taken does not seem to combine the results adequately. The error in results differ based on the temperature that the cell is operating at. This means the system accuracy may be difficult to determine in a real world application where the temperature is not consistent across the battery/cell's life. In addition, the specific profile used to develop the data set, the HPPC profile, requires special equipment to monitor the individual cell's response within the battery. Not all fielded BMS will be able to accomplish this.

Research in [28] continued this work by attempting to derive SOH through a data driven SOH forecasting model based upon a combination of sample entropy and Bayesian reasoning. This approach was refereed to as a spares Bayesian predictive model (SBPM) and was intended to capture the correlation between the capacity loss and sample entropy. The authors

performed testing on multiple lithium-ion cells against different cycle (charge/discharge) profiles and temperature environments. The sample entropy values of the reference cells under these various profiles were correlated to their respective ages. After obtaining the input target relationships from the reference cells, the SBPM approach was used to learn the underlying mapping mechanism, giving rise to the SOH estimation model. The key focus of the SBPM was on reconstruction of the static non-linear function between the capacity loss and the sample entropy. The authors identified three original contributions.

- The Bayesian scheme in a univariate or one variable form was compared with the prior polynomial model at three different temperatures.
- A multivariate Bayesian SOH estimation model was developed to analytically integrate temperature effects and a comparison with the SVM scheme was made.
- The prediction of RUL for lithium-ion batteries was performed via a combination of SBPM and a bootstrap sampling concepts.

The authors use of training data that encompassed different temperature and operational profiles provides an improved model that was more reflective of real world applications, which should provide improved accuracy once deployed. The authors also used data from different cells during their model testing to strengthen the validation of their approach. It would be recommended to validate or test the model on different operating profiles which can effect cell age and performance differently to ensure the model's robustness.

Research [29] was conducted on approximate entropy (ApEn) to estimate the SOH for lead acid batteries which are used in in multiple applications ranging from back up power for telecommunication equipment to energy buffering for wind farms. The proposed ApEn method is a statistical approach that requires an adequate amount of data for analysis. The method is used to quantify differences in the regularity of signals, with the greater the irregularity reflecting larger ApEn values. The approach looked at the end of discharge voltage of individual cells within a lead acid battery containing multiple cells that were

arranged in series electrically. Aging or degraded cells will cause other cells within the same series connection to supply more energy which leads to a decrease in overall battery capacity. In addition, the voltage of aging or degraded cells may fall below the vendor specified end of discharge voltage level which leads to an over-discharge condition. The voltage variation of an aging or degraded cell during a discharge will result in a higher ApEn value.

The authors' approach provided a straight forward methodology for identifying and determining the aging and degradation of cells within a lead acid battery. However, the described approach and experimentation seemed limited. The approach did not account for expected variations in operations that may cause errors. For example, discharge current and depth of discharge will change the discharge curves of the cells and thus make the correlation more difficult. In addition, some of the assumptions made on how cells connected in long series configurations are not accurate in all applications. This might lead to errors in the SOH or issues in applying the approach to all scenarios. Additional testing with a larger range of data may improve the system.

Another approach evaluated by several researches was the use of SVM and RVM in determining battery performance. In research conducted by Hansen and Wang [30], the authors developed an SVM for SOC estimation by choosing and processing training data, finding the optimal SVM parameters, and then selecting and processing the testing data. Training data for this paper was taken from cell testing of a lithium polymer cell that was designed for use in electrical vehicles. Processing consisted of scaling the data so that all input vector elements were in the range of 0.0 to -1.0. The training data consisted of a four-element vector which included current, voltage, the cell SOC and the change in voltage during the final part of the discharge. The training data was then used to find the optimal parameters for the SVM. A second degree polynomial kernel function, $K(a, b)$, was derived where a and b were vectors and s a linear factor and r a constant. SVM validation was conducted utilizing data from a simple SOC test followed by a dynamic SOC test based

on a profile from a hybrid electric vehicle. The proposed second degree polynomial kernel function was as follows:

$$K(a,b) = s \cdot (a \cdot b)^2 - r \tag{2.2}$$

The authors' approach provided a method to determine battery SOC utilizing an inexpensive micro-controller. However, the described SVM does not take into account several important factors that influence SOC like temperature and open circuit time which was identified as an area needing additional work by the authors. The training data used also seemed very limited compared to the wide range of possible discharge loads that the battery may see, which could cause greater errors in unique situations. The SVM has only been trained for one battery chemistry and thus may have larger errors with other battery chemistry types.

Another paper [31] utilized support vector machine for regression (SVR) as an automated learning tool with a different focus to predict the SOC of a lithium-ion cell. Their goal was to develop a method to determine battery SOC, which is an important parameter when using battery systems. The SOC provides the user/system with information on the amount of energy left in a battery compared with the energy it had when it was fully charged. This gives the user an indication of how much operational time is still available with the battery before a recharge is required.

In order to utilize SVM to solve a regression problem for data that is not linearly separable, the authors used a radial basis function (RBF) kernel. The authors were required to select the correct parameters that could then be expected to map the non-linearly separable data into a feature space where it would then be linearly separable. The SVM techniques were strongly dependent on the SVM hyper-parameters (C,δ ,γ); the regularization factor C, the hyper-parameter δ that defines the SVM type regression, and γ that represents the kernel parameter when a RBF is chosen. In this paper, each combination of hyper-parameter choices were checked using cross-validation, and the parameters with the best cross-validation accuracy were selected. Battery voltage, temperature, and current measurements were se-

lected as the input variables to the SVM model. Model training was based on laboratory experimentation on a specific cell type. A 10-fold cross validation was used to predict the fit of a model to a hypothetical validation set.

The authors approach was based on both steady state and dynamic discharge of a lithium-ion cell. However, their validation of the model and predictive approach was based on data from the same profile. Testing the approach with a different dynamic discharge profile would have shown its robustness to real operational conditions. It is unclear if the SOC prediction would work for an alternative discharge profile, or if the system has been trained for just one specific set of discharge curves.

The researchers conducted follow-on work [32] with SVR to predict the SOC of a lithium-ion cell utilizing cell voltage, current, and temperature. Sample cells were cycled with various discharge rates to obtain the training data with another set of discharge rates used to collect the test data. The voltage, current, and temperature collected during the cycling for the training data were used as inputs to the SVR model. Again, a 10-fold cross validation algorithm was used to guarantee the predictive ability of the SVR model. The cross-validation was used to predict the fit of a model to a hypothetical validation set when an explicit validation set was not available. In doing this the authors were able to find the combination of the hyper-parameters with the best performance.

However, the discussed approach did not seem to take into account all the potential variables that can impact the state of charge. Temperature can impact the electrochemical reactions within the cells and cause inefficiencies in charging and discharging. While the approach discussed does utilize temperature as an input parameter there was little discussion on how that training data or the algorithm would account for the temperature variation.

Other researchers [33] also proposed a data driven approach to battery diagnostic and health prognostics utilizing SVM. Their approach utilizes a data processing method by generating the input and output SVM vectors from measurements taken during cell discharge

testing. These methods was expected to take into account the environmental and dynamic loads seen in real operations.

The authors began by generating training data by cycling cells under a dynamic load profiles with varying temperatures. Their goal was to simulate real world operations as closely as possible. Once the data was collected, the authors processed the data by scaling it such that the input vector elements were within the range of -1.0 and 1.0. They then reduced the dimensionality of the input vector using Fisher ratio. The Fisher ratio provided information on if a particular attribute affects the regression result or not.

The authors suggested the use of "load collectives" as a new method for obtaining the training data. The authors described load collectives as a process where actual operational data is collected from a fielded unit and the frequency of a certain combination of two signal (e.g., current, state of charge, and temperature) is counted. This type of load cycle counting of two signals is also refereed to as two parameter instantaneous value (dwell time) counting. The authors also discussed and utilized another type of load counting called rain flow counting. During the training process, the SVR tries to establish a relationship between the load the battery is subjected too and a corresponding capacity decay. Utilizing these approaches allows for the use of a linear SVR kernel. The parameters of the SVR are selected using the cross-validation method.

To determine the SOH and RUL estimations the gathered data was utilized as the input vector together with the nominal value of the battery capacity. The SVR is able to estimate the actual capacity after it is has been cycled over a set time frame and then repeated over time. A future capacity, and thus SOH, can be determined by using the historical data to determine a mean load value over operational time. Other options could be to use the last several discharge cycles to project forward or select several of the deepest discharges to provide a worst case scenario.

The authors approach provided a unique methodology to determining the SOH and RUL. The approach attempts to account for the impact that the cycling and temperature variation

can have on battery and cell performance. The drawback that was identified was that the approach will require a significant amount of training data to account for the full operational scenario range that the battery could experience. If fielded systems are not equipped with the capability to monitor, record and store cell/battery performance data, then this could be difficult to obtain without significant testing.

SVR was used to determine battery SOC by other researchers [34]. Here the authors were working to obtain accurate SOC to support system operations and maximize battery life. The authors utilized SVR to find the relationship between the defined inputs (voltage and current) to a single valued output target (SOC). The authors compared both the grid search (GS) and the particle swarm optimization (PSO) methods in finding the optimal hyper parameters that are used for the objective function. These parameters are used in the objectives function to determine the best fitting curve to estimate the SOC. Training data generated from a simulated battery was used to develop the parameters. The testing data was taken from a completely different simulation on a battery with a randomly selected test load. This data was used to compare the developed projected SOC derived using the GS and PSO methods.

The authors utilized SVR to map battery parameters to the SOC of the battery. In the study they assumed that the temperature of the battery remained within a specific range. This is an issue as it should be expected that the temperature of the environment will change given the real world conditions and that the battery itself will warm and cool during cycling due to the internal exothermic electrochemical reactions.

The authors did not use actual battery performance data, but simulated data from SIMULINK. Utilizing simulated data does not provided the expected noise associated with manufacturing variation and the surrounding environment. It should not be expected that this would provide an adequate representation of an actual battery system. This approach would also fail to account for cell/battery aging over time.

Given the use of SVM and RVM for battery applications, one set of researchers [35] conducted a comparison study to examine the results of the two methods based the same data sets. The authors felt that it was necessary to examine and compare performance through an uncertainty analysis. Performance variation was evaluated by inducing small changes in the input parameters, measuring the output, and comparing the mean, standard deviation, skewness, and kurtosis of the results. The authors used a relative mean square error (RMSE) in order to compare the SVM and RVM regression approaches.

The authors began by providing a general overview of SVM and RVM. SVM is used for both classification and regression analysis. It can develop complicated decision or response functions while using a small number of sample points. SVM will construct an optimal linear decision function to separate each set of training points. For linear decision functions the approach attempts to maximize the margin between two parallel hyperplanes which separate the data into two classes. The hyperplanes were determined by solving a quadratic programming optimization problem with quadratic object function and linear constraints. For regression analysis SVM is similar to the classification approach discussed. However, to address non-linear issues, kernel functions were used for either classification or regression approaches. A kernel function shifts the data to a new hyperplane where linear separation can be found.

The RVM approach utilized a Bayesian probabilistic framework in developing the model. RVM generally utilizes fewer kernel functions than SVM. The probabilistic formulation was formed through the training data set and includes an additional noise factor. For the RVM classification, the parameter weights cannot be analytically derived and a Laplace based method was used to determine them.

The authors determined through the SVM and RVM comparison that RVM performed best in their evaluation for deterministic context and that SVM performed better for uncertainty analysis when the sample size was increasing. However, RVM provided an estimation of the data set noise in order to define the potential uncertainty of the decision function.

Utilizing an example data set the authors were able to show the results in performance between the two approaches and provide observations on the strengths and weakness. Additional detail on the data sets (e.g., sample sizes) would have provided more clarity on when it might be more appropriate to use SVM versus RVM. Their use of only one application type for the analysis limits the reliability of the authors' conclusion on the performance comparison. It would be recommended that similar data sets from different system be utilized to further validate the paper's conclusion.

The lithium-ion cells SOH was researched [36] using what was referred to as a particle swarm optimization - least square support vector regression (PSO-LSSVR) approach. The authors developed a model using an OCV model combined with a Thevenin equivalent circuit model to define the cell parameters that related to the battery state. A joint extended Kalman filter-recursive-least squares (EKF-RLS) was then used to determine the battery SOC with the model parameters and the OCV. The PSO-LSSVR approach was then used to provide an accurate SOH. The authors tested ten lithium-ion iron phosphate cells to acquire both training data and to validate the processes. They compared the results to a more common neural network approach to show an increase in accuracy with the PSO-LSSVR approach.

This approach proposed some novel input on non-traditional features to consider for battery health prognostics, specifically in looking at the charge curves associated with the cells. It also introduced the concept of the particle swarm optimization algorithm. However, upon review of the paper, there was some concern about the approach. First, the method appears to focus on electric vehicle applications or similar systems/platforms where the battery is starting from a resting or open circuit point. For battery systems that are maintained on a float charge (e.g., power grid applications) the defined model would need to be modified. Another parameter used for the analysis are derived from the recharge curve of the battery. This can become an issue as the recharge curve can vary depending on the depth of discharge

and the available currents for the recharge. Specific state of health experimentation may be required to obtain the needed performance parameters from a fielded battery.

The approach also appears to be focused on lithium-ion technologies and other battery chemistries would need different parameters to be considered for the process to be as effective. The SOC used for the SOH estimation had a high error when SOC values were above 90% and less than 20%. For some application the needs to maintain 100% SOC is important, so the inability to accurately show that the battery system is at or near 100% would be an issue in those applications. The approach also did not include parameters associated with the environment (e.g., Temperature) which can have an impact on both SOC and acceleration of performance degradation.

Researchers [37] discussed the use of RVM to predict RUL and improve operational efficiency of a lithium-ion cells. RVM utilizes a Bayesian treatment of a SVM which allows for a probabilistic prediction that increase flexibility with kernel functions. The authors obtained cell performance data from publicly available battery lifecycle testing data from an online and open sources NASA repository. The authors utilized an iterative expectation maximization (EM) algorithm for RVM training as they felt that large amounts of training data could lead to issues in deriving the relevance vectors.

The authors used capacity degradation data from the NASA open source data base as inputs and the RVM was used for regression analysis. The defined relevance vectors were used by the EM algorithm to generate a capacity prediction for a future cycle. The new predictive information was put together with the original relevance vectors to form a new training set for the RVM model.

The authors evaluations attempted to take into account variation in performance that can occur due to environmental or operational variations. The training data that was used included variation in temperature and depth of discharges. This approach improves the potential for better accuracy in fielded systems. However, additional validation testing with varying discharge profiles may be needed to ensure the impact of non-static cycle profiles.

A different group of researchers [38] evaluated the potential of combining entropy and a RVM in order to predict a battery's SOC, SOH, and State of Life (SOL). The authors proposed a data driven approach for SOH and remaining life predictions using mean entropy and RVM. The mean entropy is used to select the input variables for the RVM prediction model. RVM adopts kernel functions to project the input variables into high-dimensional feature space, so that the latent information can be extracted. RVM provides the probabilistic interpretation of its output which provides a better prediction given dynamic changes in the cycling profile and environment.

The authors collected experimental data from two cells that were cycled to failure. Each cell was operated at two different cycle profiles at ambient temperature. Cell capacities were determined through periodic full depth of discharge cycles where coulomb counting was used. The SOH was calculated using the ratio between the performed capacity test and the initial capacity.

The data collected from the experiment was processed to reduce the noise and to extract trend information which was accomplished with a Wavelet Denoising method. The denoising method was described as a "soft threshold denoising based on db5 Wavelet at decomposition level six (6) with the mixed threshold selection rule", which could be implemented in MATLAB.

An embedding dimension is required for the time series prediction. The embedding dimension was selected based on the representation structure of the measured SOH data. The algorithm for optimal embedding dimension selection began by scanning through the time series SOH data with different embedding dimensions. The mean entropy of all the data set having a specific embedding dimension was estimated and plotted in relation to the size of the data set. The mean entropy increased significantly initially and then proceeded to flatten out as an optimal embedding dimension was achieved.

The RVM model was trained by using historical data, which resulted in weights and bias. The developed RVM model was then used to make a specified number of predictions and

determine the remaining life of the battery. In general, the remaining life was used as the relevant metric for determining SOL.

Another approach researched [39] utilized a relevance vector machine-particle filter (RVM-PF) in an effort to accurately predict the battery capacity for the current and future cycles while not requiring significant computation. The authors used a Rao-Blackwellized particle filter (RBPF) in order to reduce uncertainty in the results. This RBPF approach divided the state space into deterministic and probabilistic parts and analytically solved for the former while using PFs for the latter, thus reducing the variance in the state estimate.

The approach began with an "offline" training based on historical data where the cell's operation was expressed in the form of a structural and functional model. EIS testing was conducted to identify features that could be used to estimate the internal parameters which were then used to develop an equivalent circuit battery model. For this paper the features that the authors selected were electrolyte resistance R_E and charge transfer resistance R_{CT} which the authors believed showed significant change due to the aging processes observed.

RVM regression was performed on data collected from a group of cells which were cycled over a long period of time to find representative aging curves. The use of probabilistic kernels in RVM helped to reject the effects of outliers and varying number of data points at different time steps, which could bias conventional model-fitting methods like least-square-based.

The models developed "offline" were fed into an "online" (real time battery monitoring) PF process. The input data was used to estimate the battery SOC and SOH. The PF incorporates the aging parameters of R_E and R_{CT} along with the battery capacity, at a standard discharge current as components into a state vector. The current capacity estimate was used to compute the SOC, whereas the future predictions were compared against end-of-life (EOL) thresholds to estimate SOL.

The RUL was used as the relevant metric for determining SOL. This was derived by projecting out the estimated capacity of each particle from the time of prediction into the future until they hit the predetermined EOL threshold. Both PF and RBPF prognostic frameworks

were implemented, with the latter having an estimated capacity as the deterministic state variable and the former assigning an additive zero-mean Gaussian noise of variance 0.001 to it. The battery's RUL probability density function was computed by fitting a mixture of Gaussian to the RUL values generated by the particle population.

The authors' approach provided a solid method to developing a performance based model that could be used for predicting cell aging or degradation. The approach appeared to also include a path that could account for performance variation due to temperature and various cycling profiles which can effect the aging rate. However, the parameters R_E and R_{CT} needed to predict the battery health may not be easily determined during normal battery operations. The EIS utilizes special equipment and charge/discharge profiles. To be effective, the parameters need to be determined based on normal battery operations or an easily executable charge/discharge profile. This may be difficult to apply in some applications.

A different paper [40] proposed to develop an incremental online learning strategy for RVM and an online dynamic RUL estimation framework. The authors proposed an incremental optimized relevance vector machine (IP-RVM) algorithm to support online lithium-ion battery RUL predictions. In the IP-RVM method, the algorithm only utilizes the relevance vectors and the new on-line data sample as the training data set. The authors began by choosing initial training data sets that were used to initialize the RVM model parameters and establish a prediction error bound (PEB). The RVM training was then done offline with the selected initial training data set. Sensor data from an online system was compared to the developed RVM model and the prediction error to the derived PEB. If the prediction error was larger than the established PEB, the new data was included as new training data and the RVM model was updated, otherwise the RVM was unchanged.

The authors' approach provided a potential opportunity to incorporate online data into the training set to update the RVM model in real time. This could be applied to increase the accuracy of the model when subjected to varying load profile and temperatures. However, the impact of varying load profiles or environments was not discussed. The approach still

requires training data that represents the operational profile in order to obtain an initial RVM model that can be useful in the field. This means that upfront work in developing the offline model is still required.

Research [41, 42] on battery aging and the estimation of RUL was a topic addressed as an important part of prognostics and health management (PHM) and reliability engineering. The authors proposed the use of non-linear transformation and optimization of the extracted features to enhance the correlation between the selected parameters and battery capacity. They also presented an optimized RVM algorithm to improve the accuracy and stability of RUL estimation. Their approach began with feature selection based on operational data from cycled cells. Feature selection was based on monitoring voltage, current, and cycle count (age) in each charge/discharge cycles under a constant voltage and restricted current operating profile test. They defined an equal time interval in which to normalize the data under review in order to extract the desired parameters. After the raw feature data was collected a Box-Cox transformation was applied to improve the linearity of the features and battery capacity (SOH). Finally, the optimized feature set was utilized in RUL estimation.

The authors' approach provided a sound way to identify unique features that could be used to estimate battery RUL. However, the limitation on dynamic cycling was identified as a concern that might be resolved by performing periodic maintenance cycling that provides comparable discharge performance characteristics. Operations between these points could be as dynamic as needed but still provide stable points for comparison.

A potential issue of the reviewed paper was that the evaluation data of the test cells all showed good results except for one. The authors were unable to explain why this cell's performance did not conform with the prediction model. Given this, there may be additional conditions that the selected features were unable to correlate too. This suggests that additional evaluation is needed.

The use of PF was explored by researches [21] in an effort to estimate the SOH and predicting the RUL of an energy storage devices. The authors presented an approach to es-

timating SOH and predicting RUL by utilizing particle filters while also taking into account the sudden regeneration phenomena that has been seen during lithium-ion battery operations. The authors utilized a state-space model that provides an empirical representation of the system that also includes the ability to incorporate the impact of a sudden regeneration phenomena. The model takes the output from a real time PF-based detection system which determines if a regeneration has taken place or not and modifies the predicted RUL accordingly. The PF-based detection system determines the state of the cell (regeneration or not) based on the position of the particles associated to the empirical a priori state distribution.

The authors provided a novel approach to including the influence of the sudden regeneration phenomenon. However, the impact of this issue seems small in comparison to other influences (e.g., temperature) that can effect cell/battery capacity and RUL. Based on the data provided in the paper, the overall decline curve for the cell still appears to be very constant even with the regeneration phenomenon. The approach also required a capacity test for the estimation of the SOH and RUL. The ideal desire would be to minimize capacity tests as the deeper discharge increases cell aging. The inclusion of temperature and other cycle profiles also have an impact on SOH and RUL. Those effects need to be taken into account for a battery that is deployed on various systems.

Another approach [43] that was used for battery health prognostics utilized a Verhulst model that would account for the capacity degradation of a lithium-ion cell. The model was based on data collected from constant current discharge cycling of cells to measure capacity decline over time. The Verhulst model was selected as the authors determined that it described the capacity decline trends of a cell better than other models such as a SVM or Grey Model (GM) based on experimental data.

The authors proposed an improved Verhulst model for implementation. The Verhulst model was used to fit the training data in order to obtain the model parameters. In an effort to improve the fitting precision of the model the authors utilized a PSO combined with the euclidean distancing to define the upper and lower bands of the Verhulst model

parameters. In this approach the fitting parameters are estimated by least square method. The authors proposed a method to improve the PSO algorithm by utilizing an adaptive weight method. The goal was to optimize the PSO performance by utilizing a bigger weight coefficient early within the searching, but reduce the value later in the searching to speed up the convergence. The authors noted that the fitness function was a primary factor that influenced the performance of the PSO algorithm.

The authors' approach provided increased accuracy and precision based on their comparison of the more traditional approaches they selected to review. The modifications to the Verhulst model and the use of PSO seemed to be key to these improvements. However, there may be some concern regarding the data used in the development and validation. The same data sets seemed to be utilized interchangeably for training and testing. This may lead to over training of the system. More validation testing with independent data sets would be recommended. In addition, the models do not appear to take into account external impacts to battery aging (i.e., temperature) and validation of the process at various temperature should be done.

Other researchers [44] have incorporated the use of PF in determining battery health as well. These authors established a mathematical relationship between battery age (number of cycles) and the capacity (performance) while then determining RUL through a PF algorithm. Cycling performance data from a lithium-ion cell was collected and a specific predicted starting point within the data set was selected. The data prior to this selected point was used as the training data parameters to develop the model while the follow-on information was used for validation.

