A Software Development Framework for Complete Battery Characterization: Testing, Modelling & Parameterization

Rioch D'lyma

McMaster University

A Software Development Framework for Complete Battery Characterization: Testing, Modelling & Parameterization

By
RIOCH D'LYMA

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR

THE DREGREE OF MASTER OF APPLIED SCIENCE IN

MECHANICAL ENGINEERING

Master of Applied Science (2020) McMaster University

(Mechanical Engineering) Hamilton, Ontario,

Canada

Title: A Software Development Framework for Complete Battery Characterization: Testing, Modelling & Parameterization

Author: Rioch D'lyma,

B.Tech (Automotive & Vehicle Technology)

Supervisors: Dr. Saeid Habibi

Pages: xi, 111

Abstract

Advancements in batteries, microprocessors as well as an extra emphasis being put on the environment has pushed electric vehicles to the forefront of today. Despite the many benefits of electric vehicles, range anxiety and long charge times are hurdles to overcome. These shortfalls are a result of the current battery technology regardless of the many breakthroughs over the last decade. Lithium-ion Batteries and other modern chemistries pose a number of challenges in testing and research when compared to the traditional lead acid batteries. Current test systems fall short in providing a complete testing solution with. The focus of this thesis is to develop a complete software framework for battery characterization: testing, modelling and characterization to accompany battery testing hardware developed by D&V Electronics.

The first step in battery characterization, involves battery testing in order to obtain data. This required development of the test software and a number of battery tests, including: Charge and discharge, state of charge vs. open circuit voltage curve generation, Electro-Impedance Spectroscopy, and capacity test. Research was done in order to ensure developed test procedures lined up with that of other publications. All data from the testing data is logged to a central database, allowing for the second major development, the model framework.

The model framework is composed of seven different battery models that can be parameterized with the touch of a button, using data collected from the tester. It is a software framework that is meant to be expandable by abstracting the details of a

model from the tester. This allows for new models and parameterization techniques to be integrated into the software without the need of new software development.

Lastly, all development was used to do a battery characterization of a prismatic battery cell. All tests were conducted on a battery over two hundred cycles, followed by battery parameterization using the mode framework. The battery models were then used to simulate a US06 drive profile and compared to the same profile with measurements taken from the tester. With an average root mean square error of 8 millivolts, the battery characterization using the framework proved to be a success.

Acknowledgement

I would like to thank my supervisor Dr. Habibi for his help, guidance and support throughout this endeavor. Thanks, also to Cam Fisher and the entire CMHT group for their help. I am very grateful for the opportunity to have worked with Dr. Voiko Loukanov, whose confidence in my abilities and mentorship made this possible. I would also like to thank Michael Kelly, Kalina Loukanov, Pasquale Pompa and the entire D&V Electronics team for their guidance and support.

I am deeply grateful to my parents, Wendy and Lawrence and my sister, Nicole for their constant encouragement and support throughout my life. Their love and guidance allowed me to overcome many of the hurdles that stood before me. I would also like to express my gratitude to my extended family and friends for their constant encouragement.

Lastly, I thank my wife Spruha, who's love, support and inspiration make anything possible.

Table of Contents

List of Figures

1. Introduction

The first modern-day battery was created in the early 1800s followed by the first rechargeable battery in the mid-1800s, [1]. Despite its long history, mankind's dependence on batteries has been minimal until very recently. Recent breakthroughs in electric machines and microprocessor technology have created a high demand for mobile energy storage in the form of Lithium-Ion batteries in the automotive and aerospace sectors[2-6]. This high demand and the dedication of researchers continue to push the limits of this technology and enable its use in a variety of new technologies. Over the past decade, lithium-Ion batteries have more than doubled in capacity while also being reduced in price per kilowatt, [7]. Prices per kilowatt hour are expected to are expected to continue to fall rapidly over the next decade as can be seen in Figure 1-1.

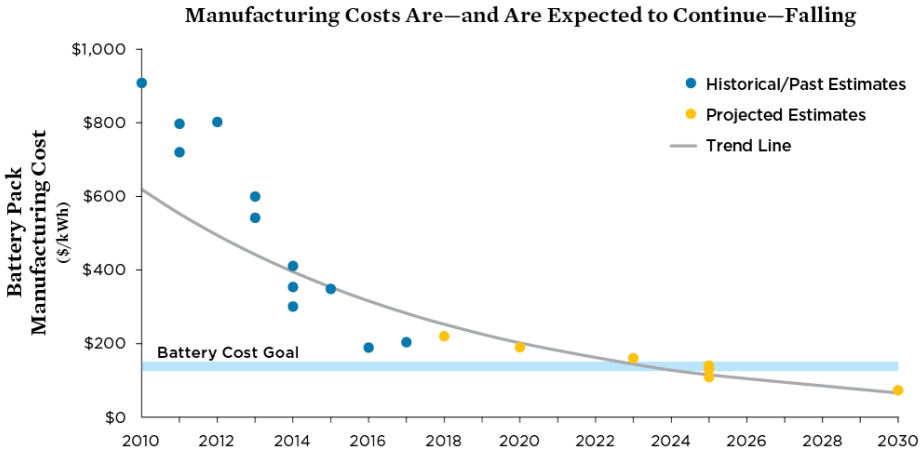

Figure 1-1: Battery Pack Manufacturing Cost Over Time[7]

The advancements in lithium-ion batteries have accelerated the development of electric vehicles at a time when an emphasis is being put on the sparing use of natural resources. Although range anxiety continues to be the number one issue with electric vehicles, sales are expected to continue to grow exponentially over the next decade, [8]. Moreover, while battery technology continues to advance, there are still a number of hurdles to be overcome.

Battery systems are currently the bottle neck for electric vehicles making them an important topic of research, by vehicle manufacturers as well as universities and independent laboratories. Alternative battery chemistries, thermal management, and control of battery systems are important research topics. A significant part of this research involves battery testing and gathering of data with tests lasting as long as months. However, the tools required to perform this research are severely lacking and often require manual work. This is the motivation for this Thesis; to develop the functionalities and software for a battery testing platform that eases data collection, automates battery model parametrization, and allows for much more complex test scenarios. The framework also allows users to run electro-impedance tests without user intervention resulting in superior battery characterization.