The PF algorithm was used to track the battery capacity prior to the predictive starting point which then updates the parameters associated with the lithium-ion cell degradation. For new data obtained after the prediction start point, each particle was iterated and then it was determined whether or not their end points where reached. If all points were not reached, the unimplemented particles are extrapolated until all particles reached an end.

The authors calculated the RUL of each particle and then performed statistical analysis to obtain a RUL distribution histogram of all the particles reaching the threshold and ending the iteration.

The authors identified some specific contributions based on their work. They showed that the prediction method based on the PF requires less prior information for individual cells and was therefore more convenient to implement. The algorithm can adapt to the degradation difference between different cells, and at the same time provided a relatively accurate residual life prediction value of individual cell capacity degradation. As more prediction data becomes available with the passage of time, the prediction error gradually decreases. Compared with the point estimation, the interval estimation of the probability density has a greater reference value for a maintenance strategy because the uncertainty factor always exists, and the probability density function provides a good potential solution.

The development of the predictive model appeared to be based on a single data set, utilizing earlier performance for model and algorithm development. The follow-on data set was then used validate the method. It did not appear that other data sets were used which would limit the model's ability to predict RUL in other operational scenarios. Also, the distribution function appears to be limited in scope and does not account for the impact of temperature or different operating profiles.

A deterministic Bayesian approach [45] was evaluated for estimating a battery's SOH as well. Training data was used to define a Gaussian prior distribution that represented the historical performance degradation of the cell performance. As real time data was collected it was combined with a "likelihood" capacity that was also modeled with a Gaussian distribution. With this information a posterior distribution function was found that lead into the predictive degradation. From this information the RUL was derived.

The authors provided an approach that could be used for SOH estimation. However, the accuracy of the approach seems fairly large early in the cell's life. It appeared that a significant amount of training data is required to develop the distribution curves and ensure

adequate SOH accuracy. The approach does not take into account the impact that the environment might have on cell/battery performance. The validation of the approach was limited due to the sample cells failing due to testing issues that lead to unusable data. The additional data may have provided better clarification into the application of the algorithm.

Other research [46] combined an electrical model with an extended Kalman filter (EKF). The authors initially developed an electrical model that could be used to represent a cell's main electrochemical characteristics. The EKF was utilized for the observation of the parameters of a lithium-ion lumped model. The authors developed the cell model by first collecting cell parameter data through EIS testing. The data collected provided information on the internal impedance and electrochemical diffusion rates. These were used to create a lumped model that represented the internal cell diffusion phenomenon. The method consisted of developing an analytically model of Warburg impedance in a series expansion of functions and combining this with an equivalent impedance composed of capacitance and resistance.

The authors than used an EKF to update the battery model. First, the internal cell resistance was determined by evaluating the cell through a constant current step charge and measuring the corresponding voltage variation. Once the value of the resistance was determined, the EKF was used to identify the usable parameters. The other model parameters were determined by the equivalent circuit relationship and the cell voltage over time.

This approach attempted to take into account important elements of battery aging (i.e., temperature and cycling) which was promising. However, the effort did not address the development/calculation of the SOC or SOH of the cells. The effort provided a way to update the impedance values of the cell over time but did not show how this information could then be applied to predict the cell's life. Additional research would be needed to take this information and apply it in a way that could support prognostic efforts.

In [47] the authors utilize a Walbot radio sensor with machine learning algorithms to determine cell/battery voltage via a contact-less monitoring. The Walbot sensor used in the research is a three-dimensional radio-frequency sensor typically used to support such efforts

as in-wall 3D imaging. The sensor was applied to a lithium-ion battery (LIB) in order to monitor the internal electrochemical changes that occur within the cells during charge or discharge with a focus on electrolyte states changes. Using the training data that linked electrolyte stage change to LIB voltages, the investigators used PCA, LDA, and stochastic gradient descent (SGD) machine learning algorithms in an effort to develop a classification approach to determine cell voltage.

The author compared the three approaches using initial training data to develop the classification modes. For the PCA method, PCA was used to support a reduction in dimensions while still trying to maintain the variance in the original data set. The developed PCs that are then used as input to a k-neighbor linear classifier in order to assign future sample data. The LDA approach is used to support dimensional reduction, but also creates a hyperplane to maximize separation between classes and minimize within class distance. In this way the assignment of new sample data can be made to the correct class with higher expected accuracy. The SGD method is a linear classifier where gradient of the voltage loss is estimated based on each sample received. The model is updated based on the new data, decreasing the learning rate.

The three machine learning approaches were compared through experiment via cycling LIB and comparing the data to a real-time voltage monitor connected to the cells. The results showed that the PCA and LDA approaches had 93% and 94% accuracy while the SGD approach was only accurate up to 31%.

While the PCA and LDA approaches showed high accuracy in determining cell voltage, the approach would most likely not be viable in fielded applications. Voltage monitoring, specifically for lithium batteries, is key to safety as overcharging cells can lead to catastrophic events. Direct monitoring of the cell/battery voltage would be preferred to ensure a mitigation can be accurately executed. The machine learning approaches do not appear to account for electrolyte or voltage changes that may be effected through temperature variation or cell aging which will change the performance of the cell.

Other research [48] attempted to use machine learning algorithms coupled with incremental capacity analysis in order to determine SOH and RUL under different temperatures and SOC. Incremental capacity analysis utilizes specific charge and discharge regimens to collect data that is transformed into $dQ/dV - V$ curves where dQ is the capacity difference and dV is the relates incremental change in voltage. The dQ/dV relationship is then plotted against the mean voltage to develop the curves. Curve features are related to transformation of a lithium cell's anode and cathode material. These features are normalized by standardizing and re-scaling the data around a mean of 0 and a standard deviation of 1 in order to apply to the machine learning algorithms.

Here the authors attempted to use PCA and LDA algorithms to classify the batteries in terms of temperature and SOC. The PCA approach was used to develop PCs, with the first two ($PC1$ and $PC2$) used in an effort to show clustering of the batteries correlated to the same temperature and SOC. However, this approach was not successfully as the first two PCs only accounted for 48% of the variance within the data set. The authors then chose to apply an LDA approach to the identified features and data set. Using the first two derived linear discriminates (LD), the LDA algorithm was able to provide a 89% accuracy for the temperature classification and 67% SOC accuracy. The analysis showed that temperature was the feature of greatest variance within the data set.

The authors work showed that the LDA approach could be used to support classification of batteries by temperature and SOC using features collected from the incremental capacity analysis. However, temperature classification was the only group with an accuracy high enough to potentially be useful. Follow on work would need to couple this approach and data with other research in order for it to provide useful information in the performance and safety management of a battery system by end users.

2.5 Maintenance Strategy Optimization

Queuing systems [49, 50] have been used to study a wide range of service systems. It is a branch of applied probability theory that is used to represent and predict operations within a system were congestion may occur. The queuing system can be described as containing a population of units or customers that are supported by a service facility. When the demand for service by the units exceeds the capabilities of the service facility(s), then a queue or waiting line develops. Multiple queuing systems can be described in order to simulate various real world systems such as production lines, service center, and computer networks by varying key parameters (e.g, arrival time, service time, service channels).

In [51] the authors utilized a $M/M/s/N$ queuing model in order to determine the optimal cost and service for plug-in electric vehicles (PEVs) at fast battery chairing stations by determining the right number of charging and waiting spaces. It was assumed that at any given time slot, a PEV could be found in one of four possible states: normal charging state (S_{nch}), fast charging state (S_{fch}), driving state (S_d) and parking state (S_p). The departure and arrival times for PEV in thge mornings and evenings (assuming both private owners and companies using PEVs) were determined to follow a normal distribution of

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.3}$$

which was tied to the expected energy consumption to the battery ($0.159kWh/km$) and the maximum battery capacity $C_b(kWh)$ where $N(\mu, \sigma^2)(\mu = 28.5$ and $\sigma = 14.7)$.

The authors utilized a Monte-Carlo simulation and Markov model to estimate the PEV travel behavior and determine charging demands (arrival rates) for the station(s). A Monte-Carlo simulation determined the PEV usage (private user or commercial group) and the individual vehicle battery capacities through the probability distributions and other defined assumptions. A Markov transition probability matrix was used to determine each PEV vehicle's state at a set incremental time interval during a given day. It was noted that

since some transition between specific state may not occur, the transition probabilities can be explicitly determined for each identified time slot. Given the known PEV probabilities for each time slot and a known PEV population (n) within an area, then the demand for charging PEVs (V) can be determined as follows:

$$V = P_n \times n \times S_{nch}^{t_i} + P_f \times n \times S_{fch}^{t_i}, \tag{2.4}$$

where $P_n \times n \times S_{nch}^{t_i}$ represents the expected normal charging demand and $P_f \times n \times S_{fch}^{t_i}$ represents the fast charging demand.

Queuing theory was then used to optimize the sizing of the fast charing stations. The arrival rate (λ) followed a Poisson distribution with the charing service time following an exponential distribution of $1/\mu$ (i.e., 30 min). Using the discussed Markov chain, the authors were able to determine the probability state of the charges were P_0 represented the probability that the charger was empty and that P_k was the probability of k PEVs at the station. Based on this information the authors were able to then determine the number of cars waiting for chairing, the number of PEVs that leave due to wait time, and the number of vehicles serviced. An analysis could then be conducted based on cost and profit information to determine the correct number and type of the charging stations to maximize profit.

The authors work provided a thorough example of how a queuing model could be used to support a battery system maintenance strategy. The determination of the arrival and departure times for the vehicle could have been better defined as not all the analysis was provided. However, the queuing analysis showed the potential benefit of simulating the environment in an effort to optimize the costs and maximize profit.

In [52] an algorithm was proposed to optimize charging facilities for PEV vehicles. The authors utilized a $M/M/C/S$ queuing optimization approach to model PEV charging facilities in order to right size the number of available charges and queuing spaces. In the presented case study, the facility contains a set number of PEV charging stations and queuing or waiting spaces. If all spaces are filled, new vehicles will leave or reject the facility for

a different location. For these rejected customers the simulation includes a penalty cost. By factoring in the various costs and service profit, the queuing simulation is used to optimize the number of charging stations and waiting spaces.

Th PEV arrival rate (λ) was based on a Poisson process which was confirmed through observational data. The battery charge time (service) was modeled from an exponential distribution with an average recharge duration of $1/\mu$. The facility will utilize multiple charging stations, which is limited by the available total power from the grid, as well as multiple waiting spaces. The facility size (S) is the defined by total number of charging (C) and waiting (W) spaces. Given a random number of vehicles ($N(t)$) at a time t, $\{N(t), t \geq\}$ is a Markovian process with finite state set $N = \{1, 2, ..., S\}$. The steady state probability of vehicles at the station (P_n) and the probability of rejected vehicles (P_S) are used to determine the number of vehicles able to enter the facility (waiting or being charged) as well as the number of vehicles rejected. By changing the number of charging station and waiting areas as well as factoring overall costs and sales, the authors are able to determine the optimal number stations and waiting spaces to maximize profit.

The authors approach provided a straight forward method to optimize a PEV charging facilities to ensure the highest expected profit is reached. The authors utilized experimental data to verify the vehicle arrival rates but made assumptions on the charging duration. The model could be expanded to include various charge time that would correspond to the different battery capacities due to vehicle types and battery ages.

Research [53] proposed an analytical model in order to optimize costs and resource selection for a wind farm maintenance support organization. The analysis utilized a queuing simulation based on a Markov chain to model the maintenance activities while investigating the impact of using alternative types of transportation. The Markov chain simulated the backlog of maintenance activities in support of performing actions on the various wind turbines. The Markov chain consisted of i number of failed turbines where the failure transitions rate was proportional to the number of operating turbines. The service rate was

proportional to the minimum number of turbines being repaired and the number of maintenance teams. Using the model, the authors were able to examine the use of different types of maintenance teams, transportation schemes to reach the turbines, and the application of preventive maintenance in an effort to minimize costs.

The authors provided a detailed analysis of their optimization of wind turbine support teams. The authors were able to clearly show the impact to the maintenance strategy in evaluating the cost and profit impacts in implementing different methods of the maintenance team staging, the number of maintenance teams, and the transportation choices.

In [54] stochastic simulations of queuing networks were utilized to evaluate staffing levels through the life cycle of a project to determine the dynamic right sizing of resources through a project and the likelihood of project success. The authors provided an overview of maintenance project related to the Y2K problem foe a finical software system. Three different models were defined in order to evaluate the staffing levels for project execution and cost; (*i*) a single-queuing model, (*ii*) a two queue system with one node dedicated to problem assessments and the other for all other actions, and (*iii*) a three queue system with one dedicated to assessments, one for technical analysis and the third for enactment and unit testing.

The authors used historical data to develop both training and test data sets for model development. Historical data was used to derive the appropriate arrival and service times by generating histograms of the data and then verifying the shapes through a Kolmogorov-Smirnov and Chi-square goodness of fit tests. Following this analysis the defined arrival time was estimated using the maximum likelihood estimator (MLE). The service time estimation required no specific estimator as it could be determined based on the statistical review of the data set (e.g., mean and standard deviation).

With the established models, the authors were then able to evaluate the different queuing systems as well as the staffing requirements, costs, and the project completion likelihood by a specific date. The simulated results were compared to the actual historical data to identify

the error in likelihood estimation between the different model approaches. The authors were able to demonstrate that the approach was highly capable of estimating the project timeline, that issues such as rework could be included into the model to determine staffing needs, and the project ramp up and shutdown could also be modeled with the appropriate simulated time sequence.

The authors provide a detailed case stated that exemplified the use of queuing theory for maintenance strategy decision making. The authors utilized historical data to develop time sequence changes and define the changing arrival and service times based on project maturity. However, the approach appeared to be developed to address the specific case study and it was unclear how projects with different resources and tasking could be modeled to provide an accurate project schedule. A different approach may be needed to assess future work and/or to monitor ongoing projects more accurately.

Research conducted in [55] modeled a finite capacity queue with a buffer partition in order to reduce end-to-end delay for network signal traffic (e.g., voice over IP, video streaming, etc). In the developed model the external traffic was defined with a compound poisson process (CPP) with traffic transmission times expressed as generalized exponential (GE) intervals. The network followed a fist-come-first-service (FCFS) service plan. In order to reduce queuing lengths which impacts delays and delay jitters, a separate buffer was given to different classes of traffic. The class arrival was determined based on state probability functions and defined constraints. A maximum entropy function was used in order to characterize a universal form of the state probability function.

The authors conducted initial validation of the approach by using video steaming and data packets for a two traffic class system. Here the arrival rates of the two classes differed while the service rate remained consistent. The initial results showed that even with a wide range of input data that the proposed model showed an accuracy similar to others models used for this purpose.

The authors provided a good overview of their approach. The model represented a two class system well with a straight forward implementation approach. However, it remains to be seen how a more complex data set with more than two classes would function. It also assumes available prior information to establish the arrival times and data classes which may not be available.

In [56] the authors utilized queuing theory to determine an optimum number of charges along a trunk road with multiple intersections where PEVs are entering and exiting. The vehicle arrival rate (λ) followed a Poisson process with the charging time (service) represented by a negative exponential distribution for μ. Given c chargers available, the service rate is described as $n\mu(n < c)$ for n vehicle population and the stations occupancy is $\rho = \lambda/(c\mu)$. The balance equation of the queuing system then provides the probability of n vehicles being charged as P_n. This allowed the authors to derive the average wait queue length

$$L_q = \sum_{n=c+1}^{\infty} (n-c)P_n, \tag{2.5}$$

an average queue length

$$L_s = L_q + \frac{\lambda}{\mu}, \tag{2.6}$$

and an average customer waiting time

$$W_q = L_q/\mu. \tag{2.7}$$

This queuing analysis was combined with a model of the expected number of vehicles that may need charging at each intersection of the trunk road. The authors determined the probability (P_k) of electric vehicles requiring a charge at a location (l_i) that corresponds to a intersection (k) along the trunk road. Based on P_k the expected (E_k) number of electric vehicles needing to be charged at an intersection k can be deduced. From this the authors determined the number of facilities needed along the defined trunk road. Utilizing the

queuing analysis already discussed, the authors could define the entire PEV charging system in order to maximize profits.

The authors provided a straight forward implementation of the queuing theory in order to determine the number of charging stations needed per facility. The approach to determining the number of facilities along the location also showed how the overall support strategy could be developed. However, there appeared to be some continuity differences between the probability distributions of the vehicles on the road needing charging and those arrival rates at the charging stations. It was unclear if the differences would cause an impact in a real world application.

Research [57] in revenue optimizations was conducted where different vehicle classes were considered in the optimization of PEV charging stations. The authors used a $M/G/S/K$ queuing system to model a PEV charging facility under different scenarios. The authors evaluated the opportunity to maximize profit under two frameworks consisting of a single-class customer and a multi-class customer scenario. In the single class system one charging stations type is used to support all customers where customers gain rewards by reaching high SoC level and battery degradation, waiting times, blocking events, and other fees were included in the model cost. In the multi-class scenario, the customers are grouped by the battery size, the class demand, and the available charger technologies (shared chargers or class specific chargers).

For single-class analysis the authors defined a battery degradation cost that is represented as a function of the charging power and the charge duration. This is due to the fact that when charging with high charge rates there is a negative impact to the battery cycle life which leads to inefficiencies. Other costs included into the analysis were the wait time and a reward cost. This scenario includes a reward (R) for the customer after the completion of service if they have paid an admission fee p. The queuing system used in the analysis assumed S chargers and r waiting spaces. PEV arrival rates (λ) followed an exponential distribution while the service or charging time was a function of the vehicles SoC and available charging

power. A new customer arriving at the facility when no charging stations are available would be blocked which was represented by blocking probability of P_K. The optimal number of chargers were defined for the system based on generating the highest revenue given the costs and sales as well as the admissions fees for the reward program.

In the multi-class analysis the authors examined the impact of specific chargers available for defined vehicle classes. The different class vehicles represent the variability in SoC, charge duration and vehicle types that could occur in actual deployed systems. The analysis conducted evaluated the impact of all vehicle classes being charged by one charger type or the opportunity to have dedicated chargers per class. The same $M/G/S/K$ queuing model was used for this analysis with varying arrival rates and service rates depending on the vehicle class. Cost differences were also applied based on the single charger type and dedicated vehicle class chargers which impacted the overall revenue. In comparing the two scenario types the authors found that in order to maximize revenue the strategy to be utilized was dependent on the traffic intensity in the area. It was determined that in a low traffic intensity environment a shared charging model was most effective while a dedicated class charger was optimal in high traffic intensity environments.

The authors showed that the application of the queuing theory could be used to define the optimal service strategy for a PEV system. This was beyond the concept of establishing the optimal number of chargers, but also the type of chargers depending on various contributing factors. The authors provided little detail on how the different class arrival and service rates were defined, but the assessment showed that by factoring in these differences an ideal strategy could be identified based on the population's needs.

In [58] queuing theory is used to model the charging demand of plug-in hybrid electric vehicles (PHEV) in order to derive the overall probabilistic power flow (PPF) required to support a charging facility or residential area. The authors choose to illustrate the approach by modeling PHEV demand at an electric vehicle (EV) charging station and within a residential community due the differences in expected usage patterns. In order to model the standalone

EV chairing facility the authors applied a $M/M/c$ queuing model where both the arrival and service times followed an exponential distribution with a maximum service capability c. The EV charging station scenario assumes an infinite vehicle population. The residential community model followed a $M/M/c/k/N_{max}$ queuing system where $c \leq k \leq N_{max}$. Here k represents the number of customers being serviced or in a queue waiting for service while N_{max} represents the total numbers of customers to be serviced. In the residential scenario the charging stations are privately owned by residents which means that there is a finite population to be serviced.

The authors also assumed that there were four PHEV classes, each with different sized batteries and quantities within the population. The population of each vehicle class was considered to be discrete distributions and thus the PHEV vehicle type was randomly selected based in its share within the population. The authors utilized a randomly generated sampling as the PHEV vehicles coming to the charging facility. It was assumed that the charging facility utilized two types of chargers for different power needs and that the charging duration (service) followed a Weibull distribution. The residential charging concepts follow the same charging distribution with chargers distributed to the the various customers. This assumed that each vehicle either had their own charging station or was sharing a station with a small subset. The queuing model was able to simulate the number of vehicles being charged and the peak power flow required to support them.

The authors used queuing theory to determine the estimated power need for a charging facility or residential area by simulating the PHEV charging needs. This was done by deriving the different attributes for the scenarios as well as accounting for varying battery types. However, the differences in usage patterns for a charging facility and a residential community did not appear to be fully explored. It would be expected that the charger types and the peak charge time would be different and may require additional detail to be added to the models.

Researchers in [59] utilized queuing theory to support a comprehensive review of optimizing power usage at an eclectic vehicle charging facilities by utilizing solar panels and a battery storage systems. The authors took a two stage approach with the first utilizing a queuing model ($M1/M2/N$) to determine the optimal number of stations to have at the facility. The authors first utilized an algorithm to determine the population of vehicles based on the constraint of maximum travel distance from one charge point to the next. The vehicle arrival rate is influenced by a spatial-temporal model that defined the expected traffic along the road way. The queuing model aids in simulating the total power demand needs at various times. The model is used to simulate the dynamic conditions of the charging facility and optimize the number of stations to minimize customer weight time.

The authors model the potential contribution from solar panels and a battery storage system in order to minimize total grid power consumption. The solar panels' contribution is dependent upon solar radiation which varies from season to season. Therefore the authors included a seasonal irradiance pattern to the total power generated from the solar panels in order to account for this variation. In addition, since it is expected that peak eclectic vehicle charging demand does not follow maximum solar power generation, the authors factored in a battery storage system to off set the power need at those times. The battery storage system is designed with a capacity able to take the excess charge from the solar radiation that is not directly used in order to apply it during the peak charging times. The combination of the various models allowed the authors to optimize a charging station that allowed for minimal wait time to customer as well as reducing grid power usage to the lowest level by defining the solar panel and battery storage system size. This leads to an overall reduction in power costs during steady state facility operations.

The authors demonstrated the benefit of utilizing queuing theory in order to optimizing an electric vehicle charging facility strategy. The authors determined the number of charging stations to support an expected need while minimizing the customer wait time as well as defining the overall power need. The authors approach then allowed for the combination

of an alternative power sources to reduce grid power usage. The authors included multiple variables that would effect the power demand and the solar power generation to provide a complete analysis.

The authors in [60] proposed a method to identify the optimal location and number of charging stations for an electric vehicle charging facility with the focus of minimizing costs through a $M/M/s_j/N$ queuing model. Unlike many other applications of a queuing model in electric vehicle charging application, the authors assumed that the queuing length would be finite with a max length of N. The authors also chose to vary the charging station quantity per location, to be dependent on expected demand at the location. They used Manhattan, New York as a case study location to demonstrate the method.

The work conducted by the authors focused on the initial construction and infrastructure costs associated with facility location and optimizing charger stations. The associated costs ($C_{station}$) under review included the facility construction costs, equipment installation, and equipment depreciation at a rate of r per year. The number of installed chargers were defined as s_j per charging facility j. The capacity at each facility site was defined as

$$N_j = s_j + [\frac{s_j}{\gamma}], \tag{2.8}$$

where N_j represents the maximum number of vehicles that can be at the location and $\frac{s_j}{\gamma}$ is the maximum queuing length. The vehicles arrival rate (λ) at the charging facility follows a Poisson distribution while the charging service follows a negative exponential distribution with a mean μ. Given a service density ρ of

$$\frac{\rho}{S} = \frac{\lambda}{\mu S}, \tag{2.9}$$

where S represents the number of chargers, the authors are able to determine the queuing length L_{jq} and the associated queuing wait time W_j per charging facility j.

The costs associated with the users were defined as the distance traveled cost and the queuing wait time cost which is used to determine if a candidate location should hold a charging facility. The authors used the distance traveled costs and potential queuing wait cost per potential location j to determine a binary decision variable for build location based on the fact that the electric vehicle can only drive a set distance based on the battery SoC.

The authors used New York State electric vehicle market data from multiple sources in order to test their approach. Using a layout of Manhattan the authors identified locations throughout the city to optimize charging facility location as well as the number of charging stations per location based on the expected populations within those areas. Through optimization and sensitivity analysis the authors were able to define a strategy to minimize total costs.

The authors approach provide a straight forward method of determining the optimal locations and the number of charging stations within a defined area. The authors approach focused on keeping initial infrastructure costs at a minimum by ensuring customers were well serviced. This analysis showed that this approach could provide a strong initial deployment strategy. However, including the operational costs of the various charging facility locations may provide greater details on location acceptability and long-term profitability.

In [61] a multi-class queuing system combined with a charging threshold strategy was applied to an electric vehicle charging facility in order to minimize customer wait time and maximize revenue. Here the authors divide the electric vehicle population into C classes based on battery size. Each class is assumed to have a specific arrival rate (λ_c) that follows a Poisson process. The charge time (T_c) per class is a function of the battery's arrival state of charge ($SoCA_C$) and departure state of charge ($SoCD_C$) when assuming a consistent charge rate. The $SoCA_C$ was defined as a random variable that follows a Cumulative Distribution Function (CDF) of $F_c(X) = P(SoCA_c \leq X)$ which allows for the derivation of the probability distribution function and the mean T_C. To determine the mean waiting time for customers, the authors chose to combine each C class into a single class by the defining the

superposed arrival time and charging time as well as the total system load. Through this approach the authors simplified the analysis and are able to determine the mean waiting time through Little's Law

$$W = \frac{L}{\lambda}, \tag{2.10}$$

where L is the number of customers waiting in the queue based on the defined queuing model and λ is the derived superposed arrival rate. The authors could then use the derived wait time to alter the number of charging stations in order to minimize the customers wait.

The authors utilized the queuing model in determining the impact on revenue by applying an incentive to customers in order to reduce service times. The authors recommended an option where customers would revive an incentive if battery charging stopped at 80% SOC. The rational being that charging beyond this point increased charge inefficiencies and increased service time. Terminating charging at 80% SOC provide adequate power for customers and reduced service time. The authors assumed that some customers would participate in the incentives while others would not, which was factored into the distribution of the service time. The inclusion of the incentive meant an increase in other parameters like the arrival rate λ_C. These additional factors were added to the queuing model in order to define a strategy (e.g., number of chargers, incentive) that would maximize the charging facilities revenue.

The authors approach attempted to validate a charging strategies to maximize revenue through incentivizing customers to reduce charge times through queuing modeling. This approach provided a straight forward method that attempted to utilize the impact of varying sized batteries by using a multi-class approach. However, the authors combined the arrival and service rates of each class to simplify the problem to a single class analysis. This may not accurately highlight the impact of the varying battery types and may not provide an accurate analysis to define the best strategy.

Chapter 3

System and Maintenance Case Description

3.1 Overview

The Navy utilizes a wide range of battery technologies to support a multitude of systems and platforms. These batteries vary in size, configuration, chemistry and operational use. However, even with the variety of battery systems there is a unifying need to improve performance, efficiency, and cost-effectiveness. The Navy's Science and Technology (S&T) Strategy [62] outlined by the Office of Naval Research (ONR) has highlighted "Power and Energy" as one of the core development areas for the Navy. Part of the strategic goal is to develop and advance "efficient power and energy systems," "increase combat capability through high energy and pulsed power systems," and develop "new materials and methods to increase reliability and reduce maintenance costs."

In order to demonstrate how an acceptance classification approach combined with maintenance optimization may advanced the Navy's goal, a large platform battery system will be described as a case study in the following chapters. An assumed finite population of 60 platforms will be considered with each platform containing one battery system. A battery system is made up a large number of cells in a parallel, series configuration. In this example, all platforms are deployed and operational. Maintenance requirements associated with the platform specifically or other non battery systems are not considered at this time. This case study leverages information obtained from fielded systems, but due to sensitivity with some aspects of the programs, the data shown here will be modified in order to protect the raw information while still demonstrating the approaches.

For this case study, a specific battery system has experienced premature cell failures that have resulted in impacts to the system lifecycle maintenance plan. The system's expected mean time before failures (MTBF) was projected to be 60 months or five (5) years based

on reliability analysis and testing conducted during system development. However, due to a significant subset of cells within the battery showing early failure, the battery MTBF has shifted to 48 months or three (4) years. This has required the execution of unplanned maintenance actions similar to those described in the previous chapter in order to extend operational use. Electrical cycling maintenance actions are conducted in order to recover declining performance, but recovery is expected to be short-lived and decline re-emerges within a short time frame. Longer recovery can be accomplished through the costly and time-consuming process of targeted cell replacements. However, the mixture of different aged cells leads to other performance issues that must then be addressed with different, significant maintenance efforts [63]. The technical issues have also resulted in the increase of scheduled maintenance actions in an effort to monitor system degradation. Platforms are required to conduct SOH tests and provide performance data to the in-service engineering team more frequently in order to monitor system readiness and to determine if corrective actions are required.

3.2 Battery System Description

The battery system used in this case study is an uninterrupted power sources for systems on a Navy platform. The battery is comprised of cells connected in a series and parallel configuration to provide the desired voltage and power requirements. The battery will provide current as required to key systems to ensure the continued operations when the main power source is not available. Upon completion of the identified task the battery is recharged. A BMS provides passive monitoring of the battery by recording battery voltage, individual cell voltages, battery current and individual cell temperatures. No active cell management is conducted by the BMS in this system. Active cell management, typically required for lithium-ion based systems, is used to ensure that all cells are fully charged and balanced safely. A conceptual diagram of a battery system is shown in Figure 3.1.

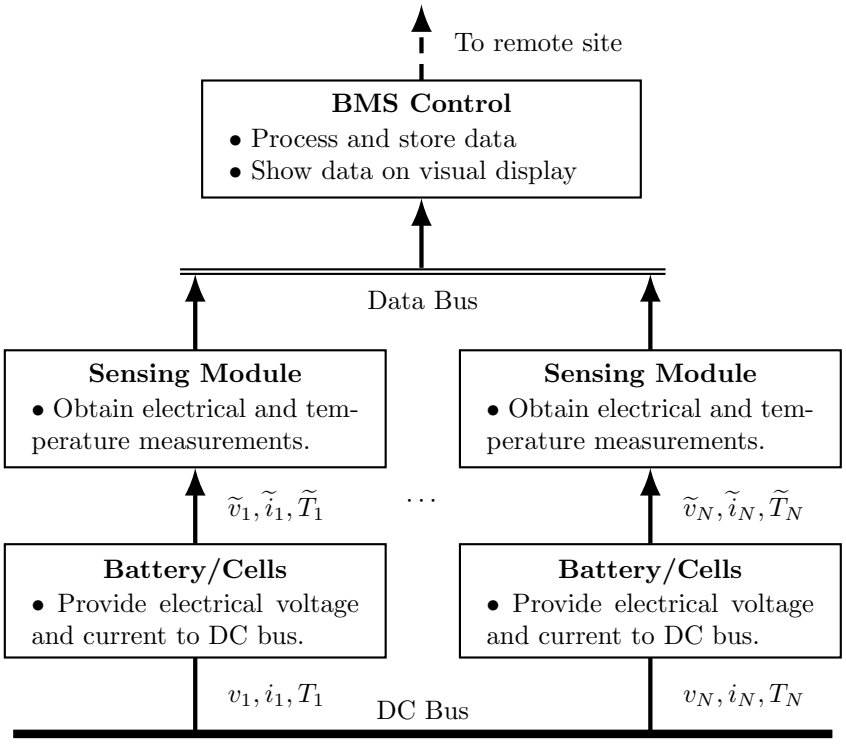

Figure 3.1: Notional diagram of battery system.

Fig. 3.1, the symbols v_k, i_k, T_k, $k \in \{1, 2, \ldots, N\}$ are the actual voltages, currents, and temperatures, respectively, of the N individual cells within the battery. Individual cells are placed in series electrically to obtain a total battery voltage. Each series set of cells will provide a current with groups placed in parallel to provide the overall battery capacity. Symbols $\widetilde{v}_k, \widetilde{i}_k, \widetilde{T}_k$ are the measured voltages, currents, and temperatures, respectively, of the corresponding cells. These measurements are collected in real-time ("online") by the BMS with basic prognostic functions to identify cells requiring actions to prevent damage through operations (e.g., overdischarge) and to ensure safety. Measurements collected by the BMS are also recorded for later inspection by an engineering organization responsible for the performance, reliability and maintenance of the battery system throughout its lifecycle. Each system includes a control interface which allows the batteries to be individually accessed, charged and discharged, and to enable maintenance tests for estimating battery SOH.

Batteries are procured and placed into storage until ready for installation on the identified platform. Prior to Government acceptance of the battery, a defined set of screening actions

that include specific electrical cycle tests are conducted on each cell to ensure that they meet the functional baseline requirements. These actions only highlight the current performance of each cell at the time of acceptance and do not forecast future life. There are some programs that will obtain multiple cell samples from a production run for lot acceptance testing (LAT), where longer term cycle testing is conducted to validate potential future expected life. However, due to the cost of cells and the battery life cycle replacement schedule this cannot be done for the system addressed in this case study. Once the battery is accepted it is placed into storage until it can be installed on the platform. Battery installation is performed by specifically identified organizations and requires unique equipment and trained personnel. Battery replacements follow a lifecycle replacement plan that is established to maximize the operational use but minimize risk of end of life performance issues. Program budgets for battery procurement and replacement are aligned to support these lifecycle milestones per platform.

For this case study, as the battery ages not all cells are performing consistently. A set number of cells within the battery are declining in performance earlier than the rest of the battery and specific performance markers have been identified to indicate "good" and "failing" cells. Premature cell failures occurs over time as the battery is used and can lead to overall battery performance issues. The performance assessment to classify the cells as "good" or "failing" is based on a review of voltage, current, and temperature collected by the BMS during normal operations and the routine SOH testing. These performance issues lead to increased maintenance actions in order to improve overall battery performance in an effort to meet platform and program requirements. However, premature cell failures have lead to earlier than planned full battery replacements due to the performance declining to unacceptable levels.

3.3 Maintenance Strategy Overview

The goal of maintenance efforts is to improve or maintain the available battery capacity in order to meet platform operational requirements. Without a battery capable of supporting the identified system requirements, the platform must operate at a reduced capability, which impacts it operational availability. Maintenance actions [1, 4] associated with most battery technologies begin with specific electrical recovery procedures. These involve applying a charge or cycle regimes (discharge and charge) on the battery in an effort to improve performance. Common maintenance actions include a constant current (CC) or constant voltage (CV) boost charges, pulse charging, and specific low rate discharge and recharge profiles.

If electrical recovery actions do not adequately improve battery performance, a targeted cell replacement may occur. In this case, failing cells that are limiting the overall battery's performance are replaced with higher preforming spares. This can be a difficult and time-consuming process depending on the cell types and the battery system design. The mixing of different aged cells is not considered a best practice as the cells will respond differently to charges and discharges which will affect the overall battery, leading to further performance issues [63]. If these actions do not maintain battery performance, an earlier than planned battery replacement may be required.

For this case study, the battery system maintenance concept is outlined in Figure 3.2. The platform and the trained users act as the organization/site maintenance level support for the battery system. The users or site maintenance personnel on the platform are provided specific guidance to perform maintenance actions on the battery by the battery vendor and/or an engineering in-service organization. A SOH cycle is conducted periodically to define battery performance at the time of the test. The SOH cycle is a defined discharge and charge routine that provides reference data on the battery's overall performance and of the individual cells within it. Data collected by the BMS is sent off the platform to the designated engineering in-service organization who tracks the battery performance for all platforms and

issues additional guidance as needed to improve operations. If battery performance were to improve, the periodicity of these action could be reduced. For this case study the assumed that scheduled maintenance conducted by the site maintenance team includes the following:

- *Constant Voltage Boost (CV).* Ensure battery is kept at a full state of charge and cells are balanced together.
- *Maintenance Cycle (MC).* Ensures that the battery is being exercised on a routine bases that helps stabilize performance.
- *State of Health (SOH) Test.* A discharge and charge cycle that provide performance information on individual cells and the overall battery. Given the performance concerns with the battery for this case study, the periodicity of the SOH test is set to obtain data as frequently as possible.

The role of the engineering in-service organization is to provide battery performance monitoring and engineering support. This organization will use the operational data form the BMS to evaluate battery performance trends and provide recommended maintenance actions to sustain battery operations. They maintain the battery operating procedures and work with the vendors to routinely update them as needed. The engineering in-service organization also conducts engineering investigations into issues related to the battery in an effort to improve system capability, monitoring vendor product quality, and implement technical program improvements. In the case of early declining performance, they provide recommendations to the platforms that take the shape of unscheduled maintenance actions that are used to maintain or improve battery performance.

If battery performance cannot be maintained utilizing the described site maintenance actions, more aggressive performance recovery efforts are required. In these cases the platform must be supported by an intermediate maintenance group that has been established to provide specific support during these unscheduled maintenance actions as well as conducted other platform support efforts like the lifecycle battery replacements. The intermediate

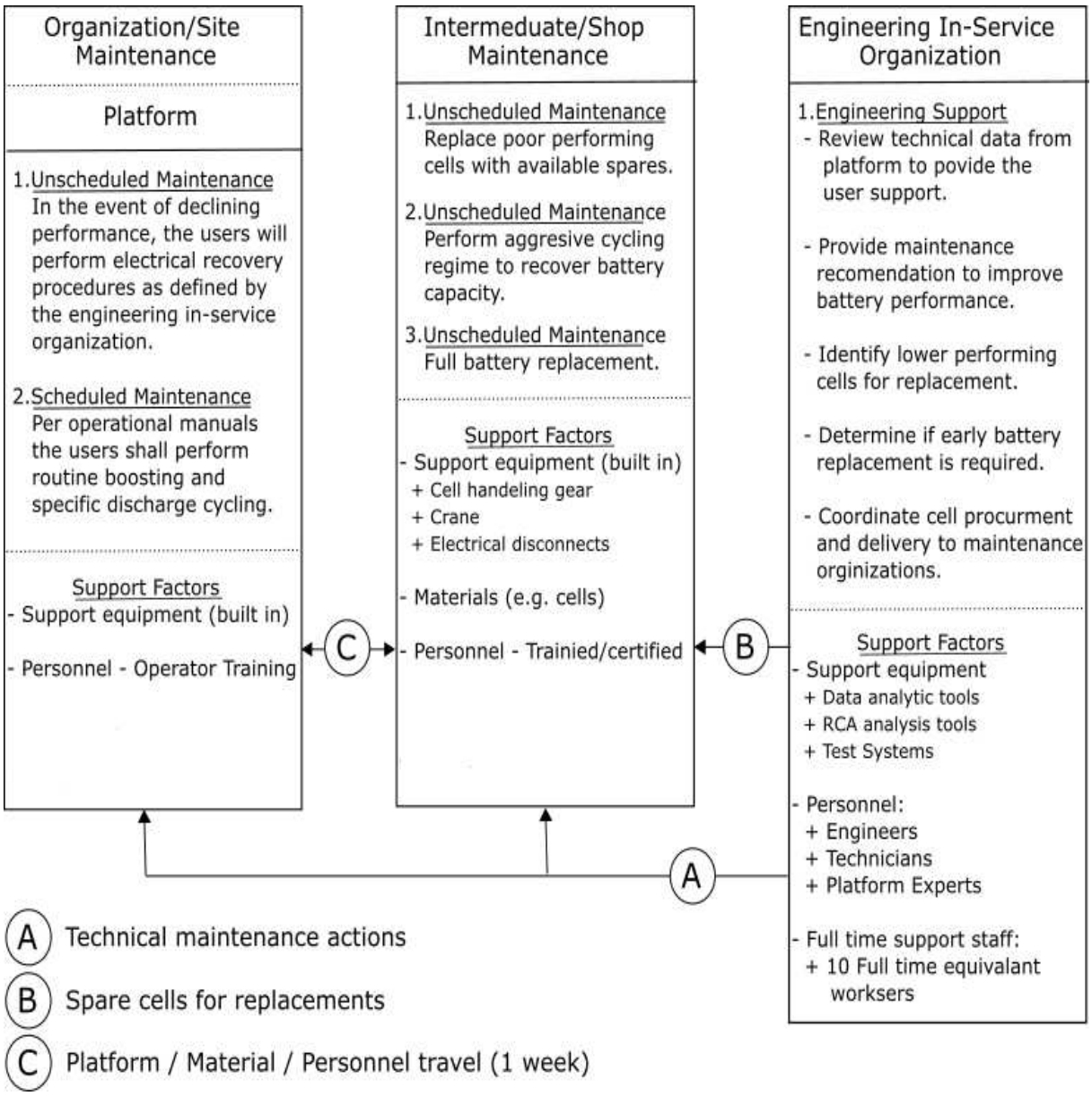

Figure 3.2: System Maintenance Concept.

maintenance team will execute actions identified by the engineering in-service maintenance organization to improve performance. Actions such as individual cell charging, aggressive recovery charging profiles [e.g., low rate discharges (LRDD), low rate charges (LRC)], and individual cell replacements must be coordinated with assigned intermediate maintenance organizations due to the need to establish conditions that cannot be done while the platform is under normal operating conditions. These actions are typically very invasive and require specific training and equipment. Performing these actions not only reduces the available operational time of the battery but also impacts the platform's availability. For this case study we will assume that the unscheduled maintenance actions carried out by the intermediate maintenance organization will include the following:

- *Constant current (CC) boost.* An aggressive CC charge used to recovery battery capacity by driving higher overcharge in order to recover capacity and balance the battery.

- *Recovery cycle regimen (cr).* A specified discharge and charge regime used to recovery battery capacity through low discharge and charge rates. The process takes several days and requires the platform to be under specific conditions that must be supported by the Intermediate Maintenance facilities.

- *Targeted cell replacement (rf).* Specific low performing cells are replaced in order to improve or maintain battery performance. Given the battery size, design, and platform integration requirements, neither a spare battery or spare cells are available on the platform. This effort requires specialized equipment and trained personnel, thus the site level personnel cannot execute the work. This effort is typically done as a last resort and in order to maintain the lifecycle replacement periodicity.

If batteries cannot be recovered or maintained by the described maintenance actions, en early or out of cycle battery replacement must be conducted to ensure platform operational readiness. This is usually due to the fact that a significant number of cells or the overall battery has declined in performance due to some issue and the performance cannot

be recovered or sustained. Most battery lifecycle replacement plans define the replacement periodicity based on the expected battery end of life. The period of performance is set to maximize battery useful life while minimizing the risk that performance will fall below the identified operational thresholds. Battery replacements and/or refurbishment occurs typically at the depot maintenance level for some systems. However, in this case study a majority of the removed cells are disposed of while the best performing cells placed in a spare pool.

3.4 Program Cost and Operational Availability

This section describes the maintenance costs and operational availability calculations used in this research. The current total maintenance cost, C_{Total}, per battery set was derived by determining the scheduled, C_{sm}, and unscheduled, C_{us}, maintenance costs to the program. The scheduled maintenance actions for this case study are well-defined and documented. For this research the identified unscheduled maintenance actions were determined by reviewing the actions conducted on multiple platforms, the resource and time required to execute them, and the average number of actions completed per the life of a battery set. These actions included efforts conducted by the user (site maintenance) or intermediate maintenance teams. These costs do not include logistical and administrative actions associated with those efforts.

The scheduled maintenance cost for a platform over the battery's life, C_{sm}, is:

$$C_{sm} = C_{cv} + C_{dd} + C_{sh} \tag{3.1}$$

where C_{CV} is the cost to conduct a CV maintenance boost, C_{dd} is the cost associated with conducting a maintenance cycle, and C_{sh} accounts for the cost to conduct a monthly SOH test. The SOH test is considered as part of the maintenance actions as it requires the battery be operated a specific way and provides information required to asses the battery.

The cost parameter for each scheduled maintenance actions is based on the number of actual hours to conduct the action, t_a, the number of actions conducted per year, N_{AY}, the

number of personnel required to execute the action, N_P, the associated labor rates, R_L, and the expected number of years of planned use, N_Y. The cost for each maintenance action was determined by the following:

$$C_{cv} = t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y \tag{3.2}$$

$$C_{dd} = t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y \tag{3.3}$$

$$C_{sh} = t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y \tag{3.4}$$

The average unscheduled maintenance cost for a platform over the full cycle life of a battery, C_{um}, is:

$$C_{um} = C_{cc} + C_{cr} + C_{rf} \tag{3.5}$$

where C_{cc} is the cost to conduct the CC maintenance boost, C_{cr} is the cost associated with conducting the recovery cycle regimen, and C_{rf} accounts for the cost to conduct a targeted cell replacement.

For the unscheduled maintenance actions the cost parameters are based on an analysis of the unplanned maintenance actions over multiple platforms. The cost for the CC boost charge, C_{cc}, and the recovery cycle regimen, C_{cr}, is:

$$C_{cc} = t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y \tag{3.6}$$

$$C_{cr} = t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y \tag{3.7}$$

where t_a is the number of actual hours to conduct the action, N_{AY} is the number of actions conducted per year, N_P the number of personnel required to execute the action, R_L the associated labor rates, and the expected number of years of planned use, N_Y.

A review of multiple platforms showed that for a battery to reach the targeted replacement plan, that at least one (1) targeted cell replacement would be need during a projected 60

month life. However, the significant cost associated with this action must be accounted for due to the high impact to the maintenance budget since the actions are typically unplanned. The targeted cell replacement cost, C_{rf}, is:

$$C_{rf} = (t_a \cdot N_{AY} \cdot N_P \cdot R_L \cdot N_Y) + M \tag{3.8}$$

where t_a is the number of actual hours to conduct the action, N_{AY} is the number of actions conducted per year, N_P the number of personnel required to execute the action, R_L the associated labor rates, N_Y the expected number of years of planned use, and M is the material cost need for the replacements.

Operational availability, A_o, is the percentage of actual operating hours to the maximum nominal operating hours [49]. The A_o per battery system is determined similar to that of the maintenance cost already described. The maximum nominal operating hours, T_{max}, is determined through a review of operational logs from multiple historic platforms. The total maintenance time T_{total} is determined by summing the total time required to support all scheduled, T_s, and unscheduled, T_{us}, maintenance actions.

The scheduled maintenance time for a platform over the battery life, T_s is

$$T_{sm} = T_{cv} + T_{dd} + T_{sh} \tag{3.9}$$

where T_{CV} is the time to conduct a CV maintenance boost, T_{dd} is the time associated with conducting a maintenance cycle (DD), and T_{sh} accounts for the time to conduct a SOH test. These efforts are standard cycling regimes placed on the battery and do not require administrative or logistic actions that would add to the maintenance time.

The time parameter for each scheduled maintenance actions is based on the number of total hours to conduct the action including administrative and logistic actions, t_{al}, the number of actions conducted per year, N_{AY}, and the expected number of years of planned use, N_Y. The cost for each maintenance action was determined by the following:

$$T_{cv} = t_{al} \cdot N_{AY} \cdot N_Y \tag{3.10}$$

$$T_{dd} = t_{al} \cdot N_{AY} \cdot N_Y \tag{3.11}$$

$$T_{sh} = t_{al} \cdot N_{AY} \cdot N_Y \tag{3.12}$$

The average unscheduled maintenance cost for a platform over the full cycle life of a battery, T_{um}, is:

$$T_{um} = T_{cc} + T_{cr} + T_{rf} \tag{3.13}$$

where T_{cc} is the time to conduct the CC maintenance boost, T_{cr} is the time associated with conducting the recovery cycle regimen, and T_{rf} accounts for the time to conduct a targeted cell replacement.