1.1. Electric Vehicle Overview

Electric vehicles improve on gasoline-powered vehicles in many different ways that benefit the consumers as well as their community and environment. Electric vehicles

date back to the 1800s, however over the past few years electric vehicle ownership continues to have year over year growth, [9]. While there are still many challenges, electric vehicles have a number of advantages that make them an appealing replacement for the conventional internal combustion vehicles.

A major motivation for the adoption of electric vehicles has been the environment as well as the world's quickly depleting storage of fossil fuels that currently power combustion engines, [10]. Electric vehicles are powered by the electric energy stored in their battery systems usually charged using the current electric grid. Well-to-wheel analysis or source to wheel for electric vehicles, seen in Figure 1-2, are done to accurately compare the efficiency of using an electric versus a gasoline vehicle, where the resources associated with the production of the fuel is also evaluated. This includes fuel production, from feedstock to the pump, as well as a vehicle's operation, tank to wheel, [11].

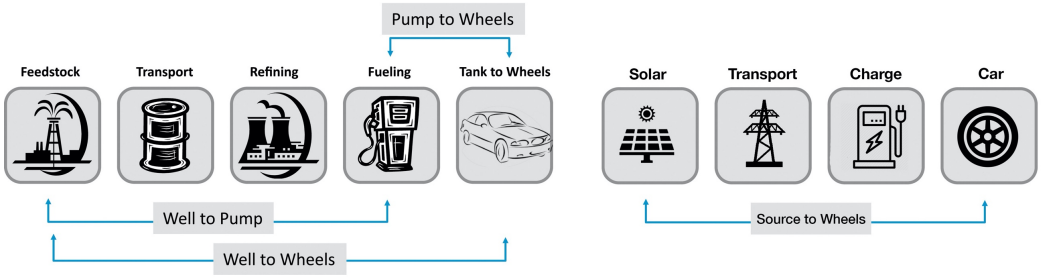

Figure 1-2: Well to Wheel & Source to Wheel Illustration

Conventional Vehicles with gasoline engines on average have a well to wheel efficiency of around 15%, whereas electric vehicles are capable of source to wheel efficiencies that could be as high as 70%, [12]. Their source to wheel efficiency and overall emissions is dependent on where the electric energy was produced. Electric

energy used to charge an electric vehicle battery, that originates from a coal factory results in a much higher source to wheel emissions than that of energy originating from solar panels. Their efficiencies also differ. As can be seen in Table 1, based on the current mixed sources of electricity production in Canada, on average, an electric vehicle operated in Canada is estimated to have 50% source to wheel efficiency. This along with the vastly reduced tailpipe emissions is one of the primary motivators for electric vehicle ownership.

	Well to Wheel Efficiency			
	w/ Ontario Electricity Grid	w/ Canada Electricity Grid	w/ USA Grid	w/ Residential Solar Power*
Electric	36%	51%	32%	70%
Gas	12%			
Diesel	18%			
Fuel Cell	25%			

Table 1: Well to Wheel Efficiency Comparison, [13]

Electric vehicles in many ways are also simpler systems overall, mechanically, electrically and concerning their control systems. In comparison to gasoline engines, they are made of a fraction of moving parts. The powertrain of an electric vehicle consists of very few elements when compared to the modern-day internal combustion vehicle. These systems include the battery pack, power inverter, electric motors, and transmission. However, while simpler, there are still complex hurdles that need to be overcome.

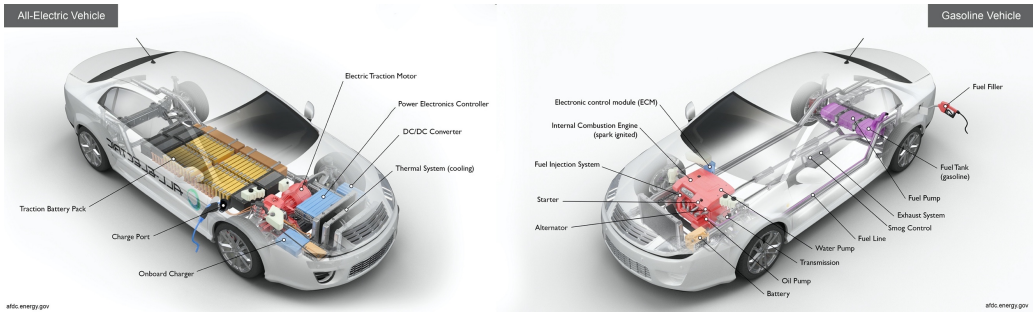

Figure 1-3: Gasoline vs. Electric Powertrain Components, [14]

The current drawback of electric vehicles originates from the battery packs that store the energy. The Tesla Model S is capable of 540 km's on a single charge, the most extended range of any other electric vehicle currently on the market. While a similar range to a combustion engine, the charging duration is a major setback as multiple hours may be required to recover much of the range. As a result, range anxiety along with the higher costs associated with electric vehicles is one of the obstacles in the widespread adoption of electric vehicles.

1.2. Battery Overview

The battery pack in an electric vehicle can be compared to that of the gasoline tank in an internal combustion engine in that it serves as a form of energy storage for the vehicle. Most electric vehicles currently use Li-Ion batteries due to their advantages over other chemistries including their high energy density, ability to deliver higher currents during a shorter period and average self-discharge over other chemistries, [15]. However, a shortcoming that spurs a significant area of research is related to the control of these battery systems, [16].

While each manufacturer does not publicize the exact chemistry of their batteries, many have decided on two types physically, cylindrical cells, Figure 1-5, and prismatic cells, Figure 1-4. Each battery pack is usually made up of a substantial number of these cells ranging from a few to as many as 7,100 cylindrical cells as found in a Tesla battery pack.

Figure 1-4: Cylindrical Battery Cell

Figure 1-5: Prismatic Battery Cell

Both of these types of cells are internally made up of 3 elements; an anode, cathode, and an electrode. The anode and cathode are separated by the electrode and connected through the load. Lithium ions flow from the anode to the cathode through the electrolyte which causes electrons to flow from the anode to the cathode through the external circuit resulting in power generated. This process is reversed when the battery is being charged.