For the unscheduled maintenance actions the cost parameters are based on an analysis of the unplanned maintenance actions over multiple platforms. The time taken for the CC boost charge, T_{cc}, the recovery cycle regimen, T_{cr}, and the targeted cell replacement, T_{rf} is:

$$T_{cc} = t_a \cdot N_{AY} \cdot N_Y \tag{3.14}$$

$$T_{cr} = t_a \cdot N_{AY} \cdot N_Y \tag{3.15}$$

$$T_{cr} = t_a \cdot N_{AY} \cdot N_Y \tag{3.16}$$

where t_{al} is the total number of hours to conduct the action including administrative and logistic efforts, N_{AY} is the number of actions conducted per year, and the expected number of years of planned use, N_Y.

The Operational availability, A_o, was thus calculated by

$$A_o = \frac{T_{max} - T_{total}}{T_{max}} \tag{3.17}$$

3.5 Maintainability in the Baseline Maintenance Plan

The following section describes the program and maintenance impacts associated with the baseline maintenance plan and a scenario where early failure prone cells are present. The presence of early failing cells drive additional unscheduled maintenance actions which have impacts to systems lifecycle costs. These unplanned events impact operational availability and require intermediate maintenance resources to be reassigned from ongoing work to support efforts to improve battery performance, impacting other programs.

The maintainability metrics of lifecycle costs and operational availability are provided for the baseline maintenance plan. Herein, it is assumed that the number of battery systems deployed is 60 and the cost to a platform waiting for service or while being repaired during unplanned maintenance is \$100k per month. The maintenance system contains a single repair channel with a baseline average number of units waiting or in service at 13. The MTBF of a battery system is 48 months with a mean time to repair (MTTR) of a battery system being one (1) month, which includes downtime hours by the maintenance team, limited to one 12 hrs shift per day. The intermediate team labor costs are \$200 per hour. The nominal maximum operating hours for a battery system is $6,900$ hrs per year. Average scheduled and unscheduled maintenance actions over a five year expected lifetime, their maintenance costs (including labor, materials, and waiting costs) and operational downtime in the baseline plan are shown in Tables 3.1 and 3.2.

Table 3.1: Scheduled Maintenance Actions

Action	Maintenance Duration (hrs)	Actions per Year	Personnel Required	Material Cost (\$k)	Operational Time Lost (hrs)	Maintenance Cost (\$k)
CV Boost	12	12	2	–	720	\$288
Maintenance Cycle	6	50	2	–	1500	\$600
SOH Test	6	12	2	–	360	\$144
Total					**2580**	**\$1,032**

From Tables 3.1-3.2, implementation of the baseline maintenance plan has total maintenance costs of $\$1,876$k per system. The total cost penalty for systems being non-operational

Table 3.2: Unscheduled Maintenance Actions

Action	Maintenance Duration (hrs)	Actions per Year	Personnel Required	Material Cost ($k)	Operational Time Lost (hrs)	Maintenance Cost ($k)
CC Boost	24	1	2	0	456	$48
Recover Cycle	120	0.8	2	0	816	$192
Targeted Cell Replacement	252	0.2	12	100	1056	$605
Total					**2328**	**$844**

is $\$1,300$k, resulting in an LCC of $\$3,176$k per system. Operational availability (percentage of actual operating hours to maximum nominal operating hours) in the baseline maintenance plan is 86%.

Chapter 4

Classifier Reviews

4.1 Overview

This chapter describes the classification methods considered for pre-screening batteries as the initial effort to improve reliability and lifecycle maintenance improvement. The overall approach is to utilize (offline) pattern recognition method(s) to screen cells that are predicted to become poor performers (failure prone) prior to acceptance and deployment in a battery system. Pre-acceptance classification is to be performed with features based on minimally invasive cell measurements collected during the manufacturing process and initial acceptance testing conducted by the vendors. The selected algorithm should provide high classification accuracy, be mathematically simple and computationally efficient.

The classification algorithms selected for evaluation were (*i*) SGC, (*ii*) SVM, (*iii*) LDA, and (*iv*) PCALDA. The SGC method was selected due to the statistical bases of the approach and that ability to identify significant features correlated to failure prone cells. The SVM method was selected for its ability to define optimal margin between two class in a linear or non-linear data sets. The SVM is a method that classifies sample vectors through the determination of hyperplanes that maximize the separation of training data into classes [26–28,30–33,38,39,43]. SVM can be used to solve both linear and non-linear binary classification problems. The LDA method was selected as it has shown good accuracy in classification efforts by defining the scatter of data within each class and then maximizing the distance between the two classes while minimizing the within class space [9, 15, 47, 48, 64]. PCA dimensional reduction followed by LDA, referred to herein as PCALDA, was selected based on the potential increase in overall accuracy identified in literature [16, 17].

In the description which follows, bold faced letters are used to denote matrices or vectors; non-bold letters are used for scalar quantities; for a matrix $\mathbf{X}, \mathbf{X}_{(i;j)}$ represents the element

in row i and column j; for vector $\mathbf{x}, \mathbf{x}_{(i)}$ represents the element in row i; for an $n \times 1$ vector $\mathbf{x}$, diag($\mathbf{x}$) is an $n \times n$ matrix with $\mathbf{x}_{(i)}$ in the $(i;i)$ positions and zeros in the $(i;j)$; $\forall_i \neq j$ positions; $\{\cdot\}^T$ represents the matrix transpose operator; $|\cdot|$ symbolizes the absolute value of a number or cardinality of a set; $g(x;\alpha)$ symbolizes the evaluation of function g with variable input x and fixed parameter(s) α.

4.2 Simple Generalized Classifier Method

The simple generalized classifier (SGC) is a classification algorithm that assigns samples to classes based on their distance from the expected value of each class. While the SGC employs a simple procedure, it is able to distinguish between more than two classes of data, allows the creation of non-correlated data features from a smaller set of features, and can include rules based on expert system knowledge [12].

The SGC employs the notion of critical points, which specify boundaries between the classes. To illustrate the concept, Figure 4.1 shows the overlap of data from two distinct classes. While the distributions depicted in Figure 4.1 appear statistically similar and normally distributed, this is not required in the SGC algorithm.

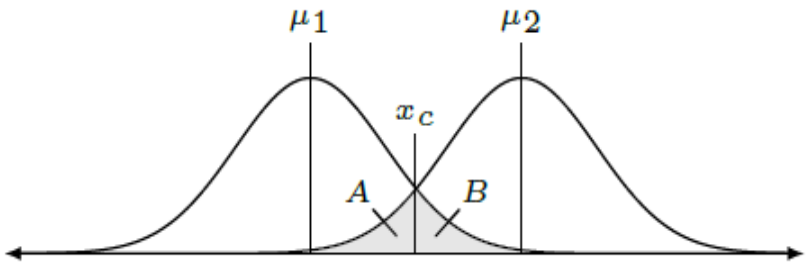

Figure 4.1: Conceptual diagram of data from two classes.

In 4.1, μ_1 and μ_2 are the expected mean values for class 1 (left curve) and class 2 (right curve) respectively; x_c is the critical point between classes; the shaded region A represents the "error area" of miss classification of data actually from class 2 to class 1; the shaded area

B represents the error area from miss classification of data actually from class 1 to class 2. The sum of the error areas is minimized when x_c is the same multiple of standard deviations away from the means of both classes, when expressed in standard deviations specific to each class. At point x_c the number of standard deviations away from the mean(s), η_σ, is given as

$$\eta_\sigma = \frac{\mu_2 - \mu_1}{\sigma_1 + \sigma_2}, \mu_1 < \mu_2 \tag{4.1}$$

where μ_1 and μ_2 are the standard deviations of the data from classes 1 and 2, respectively. The critical point is therefore located at

$$x_c = \mu_1 + \eta_\sigma \sigma_1 = \mu_2 - \eta_\sigma \sigma_2 \tag{4.2}$$

Here N_c denotes the number of possible data classes associated with the cells. The identifications of the classes and the assignment of training data is based on expert knowledge from the review of cell performance during laboratory testing or results from deployed systems. For each class, N_s training samples for the N_f data features have been collected and assigned.

The sample data is stored for the k'th class in a $N_s \times N_f$ matrices denoted as $\mathbf{D}^k$. Expert knowledge on cell manufacturing and the electrochemical processes can be used to develop new features from these existing ones. These new features are created by augmenting the original features' data by performing mathematical operations (e.g., multiplying, dividing, etc.) on the columns of $\mathbf{D}^k$. For example, the ratio of positive active material (PAM) and negative active material (NAM) used within the cell could be a new feature (PAM/NAM) used by the classifier. These augmented features are collected in matrices $\mathbf{A}^k$ and assigned $\mathbf{D}^k \leftarrow [\mathbf{D}^k \mid \mathbf{A}^k]$ if available.

For each class, the mean, μ, and standard deviation, σ, of each feature data set (consisting of N_s sample observations) is determined. These values are collected into $N_c \times N_f$ matrices μ

and σ. The means are then placed in order based on the class number to ensure the correct inequality for pairwise evaluations when calculating the critical point.

The critical point, x_c, and number of standard deviations between the $N_c - 1$ contiguous pairs of ordered features are determined by using the correctly order pairs. The defined standard deviations are collected into a $N_f \times (N_c - 1)$ matrix, $\mathbf{N}$. For each pair-wise feature comparison identified, a t-statistic according to a two-sided t-test is performed with $df = 2(N_s - 1)$ degree of freedom. Each associated p-value for each t-statistic is collect in $N_f \times (N_c - 1)$ matrix $\mathbf{P}$.

In order to determine the critical features, sum the p-values for each feature by summing the rows in $\mathbf{P}$. Down-select features with p-values for which distinction by class is statistically significant (e.g., $p < 0.05$). Rank order the down-selected features, with the highest rank assigned to the lowest p-value, second highest rank to next lowest p-value, etc. Of the rank ordered features, one may select a subset if fewer features are desired. Denote the final number of "critical features" is $N_{cf} \leq N_f$. With the critical features selected, prune the matrices η, σ, N to contain only the N_{cf} rows of critical feature data.

The identified critical features are then assigned a weight in order to ensure features with higher potential accuracy provide greater influence on the classification. The $N_{cf} \times 1$ elements of the feature weight vector w is determined as

$$w_{(i)} = \frac{\sum_j (N_{(i,j)})^2}{\sum_i \sum_j (N_{(i,j)})^2} \tag{4.3}$$

The weighted critical features can now be applied to newly obtained data for classification. For a single test sample example, the new data collected for a cell is reviewed and any new data generated from augmentation of the sample vector is conducted. The data set is then pruned to only the critical factors which were determined in training. Let x denote the resulting $N_{cf} \times 1$ sample vector.

A z-score (distance) of each sample feature to the recorded training feature means for all classes is calculated. The parameterized z function is:

$$z(y:\mu,\sigma,\eta) = \frac{\mid y-\mu \mid}{\sigma/\sqrt{\eta}} \tag{4.4}$$

where y is the input data; μ,σ, and η are parameters specifying the mean, standard deviation, and number of samples used to compute the mean and standard deviation parameters. The z-score for each feature is computed and collected in a $N_c \times N_{cf}$ z-matrix as

$$\mathbf{Z}_{(i,j)} = z(X_{(i)}; \boldsymbol{\mu}^T_{(i,j)}, \boldsymbol{\sigma}^T_{(i,j)}, N_s), \tag{4.5}$$

where it is noted that $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and N_s are associated with the training samples.

An element-wise evaluation of function $f(z)$ is applied to each entry of $\mathbf{Z}$ and the values are collected in matrix $\mathbf{F}$. For example, the function $f(z) = \frac{1}{z^2}$ may be used to achieve increasing function values with decreasing z-scores, i.e., as the samples get closer to the feature means. $\mathbf{F}$ is normalized by the column. The weighting factors $w_{(i)}$ determined during training are applied to $\mathbf{F}$ by assigning $\mathbf{F} \leftarrow \mathbf{F}\cdot \text{diag}(w)$. The weighted values are summed for each class to obtain the $N_c \times 1$ vector $s_{(i)} = \sum_j \mathbf{F}_{(i;j)}$. The classification decision is obtained by selecting the row of s with the largest value; this row number corresponds to the assigned class number.

4.3 Support Vector Machine

The SVM is a method that classifies sample vectors through the determination of hyperplanes that maximize the separation of training data into classes [9, 10, 15, 20, 30–32, 34–36, 40, 42]. SVM can be used to solve both linear and non-linear binary classification problems.

In linear binary classification problems the training set to the SVM consists of pairs denoted $\{\mathbf{x}_i, y_i\}$, $i = 1, 2, \ldots, N_s$, $y_i \in \{1, -1\}$ where, $\mathbf{x}_i$ is the $N_f \times 1$ vector for training sample i and y_i is the class indicator for $\mathbf{x}_i$. The SVM algorithm is used to determine hyperplanes that separate the data into the two classes, described by the equations $\mathbf{w}^T\mathbf{x}_i + b \geq$

$+1$ (for $y_i = +1$) and $\mathbf{w}^T\mathbf{x}_i + b \leq +1$ (for $y_i = -1$), where $\mathbf{w}$ is a vector orthogonal to both hyperplanes and b is the bias, as depicted in Figure 4.2.

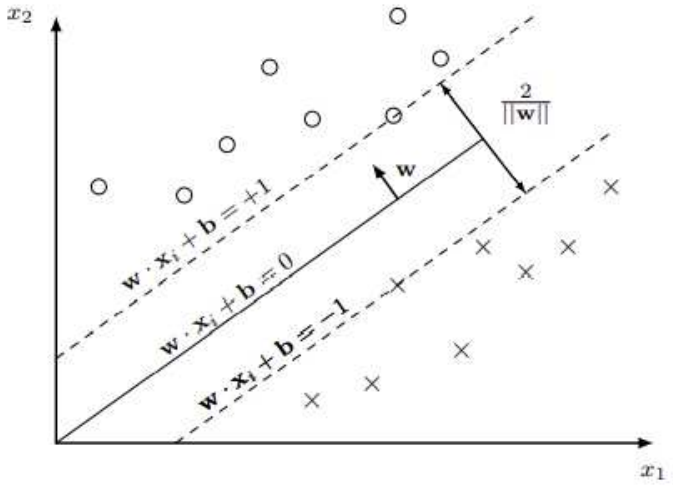

Figure 4.2: Optimal Separating Hyperplane with Support Vectors.

Identification of the hyperplanes that maximize the separation distance between the classes is determined by solving the constrained optimization problem:

$$\min \left\{ \frac{1}{2}||\mathbf{w}||^2 \right\}$$
$$\text{s.t.} \quad y_i \left(\mathbf{w}^T\mathbf{x}_i + b \right) - 1 \geq 0. \tag{4.6}$$

The 'support vectors' lie along the hyperplanes described by the equations $\mathbf{w}^T\mathbf{x}_i + b \pm 1$ and have separation distance $2/|||\mathbf{w}|$. In nonlinear separable problems slack variables, ξ_i, are used to augment the constraints in (4.6) as

$$y_i \left(\mathbf{w}^T\mathbf{x}_i + b \right) > 1 - \xi_i, \tag{4.7}$$

and the optimal separation distance is determined through solution of the optimization problem:

$$\min \left\{ \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N_f} \xi_i \right\}, \tag{4.8}$$

with constraints given in (4.7), and where the second term in (4.8) is a cost penalty. SVM can also be applied to nonlinear classification problems by introducing nonlinear functions that map the vector space containing the training vectors to a higher-dimensional space that allows linear classification; symbolically, let $\boldsymbol{\varphi} : \mathbb{R}^{N_f} \to \mathbb{R}^{N_k}$ denote the mapping where $N_k > N_f$ is the dimension of the higher-dimensional space.

A positive definite 'kernel' function $k : \mathbb{R}^{N_f} \times \mathbb{R}^{N_f} \to \mathbb{R}$ is defined as:

$$k(\mathbf{x}, \mathbf{x}_i) := \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}_i) \rangle \tag{4.9}$$

which is then used to form the kernelized decision classification function:

$$\hat{y}(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{N_s} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b \right), \tag{4.10}$$

where $\hat{y}$ is the estimated class and α_i are scalar coefficients with upper-bound C [65].

4.4 Linear Discriminant Analysis

The LDA method is used to determine a linear combination of the data features that can then be used to separate input samples into two or more classes [9, 15, 47, 48, 64]. As with SVM, the true classifications of the training vectors are assumed to be known *a priori*. Let K denote the number of possible classes of the data set, n_k the number of possible classes and $\Pi_k = \{x_j\}$, $j = 1, 2, \ldots, n_k$ contain the data vectors assigned to each class. The total number of samples in all classes $N_s = \sum_{k=1}^{K} n_k$. The mean features for each class are computed as:

$$\mu_{\mathbf{k}} = \frac{1}{n_k} \sum_{j \in \Pi_k} \mathbf{x_j}, \tag{4.11}$$

The covariance matrix for vectors $\mathbf{x_j} \in \Pi_k$ is computed as:

$$\mathbf{\Sigma_k} = cov(\mathbf{x_j}, \mathbf{x_j}) = \mathbf{E}[(\mathbf{x_j} - \mu_k), (\mathbf{x_j} - \mu_k)^T], \tag{4.12}$$

where $\mathbf{E}[\cdot]$ is the expectation function. In the case of $K = 2$ classes, each input vector $\mathbf{x}$ (including the training vectors) are assumed to be a member of class $y = 0$ or $y = 1$. The LDA approach assumes that the conditional probability density functions $p(\mathbf{x} \,|\, y = 1)$ and $p(\mathbf{x} \,|\, y = 2)$ are normal distributions with mean and covariance parameters $(\mu_\mathbf{1}, \mathbf{\Sigma_1})$, $(\mu_\mathbf{2}, \mathbf{\Sigma_2})$, respectively. Under the additional assumption that the two covariance matrices are equal, i.e., $\mathbf{\Sigma_1} = \mathbf{\Sigma_2} = \mathbf{\Sigma}$, the classification decision function can be expressed:

$$\mathbf{w^T x} > \mathbf{c}, \tag{4.13}$$

where

$$\mathbf{w} = \mathbf{\Sigma^{-1}}(\mu_\mathbf{2} - \mu_\mathbf{1}), \tag{4.14}$$

and c is a scalar threshold computed as:

$$c = \frac{1}{2}\mathbf{w^T}(\mu_\mathbf{2} - \mu_\mathbf{1}). \tag{4.15}$$

4.5 PCALDA

The proposed PCALDA approach combines the LDA approach described in the previous section with PCA. n particular, the dimension of the data are first reduced using PCA by computing the PCs in the manner described in this section. The transformed data are then used as inputs to the LDA algorithm which performs the classification according to [31].

PCA is a method used to transform data to 'principal component' (PC) bases, where the basis vectors are in the direction of maximum variance of the underlying data set. PCA can often be used to reduce the dimensionality of the data, by selecting a subset of computed basis vectors while maintaining sufficient variance within the data set [9, 15, 64].

To determine the PCs from training samples, the raw sample data is first collected in $N_s \times N_f$ matrices, denoted as $\mathbf{X}$. Expert knowledge can be applied to develop augmented features by performing mathematical operations on the original data sets. These augmented features are collected in matrix $\mathbf{A}$ and the sample matrix is updated as $\mathbf{X} \leftarrow [\mathbf{X} \mid \mathbf{A}]$. The mean for each feature is computed and subtracted from the corresponding feature in the sample matrix, resulting in a matrix denoted $\mathbf{B}$. The covariance matrix C is then calculated as

$$\mathbf{C} = \frac{1}{N_f}\mathbf{B}^T\mathbf{B}. \tag{4.16}$$

A reduction in dimensions is accomplished by determining the eigenvalues and eigenvectors of $\mathbf{C}$ in the equation $\mathbf{E}^{-1}\mathbf{C}\mathbf{E} = \mathbf{\Lambda}$, where the columns of $\mathbf{E}$ contain the eigenvectors of $\mathbf{C}$ and the eigenvalues are contained along the principal diagonal of (diagonal) matrix $\mathbf{\Lambda}$. The eigenvectors are ordered according to their corresponding eigenvalues; eigenvalues less than a predefined threshold are removed, and those above the threshold are assigned as the remaining, PC vectors.

The selected PCs are used to transform the data set to which is then used as input to the LDA method. The LDA method finds a linear combination of the data features that can be used to separates input samples into two or more classes [9, 15, 47, 48, 64]. LDA implementation will follow the same process as outlined in the previous section.

Chapter 5

Classifier Implementation and Comparison

5.1 Data Preparation

In this chapter the classification approaches are evaluated for implementation using sample battery data from the case study system outlined earlier. The data used in this study represents measurements of various battery cell features taken prior to acceptance and where eventual maintenance records and end-of-life (EOL) performance were known. Feature data for the batteries include pre-acceptance test measurements of cell attributes such as: cell weight, acid concentration, formation temperature, open circuit cell voltage, initial capacity, etc. Using the maintenance records and EOL performance information for the deployed systems, selected sample cells were classified into 'good' (class 1) and 'failure-prone cells' (class 2).

For the evaluation an acceptable sample size was determined through a statistical review of the total available population [66]. For each feature a sample size n was determined per

$$n = \frac{(z_{\alpha/2})^2 \sigma^2}{E^2}, \tag{5.1}$$

where $(z_{\alpha/2})$ is the z-value correlated to the desired confidence value, σ is the estimated feature population standard deviation, and E is the width of the confidence interval. For this analysis the desire is to achieve a 95% confidence interval ($\alpha = 0.05$) where the standard deviation (σ) was derived from the total available feature population. The confidence interval error is expressed as $E = W/2$ where W is the tolerable error (W) for the confidence interval of each feature.

The sample size was further refined by conducting a power of the test analysis for each feature [66]. Power is considered the probability of rejecting H_0 (the null hypothesis), when H_0 is in fact false. Power is defined as

$$Power = 1 - \beta, \tag{5.2}$$

where

$$\beta \approx P(z \leq z_{\alpha/2} - \frac{|\mu_0 - \mu_a|}{\sigma/\sqrt{n}}). \tag{5.3}$$

Here μ_0 denotes the null value of μ and μ_a denotes the actual value of the mean in H_a (the alternative hypothesis) with the other variables previously described [66]. A power of 85% or greater was selected in order to provide confidence in the data set and follow on analysis. If a power value less than 85% was calculated, the sample size was increased until the minimum power requirement was achieved. Once the sample values were calculated for all features, the optimal sample size was selected so that each feature encompassed the same number of samples. For the efforts conducted in this research 200 cells was selected as the sample size with eight identified features for classification training. The 200 samples consisted of 100 class 1 and 100 class 2 cells that were classified based on known historic performance and end of life failure mode. For validation, 200 additional cells were selected consisting of the same class 1 and class 2 population sizes. This is an increase in the number of samples used in previous research [67]. The same sample sets were used in the training and testing of each classification model for consistency and comparison.

To better visualize the actual data set used, Figure 5.1 shows the raw class 1 and class 2 training data within a comparison of each feature used in this research. The figures highlight the presence of some clustering or possible spacial separation between the classes when comparing individual feature together. This initially suggested that a classification technique may aid in identifying class 1 and class 2 cells. A detailed view of example comparisons can be seen in Figure 5.2 and Figure 5.3.

Figure 5.1: Individual comparison of all features.

Figure 5.2: Feature 1 vs Feature 2 comparison of classes.

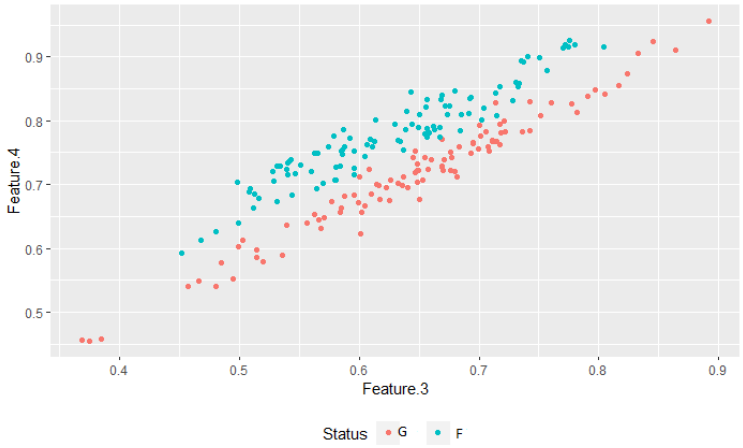

Figure 5.3: Feature 3 vs Feature 4 comparison of classes.

Ten example training vectors used to illustrate classes 1 and class 2 are shown in Tables 5.1 and 5.2 respectively.

Table 5.1: Class 1 Example Training Data

n	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	2.0	1.8	1.2	2.2	2.8	2.5	1.9	2.4
2	3.0	0.1	0.9	0.8	0.4	0.3	0.6	1.0
3	0.0	0.8	2.5	0.7	1.3	2.2	2.0	1.8
4	1.7	1.0	2.1	0.2	1.7	1.0	0.9	0.9
5	2.0	0.5	0.1	2.5	5.1	1.3	1.7	5.0
6	4.6	2.4	4.2	1.5	6.4	0.3	0.7	1.1
7	0.7	1.1	1.3	0.3	0.0	1.8	18.2	1.9
8	5.3	8.3	4.0	1.9	12.0	33.6	3.6	0.8
9	0.9	1.0	1.4	1.4	1.6	1.4	1.4	1.4
10	1.3	0.6	0.2	0.2	0.3	0.9	0.1	0.0

Table 5.2: Class 2 Example Training Data

n	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	1.9	3.8	2.4	2.7	3.2	2.0	3.4	2.3
2	3.7	0.7	1.8	1.6	1.0	0.8	1.4	1.4
3	1.7	1.4	1.4	2.6	1.4	2.0	2.0	3.5
4	2.2	3.2	2.8	1.4	6.9	3.8	2.6	2.7
5	4.9	2.7	5.1	5.0	2.5	10.0	3.4	5.5
6	7.0	5.3	5.1	9.3	10.3	1.0	4.8	2.3
7	1.7	4.7	2.2	2.5	5.5	3.8	2.5	2.1
8	2.8	3.8	1.5	2.4	2.0	1.7	2.7	1.4
9	1.6	1.4	1.7	0.6	2.2	1.1	0.9	1.3
10	0.7	1.1	0.5	0.4	0.4	0.9	0.5	0.6

In Tables 5.1 and 5.2, f_i correspond to the i'th data associated with the specific feature for the $n = 1, \ldots, 10$ cell samples. The $N_f = 8$ features include six "raw" features, f_i, $i \in \{1, 2, 3, 4, 5, 6\}$, based on direct measurements and two augmented features, f_i, $i \in \{7, 8\}$, derived using pair-wise multiplication and division of raw features selected based on expert knowledge.

The test vectors, which were applied to the classifier models after training, were prepared in the same manner as the training vectors, i.e., they contained the same raw and augmented features in the same order. An example test vector is shown in Table 5.3

For the classifier evaluation the same data sets with augmented features were used in all cases. Depending on the approach taken, the vectors were augmented further as required by the approach discussed.

Table 5.3: Sample Test Data.

n	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	2.2	3.3	3.7	2.0	4.0	3.5	3.2	3.4
2	1.4	2.0	1.7	1.1	2.9	1.3	2.6	1.9
3	2.7	1.9	0.6	1.9	1.3	1.5	1.2	1.0
4	0.9	1.7	1.3	1.7	2.6	2.0	1.6	3.1
5	2.3	1.5	2.7	4.3	2.0	7.0	2.6	5.8
6	2.5	0.3	0.0	0.9	0.6	0.5	0.4	0.8
7	3.3	5.5	2.0	4.7	2.9	5.6	9.7	3.9
8	0.1	1.0	0.8	2.2	2.3	1.6	2.0	1.2
9	1.4	0.8	1.7	0.5	0.1	1.6	0.7	1.5
10	2.2	1.4	0.3	2.5	1.1	4.5	4.1	1.8

5.2 SGC Implementation

The SGC classifier was coded in MATLAB® R2020b, and implemented the algorithm described in chapter 4, section 4.2. As stated, Tables 5.1 and 5.2 represent the class 1 and class 2 training data sets. Here f_i correspond to the i'th data associated with the specific feature for the $n = 1, \ldots, 10$ data samples. The $N_f = 8$ features include six "raw" features, f_i, $i \in \{1, 2, 3, 4, 5, 6\}$, based on direct measurements and two augmented features, f_i, $i \in \{7, 8\}$, derived using pair-wise multiplication and division of raw features selected based on expert knowledge. However, note that any operation or mathematical function of the raw features could have been used to obtain the augmented features (a stated advantage of the SGE algorithm [12]). The augmented feature data in Tables 5.1 and 5.2 constitute training data matrices $\mathbf{D}^1$ and $\mathbf{D}^2$ as discussed in the previous section.

Applying the classifier training process yielded mean and standard deviation matrices

$$\mu = \begin{bmatrix} 2.1 & 2.8 & 0.5 & 1.3 & 1.7 & 2.3 & 1.2 & 3.8 \\ 6.5 & 0.8 & 3.1 & 9.0 & 2.3 & 1.4 & 1.3 & 0.6 \end{bmatrix},$$

$$\sigma = \begin{bmatrix} 0.6 & 0.7 & 0.4 & 0.4 & 0.6 & 0.7 & 0.8 & 1.7 \\ 2.7 & 0.5 & 1.5 & 10.1 & 0.7 & 0.5 & 0.4 & 0.4 \end{bmatrix},$$

respectively. The first three critical features were selected by rank-ordered p-value, and were identified as features f_2, f_4 and f_6. The associated feature weighting vector was computed as $\mathbf{w} = [\,0.32\ 0.34\ 0.35\,]^T$.

The trained classifier was then applied to measurement data obtained during cell screening evaluations, similar to the example vectors in Table 5.3. Here the augmented features f_7 and $f,8$ were calculated in the same manner as described above. The determined critical features (f_2, f_4 and f_6) of the test data were evaluated and assigned the appropriate computed vector weights in order to classify the new samples. The results of the classifier using the full data set of 200 test cells, 100 class 1 and 100 class 2 samples, is summarized in the 'confusion' matrix shown in Table 5.4.

Table 5.4: SGC Confusion Matrix

		Classifier Output	
		Class 1	Class 2
True	Class 1	76	24
Class	Class 2	1	99

As shown in Table 5.4, the classifier correctly identified 76 of the 100 samples actually from class 1 as being in class 1 and incorrectly assigned 24 class 1 samples to class 2 (76.0% accuracy for classifying class 1 samples). The classifier correctly identified 99 of the 100 samples actually from Class 2 as being in Class 2 and incorrectly assigned 1 class 2 samples to class 1 (99.0% accuracy for classifying class 2 samples). Thus the algorithm provided an overall accuracy of 87.5%.

5.3 SVM Implementation Method

The SVM classifier was developed and executed using R programming language, version 3.5.1 which followed the implementation outlined in chapter 4, section 4.3. When train-

ing the SVM classifier, the `svm()` function from the `e1071` package was used to develop a classification model with the following parameter options: the type option was set to "C-Classification" (to perform classification), the kernel function type was set to linear, and cost (corresponding to C in function 4.8) was initially set to high value (>1) with all other options used as default values. A tuning analysis was conducted to optimize the cost function in order to maximize accuracy.

The training vectors were applied to the `svm()` function for training purposes. The training vectors were applied to the classification model as new inputs according to 4.10 using the `predict()` function in order to optimize the cost value. The initial classification accuracy was low given, which was expected given the high cost rate. To improve the overall accuracy the optimal cost defined in equation 4.8 by completing a sensitivity analysis to minimize the classification error (Class_Error) which can be seen in Figure 5.4. It shows that the optimal cost value was determined to be 0.95 which was then used to update the parameters of the `svm()` function and retrain the model.

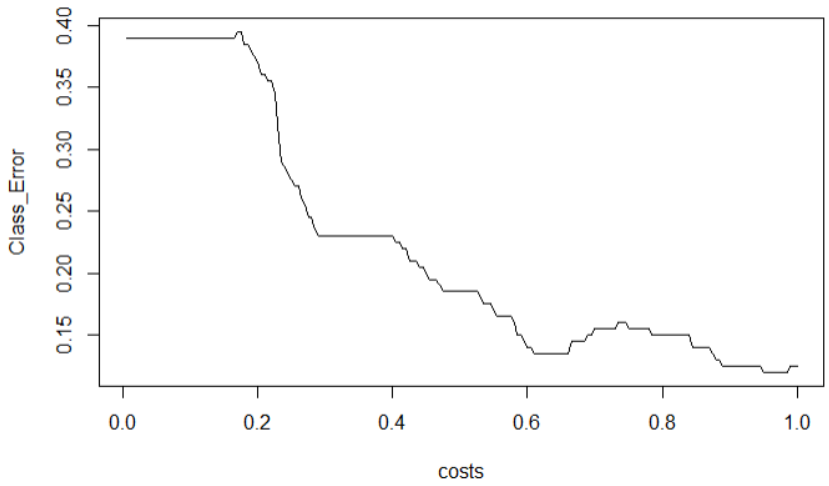

Figure 5.4: Sensitivity Analysis for Best Cost Value.

The test vectors were then classified with the trained classifier model also using the `predict()` function from the R stats package. The results of the classifier using the full

test vectors of the 200 test cells, 100 class 1 and 100 class 2 samples, is summarized in the 'confusion' matrix shown in Table 5.5.

Table 5.5: SVM Confusion Matrix

		Classifier Output	
		Class 1	Class 2
True	Class 1	91	9
Class	Class 2	10	90

Table 5.5 shows that the SVM classifier correctly identified 91 of the 100 samples actually from class 1 as being in class 1 and incorrectly assigned 9 class 1 samples to class 2 (91.0% accuracy for classifying class 1 samples). The classifier correctly identified 90 of the 100 samples actually from class 2 as being in class 2 and incorrectly assigned 10 class 2 samples to class 1 (90.0% accuracy for classifying class 2 samples). Thus the algorithm provided an overall accuracy of 90.5%.

5.4 LDA Implementation Method

The LDA classifier was developed and executed using R programming language, version 3.5.1 which followed the implementation outlined in chapter 4, section 4.4. When training the LDA classifier, the `lda()` function from the `MASS` package was used to compute w and c in the classification function 4.13 with the standard R default parameter options selected.

Applying the training vectors to the `lda()` function the means for each feature was calculated as defined in function 4.11. Given that only two class are being compared, only one LD is required to be defined. The coefficients of the LD for the actual training data set consisting of 100 class 1 and 100 class 2 cells were defined as shown in Table 5.6.

Table 5.6: Derived Coefficients of Linear Discriminant

Feature	**LD1 Coefficients**
f_1	$3.91e+01$
f_2	$-1.85e+03$
f_3	$3.67e+03$
f_4	$-3.67e+03$
f_5	$-6.29e-02$
f_6	$2.54e+01$
f_7	$1.04e+02$
f_8	$1.70e-01$

The test vectors were then evaluated through the trained classification model that applied the function 4.13 through the `predict()` function in the R stat package. The results of the classifier using the full data set of 200 test cells, 100 class 1 and 100 class 2 samples, is summarized in the 'confusion' matrix shown in Table 5.7.

Table 5.7: LDA Confusion Matrix

		Classifier Output	
		Class 1	Class 2
True	Class 1	93	7
Class	Class 2	2	98

As shown in Table 5.7, the classifier correctly identified 93 of the 100 samples actually from class 1 as being in class 1 and incorrectly assigned 7 class 1 samples to class 2 (93.0% accuracy for classifying class 1 samples). The classifier correctly identified 98 of the 100 samples actually from class 2 as being in class 2 and incorrectly assigned 2 class 2 samples

to class 1 (98.0% accuracy for classifying class 2 samples). Thus the algorithm provided an overall accuracy of 95.5%.

5.5 PCALDA Implementation Method

The PCALDA classifier was developed and executed using R programming language, version 3.5.1 which followed the implementation outlined in chapter 4, section 4.5. When performing dimension reduction using PCA, the `prcomp()` function from the stats package was used to perform the operation with the following parameter options: centering was set to "true", and scale was set to "true"; all other options used the default values. In this case, centering was used to give zero mean of the data and scaling was done to ensure the data were scaled to have unit variance.

The training vectors are transformed via PCA for dimensional reduction and as inputs to the LDA classifier. As described, the mean value of each feature is determined and then subtracted from each feature set to center and scale the vectors, thus normalizing the data which forms the new matrix **B** as discussed in chapter 4. The PCs scale and format the data to be represented in a new plane, with PC1 explaining the largest variation within the matrix. The standard deviations and variances for each PC is defined in Table 5.8. The table shows that the first five PCs account for 92% of the variation within the data set. The derived PC coefficients using the `prcomp()` function are defined in Table 5.9.

Table 5.8: PCs Variance and Standard Deviation

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**
Standard Deviation	1.747	1.374	0.993	0.9189	0.769	0.615	0.511	0.000
Proportion of Variance	0.382	0.236	0.123	0.106	0.074	0.047	0.033	0.000
Cumulative Proportion	0.382	0.617	0.741	0.846	0.920	0.967	1.000	1.000

Table 5.9: Principal Component Coefficients

Feature	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
f_1	0.471	0.052	-0.048	-0.189	-0.174	0.840	-0.048	-2.27e-06
f_2	-0.514	0.041	-0.037	0.030	0.224	0.298	-0.688	-0.345
f_3	-0.149	0.698	-0.034	0.063	-0.002	0.043	-0.163	0.676
f_4	0.119	0.702	-0.016	0.050	-0.121	-0.114	0.195	-0.651
f_5	-0.103	0.046	0.968	-0.206	-0.060	0.052	0.022	-8.81e-06
f_6	0.400	0.093	0.0878	-0.047	0.904	-0.050	-0.028	9.56e-06
f_7	0.478	0.004	0.006	-0.307	-0.272	-0.431	-0.646	3.26e-06
f_8	0.279	-0.061	0.222	0.904	-0.096	0.032	-0.202	2.76e-05

The variance for each PC is plotted in Figure 5.5 which includes a boundary line for an eigenvalue equal to one. Typically, PCs with an eigenvalue less than one would be discarded since it suggests that the PC explains less then what one variable would [14]. However, Figure 5.6 and Table 5.8 confirms that using only the first three PCs would only provide an explanation for a cumulative variance of 78%. In order to achieve higher accuracy in the analysis, the first five PCs were selected and explain 92% of the variation.

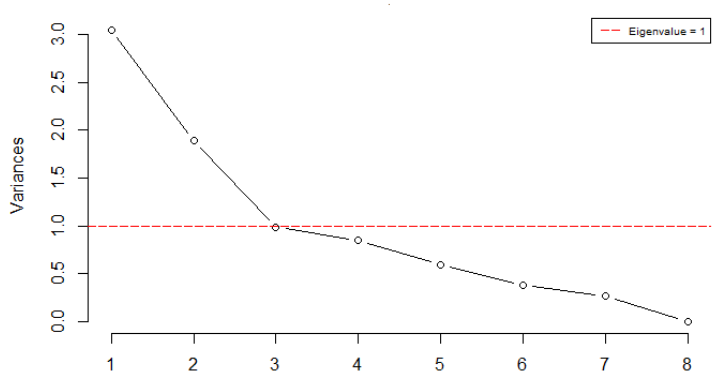

Figure 5.5: Eigenvalues for the associated PCs.

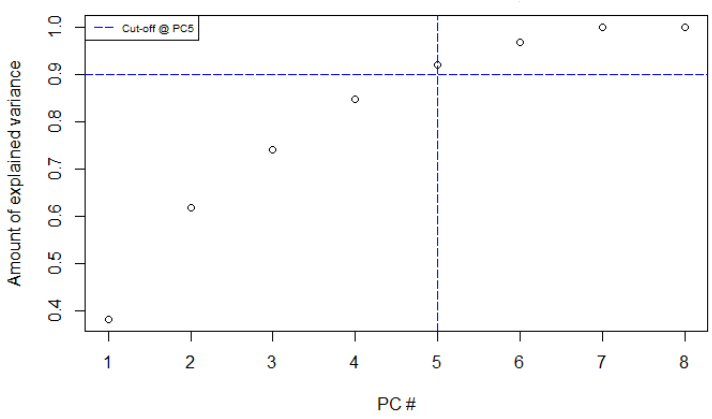

Figure 5.6: Cumulative variance of the PCs.

Using the defined PCs, the original training vectors are transformed with reduced dimensionality to be used as the training set for the LDA model. The transformed training vectors are used as inputs to the `lda()` function from the `MASS` package. The coefficients of the LD for the actual transformed training vectors for the 100 class 1 and 100 class 2 cells is defined in Table 5.10.

Table 5.10: Derived LD from PC Input

Principal Components	**LD1 Coefficients**
PC_1	-1.529
PC_2	-0.279
PC_3	-0.090
PC_4	0.130
PC_5	0.358

With the LDA algorithm trained, the test vectors are transformed using the identified five PCs. The transformed test vectors were then evaluated using the trained model and test vectors as inputs to the `predict()` function in the R stat package. The results of the

classifier using the full data set of 200 test cells, 100 class 1 and 100 class 2 samples, are summarized in the 'confusion' matrix shown in Table 5.11.

Table 5.11: PCALDA Confusion Matrix

		Classifier Output	
		Class 1	Class 2
True	Class 1	99	1
Class	Class 2	5	95

As shown in Table 5.11, the classifier correctly identified 99 of the 100 samples actually from class 1 as being in class 1 and incorrectly assigned 1 of the 100 samples actually in class 1 to class 2 (99.0% accuracy for classifying class 1 samples). The classifier correctly identified 95 of the 100 samples actually from Class 2 as being in Class 2 and incorrectly assigned 5 of the 100 samples actually in class 2 to class 1 (95.0% accuracy for classifying class 2 samples). Thus, the algorithm provided an overall accuracy of 97.0%.

5.6 Classification Comparison Summary

Per Table 5.4, the SGC approach provided an overall accuracy of 87.5% with 76 out of 100 Class 1 samples (76.0%) and 99 out of 100 Class 2 sample (99.0%) accurately classified. Table 5.5 shows that the SVM classifier provided an overall accuracy of 90.5% with 91 out of 100 Class 1 samples (91.0%) and 90 out of 100 Class 2 sample (90.0%) accurately classified. The LDA classification method, as shown in Table 5.7, provided an overall accuracy of 95.5% with 93 out of 100 Class 1 samples (93.0%) and 98 out of 100 Class 2 sample (98.0%) accurately classified. Finally, the PCALDA method provided an overall accuracy of 97.0% based on accurately classifying 194 out of 200 cells from the test data set, as shown in Table 5.11. As can be seen, the PCALDA approach correctly classified 99 out of 100 Class 1 samples

(99.0%) and 95 out of 100 Class 2 sample (95.0%). Classifier accuracy is summarized in Table 5.12.

Table 5.12: Classifier Accuracy

Classifier	**Overall**	**Class 1**	**Class 2**
SGC	87.5%	76%	99%
SVM	90.5%	91%	90%
LDA	95.5%	93%	98%
PCALDA	97.0%	99%	95%

PCALDA, one possible fusion between algorithms, yielded the highest overall accuracy and the highest success rate in correctly classifying good (Class 1) cells. In pre-acceptance screening, a small percentage of failure- prone cells may be deployed, but the smaller expected size of this population after pre-screening would be expected to result in decreased corrective maintenance actions and a subsequent decrease in unscheduled maintenance costs and operational downtime. In practice, the SVM, LDA and PCALDA approaches were easier to implement since they used built-in R functions. However, one benefit of the SGC approach over the others is that features impacting cell performance can be more readily identified; this information can also potentially be used to inform product or process improvements.

It should also be note that the SGC and LDA approaches showed the highest accuracy in correctly classifying failure-prone or Class 2 cells. Depending on the system, platform, and lifecycle management priorities, this accuracy may be more important than the overall classifier accuracy. For example, lifecycle management priorities may require an emphasis on reliability or reduction of maintenance cost, at the expense of a higher procurement cost associated with identifying and selecting a greater set of good cells, while rejecting others.

5.7 Maintainability After Pre-Screening

This section considers LCC and operational availability impacts where the baseline maintenance plan is augmented with pre-screening using the PCALDA classification algorithm described above. Herein, it is assumed that the pre-screening classifier provides 97% accuracy in identifying failure-prone cells (cf. [68]). In this scenario, cost and operational availability improvements result from the increase in MTBF of the in-service battery population after pre-screening. In particular, there are significant reductions in required unscheduled and scheduled maintenance actions and the associated costs for waiting and repairs (labor and materials). For example, the need to conduct SOH testing is diminished as battery reliability improves. In addition, while not all failure prone cells may be eliminated from the population, the number is sufficiently reduced by the high classifier accuracy that targeted cell replacement efforts—the most expensive and time-consuming action—are effectively eliminated.

In this study, it is assumed that pre-screening results in a MTBF of the battery population of 60 months, and there are 6 units waiting or being repaired at any given time. A summary of computed costs under this maintenance plan are shown in Tables 5.13 and 5.14.

Table 5.13: Scheduled Maintenance Actions with Screening

Action	Maintenance Duration (hrs)	Actions per Year	Personnel Required	Material Cost ($k)	Operational Time Lost (hrs)	Maintenance Cost ($k)
CV Boost	12	12	2	–	720	$288
Maintenance Cycle	6	50	2	–	1500	$600
SOH Test	6	6	2	–	180	$72
Total					**2400**	**$960**

Table 5.14: Unscheduled Maintenance Actions with Screening

Action	Maintenance Duration (hrs)	Actions per Year	Personnel Required	Material Cost ($k)	Operational Time Lost (hrs)	Maintenance Cost ($k)
CC Boost	24	0.8	2	0	432	$38
Recover Cycle	120	0.6	2	0	696	$144
Targeted Cell Replacement	–	–	–	–	–	–
Total					**1128**	**$182**

From Tables 5.13-5.14, implementing the baseline maintenance plan with pre-screening has a total maintenance costs of $\$1,142$k per system. The total cost penalty for non-operational systems is \$600k, resulting in an LCC of $\$1,742$k per system (a 45% reduction from baseline). Operational availability (percentage of actual operating hours to maximum nominal operating hours) of 90% (a 4% increase from baseline).

Chapter 6

Maintenance Plan Parameter Optimizations

6.1 Overview

In this chapter it is demonstrated how a combination of pre-screening with optimization of maintenance plan parameters using a finite queuing system model are used to further enhance the maintainability of a fleet of battery systems. In these studies, the finite queuing system model described in section 6.2 was used to simulate stochastic part failures and repairs. Using the nominal costs and lost operational time for scheduled and unscheduled maintenance actions given in 3.5 and 5.7, the queuing system dynamics (e.g., part failures, queuing, exit of repaired parts) were used to compute simulated costs for various maintenance plans. These studies focus on the optimization of the statistical expectation of lifecycle cost, $E[LCC]$, obtained over all study runs. Analysis of the impact of pre-screening is also performed. In these studies, fleet MTBFs of 48 and 60 hrs were used for non-screened and pre-screened battery systems, respectively.