Unlike a gasoline tank in internal combustion engines, batteries do not have any sensor that is able to determine their state of charge or state of health quickly. The terminal voltage of a battery, usually obtained once a battery is left in a non-operational state, is the only measurement able to provide an indication of a battery's state of

charge. This has resulted in the use of models to simulate a battery in order to estimate the state of charge in dynamic situations when a vehicle is in use. Other models are used in order to estimate a battery's state of health.

1.3. Battery Research/Testing Overview

Due to their complex nature, there are a number of factors to focus on when testing batteries. Traditional lead-acid batteries are usually quantified using three parameters: capacity, internal resistance and self-discharge. However, when placed in electric vehicles, Li-ion and other modern chemistries require a more in-depth look at batteries. In addition to this and the requirement for battery models, a lot of data is required from modern battery testing equipment as well as post-processing techniques.

Tests can range from minutes to as long as months in order to simulate battery aging over time. During this period, it is essential that tests are not interrupted, and data is easily accessible for continued processing throughout the test. One of the goals of battery testing in the research field is to collect data, in order to use for things like battery modeling, parameterization, and validation of new algorithms. Likewise, tests are also completed in order to evaluate the development of these new algorithms and to test their outcomes.

1.4. Motivation

While new battery chemistries are continually being tested, models continually improved, and controls repeatedly optimized, it is vital for researchers to have the right tools to test batteries, collect data and efficiently post process and parameterize battery models. Current battery test solutions are unable to provide the requirements in hardware and software to meet the steadily growing requirements of researchers.

1.5. Scope & Objective

As part of this thesis, a complete software platform was developed to complement battery testing hardware functionality, address data collection as well as close the gap between data collection and battery model parameterization. These objectives include:

I. Using a new test cell developed by D&V Electronics and McMaster for battery cell testing and developing software to implement new testing and characterization strategies.

II. Developing a comprehensive framework capable of bridging the gap between Simulink and the tester software. This includes:
    - Being able to take a Simulink model and easily parameterize battery models using data obtained from the above-mentioned battery cell tester.
    - Implementing a library of battery models to provide a starting point for users.

- Enabling easy integration of future models into the tester software.
- Using a genetic algorithm to parameterize models.
- Allowing for expansion for other optimization techniques in the future.
- Providing a comparative overview of the different battery models.

III. A tester software platform developed to work with hardware developed by D&V Electronics and capable off:

- Handling multiple tests simultaneously, up to 16 battery cells at a time.
- Managing automatic switching between multiple units.
- Customizing tests based on user profiles.
- Storing data on a remote server.

IV. Provide a suggested path forward for complete automation of Simulink models.

1.6. Outline

This thesis is organized into 7 Chapters that progressively goes through the process of design and development of the tools required for testing and development of modern battery cells, battery models and battery systems. Chapter 2 provides a literature review of used battery models as well as common techniques used for parameterizing these models. Chapter 3 provides an overview of the battery testing hardware jointly developed by D&V Electronics and McMaster researchers. It also provides an in-depth look at the software architecture, the development of communication drivers, and user interface design. Chapter 4 includes an overview of the model framework developed, along with implementation details with the battery tester software. Chapter 5 outlines the testing strategies developed, their implementation and goes over the results collected. Chapter 6 provides a summary of the contributions and recommendation for future research.

2. Review of Battery Characterization and Modelling

Battery testing, characterization and modelling play an important role with the development of new battery chemistries. Having a good understanding of the different types of models will assist in the integration of them into the software architecture. Whereas an understanding of battery testing mythologies will yield the creation of accurate testing methods.

2.1. Battery Modelling

Battery modeling is a crucial part of control and analysis of battery cells and packs as they become widely used. They are primarily used for state of charge and state of health estimation. Battery models have a wide range of complexity which has a direct link to their accuracy. Models are usually categorized under:

1. Behavioral Models.
2. Equivalent Circuit Models (ECM).
3. Electro-Chemical Models.

Each present a trade-off from the other in terms of complexity, accuracy or computational power required to evaluate the models.

2.1.1. Behavioral Models

Behavioral models are usually derived from measured data with little to no knowledge of the system. Their empirical roots allow them to be easily built upon for more accuracy at the cost of computational complexity, [17].

Behavioral battery models revolve around 3 of the most popular models that include, [18, 19]:

1- Shepherd:

$$y_k = E_0 - Ri_k - \frac{K_i}{z_k} \tag{2.1}$$

2- Unnewehr Universal Model:

$$y_k = E_0 - Ri_k - K_i z_k \tag{2.2}$$

3- Nernst:

$$y_k = E_0 - Ri_k + K_2 \ln(z_k) + K_3 \ln(1 - z_k) \tag{2.3}$$

Where:

- y_k is the terminal voltage;
- E_0 is the initial voltage;
- R is the internal resistance;
- z_k is the stored charge; and
- K_1, K_2, K_3 are constants used to fit the data.

The Nernst model has proven to provide the best accuracy amongst the few overall, however still underperforms in dynamic scenarios, [20]. The three models work best when combined as seen below.

Combined Model

The Combined Model incorporates the models defined above to produce a model more accurate than each individually. This model is represented by the following equations:

$$z_{k+1} = z_k - \left(\frac{\eta_i \Delta t}{C}\right) i_k \tag{2.4}$$

$$y_k = K_0 - Ri_k - \frac{K_1}{z_k} + K_2 z_k + K_3 \ln(z_k) + K_4 \ln(1 - z_k) \tag{2.5}$$

Using parameterization techniques, constants K_0, K_1, K_2, K_3 and K_4 can be obtained to fit any set of battery data.

Simple Model

The Simple Model is a simplification of the combined model, where the model is isolated into two separate parts, [17]. From the combined model, separating the state of charge from the current part, yields the following equations, [17]:

$$f(z_k) = K_0 - \frac{K_1}{z_k} + K_2 z_k + K_3 \ln(z_k) + K_4 \ln(1 - z_k) \tag{2.6}$$

$$f(i_k) = Ri_k \tag{2.7}$$

Further simplification results in the two equations below.

$$z_{k+1} = z_k - \left(\frac{\eta_i \Delta t}{C}\right) i_k \tag{2.8}$$

$$y_l = OCV(z_k) - Ri_k \tag{2.9}$$

One State Hysteresis Model

The combined and simple models can capture many of the battery dynamics but overlook the hysteresis experienced by the battery's voltage between charging and discharging. Some types of batteries experience a hysteresis effect, resulting in their voltage for a specific state of charge differing based on whether they are being charged or discharged, [21]. This can be witnessed by plotting the voltage of a battery in relation to its state of charge during charging and discharging as seen in Figure 2-1.