Case Study I demonstrates the use of the finite queuing system model to optimize the number of repair channels in the maintenance concept, and the specific impact of pre-screening on maintainability in this scenario. Case Study II demonstrates the optimization of mean service time in the maintenance concept and the impact of pre-screening.

6.2 Finite Queuing System Model

This section describes the finite queuing system model that was used to represent battery pre-screening, in-service, failure and repair dynamics. The system is depicted in Fig. 6.1. In Fig. 6.1, N_o is the number of batteries that are pre-screened (the "Initial Population"). The N batteries that are classified as "good" enter the in-service population ("Screened Population"), while batteries classified as "failure prone" do not enter the service population

("Rejected"). After batteries are placed in service, they fail and enter the repair queue according to the probability density function of their *actual* classification (good or failure prone). Failed batteries enter the repair queue to wait upon the next available repair channel. After repairs are attempted, batteries are either stored in inventory (if repairable) or wasted (if not repairable). In this research it was assumed that all parts could be repaired, and all repaired parts immediately reentered the service population.

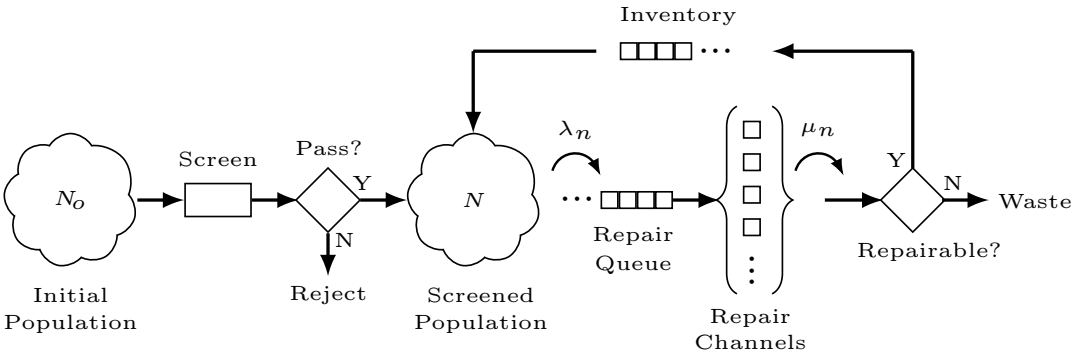

Figure 6.1: Battery service and repair queuing system model.

In the queuing system model, the number of failed parts currently in either the repair queue or a repair channel are enumerated by the symbol $0 \leq n \leq N$. A new part failure was represented as an "arrival" into the system, modeled as a random variable described by a Poisson process where:

1. Probability of more than one new failure in time interval $(t, t + \Delta t)$ as $\Delta t \to 0$ is zero;

2. Probability of a single new failure in time interval $(t, t + \Delta t)$ is $\lambda_n \Delta t$.

The above conditions result in stochastic arrival times—times between part failures—that are described by an exponential distribution. Herein, the time required to complete an individual repair was also modeled as a random variable with exponential distribution. Under the assumption of Poisson arrivals into the system and exponentially distributed repair times, it can be shown (cf. [69]) that the probability of either an arrival or exit (completed repair) from the system does not depend on the time associated with the preceding event, i.e., arrivals

and exits are "memoryless" processes. Viewing n as a 'state' of the queuing system, state transitions were therefore modeled using a Markov chain ("birth-death") process, depicted in Fig. 6.2.

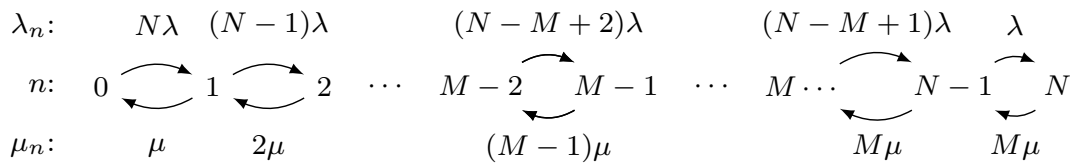

Figure 6.2: Markov chain model for finite queuing process.

In Fig. 6.2, quantities shown above the arrows pointing from left-to-right are failure rates for parts entering the queuing system; quantities shown below the arrows pointing from right-to-left are repair rates parts exiting the queuing system and reentering the population. Letting the failure rate of a single battery be denoted $\lambda = 1/\text{MTBF}$, the failure rate of the population after n items are in queue or being repaired is:

$$\lambda_n = (N - n)\,\lambda, \quad 0 \leq n \leq N. \tag{6.1}$$

Letting the repair rate of a single channel be denoted $\mu = 1/\text{MTTR}$, where MTTR is the mean time to repair, the repair rate of the entire facility after n items have already failed is:

$$\mu_n = \begin{cases} n\mu, & n = 1, 2, \ldots, M-1 \\ M\mu, & n = M, M+1, \ldots, N \end{cases}, \tag{6.2}$$

where M is the number of service channels.

When simulating the queuing system dynamics, state transitions were determined in the following way. Let $\mathbf{t}_f \sim \exp(\lambda_n)$ be a random variable representing the time of the next part failure, sampled from an exponential distribution with rate parameter λ_n; a specific occurrence of the random variable is denoted t_f. Similarly, let $\mathbf{t}_r \sim \exp(\mu_n)$ be a random

variable representing the time of the next part exit from repair with rate parameter μ_n, with specific occurrence t_r. At time t, with the system in state $0 \leq n \leq N$, samples are obtained for $\mathbf{t}_f, \mathbf{t}_r$ using inverse transform sampling. If $t_f \leq t_r$, then n is incremented by one; otherwise, n is decremented by one.

To validate the code (written in MATLAB®) used in this research to represent the finite queuing system, simulations of the stochastic system behavior of the system were compared to theoretical expressions. The theoretical steady-state probability of there being n items in the finite multi-channel queuing system, denoted P_n, can be expressed as:

$$P_n = P_0 C_n, \tag{6.3}$$

where P_0 is the probability that zero units have failed:

$$P_0 = \left(\sum_{n=0}^{N} C_n \right)^{-1}, \tag{6.4}$$

where

$$C_n = \begin{cases} \frac{N!}{(N-n)!n!}(\frac{\lambda}{\mu})^n, & n = 0, 1, 2, \ldots, M \\ \frac{N!}{(N-n)!M!M^{(n-M)}}(\frac{\lambda}{\mu})^n, & n = M+1, M+2, \ldots, N \end{cases}. \tag{6.5}$$

The expressions in (6.3)–(6.5) describe the probability mass function associated with observing n parts in the system. To quantify the degree of similarity between theoretical and simulated probability distributions in this research, an overlapping index was used [70]. For two probability density functions $f_A(x), f_B(x)$, with random variable $x \in \mathbb{R}$, the overlapping index $\eta : \mathbb{R} \times \mathbb{R} \to [\,0, 1\,]$ is defined as:

$$\eta(A, B) = \int_{\mathbb{R}} \min [\, f_A(x), f_B(x) \,]\, dx, \tag{6.6}$$

where $\eta(A, B) = 0$ indicates distinct distributions, and $\eta(A, B) = 1$ indicates identical distributions. In the discrete case, the density functions in (6.6) are represented by probability

mass functions and the integration reduces to a summation. Fig. 6.3 shows histograms of the theoretical vs. simulated number of parts (denoted by x) in the system in an example study. In this validation effort, $\lambda = 1/32\,\text{units/s}$, $\mu = 1/8\,\text{units/s}$, $N = 20$, $M = 4$ and the queuing system was simulated over 1000 s.

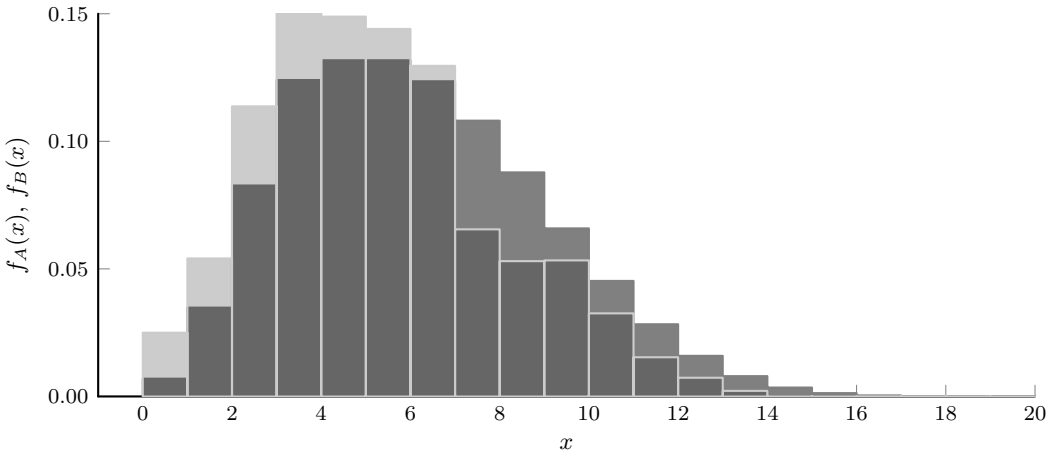

Figure 6.3: Histogram of theoretical (dark gray) vs. simulated (light gray) number of parts in the system, over 1000 simulated seconds.

In Fig. 6.3, the theoretical histogram is shown in dark gray, simulated histogram in light gray; overlapping is shown in medium gray. In this study, the overlapping index was computed as 86.4%. Fig. 6.4 shows the results with the same study parameters, except simulated over $100,000$ simulated seconds.

The overlapping index of the histograms in Fig. 6.4 was computed as 99.5%. These studies show that the finite queuing system model produces stochastic behavior that converges to the theoretical probability distribution as number of the number of queuing operations (i.e., length of time) increases, as expected.

6.3 Case Study I: Optimization of Service Channels

This case study considers potential maintainability improvements by selecting the number of intermediate maintenance locations or service channels (repair bays) in the battery system

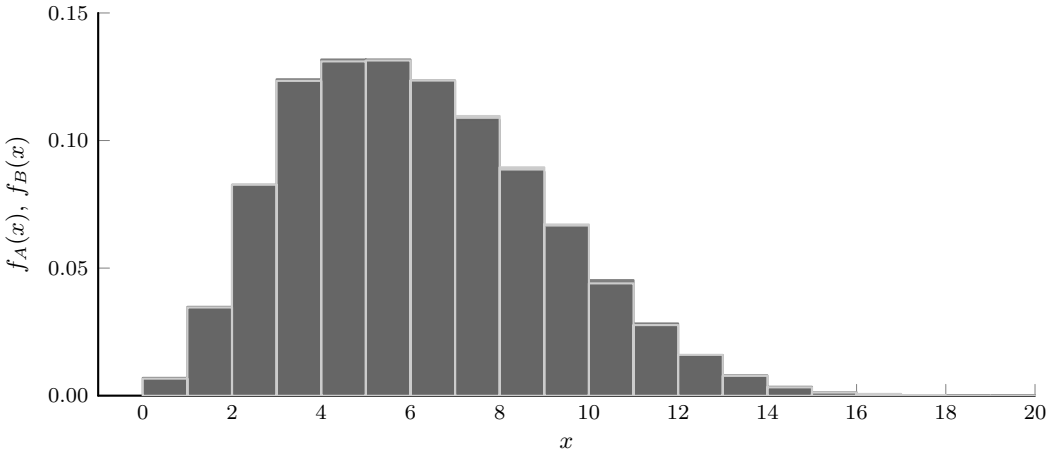

Figure 6.4: Histogram of theoretical (dark gray) vs. simulated (light gray) number of parts in the system, over $100,000$ simulated seconds.

maintenance concept. If the number of service channels results in excess waiting times, the maintenance costs and operational availability will be negatively impacted. However, if the number of channels is greater than what is required, additional resource costs will be required. The minimal number of channels must therefore be determined to support maintenance activities while minimizing overall cost.

To determine the optimal number of service channels, the finite queuing system was used to simulate the maintenance plan over the five year (60 month) battery lifecycle using a range of service channels, $M \in \{1, 2, 3, 4\}$. The study was repeated $2,000$ times and the results were averaged to yield a total of $120,000$ queuing simulation months. The total maintenance cost from the studies was computed, and the alternative with the lowest expected LCC was designated as the optimal channel number, M^*.

Table 6.1 presents expected unscheduled maintenance costs versus service channels without pre-screening. In Table 6.1, queue length, non-operational unit costs and service costs are averaged over all studies. It can be seen from Table 6.1 that $M^* = 2$. With two channels, the battery systems are able to be fully serviced with a small average number of units waiting in the queue (0.6). Table 6.1 further shows that with service channels beyond two, queue length decreases, but maintenance costs increase. Implementing two channels without pre-

screening results in an expected LCC of $\$2,906$k, a cost reduction of \$270k versus baseline (a 9% improvement). Operational availability was computed as 86% per system.

Table 6.1: Expected Maintenance Costs Versus Number of Service Channels (MTBF $= 48$)

Service Channels	**Avg Queue Length**	**Non-Operating Unit Cost (\$k)**	**Service Cost (\$k)**	$\boldsymbol{E[LCC]}$ **(\$k)**
1	12.2	1,314	844	3,190
2	0.6	186	1,688	2,906
3	0.1	134	2,532	3,698
4	0.0	127	3,376	4,535

Table 6.2 summarizes the costs versus number of service channels while also implementing the battery pre-screening stage. Here the optimal number of channels to support the fleet is also $M^* = 2$. However, comparing Tables 6.1 and 6.2 show that the implementation of pre-screening in combination with channel optimization also results in a lower expected LCC versus channel optimization alone. In this case, the expected LCC is $\$1,452$k per system (a 54% reduction from baseline). Operational availability in this scenario is 90%.

Table 6.2: Expected Maintenance Costs Versus Number of Service Channels (MTBF $= 60$)

Service Channels	**Avg Queue Length**	**Non-Operating Unit Cost (\$k)**	**Service Cost (\$k)**	$\boldsymbol{E[LCC]}$ **(\$k)**
1	4.78	569	182	1,711
2	0.3	128	364	1,452
3	0.1	105	546	1,611
4	0.0	101	728	1,789

6.4 Case Study II: Optimization of Mean Service Time

This case study considers potential maintainability improvements by adjusting the mean service time to find the optimal service rate, μ^*. In this study the same number of queuing simulations as Case Study I were performed, over a range of service times $1/\mu \in \{0.25, 0.50, \ldots, 1.25\}$. Mean service time can be changed by altering the number of ser-

vice personnel, tooling, training and other resources. In practice, these resources contribute to additional unscheduled maintenance costs, which are a function of the service rate. Herein, the cost function used to model this relationship was: $C_{us}(\mu) = C_{usn}/\mu + e^{2.5\mu^2}$, where C_{usn} is the nominal unscheduled maintenance cost. The sharp cost increase at $1/\mu = 0.25$ is due to the duplication of specialized equipment and personnel not required prior to that point.

To focus only on the impact of service time, the number of service channels was fixed at $M = 1$ in these studies. Table 6.3 shows the computed unscheduled maintenance costs without implementation of pre-screening. From Table 6.3, it can be seen that $\mu^* = 1/0.75 = 3$ per month. Table 6.3 also indicates that implementation of this plan results in an expected LCC of $\$2,626$k per system (an 18% reduction versus baseline). Operational availability in this scenario is 86% per system.

Table 6.3: Expected Maintenance Costs Versus Service Times (MTBF = 48)

Mean Service Time (Months)	**Avg Queue Length**	**Non-Operating Unit Cost ($k)**	**Service Cost ($k)**	$\boldsymbol{E[LCC]}$ **($k)**
0.25	0.0	45	3,978	5,055
0.5	0.5	154	1,693	2,879
0.75	3.7	466	1,127	2,626
1.0	11.8	1,277	845	3,155
1.25	21.0	2,197	676	3,905

Table 6.4 shows computed costs versus service time when also implementing the pre-screening stage. As shown in Table 6.4, $\mu^* = 1/0.5 = 2$ per month in this scenario is optimal, with additional reduction in costs compared to Table 6.3. Implementation of this plan results in an expected LCC of $\$1,425$k per system (a 55% cost reduction versus baseline). Operational availability in this scenario is 90%.

As demonstrated in these case studies, the implementation of pre-screening with maintenance plan parameter optimization resulted in significant cost reductions and improvements in operational availability compared to the baseline maintenance plan, with optimization of channel number or service time yielding similar maintainability metrics.

Table 6.4: Expected Maintenance Costs Versus Service Times (MTBF = 60)

Mean Service Time (Months)	**Avg Queue Length**	**Non-Operating Unit Cost ($k)**	**Service Cost ($k)**	$\boldsymbol{E[LCC]}$ **($k)**
0.25	0.0	33	1,330	2,323
0.5	0.0	96	369	1,425
0.75	1.4	240	245	1,444
1.0	5.1	605	182	1,748
1.25	11.6	1,234	147	2,371

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This research was proposed in order to identify an approach to improving battery system reliability in an effort to reduce lifecycle maintenance costs and improve system operational availability. Literature review conducted in this area identified several "on-line" methods to predict battery health prognostics. However, these approaches focused on batteries already deployed into the field where system reliability could not be changed. Maintenance optimization techniques for batteries were also reviewed as correctly sizing maintenance resources to ensure fleet operational availability supports reducing lifecycle costs while ensuring an fielded fleet. The majority of the battery research in this area focused on optimizing charging facilities for electric vehicles. The research in this area focused on maximizing profit with acceptable wait times. While this approach is justifiable for many applications, the scenarios discussed in this dissertation focus on minimizing the wait times of platforms as well as reducing lifecycle maintenance costs.

The research conducted in this dissertation proposed a two-faceted approach for enhancing the overall maintainability of battery systems, measured by total LCC and operational availability. The approach included a battery classification pre-screening stage with the optimization of maintenance design parameters using a finite queuing system model. Queuing system simulation outputs were used to compute cost and availability metrics under alternative design options. To isolate the impact of pre-screening, optimizations of the maintenance parameters were also analyzed with and without the screening stage.

The initial efforts evaluated the potential effectiveness of classification algorithms for pre-acceptance screening of batteries [67, 68]. Ultimatly, four classification methods (SGC, SVM, LDA, and PCALDA) were evaluated utilizing easily measurable physical and electrical

features in an effort to conduct pre-acceptance screening of batteries. It was determined that the PCALDA approach yielded the greatest overall classifier accuracy (97%), while SGC showed the greatest accuracy (99%) in identifying failure-prone cells. It was additionally noted that the SVM, LDA and PCALDA approaches were easier to implement over SGC as they are available as built-in functions in some software tools (e.g., R programming language). However, it was observed that the SGC approach can be used to readily-identify features affecting cell performance and potentially inform product or process improvements to further increase product reliability.

Optimization of the number of intermediate maintenance facilities (service channels) showed that the optimal number of both with and without the cell screening process was two (2). However, the addition of the pre-screening stage resulted in a cost savings of $\$1,454$k, while ensuring the majority of the fleet was operational. It was also shown that pre-screening resulted in an optimal mean service rate of 2 per month, versus 3 per month required without pre-screening, while providing a cost savings of $\$1,201$k and waiting cost reduction of \$370k. These results show that implementation of the pre-screening classification method combined with optimization of maintenance plan design parameters can be used to achieve a reduced LCC compared to pre-screening alone, while maintaining or increasing operational availability.

7.2 Future Work

The acknowledged limitations of this research are: the methodology was demonstrated on only four scenarios (two case studies with non-screened and pre-screened batteries). The research only considered one type of pre-screening classification algorithm; batteries were assumed to always be repairable as well as the times for the next failure and repair times were assumed to be exponentially distributed.

Future research includes further refinements in the queuing system model to include repairability of batteries and different failure and repair time density functions. Additional

investigation of coupled optimization studies (e.g., combined channel number and service time) and other maintenance plan parameters. Additional work would also focus on improving the pre-screening algorithms by reviewing additional classification methods as well as the combination of algorithms to improve accuracy (an ensemble approach).

Bibliography

[1] D. Linden and T. Reddy, *Handbook of Batteries, 3rd ed.* Two Penn Plaza, New York, New York: McGraw-Hill, 2001.

[2] D. Pavlov, *Lead Acid Batteries, Science and Technology, 2nd ed.* 1000 AE Amsterdam, The Netherlands: Elsevier, 2017.

[3] D. Rand, P. Moseley, J. Garche, and C. Parker, *Valve Regulated Lead Acid Batteries.* 1000 AE Amsterdam, The Netherlands: Elsevier, 2004.

[4] D. Berndt, *Maintenance Free Batteries: Based on Aqueous Electrolyte, 3rd ed.* Baldock, Hertfordshire, England: Research Studies Press LTD., 2003.

[5] S. M. Rezvanizaniani, Z. Liu, Y. Chen, and J. Lee, "Review and recent advances in battery health monitoring and prognostics technologies for electric vehicle (ev) safety and mobility," *Journal of Power Sources*, vol. 256, pp. 110–124, 2014.

[6] R. Prengman and A. Siegmund, "Premature capacity loss in vrla batteries for telecom applications," *Battery Conference on Applications and Advances*, vol. IEEE Xplore, 2001.

[7] Y. Okada, Y. Tsuboi, M. Shiomi, and S. Osumi, "Premature capacity loss in vrla batteries for telecom applications," *International Telecommunications Energy Conference*, vol. IEEE Xplore, 2003.

[8] S. Asht, R. Dass, A. Fasllis, and et al, "Pattern recognition techniques: A review," *Journal of Chemical Information and Modeling*, vol. 53, no. 9, pp. 1689–99, 2013.

[9] P. Sharma and M. Kaur, "Classification in pattern recognition: A review," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 2277–128, 2013.

[10] J. Zhang, S. O. Williams, and H. Wang, "Intelligent computing system based on pattern recognition and data mining algorithms," *Science Direct - Sustainable Computing: Informatics and Systems*, 2017.

[11] N. Kaur and U. Kaur, "Survey of pattern recognition methods," *Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 2, pp. 317–19, 2013.

[12] S. Simske, *Meta-analytics: Consensus approaches and system patterns for data analysis*. Cambridge, MA: Morgan Kaufmann, 2019.

[13] S. J. Phillipsa, R. P. Andersonb, and R. E. Schapirer, "Maximum entropy modeling of species geographic distributions," *Ecological Modelling*, vol. 190, pp. 231–259, 2006.

[14] M. Richardson, "Principal component analysis," 2009.

[15] S. Marsland, *Machine Learning: An Algorithmic Perspective, 2nd ed.* 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL: CRC Press, 2015.

[16] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant analysis of principal components for face recognition," *IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

[17] J. Yang and J. yu Yang, "Why can lda be performed in pca transformed space?" *Pattern Recognition*, vol. 36, no. 2, pp. 563–566, 2003.

[18] S. Simske, "Low-resolution photo/drawing classification: metrics, method and archiving optimization," in *Proceedings of the IEEE International Conference on Image Processing, vol. 2*, Genoa, Italy, 2005, pp. 534–537.

[19] L. Liao and F. Köttig, "Review of hybrid prognostics approaches for remaining useful life prediction of engineering systems, and an application to battery life prediction," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 191–207, 2014.

[20] H. Pan, Z. Lu, H. Wang, H. Wei, and L. Chen, "Novel battery state-of-health online estimation method using nultiple health indicators and an extreme learning machine," *ScienceDirect - Energy*, vol. 160, pp. 466–477, 2018.

[21] B. E. Olivares, M. A. C. Muñoz, M. E. Orchard, and J. F. Silva, "Particle-filtering-based prognosis framework for energy storage devices with a statistical characterization of state-of-health regeneration phenomena," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 364–376, 2013.

[22] S. Barsali and M. Ceraolo, "Dynamical models of lead-acid batteries: Implementation issues," *IEEE Transactions on Energy Conversion*, vol. 17, no. 1, pp. 16–23, 2002.

[23] M. Chen and G. A. Rinc´on-Mora, "Accurate electrical battery model capable of predicting runtime and i–v performance," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 504–511, 2006.

[24] K. Goebel, B. Saha, A. Saxena, J. Celaya, and J. Christophersen, "Prognostics in battery health management," *IEEE Instrumentation & Measurment Magazine*, pp. 33–40, August 2008.

[25] J. Lu, L. Wei, M. M. Pour, Y. Mekonnen, and A. I. Sarwat, "Modeling discharge characteristics for predicting battery remaining life," *IEEE Transportation Electrification Conference and Expo (ITEC)*, pp. 468–473, 2017.

[26] A. Widodo, M.-C. Shim, W. Caesarendra, and B.-S. Yang, "Intelligent prognostics for battery health monitoring based on sample entropy," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11 763–11 769, 2011.

[27] X. Hu, S. E. Li, Z. Jia, and B. Egardt, "Enhanced sample entropy-based health management of li-ion battery for electrified vehicles," *Science Direct - Energy*, vol. 64, pp. 953–960, 2014.

[28] X. Hu, J. Jiang, D. Cao, and B. Egardt, "Battery health prognosis for electric vehicles using sample entropy and sparse bayesian predictive modeling," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2645–2656, 2016.

[29] Y.-H. Sun, H.-L. Jou, and J.-C. Wu, "Aging estimation method for lead-acid battery," *IEEE Transactions on Energy Conversion*, vol. 26, pp. 1–4, 2011.

[30] T. Hansen and C.-J. Wang, "Support vector based battery state of charge estimator," *ScienceDirect - Journal of Power Sources*, vol. 141, pp. 351–358, 2005.

[31] J. C. A. Anton, P. J. G. Nieto, C. B. Viejo, and J. A. V. Vilan, "Support vector machines used to estimate the battery state of charge," *IEEE Transactions on Power Electronics*, vol. 28, no. 12, pp. 5919–5926, 2013.

[32] J. Álvarez Antón, P. G. Nieto, F. de Cos Juez, F. S. Lasheras, M. G. Vega, and M. R. Gutiérrez, "Battery state-of-charge estimator using the svm technique," *Applied Mathematical Modelling*, vol. 37, no. 9, pp. 6244–6253, 2013.

[33] A. Nuhic, T. Terzimehic, T. Soczka-Guth, M. Buchholz, and K. Dietmayer, "Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods," *Journal of Power Sources*, vol. 239, pp. 680–688, 2013.

[34] V. Surendar, V. Mohankumar, S. Anand, and D. P. Vadana, "Estimation of state of charge of a lead acid battery using support vector regression," *Procedia Technology*, vol. 21, pp. 264–270, 2015.

[35] Y. Shi, F. Xiong, R. Xiu, and Y. Liu, "A comparative study of relevant vector machine and support vector machine in uncertainty analysis," in *2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, 2013, pp. 469–472.

[36] D. Yang, Y. Wang, R. Pan, R. Chen, and Z. Chen, "State-of-health estimation for the lithium-ion battery based on support vector regression," *ScienceDirect - Applied Energy*, vol. 227, pp. 273–283, 2018.

[37] M. Andoni, W. Tang, V. Robu, and D. Flynn, "Data analysis of battery storage system," *24th International Conference & Exhibition on Electricity Distribution (CIRED)*, vol. 2017, no. June, pp. 96–99, 2017.

[38] H. Li, D. Pan, and C. L. P. Chen, "Intelligent prognostics for battery health monitoring using the mean entropy and relevance vector machine," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7, pp. 851–862, 2014.

[39] B. Saha, K. Goebel, S. Poll, and J. Christophersen, "Prognostics methods for battery health monitoring using a bayesian framework," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, pp. 291–296, 2009.

[40] J. Zhou, D. Liu, Y. Peng, and X. Peng, "An optimized relevance vector machine with incremental learning strategy for lithium-ion battery remaining useful life estimation," *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, no. 1, pp. 561–565, 2013.

[41] D. Liu, J. Zhou, and H. Liao, "A health indicator extraction and optimization framework for lithium-ion battery degradation modeling and prognostics," *EEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 6, pp. 915–928, 2015.

[42] D. Liu, J. Zhou, D. Pan, Y. Peng, and X. Peng, "Lithium-ion battery remaining useful life estimation with an optimized relevance vector machine algorithm with incremental learning," *Science Direct - Measurement*, vol. 63, pp. 143–151, 2015.

[43] W. Xian, B. Long, M. Li, and H. Wang, "Prognostics of lithium-ion batteries based on the verhulst model, particle swarm optimization and particle filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 1, pp. 2–17, 2014.

[44] L. Zhang, Z. Mu, and C. Sun, "Remaining useful life prediction for lithium-ion batteries based on exponential model and particle filter," *IEEE Access*, vol. 6, pp. 17 729–17 740, 2018.

[45] F. Zheng, J. Jiang, M. A. Zaidan, W. He, and M. Pecht, "Prognostics of lithium-ion batteries using a deterministic bayesian approach," *2015 IEEE Conference on Prognostics and Health Management (PHM)*, pp. 1–4, 2015.

[46] D. V. Do, C. Forgez, K. E. K. Benkara, and G. Friedrich, "Impedance observer for a li-ion battery using kalman filter," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 3930–3937, 2009.

[47] Y. Wang, H. Niu, T. Zhao, X. Liao, L. Dong, and Y. Chen, "Contact-less li-ion battery voltage detection by using walabot and machine learning," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2019)*, 2019.

[48] J. Lai, D. Choa, A. Wu, and C. Wang, "Combining machine learning algorithms and an incremental capacity analysis on 18650 cell under different cycling temperature and soc range," *Conference on Power, Energy and Electrical Engineering (CPEEE 2022)*, 2020.

[49] B. Blanchard and W. Fabrycky, *Systems Engineering and Analysis, 5th ed.* Upper Saddle River, NJ: Pearson Hall, 2011.

[50] H. Srivastava, "Queueing theory," in *Encyclopedia of Physical Science and Technology (Third Edition)*, third edition ed., R. A. Meyers, Ed. New York: Academic Press, 2003, pp. 481–495. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0122274105006323

[51] Q. Yang, S. Sun, S. Deng, Q. Zhao, and M. Zhou, "Optimal sizing of pev fast charging stations with markovian demand characterization," *IEEE Transactions On Smart Grid*, vol. 10, no. 4, 2019.

[52] M. Ismail, I. S. Bayram, M. Abdallah, E. Serpedin, and K. Qaraqe, "Optimal planning of fast pev charging facilities," in *2015 First Workshop on Smart Grid and Renewable Energy (SGRE)*, 2015, pp. 1–6.

[53] F. Besnard, K. Fischer, and L. B. Tjernberg, "A model for the optimization of the maintenance support organization for offshore wind farms," *IEEE Transactions On Sustainable Energy*, vol. 4, no. 2, 2013.

[54] G. Antoniol, A. Cimitile, G. Di Lucca, and M. Di Penta, "Assessing staffing needs for a software maintenance project through queuing simulation," *IEEE Transactions on Software Engineering*, vol. 30, no. 1, pp. 43–58, 2004.

[55] S. Ahmad, I. Awan, and B. Ahmad, "Performance modelling of finite capacity queues with complete buffer partitioning scheme for bursty traffic," in *First Asia International Conference on Modelling Simulation (AMS'07)*, 2007, pp. 264–269.

[56] L. Feng, S. Ge, H. Liu, L. Wang, and Y. Feng, "The planning of charging stations on the urban trunk road," in *IEEE PES Innovative Smart Grid Technologies*, 2012, pp. 1–4.

[57] C. Kong, I. S. Bayram, and M. Devetsikiotis, "Revenue optimization frameworks for multi-class pev charging stations," *IEEE Access*, vol. 3, pp. 2140–2150, 2015.

[58] G. Li and X.-P. Zhang, "Modeling of plug-in hybrid electric vehicle charging demand in probabilistic power flow calculations," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 492–499, 2012.

[59] N. Kumar, T. Kumar, S. Nema, and T. Thakur, "A comprehensive planning framework for electric vehicles fast charging station assisted by solar and battery based on queueing theory and non-dominated sorting genetic algorithm-ii in a co-ordinated transportation and power network," *Journal of Energy Storage*, vol. 49, p. 104180, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X22002134

[60] D. Xiao, S. An, H. Cai, J. Wang, and H. Cai, "An optimization model for electric vehicle charging infrastructure planning considering queuing behavior with finite queue length," *Journal of Energy Storage*, vol. 29, p. 101317, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X19309053

[61] I. Zenginis, J. S. Vardakas, N. Zorba, and C. V. Verikoukis, "Analysis and quality of service evaluation of a fast charging station for electric vehicles," *Energy*, vol. 112, pp. 669–678, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544216308404

[62] "Naval S&T Strategy," Office of Naval Research, https://defenseinnovationmarketplace.dtic.mil, Tech. Rep., 2015.

[63] X. Muneret, V. Gobe, and C. Lemoine, "Influence of float and charge voltage adjustment on the service life of agm vrla batteries depending on the conditions of use," *Journal of Power Sources*, vol. 144, pp. 322–328, 2005.

[64] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Communications*, vol. 30, no. 2, pp. 169–190, 2017.

[65] A. Mathur and G. M. Foody, "Multiclass and binary svm classification: Implications for training and classification users," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 2, pp. 241–245, 2008.

[66] R. L. Ott and M. Longnecker, *An Introduction to Statistical Methods & Data Analysis*. 20 Channel Center Street Boston, MA 02210: Cengage Learning, 2016.

[67] R. Pirani and J. L. Cale, "A pattern recognition approach for enhancing lifecycle maintainability of battery systems," *2019 International Symposium on Systems Engineering (ISSE)*, pp. 1–8, 2019.

[68] ——, "Comparison of pattern recognition approaches for identification of failure prone battery cells," *2022 International Symposium on Systems Engineering (ISSE)*, pp. 1–8, 2022.

[69] B. S. Blanchard and W. J. Fabrycky, *Systems Engineering and Analysis, 5th ed.* 1 Lake Street, Upper Saddle River, New Jersey: Prentice Hall, 2011.

[70] M. Pastore and A. Calcagnì, "Measuring distribution similarities between samples: A distribution-free overlapping index," *Frontiers in Psychology*, vol. 10, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2019.01089

Appendix A

Modeling and Analysis Code

A.1 Simple Generalized Classifier MATLab Program

```
%-------------------------------------------------------------%
% Description: this script trains a simple binary classifier
%    (SBC), See Chapter 4.
%
% Code developed using MatLab
%
% Inputs:
%   - trainingdata.csv : feature data for N classes
% Outputs:
%   - D : data structure including
%         D.fw = feature weights
%   - Cunfusion matrix of test data
%
% Written by:
% Rudy Pirani
% Colorado State University
% Contact: rudy.pirani@colostate.edu
% James Cale, Ph.D.
% Colorado State University
% Contact: jcale@colostate.edu
%
```

```
% References: [12] S. Simske, "Meta-analytics: consensus
% approaches and system patterns for data analysis,"
% Massachusetts, MA: Morgan Kaufman, 2019.
%
% Revision Date: 19 April 2019
% Revision Notes:
%   - 19 April 2019: initial release
%-------------------------------------------------------------%

clear all; clc; close all;

%-------------------------------------------------------------%
%  Collect Feature Data for All Classes From Training Data
%-------------------------------------------------------------%

% import training data (skip first 2 lines of header)
Fa = csvread('./trainingdata/trainingdataClassA.csv', 2);
Fb = csvread('./trainingdata/trainingdataClassB.csv', 2);

NumSamples  = size(Fa,1);
NumFeatures = size(Fa,2);
NumClasses  = 2;

%-------------------------------------------------------------%
%   Compute Derived Features
%-------------------------------------------------------------%
```

```
Ic = combnk(1:NumFeatures,2);
% n choose NumClasses (indices of all combos)

% Class A
% add features for products of all combos
Fap = [];
for k = 1:size(Ic,1)
    Fap = cat( 2, Fap, Fa(:,Ic(k,1)).*Fa(:,Ic(k,2)) );
end

% add features for divisions of all combos
Fad = [];
for k = 1:size(Ic,1)
    Fad = cat( 2, Fad, Fa(:,Ic(k,1))./Fa(:,Ic(k,2)) );
end

% concatenate all feature data for Class A
Fa = cat( 2, Fa, Fap, Fad);

% Class B
% add features for products of all combos
Fbp = [];
for k = 1:size(Ic,1)
    Fbp = cat( 2, Fbp, Fb(:,Ic(k,1)).*Fb(:,Ic(k,2)) );
end

% add features for divisions of all combos
```

```
Fbd = [];
for k = 1:size(Ic,1)
    Fbd = cat( 2, Fbd, Fb(:,Ic(k,1))./Fb(:,Ic(k,2)) );
end

% concatenate all feature data for Class B
Fb = cat( 2, Fb, Fbp, Fbd);

%-------------------------------------------------------------%
%   Compute Sample Statistics
%-------------------------------------------------------------%

% Class A
mua  = mean(Fa);
siga = std(Fa);

% Class B
mub  = mean(Fb);
sigb = std(Fb);

% collect all means
MU  = cat(1,mua,mub);
SIG = cat(1,siga,sigb);

%-------------------------------------------------------------%
%   Order All Populations By Means
%-------------------------------------------------------------%
```

```
% obtain indices for sorted means
[ MUs, I] = sort(MU,'ascend');

%-------------------------------------------------------------%
%   Compute Critical Points
%-------------------------------------------------------------%

NumCpt = NumClasses - 1;
NumFeatures = size(MU,2);

CPt  = zeros( NumFeatures, NumCpt);
NSTD = zeros( NumFeatures, NumCpt);

% calculate CPt's and nSTD-CPt's
for m = 1:NumFeatures
        NSTD(m,1) = ( MU( I(2, m), m ) - MU( I(1, m), m ) ) /...
            ( SIG( I(2, m), m ) + SIG( I(1, m), m ) );
        CPt(m,1) = MU( I(1, m), m ) + NSTD(m,1)*SIG( I(1, m), m );
end

% obtain sums of nSTD-CPt's
sumNSTD = sum(NSTD,2);

% obtain indices for sorted nSTD-CPt's
[ ~, InSTD] = sort(sumNSTD,'descend');
```

```
%-------------------------------------------------------------%
%   Compute t-statistics and their p-values
%-------------------------------------------------------------%

% define inline functions for t-statistic
t = inline ( '(u1-u2)/(sqrt( (
    (n1-1)*s1^2+(n2-1)*s2^2)/(n1+n2-2))*sqrt((1/n1)+(1/n2))
    )', 'u1','u2','s1','s2','n1','n2');
p4tdist2T = @(t,v) (betainc(v/(v+t^2),v/2,0.5)/2);
% p-value for 2-tailed t-distribution

T  = zeros( NumFeatures, NumCpt);
P  = zeros( NumFeatures, NumCpt);

n1 = NumSamples;
n2 = NumSamples;
df = n1 + n2 - 2;

% calculate t-statistics
for m = 1:NumFeatures

       u2 =  MU( I(2, m), m );
       u1 =  MU( I(1, m), m );
       s1 = SIG( I(2, m), m );
       s2 = SIG( I(1, m), m );

       T(m,1) = t( u2, u1, s2, s1, n2, n1);
```

```
       P(m,1) = p4tdist2T(T(m,1),df);

end

%-------------------------------------------------------------%
%   Compare nSTD-CPt's and t-statistic p-values
%-------------------------------------------------------------%

TnP = [sum(NSTD,2) sum(P,2) ]

% obtain indices for sorted nSTD-CPt's
[ ~, ITnP] = sort(sum(P,2),'ascend');

%-------------------------------------------------------------%
%   Compute Weighting Factors
%-------------------------------------------------------------%

Wds1 = NSTD(ITnP(1:3),1);
Wsqr = Wds1.^2;
W = Wsqr / sum(Wsqr);

Icc = [ zeros(12,2); Ic; Ic;];

% display factor numbers and weights
W = cat(2,ITnP(1:3),W)

Icc(W(1:3),:)
```

```
% save testing + validation data statistics
MU_testing = MU; SIG_testing = SIG; W_testing = W;

save MU_testing MU_testing
save SIG_testing SIG_testing
save W_testing W_testing

%-------------------------------------------------------------%
%   Collect Feature Data for All Classes From Test Data
%-------------------------------------------------------------%

% import training data (skip first 2 lines of header)
Fa = csvread('./testingdata/testingdataClassA.csv', 2);
Fb = csvread('./testingdata/testingdataClassB.csv', 2);

NumSamples  = size(Fa,1);
NumFeatures = size(Fa,2);
NumClasses  = 2;

%-------------------------------------------------------------%
%   Compute Derived Features
%-------------------------------------------------------------%

Ic = combnk(1:NumFeatures,2);
% n choose NumClasses (indices of all combos)
```

```
% Class A
% add features for products of all combos
Fap = [];
for k = 1:size(Ic,1)

    Fap = cat( 2, Fap, Fa(:,Ic(k,1)).*Fa(:,Ic(k,2)) );

end

% add features for divisions of all combos
Fad = [];
for k = 1:size(Ic,1)

    Fad = cat( 2, Fad, Fa(:,Ic(k,1))./Fa(:,Ic(k,2)) );

end

% concatenate all feature data for Class A
Fa = cat( 2, Fa, Fap, Fad);



% Class B
% add features for products of all combos
Fbp = [];
for k = 1:size(Ic,1)

    Fbp = cat( 2, Fbp, Fb(:,Ic(k,1)).*Fb(:,Ic(k,2)) );
```

```
end

% add features for divisions of all combos
Fbd = [];
for k = 1:size(Ic,1)

    Fbd = cat( 2, Fbd, Fb(:,Ic(k,1))./Fb(:,Ic(k,2)) );

end

% concatenate all feature data for Class B
Fb = cat( 2, Fb, Fbp, Fbd);

%-------------------------------------------------------------%
%   Compute Sample Statistics
%-------------------------------------------------------------%

% Class A
mua  = mean(Fa);
siga = std(Fa);

% Class B
mub  = mean(Fb);
sigb = std(Fb);

% collect all means
```

```
MU  = cat(1,mua,mub);
SIG = cat(1,siga,sigb);

%-------------------------------------------------------------%
%   Ensure Data Has Not Drifted
%-------------------------------------------------------------%

load MU_testing
load SIG_testing

% define function for computing t-statistic
t = inline ( '(u1-u2)/(sqrt( (
    (n1-1)*s1^2+(n2-1)*s2^2)/(n1+n2-2))*sqrt((1/n1)+(1/n2)) )',
    'u1','u2','s1','s2','n1','n2');

% define function for computing p-value for t-statistic
%(two-sided)
p4tdist2T = @(t,v) (betainc(v/(v+t^2),v/2,0.5)/2);

T  = zeros( NumClasses, NumFeatures );
P  = zeros( NumClasses, NumFeatures );

n1 = NumSamples;
n2 = NumSamples;
df = n1 + n2 - 2;

% calculate t-statistics
```

```
for m = 1:NumClasses
    for n = 1:NumFeatures

        u2 =  MU( m, n );
        u1 =  MU_testing( m, n );
        s1 = SIG( m, n );
        s2 = SIG_testing( m, n );

        T(m,n) = t( u2, u1, s2, s1, n2, n1);
        P(m,n) = p4tdist2T(T(m,n),df);

    end
end

%-------------------------------------------------------------%
%   Compute z-values
%-------------------------------------------------------------%

% obtain dominant factors from testing
load W_testing;
criticalFactors = sort(W_testing(:,1),'ascend');

z = inline ( 'sqrt(n)*abs(x-mu)/sigma', 'x','mu','sigma','n');

% calculate z-values for Classes
for m = 1:length(criticalFactors)
    for n = 1:NumSamples
```

```
        zA_A(n,m) = z( Fa(n,criticalFactors(m)),
        MU_testing(1,criticalFactors(m)),
            SIG_testing(1,criticalFactors(m)), NumSamples );
        zB_A(n,m) = z( Fa(n,criticalFactors(m)),
        MU_testing(2,criticalFactors(m)),
            SIG_testing(2,criticalFactors(m)), NumSamples );

        zA_B(n,m) = z( Fb(n,criticalFactors(m)),
        MU_testing(1,criticalFactors(m)),
            SIG_testing(1,criticalFactors(m)), NumSamples );
        zB_B(n,m) = z( Fb(n,criticalFactors(m)),
        MU_testing(2,criticalFactors(m)),
            SIG_testing(2,criticalFactors(m)), NumSamples );

    end
end

%-------------------------------------------------------------%
%   Compute f(z) values and normalize
%-------------------------------------------------------------%

fzA_A = 1./zA_A.^2;
fzB_A = 1./zB_A.^2;

fzA_B = 1./zA_B.^2;
fzB_B = 1./zB_B.^2;
```

```
% collect f(zA), f(zB) for each feature - Class A
fz2_A = [fzA_A(:,1) fzB_A(:,1) ];
fz6_A = [fzA_A(:,2) fzB_A(:,2) ];
fz7_A = [fzA_A(:,3) fzB_A(:,3) ];

% collect f(zA), f(zB) for each feature - Class B
fz2_B = [fzA_B(:,1) fzB_B(:,1) ];
fz6_B = [fzA_B(:,2) fzB_B(:,2) ];
fz7_B = [fzA_B(:,3) fzB_B(:,3) ];

% normalize
for n = 1:NumSamples
   fz2_A(n,:) = fz2_A(n,:) / sum(fz2_A(n,:));
   fz6_A(n,:) = fz6_A(n,:) / sum(fz6_A(n,:));
   fz7_A(n,:) = fz7_A(n,:) / sum(fz7_A(n,:));

   fz2_B(n,:) = fz2_B(n,:) / sum(fz2_B(n,:));
   fz6_B(n,:) = fz6_B(n,:) / sum(fz6_B(n,:));
   fz7_B(n,:) = fz7_B(n,:) / sum(fz7_B(n,:));

end

% matrix of normalized f(z) values for each class
fzAn = [fz2_A fz6_A fz7_A];
fzBn = [fz2_B fz6_B fz7_B];
```

```
%-------------------------------------------------------------%
%   Weight f(z) values by features weights
%-------------------------------------------------------------%

W = W_testing(:,2);
[~,indices] = sort(W_testing(:,1),'ascend');
W = W(indices);

fzAw = [fz2_A*W(1) fz6_A*W(2) fz7_A*W(3)];
fzBw = [fz2_B*W(1) fz6_B*W(2) fz7_B*W(3)];

%-------------------------------------------------------------%
%   Sum weighted feature f(z) values for each class
%-------------------------------------------------------------%

for k = 1:NumSamples

    sumWa_A(k,1) = sum(fzAw(k,1:2:6));
    sumWb_A(k,1) = sum(fzAw(k,2:2:6));

    sumWa_B(k,1) = sum(fzBw(k,1:2:6));
    sumWb_B(k,1) = sum(fzBw(k,2:3:6));

end
```

```
swA = [ sumWa_A sumWb_A ];
swB = [ sumWa_B sumWb_B ];

%-------------------------------------------------------------%
%   Classify
%-------------------------------------------------------------%

SA = zeros(NumSamples, 1);
SB = zeros(NumSamples, 1);

for k = 1:NumSamples

   if (find(swA(k,:)==max(swA(k,:))) == 1)
      SA(k) = 1;
   end
   if (find(swB(k,:)==max(swB(k,:))) == 2)
      SB(k) = 1;
   end

end

S = [ SA SB ];

%-------------------------------------------------------------%
%   Testing Classifier on Testing Data
%-------------------------------------------------------------%
```

```
% overall accuracy with testing data
indices = find(S(:)==1);
Accuracy = length(indices)/length(S(:))

% build confusion matrix
for k = 1:NumSamples

   index = find(swA(k,:)==max(swA(k,:)));
   Ct(k,1) = index;
   index = find(swB(k,:)==max(swB(k,:)));
   Ct(k,2) = index;

end

for k = 1:NumClasses

   C(k,1) = length(find(Ct(:,k)==1));
   C(k,2) = length(find(Ct(:,k)==2));

end

C

% compute recalls
Ra = C(1,1)/sum(C(1,:));
Rb = C(2,2)/sum(C(2,:));
```

```
% compute precisions
Pa = C(1,1)/sum(C(:,1));
Pb = C(2,2)/sum(C(:,2));
```