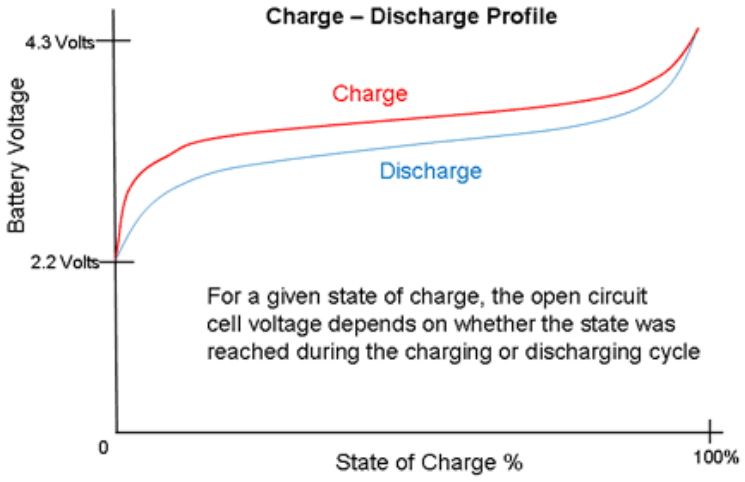

Figure 2-1: Hysteresis Effect During Charge/Discharge, [21]

A Zero-State Hysteresis model can be used to imitate this behavior, however, it is not able to capture the slow transition that the one-state hysteresis model is able to [19].

The Hysteresis effect can be modeled as follows [17]:

$$h_{k+1} = \exp\left(-\left|\frac{\eta_1 i(t)\beta}{C_n}\right|\right) h_k + \left(1 - \exp\left(-\left|\frac{\eta_1 i(t)\beta}{C_n}\right|\right)\right) M(z, \dot{z}) \tag{2.10}$$

$$h_{k+1} = F(i_k)\, h_k + (1 - F(i_k)) M(z, \dot{z}) \tag{2.11}$$

Where:

$$F(i_k) = \exp\left(-\left|\frac{\eta_1 i(t)\beta}{C_n}\right|\right) \tag{2.12}$$

Where the maximum hysteresis is represented by M(z,ż). The difference between M(z,ż) and h(z) represents the change in hysteresis voltage over time, proportional to the distance from the main hysteresis loop, [17].

Enhanced Self-Correcting Model

The enhanced self-correcting model excels where the previous models fall short at capturing all of a battery's dynamics. In addition to considering a battery's ohmic losses, hysteresis and polarization time constants, it is also able to capture the relaxation effect, [22]. The relaxation effect refers to the voltage response after a charge or discharge current is applied, as seen in Figure 2-2. The time taken for the voltage to transition to its steady state is referred to as relaxation time.

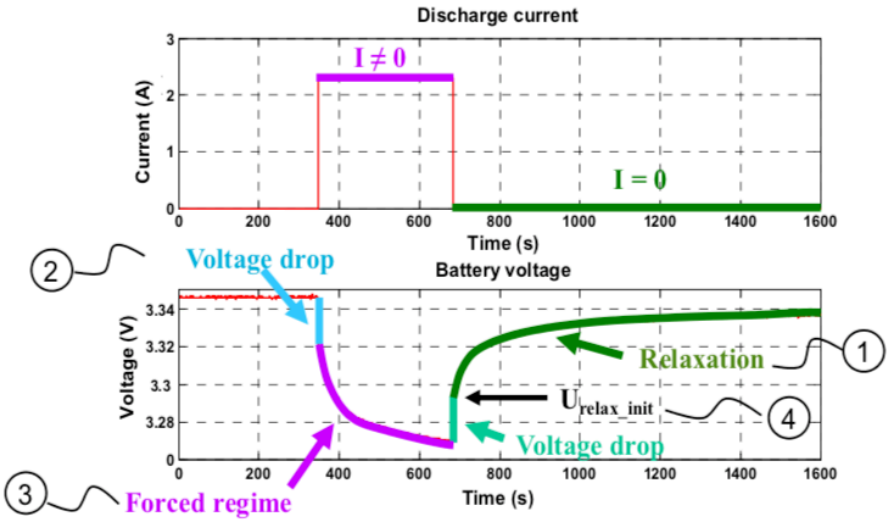

Figure 2-2: Voltage Response of a battery after discharge, [23]

The model can be represented by the following equations, [19]:

$$\begin{bmatrix} f_{k+1} \\ h_{k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} diag(\propto) & 0 & 0 \\ 0 & F(i_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_k \\ h_k \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} & 0 \\ 0 & (1 - F(i_k) \\ -\frac{\tau_i \Delta t}{C_n} & 0 \end{bmatrix} \begin{bmatrix} i_k \\ M(z, \dot{z}) \end{bmatrix} \quad (2.13)$$

$$\gamma_k = OCV(z_k) - Ri_k + h_k + Gf_k \quad (2.14)$$

Where:

- y_k is the terminal voltage;
- f_k is the low pass filter state of i_k;
- $\propto$ is the poles of the low pass filter;

- h_k is the hysteresis state;
- R is the internal resistance; and
- i_k is the battery current

2.1.2. Equivalent Circuit Models

Equivalent circuit models make use of electric circuit elements to simulate the complex electrochemical behavior of a Li-Ion battery, [24]. These models are used often due to their simplicity and ease of processing for real-time applications such as a battery management system, [22, 25]. There are a number of ways to implement equivalent circuit models, some of which are outlined below.

First Order RC Model

The most common equivalent circuit model is the first order R-RC models and acts as a base for second and third order R-RC models. The model, which can be seen in Figure 2-1 consists of a resistor in series with a resistor and capacitor in parallel. The model's simplicity makes it ideal for real-time applications while still mimicking most battery dynamics.