A.2 SVM R Program

```
#------------------------------------------------------------
# Description: this script trains a simple binary classifier
#    (SVM), See Chapter 5.
#
# Code developed using R Studio and documented in R Markdown
#
# Inputs:
#   - trainingdata.csv : feature data for N classes
#   - testingdata.csv : feature data for N Classes
#
# Outputs:
#   - Confusion matrix of training Data
#
# Modifed by:
# Rudy Pirani
# Colorado State University
# Contact: rudy.pirani@colostate.edu
#
# References: R Documentation
# https://www.rdocumentation.org/packages/e1071/versions/1.7-9/topics/svm
#
```

```
# Revision Date: 8 March 2022
#-------------------------------------------------------------
```{r}
# Imported Libraries
library(e1071)
library(ggplot2)
library(factoextra)
library(caret)
library(kernlab)
```


```{r}
# Importing the training data set
dataset <- read.csv("C:/Users/b_pir/Desktop/SVM/...
    SVM Train May 2021.csv", quote = "'")
str(dataset)
```


```{r}
# Check training data has been received
dataset$Status <- factor(dataset$Status)
str(dataset)
```


```{r}
# Importing the Test data set
dataset2 <- read.csv("C:/Users/b_pir/Desktop/SVM/...
```
```

```
    SVM TestB May 2021.csv", quote = "'")
str(dataset2)
```

```{r}
# Check test data has been received
dataset2$Status <- factor(dataset2$Status)
str(dataset2)
```

```{r}
# Train SVM model
svm.model<- svm(formula = Status ~ ., data = dataset,...
    type = 'C-classification', kernel = 'linear', cost = 100, gamma = 1 )

# Test model with training data
svm.pred <- predict(svm.model, dataset[,1:9])
svm.pred
```

```{r}
# Output Confusion Matrix for training data
classificationTable=table(pred = svm.pred, true = dataset2$Status)
classificationTable
```

```{r}
```

```
# Determine optimal cost
costs = seq(from=0.005, to=1,by=0.005)
pseudor2 = double(length(costs))

for (c in 1:length(costs)){
  epsilon.svr = svm(Status ~ ., data=dataset,cost=costs[c], gamma = 1)
svm.pred <- predict(epsilon.svr, dataset2[,2:9])
classificationTable=table(pred = svm.pred, true = dataset2$Status)
correctRate[c] = sum(svm.pred==dataset2$Status)/length(dataset2$Status)
misRate[c]=1-correctRate[c]
}

Class_Error = misRate

plot(costs,Class_Error, type="l")
```

```{r}
# Print cost associated with minimum error rate
k=which.min(misRate)
print(costs[k])
```

```{r}
# Retrain model with new cost value
svm.model<- svm(formula = Status ~ ., data = dataset, type = 'C-classification',
    kernel = 'linear', cost = 0.95, gamma = 1 )
```

```
# Run model with test data
svm.pred <- predict(svm.model, dataset2[,1:9])
svm.pred
```

```{r}
# Test data confusion matrix
confusionMatrix(table(svm.pred, dataset2$Status))

```

A.3 LDA RStudio

```
#------------------------------------------------------------
# Description: this script trains a simple binary classifier
#    (LDA), See Chapter 5.
#
# Code developed using R Studio
#
# Inputs:
#   - trainingdata.csv : feature data for N classes
#   - testingdata.csv : feature data for N Classes
#
# Outputs:
#   - Test_Result_Mar 2021A.csv: Classification results
#
# Written by:
```

```
# Rudy Pirani
# Colorado State University
# Contact: rudy.pirani@colostate.edu
#
# Refernce: Linear Discriminant Analysis (LDA) 101,
# using R, Peter Nistrup - https://towardsdatascience.com/
#    linear-discriminant-analysis-ld#a-101-using-r-6a97217a55a6
#
# Revision Date: 8 March 2022
#
#-------------------------------------------------------------

```{r}
# Imported Libraries
library(plyr)
library(dplyr)
library(emmeans)
library(tidyverse)
library(broom)
library(BSDA)
library(epitools)
library(lawstat)
library(metafor)
library(car)
library(coin)
library(MASS)
```
```

```{r}
# Importing the training data set
MTrain <- read.csv("C:/Users/b_pir/Desktop/Train May 2021.csv", quote = "'")
str(MTrain)
```

```{r}
# Importing the Test data set
MTest <- read.csv("C:/Users/b_pir/Desktop/Test May 2021.csv", quote = "'")
str(MTest)
```

```{r}
# Convert data in matrix to factors
St <- factor(MTrain[,1])
str(St)
```

```{r}
# Format all training data
train.df <- as.data.frame(MTrain)
str(train.df)
```

```{r}
# Format all test data
test.df <- as.data.frame(MTest)
```

```
str(test.df)
```

```{r}
# Train lda model
MidChk.lda <- lda(Status ~ . , data = train.df)

# Apply test data to LDA model
Midchk.lda.predict <- predict(MidChk.lda, newdata = test.df)

# Add the classification to the new test matrix
Midchk.lda.predict$class

# Format assigned class values
Midchk.lda.predict.show <- as.data.frame(Midchk.lda.predict$class)

# Show classification of test data
MTest.show <- cbind(test.df, as.numeric(Midchk.lda.predict$class)-1)
colnames(MTest.show)[9] <- "Class"
```

```{r}
# Write data to new file
write.csv(MTest.show, "C:/Users/b_pir/Desktop/LDA/Test_Result_Mar 2021A.csv")
```

```{r}


```

```
# Show LD Function
MidChk.lda
```

A.4 PCALDA RStudio

```
#------------------------------------------------------------
# Description: this script trains a simple binary classifier
#    (LDA with PCA), See Chapter 5.
#
# Code developed using R Studio
#
# Inputs:
#   - trainingdata.csv : feature data for N classes
#   - testingdata.csv : feature data for N Classes
#
# Outputs:
#   - PCs for training and testing
#   - Plot of eigenvalues associated with the PCs
#   - Plot of cumulative variance of the PCs
#   - 2D plot of PCs 1 vs PC2
#   - 3D plot of PCs 1 vs PC2 vs PC3
#   - Output classification
#
# Written by:
# Rudy Pirani
# Colorado State University
```

```
# Contact: rudy.pirani@colostate.edu
#
# References: [49] A. Tharwat, T. Gaber, A. Ibrahim, and
# A. E. Hassanien, ‘‘Linear discriminant analysis: A
# detailed tutorial,’’AI Communications, vol. 30,
# no. 2, pp. 169-190, 2017.
#
# Refernce: Linear Discriminant Analysis (LDA) 101,
# using R, Peter Nistrup - https://towardsdatascience.com/
#    linear-discriminant-analysis-ld#a-101-using-r-6a97217a55a6
#
# Revision Date: 25 March 2022
#
#-------------------------------------------------------------

‘‘‘{r}
# Import Library
library(plyr)
library(dplyr)
library(emmeans)
library(tidyverse)
library(broom)
library(BSDA)
library(epitools)
library(lawstat)
library(metafor)
library(car)
```

```
library(coin)
library(MASS)
library(psych)
```

```{r}
MTrain <- read.csv("C:/Users/b_pir/Desktop/Train Jan 2021.csv", quote = "'")
str(MTrain)
```

```{r}
MTest <- read.csv("C:/Users/b_pir/Desktop/Test Jan 2021.csv", quote = "'")
str(MTest)
```

```{r}
# Create principal components (PC) and show summary information
MTrain.pr <- prcomp(MTrain[c(2:9)], center = TRUE, scale = TRUE)
summary(MTrain.pr)
```

```{r}
# Show PC coefficients and standard deviations
print(MTrain.pr)
```

```{r}
```

```
# Plot variance for the PCs
screeplot(MTrain.pr, type = "l", npcs = 8, main = "Screeplot of the first 8 PCs")
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),
       col=c("red"), lty=5, cex=0.6)

# Plot cumulative variance for PCs
cumpro <- cumsum(MTrain.pr$sdev^2 / sum(MTrain.pr$sdev^2))
plot(cumpro[0:8], xlab = "PC #", ylab = "Amount of explained variance",
    main = "Cumulative variance plot")
abline(v = 5, col="blue", lty=5)
abline(h = 0.9, col="blue", lty=5)
legend("topleft", legend=c("Cut-off @ PC5"),
       col=c("blue"), lty=5, cex=0.6)
```

```{r}
# Reduce PCs to be used to the first 5 PCs and attach the known classification
# for training
MTrain.pcst <- MTrain.pr$x[,1:5]
MTrain.pcst <- cbind(MTrain.pcst, as.numeric(MTrain$Status)-1)
colnames(MTrain.pcst)[6] <- "Status"
str(MTrain.pcst)
```

```{r}
# Transform training data using selected PC
```

```
train.df <- predict(MTrain.pr, newdata = MTrain)
train.df <- as.data.frame(MTrain.pcst)
str(train.df)
```

```{r}
# Transform test data using selected PC
test.df <- predict(MTrain.pr, newdata = MTest)
test.df <- as.data.frame(test.df)
test.df <- test.df[1:5]
str(test.df)
```

```{r}
# Train LDA classifier with PCs and transformed training data
MidChk.lda <- lda(Status ~ PC1 + PC2 + PC3 + PC4 + PC5, data = train.df)

# Classify test data using trained LDA model
Midchk.lda.predict <- predict(MidChk.lda, newdata = test.df)

# Pull assignments
Midchk.lda.predict$class

# Format assignment data
Midchk.lda.predict.show <- as.data.frame(Midchk.lda.predict$class)

# Show assignments of the test data
```

```
MTest.show <- cbind(MTest.pcst, as.numeric(Midchk.lda.predict$class)-1)
colnames(MTest.show)[8] <- "Class"
```

```{r}
# Write modeling results to file
write.csv(MTest.show, "C:/Users/b_pir/Desktop/Test_Result_Jan 2021I.csv")
```

A.5 Service Channel Optimization Finite Queuing Model

```
%-------------------------------------------------------------------------%
% Description: This evaluates the Number of Service Channels under control
% in both the simulated environment and theoretical as defined in section
% 10.6.3 of Blanchard. This follows the book to validate the code.
%
% Inputs:
%    - none
%
% Outputs:
%    - Table of Simulated Preventative Maintenance Polices
%    - Table of Theoretical Preventative Maintenance Polices
%
% Written by: James Cale & Rudy Pirani
% Systems Engineering Department
% Colorado State University
% Email: jcale@colostate.edu; pirani@colostate.edu
```

```
%
% Revision Date: 19 December 2021
%-------------------------------------------------------------------------%

clear all; clc; close all; format long;

% define fixed parameters
params.lambdab = (1/48); % base arrival rate, per unit [units/month]
params.mub     = 1/1;  % base service time, per unit, per channel [units/month]
params.Npop    = 60;   % number of units in the population
M              = 4;    % number of service channels



%define time array
dt = 1e-2;           % time increment [s]
t = 0:dt:100000;     % time array (secs in one day)
ArriveT = 0;         % the arrival time
ServiceT = 0;        % the sercive time completion

% initialize discrete state variables
n  = 0;     % total number of units in system
nb = 0;     % number of units in buffer
ns = 0;     % number of units in service

% allocate memory for saved states
B = zeros(1,length(t));     % array to store buffer length
N = zeros(1,length(t));     % array to store total units in system
```

```
Ns = zeros(1,length(t));    % array to store units in service
NB_S = zeros(1,length(t));  % array to store total of service and queue
Wc = zeros(1,length(t));    % array to store waiting cost
Sc = zeros(1,length(t));    % array to store service cost
Tc = zeros(1,length(t));    % array to store total cost

M_Table = zeros(1,length(M));   % array for channel number in table
B_Table = zeros(1,length(M));   % array for buffer length per # of channels
Ns_Table = zeros(1,length(M));  % array for units in service per # of channels
MB_sTable = zeros(1,length(M)); % array for service and queue per # of channels
Wc_Table = zeros(1,length(M));  % array for Wait Cost per # of channels
Sc_Table = zeros(1,length(M));  % array for Service cost per # of channels
Tc_Table = zeros(1,length(M));  % array for Total cost per # of channels
results = zeros(1,length(M));  % array for Total cost per # of channels

Arrival_Time = zeros(1,length(t));  % array of arrival times
Service_Time = zeros(1,length(t));  % array of service times compelted

%--------------------------------------------------------------------------
%                       Process Queue
%--------------------------------------------------------------------------

for i = 1:M

    params.M = i;

for k = 1:length(t)
```

```
% compute failure rate (i.e., from population into system)
lambda_n = (params.Npop-n) * params.lambdab;

% compute service rate (i.e., from service channel(s) into population)
if n <= params.M-1
    mu_n = params.mub * n;
else
    mu_n = params.M * params.mub;
end

% check for arrivals/deparatures
if rand <= dt*lambda_n % arrival from population
   n = n + 1;
   ArriveT = ArriveT + dt*lambda_n; % Incementing arrival times
   Arrival_Time(k) = ArriveT;       % Tracking arrival times
   Service_Time(k) = 0;             % No Service complete
elseif (rand <= dt*mu_n) && (n >= 1) % departure from service
   n = n - 1;
   Arrival_Time(k) = 0; % No Arrival
   ServiceT = ServiceT + dt*mu_n;
   Service_Time(k) = ServiceT;
end

% update state counters
if n == 0
    nb = 0;
```

```
        ns = 0;
    elseif (n >= 1) && (n<=params.M)
        nb = 0;
        ns = n;
    elseif (n>params.M)
        nb = n-params.M;
        ns = params.M;
    end

    % save states
    B(k) = nb;
    N(k) = n;
    Ns(k) = ns;         %added to track service numbers
    NB_s(k) = nb+ns;    %added to track queue and service numers
    Wc(k) = (nb+ns) * 100;    %added provides waiting cost
    Sc(k) = params.M * 844;  %added to track service cost
    Tc(k) = ((nb+ns) * 100) + (params.M * 844) + 1032; %add to track total cost

end

%--------------------------------------------------------------------------
%               Calculate Statistics and Compare to Theory
%--------------------------------------------------------------------------

% compute supplementary probability coefficents
C = zeros(1,params.Npop+1);
P = zeros(1,params.Npop);
```

```
for n = 0:params.M
    index = n + 1;
    C(index) = (factorial(params.Npop)/(factorial(params.Npop-n)...
        *factorial(n)))*(params.lambdab/params.mub)^n;
end
for n = params.M:params.Npop
    index = n + 1;
    C(index) = (factorial(params.Npop)/(factorial(params.Npop-n)...
        *factorial(params.M)*params.M^(n-params.M)))...
        *(params.lambdab/params.mub)^n;
end

% compute probability of no failures in system
P0 = 1/sum(C);

% compute probability of n>0 failures in system
for n = 1:params.Npop
   P(n) = P0 * C(n+1);
end

P = cat(2,P0,P);

% compute expected number of failures in system
n_hat = 0;

for n = 1:params.Npop
```

```
   n_hat = n_hat + P(n+1) * n
end

%find points where the queue is zero (no failures)
indices_nofailures = find(N(:)==0);

%find points where the queue has 4 failures (P_4)
indices_4failures = find(N(:)==4);

Sim_Table(i) = round(n_hat,2);
N_Table(i) = round(mean(N),2);

M_Table(i) = params.M;
B_Table(i) = round(mean(B),2);
Ns_Table(i) = round(mean(Ns),2);
NB_sTable(i) = round(mean(NB_s),2);
Wc_Table(i) = round(mean(Wc),2);
Sc_Table(i) = round(mean(Sc),2);
Tc_Table(i) = round(mean(Tc),2);
ServT(i) = round(mean(Service_Time),5);

end

disp(sprintf('Table Showing Simulated Data'));
T = table(M_Table', B_Table', Ns_Table', Wc_Table', Sc_Table', Tc_Table');
T.Properties.VariableNames = {'Number_of_Channels','Avg_Queue_Length',...
    'Avg_Being_Serviced','Waiting_Cost','Service_Cost','Total_Cost'}
```

```
[M,I] = min(Tc_Table);

Min_Channel = M_Table(I);

disp(sprintf('Table Showing Simulated vs Theoretical Data'));
T = table(M_Table', N_Table', Sim_Table');
T.Properties.VariableNames = {'Number_of_Channels',...
    'Simulated_Failures','Theoretical_Failure'}

disp(sprintf('Number of channels providing the least ...
    total cost (simulated): %1.0f', Min_Channel));
```

A.6 Service Time Optimization Finite Queuing Model

```
%-------------------------------------------------------------------------%
% Description: This evaluates the Mean service time under control
% in both the simulated environment and theoretical as defined in section
% 10.6.4 of Blanchard. This follows the book to validate the code.
%
% Inputs:
%    - none
%
% Outputs:
%    - Table of Simulated Preventative Maintenance Polices
%    - Table of Theoretical Preventative Maintenance Polices
%
```

```
% Written by: James Cale & Rudy Pirani
% Systems Engineering Department
% Colorado State University
% Email: jcale@colostate.edu; pirani@colostate.edu
%
% Revision Date: 19 December 2021
%-------------------------------------------------------------------------%

clear all; clc; close all; format long;

% define fixed parameters
params.lambdab = (1/60); % base arrival rate, per unit [units/sec]
params.mub     = 1/0.25;  % base service time, per unit, per channel [units/sec]
params.Npop    = 60;   % number of units in the population
M              = 1;    % number of service channels
S_T            = 5;    % maxiumum minutes per unit servissed

%define time array
dt = 1e-2;           % time increment [s]
t = 0:dt:100000;       % time array (secs in one day)
ArriveT = 0;         % the arrival time
ServiceT = 0;        % the sercive time completion

% initialize discrete state variables
n  = 0;     % total number of units in system
nb = 0;     % number of units in buffer
ns = 0;     % number of units in service
```

```
% allocate memory for saved states
B = zeros(1,length(t));     % array to store buffer length
N = zeros(1,length(t));     % array to store total units in system
Ns = zeros(1,length(t));    % array to store units in service
NB_S = zeros(1,length(t));  % array to store total of service and queue
Wc = zeros(1,length(t));    % array to store waiting cost
Sc = zeros(1,length(t));    % array to store service cost
Tc = zeros(1,length(t));    % array to store total cost
Fld = zeros(1,length(t));   % array to store number of units still operating
GrosP = zeros(1,length(t));   % array to store gross profit of operating units
NetP =  zeros(1,length(t));   % array to store net profit of operating units

S_T_Table = zeros(1,length(S_T)); % array for service time in table
M_Table = zeros(1,length(S_T));   % array for channel number in table
B_Table = zeros(1,length(S_T));   % array for buffer length per # of channels
Ns_Table = zeros(1,length(S_T));  % array for service length per # of channels
MB_sTable = zeros(1,length(S_T)); % array for servce & queue per # of channels
Wc_Table = zeros(1,length(S_T));  % array for Wait Cost per # of channels
Sc_Table = zeros(1,length(S_T));  % array for Service cost per # of channels
Tc_Table = zeros(1,length(S_T));  % array for Total cost per # of channels
Fld_Table = zeros(1,length(S_T));  % array for fielded untis
GrosP_Table = zeros(1,length(S_T));  % array for gross profit
NetP_Table = zeros(1,length(S_T));  % array for net profit
SF =  zeros(1,length(S_T));  % array service Factor

Arrival_Time = zeros(1,length(t));  % array of arrival times
```

```
Service_Time = zeros(1,length(t));  % array of service times compelted



%------------------------------------------------------------------------
%                       Process Queue
%------------------------------------------------------------------------

for i = 1:S_T

    params.M = M;



for k = 1:length(t)

    % compute failure rate (i.e., from population into system)
    lambda_n = (params.Npop-n) * params.lambdab;

    % compute service rate (i.e., from service channel(s) into population)
    if n <= params.M-1
        mu_n = ((params.mub)/i) * n;
    else
        mu_n = params.M * ((params.mub)/i);
    end

    % check for arrivals/deparatures
    if rand <= dt*lambda_n % arrival from population
```

```
   n = n + 1;
   ArriveT = ArriveT + dt*lambda_n; % Incementing arrival times
   Arrival_Time(k) = ArriveT;       % Tracking arrival times
   Service_Time(k) = 0;             % No Service complete
elseif (rand <= dt*mu_n) && (n >= 1) % departure from service
   n = n - 1;
   Arrival_Time(k) = 0; % No Arrival
   ServiceT = ServiceT + dt*mu_n;
   Service_Time(k) = ServiceT;
end

% update state counters
if n == 0
    nb = 0;
    ns = 0;
elseif (n >= 1) && (n<=params.M)
    nb = 0;
    ns = n;
elseif (n>params.M)
    nb = n-params.M;
    ns = params.M;
end

% save states

params.S = 1/((params.mub)/i);
```

```
    %+(161*((1-params.S)))//(161*((exp(params.S))/100))

    B(k) = nb;          % store buffer length
    N(k) = n;           % store total units in system
    Ns(k) = ns;         %added to track service numbers
    NB_s(k) = nb+ns;    %added to track queue and service numers
    Wc(k) = (nb+ns) * 100;    %added provides waiting cost
    Sc(k) = (182 / params.S)+(exp(1/(((params.S)^2)*2.5)));
        %added to track service cost
    Tc(k) = ((nb+ns) * 100) + (Sc(k)) + 960; %add to track total cost
    Fld(k) = params.Npop - (nb+ns);
        %add to number of fieled units still operating
    GrosP(k) = (params.Npop - (nb+ns))*100;
        % Gross profit of operaiting units
    NetP(k) = ((params.Npop - (nb+ns))*100) - (((nb+ns) * 100) + (Sc(k)));
        %Incldes Waiting and Service Cost

end

S_T_Table(i) = params.S;
M_Table(i) = params.M;
B_Table(i) = round(mean(B),2);
Ns_Table(i) = round(mean(Ns),2);
NB_sTable(i) = round(mean(NB_s),2);
Wc_Table(i) = round(mean(Wc),2);
Sc_Table(i) = round(mean(Sc),2);
Tc_Table(i) = round(mean(Tc),2);
```

```
ServT(i) = round(mean(Service_Time),5);
Fld_Table(i) =  round(mean(Fld),2);
GrosP_Table(i) =  round(mean(GrosP),2);
NetP_Table(i) = round(mean(NetP),2);

end

disp(sprintf('Table Showing Simulated Data'));
T = table(S_T_Table', NB_sTable', Wc_Table', Sc_Table', NetP_Table',...
    Tc_Table');
T.Properties.VariableNames = {'Service_Time','Queue','Wait_Cost',...
    'Service_Cost','Net_Profit','TC_Cost'}

[M,I] = max(NetP_Table);

Max_Net_Profit = S_T_Table(I);

disp(sprintf('Number of channels providing the maximum net profit...
    (simulated): %1.0f', Max_Net_Profit));
```