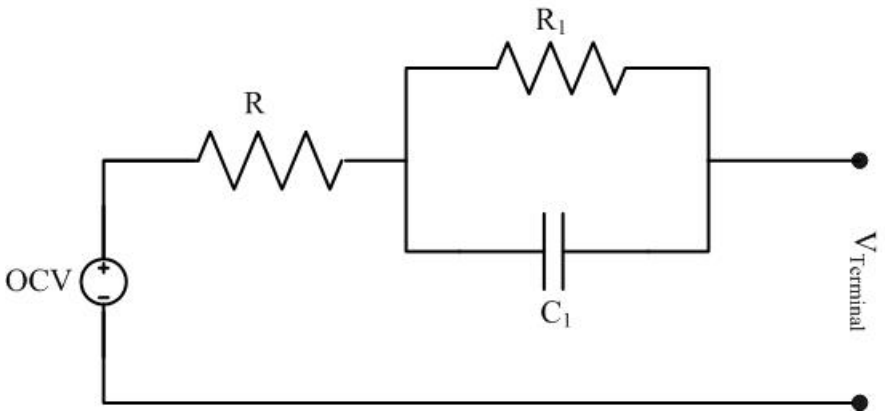

Figure 2-3: First Order RC Model

The model is represented by the following equations

$$\begin{bmatrix} V_{1,k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} 1 - \frac{\Delta t}{R_1 C_1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} \\ -\frac{\eta_1 \Delta t}{C_n} \end{bmatrix} i_k \tag{2.15}$$

$$\gamma_k = OCV(z_k) - Ri_k - V_k \tag{2.16}$$

The output, γ_k, represents the terminal voltage of the battery and the current input by i_k. The capacitance and resistance in parallel are stated as R_1 and C_1, with the series resistor represented by R.

First Order RC Model with Hysteresis

This R-RC model is further developed with the addition of a hysteresis state outlined in the previous section 2.1.1. This allows the model to capture the hysteresis effect observed when charging and discharging a battery.

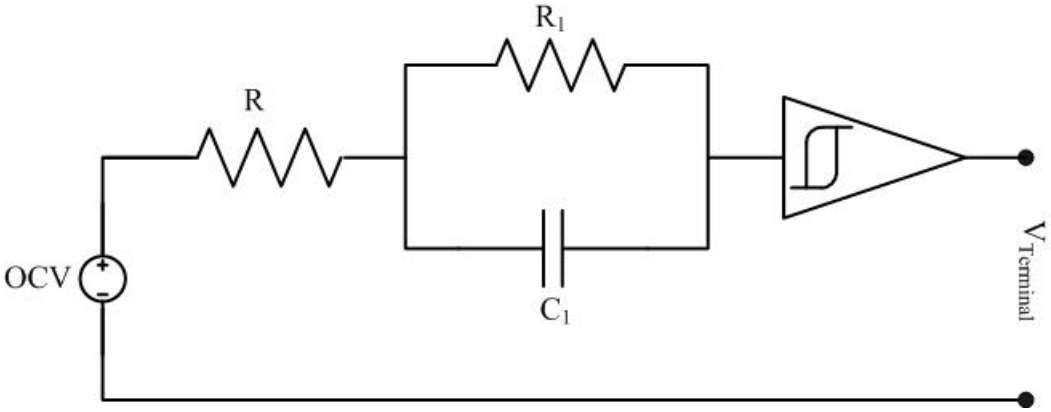

Figure 2-4: First Order RC Model with Hysteresis

$$\begin{bmatrix} V_{1,k+1} \\ h_{k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} 1 - \frac{\Delta t}{R_1\, C_1} & 0 & 0 \\ 0 & F(i_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ h_k \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} & 0 \\ 0 & (1 - F(i_k) \\ \frac{\eta_i \Delta t}{C} & 0 \end{bmatrix} \begin{bmatrix} i_k \\ M(z, \dot{z}) \end{bmatrix} \tag{2.17}$$

$$\gamma_k = OCV(z_k) - Ri_k - V_k + h_k \tag{2.18}$$

The formulas for this model are similar to the R-RC with the exception of the added hysteresis state, represented by h_k.

Second Order RC Model

The second order RC Model builds upon the first with the addition of an additional resistor and capacitor in parallel, as can be seen in Figure 2-5. The additional components further aid in modeling the dynamics of a battery's voltage recovery, [17].

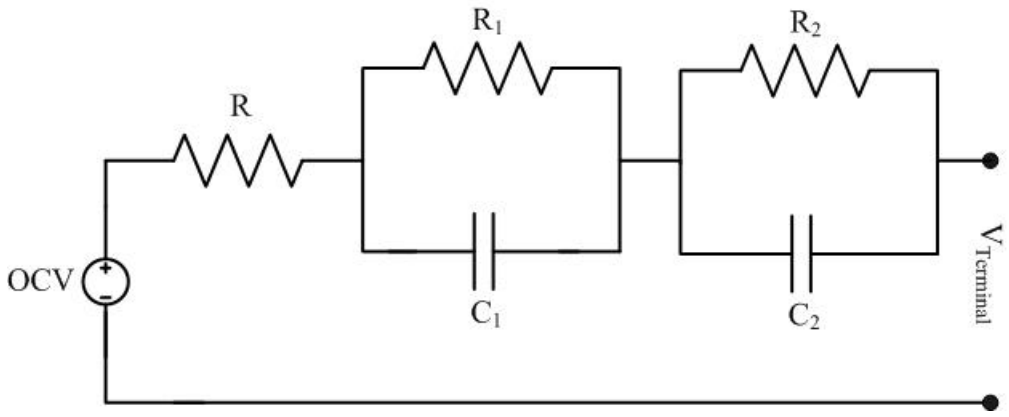

Figure 2-5: Second Order RC Model

The additional RC branch is represented by, $V_{2,k}$ in the equation below.

$$\begin{bmatrix} V_{1,k+1} \\ V_{2,k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} 1 - \frac{\Delta t}{R_1\, C_1} & 0 & 0 \\ 0 & 1 - \frac{\Delta t}{R_2\, C_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ V_{2,k} \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} \\ \frac{\Delta t}{C_2} \\ -\frac{\eta_i \Delta t}{C} \end{bmatrix} [i_k] \tag{2.19}$$

$$\gamma_k = OCV(z_k) - Ri_k - V_{1,k} + V_{2,k} \tag{2.20}$$

Second Order RC Model with Hysteresis

Similar to the first order RC model with hysteresis, this model encompasses the second order RC model described above with the addition of hysteresis.

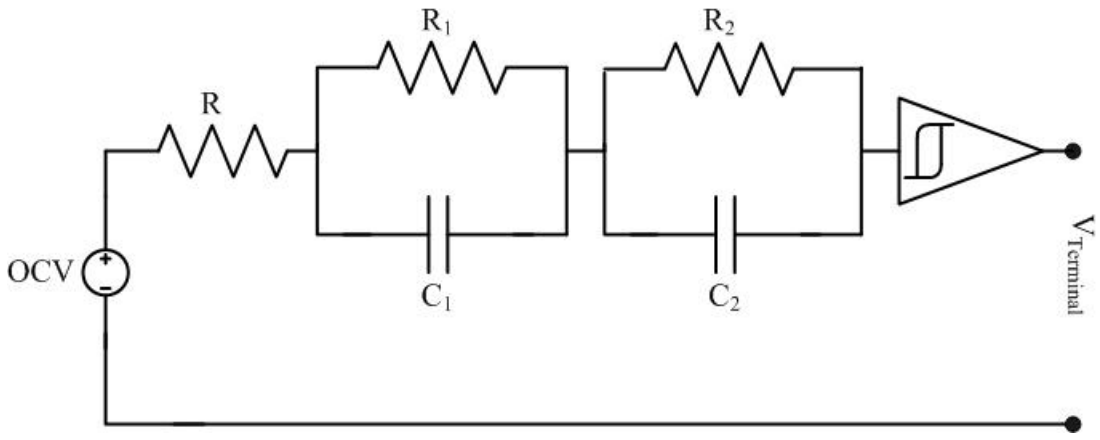

Figure 2-6: Second Order RC Model with Hysteresis

The hysteresis state is again represented by h_k in the equation representing the model.

$$\begin{bmatrix} V_{1,k+1} \\ V_{2,k+1} \\ h_{k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} 1-\frac{\Delta t}{R_1\,C_1} & 0 & 0 & 0 \\ 0 & 1-\frac{\Delta t}{R_2\,C_2} & 0 & 0 \\ 0 & 0 & F(i_k) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ V_{2,k} \\ h_k \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} & 0 \\ \frac{\Delta t}{C_2} & 0 \\ 0 & (1-F(i_k)) \\ -\frac{\eta_i \Delta t}{C} & 0 \end{bmatrix} \begin{bmatrix} i_k \\ M(z,\dot{z}) \end{bmatrix} \quad (2.21)$$

$$\gamma_k = OCV(z_k) - Ri_k - V_{1,k} - V_{2,k} + h_k \quad (2.22)$$

Third order RC Model

The third order RC model furthers the model's ability to capture the dynamics of the battery's voltage, able to better capture the slow and fast time constants, [17]. The addition of the third Resistor and Capacitor reportedly provides an improvement on the average RMS error of up to 2.8%, [26].

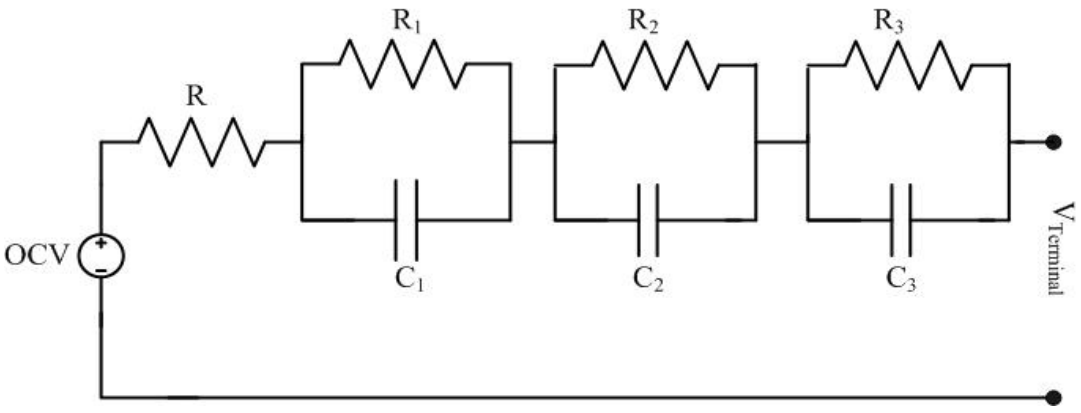

Figure 2-7: Third Order RC Model

The model's equivalent equations are represented below, with the addition of the voltage drop across $V_{3,k}$.

$$\begin{bmatrix} V_{1,k+1} \\ V_{2,k+1} \\ V_{3,k+1} \\ Z_{k+1} \end{bmatrix} = \begin{bmatrix} 1-\frac{\Delta t}{R_1\,C_1} & 0 & 0 & 0 \\ 0 & 1-\frac{\Delta t}{R_2\,C_2} & 0 & 0 \\ 0 & 0 & 1-\frac{\Delta t}{R_3\,C_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ V_{2,k} \\ V_{3,k} \\ Z_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{C_1} \\ \frac{\Delta t}{C_2} \\ \frac{\Delta t}{C_3} \\ -\frac{\eta_i \Delta t}{C} \end{bmatrix} [i_k] \quad (2.23)$$

$$\gamma_k = OCV(z_k) - Ri_k - V_{1,k} - V_{2,k} - V_{3,k} \quad (2.24)$$

Third Order RC Model with Hysteresis

Similar to the first and second order, a hysteresis state can be added to the third order model to take in to account its effect.

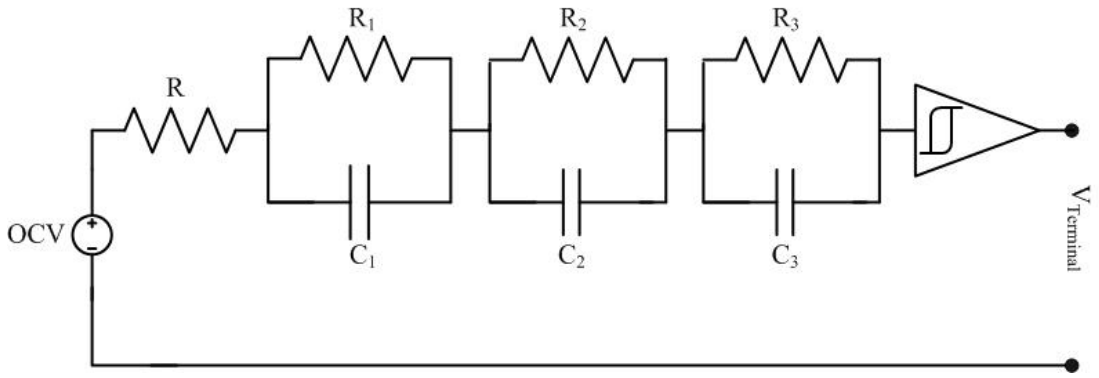

Figure 2-8: Third Order RC Model with Hysteresis

$$\begin{bmatrix} V_{1,k+1} \\ V_{2,k+1} \\ V_{3,k+1} \\ h_{k+1} \\ Z_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 1-\frac{\Delta t}{R_1\,C_1} & 0 & 0 & 0 & 0 \\ 0 & 1-\frac{\Delta t}{R_2\,C_2} & 0 & 0 & 0 \\ 0 & 0 & 1-\frac{\Delta t}{R_3\,C_3} & 0 & 0 \\ 0 & 0 & 0 & F(i_k) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,k} \\ V_{2,k} \\ V_{3,k} \\ h_k \\ Z_k \end{bmatrix} \tag{2.25}$$

$$+ \begin{bmatrix} \frac{\Delta t}{C_1} & 0 \\ \frac{\Delta t}{C_2} & 0 \\ \frac{\Delta t}{C_3} & 0 \\ 0 & (1-F(i_k) \\ -\frac{\eta_i \Delta t}{C} & 0 \end{bmatrix} \begin{bmatrix} i_k \\ M(z,\dot{z}) \end{bmatrix}$$

$$\gamma_k = OCV(z_k) - Ri_k - V_{1,k} - V_{2,k} - V_{3,k} + h_k \tag{2.26}$$

Impedance Model

In order for a model to accurately imitate the battery when doing an Electro-Impedance Spectroscopy, a different approach is required for modeling. While other models work in the time domain, the impedance model is required to imitate the battery's impedance characteristics in the frequency domain. This is accomplished

with the inclusion of up to two constant phase elements, [27], as well as what is referred to as the Warburg element.

$$Z_w = \frac{\sigma}{\omega^{\frac{1}{2}}} - j\frac{\sigma}{\omega^{\frac{1}{2}}} \tag{2.27}$$

The Warburg element is defined in equation 2.28, where σ is the Warburg coefficient and ω is the angular frequency. The constant phase element is usually added in series to the existing n[th] order RC Models as seen in Figure 2-9.

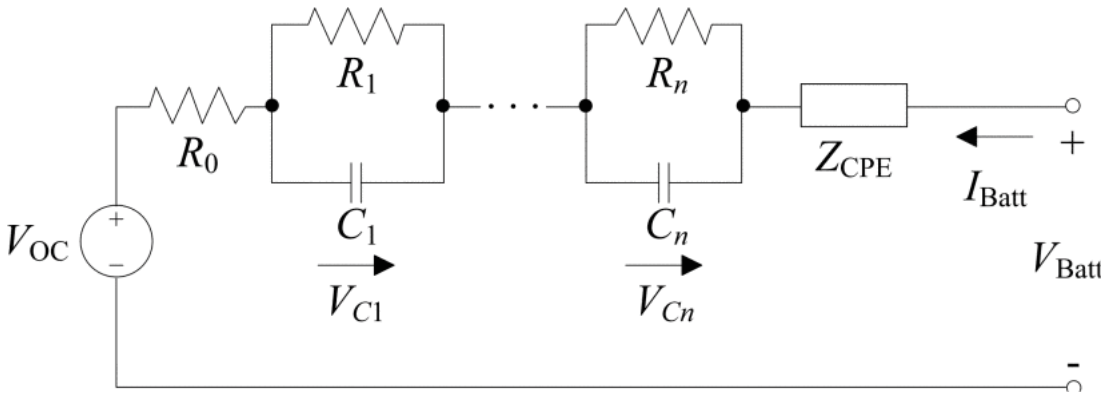

Figure 2-9: EIS Model with constant phase element[27]

2.1.3. Parameterization

While the battery models described in the previous section can be an excellent tool for state of charge and health estimation, they still require all the parameters to be defined in order to be functional. These parameters can vary between batteries of the same make, and finding the right values is not an easy task. Parameterization plays a huge roll in fitting these models correctly.

This is done by first collecting voltage, current and state of charge data from a physical battery. This data is then used in order to estimate the parameters of each model. Each model is simulated with the collected current data and the measured outputs (terminal voltage and derived state of charge) are compared to the model's simulated Voltage and state of charge output. Based on the error, an optimization strategy is used and would proceed to pick a new set of parameters. The optimization process is iterative and would continue until a stopping criterion, such as time, number of attempts, or percentage change are reached. One of the most commonly used optimization strategies for parameterization is the Genetic Algorithm as described below.

Genetic Algorithms

Genetic Algorithms are optimization strategies inspired by Charles Darwin's theory of natural evolution, [28]. They are based on chromosomes which are a representation of a solution to the given problem. Each chromosome is part of a larger population and feeds in a fitness function. The fitness function is used to evaluate the model or in this case the fitness of parameters associated with a particular chromosome. Based on the outcome of the fitness function and following an iterative search, the fittest chromosomes are chosen as members of a population to be retained and reproduce as part of a Selection step.

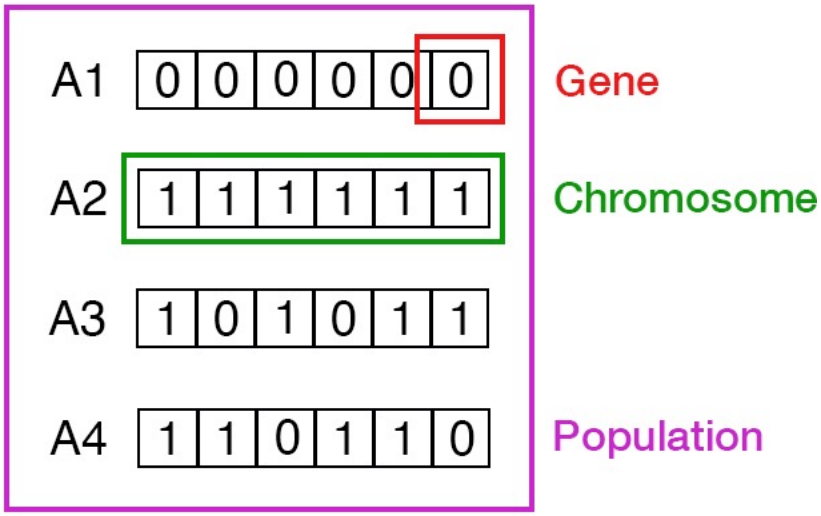

Figure 2-10: Gene, Chromosome And Population, [28]

Following their selection, these chromosomes are then used during repopulation to create what is called off springs. A crossover point is randomly chosen, and the offspring are then created through the exchanging of genes between the chromosomes. The crossover point is the red line in Figure 2-11 and the exchanging of genes witnessed by the green lines.

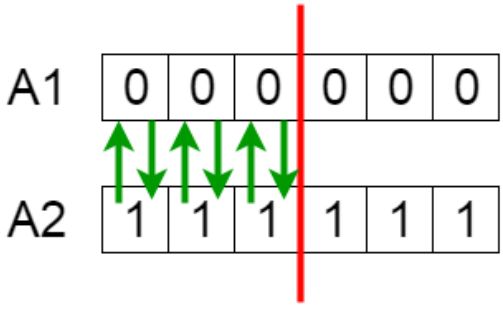

Figure 2-11: Chromosomes with Crossover Point, [28]

The introduction of diversity is through mutations. Bit-flip mutations, as seen in Figure 2-12, are among the most common, where one of the bits of the offspring is modified to produce the new offspring.

Figure 2-12: Bit-Flip Mutation

As such a new population is created and the process is repeated until the point of Termination. The Termination point can be defined as the point of convergence where there is not enough significant difference in terms of the best fitness level of the new population compared to the previous one. Other termination points are defined by the number of generations allowed as well as time[29]. The resulting chromosome is translated into parameters for the model.

2.2. Battery Testing

Battery testing techniques continue to be a heavily researched area with not many standard practices as of yet. Many testing techniques aim to estimate the impedance or resistance of the battery cell. Some of these tests are:

- Hybrid Pulse Power Characterization (HPPC)
- Electrochemical Impedance Spectroscopy (EIS)
- Coulombic Efficiency Test

Another test strives to provide an overview of a cell's performance during real-world situations by running a current load profile based on drive cycle profiles.

Lastly, aging tests attempt to capture battery degradation over time.

2.2.1. Hybrid Pulse Power Characterization Test

The Hybrid Pulse Power Characterization (HPPC) test provides a measure of the battery's impedance as well as the power characteristics of the cell at a specific point in time, [30]. The test involves a discharge pulse, followed by a rest and a charge pulse as seen in Figure 2-13. The magnitude of the charge, discharge and the length of the pulse vary between publications ranging from 3C, [31] up to 5C, [30].

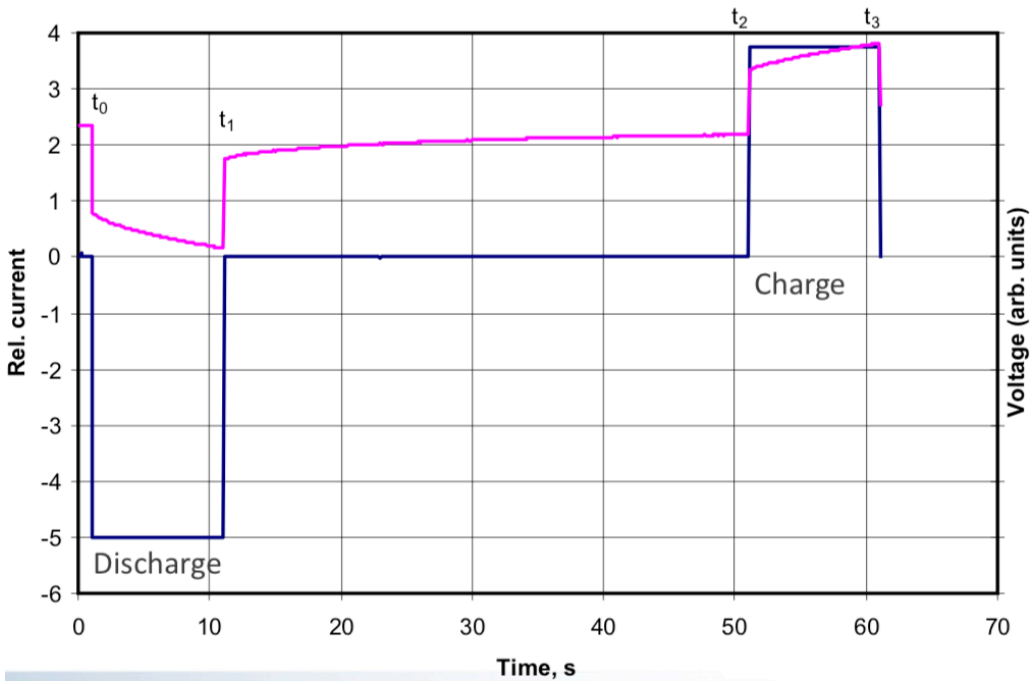

Figure 2-13: Hybrid pulse power characterization profile [32]

Based on the data collection, the impedance of the battery is calculated as well as the discharge power capability by the following equations[30, 32].

$$R = \frac{\Delta V}{\Delta I} = \frac{(V_{t1} - V_{t0})}{(I_{t1} - I_{t0})} \tag{2.28}$$

$$Discharge\ pulse\ power\ capability = \frac{V_{\min}\,(OCV - V_{min})}{R_{discharge}} \tag{2.29}$$

2.2.2. Electro Impedance Spectroscopy Test

An Electro Impedance Spectroscopy acts as another measure of impedance while providing additional data points that vary based on the state of charge, health, and temperature, [30]. The test is performed by imposing a sinusoidal signal at varying frequencies ranging between 0.02Hz up to 50kHz. The amplitude varies based on the type of cell and should be large enough that the voltage response of the current is observable, but small enough to operate the battery to within a piece-wise linear region. A Nyquist plot is used to display the results from an EIS test as seen in Figure 2-9. It compares three different cells from the beginning of life (BOL) to their end of life (EOL). The resistance of the cell is represented by the intercept on the x-axis.

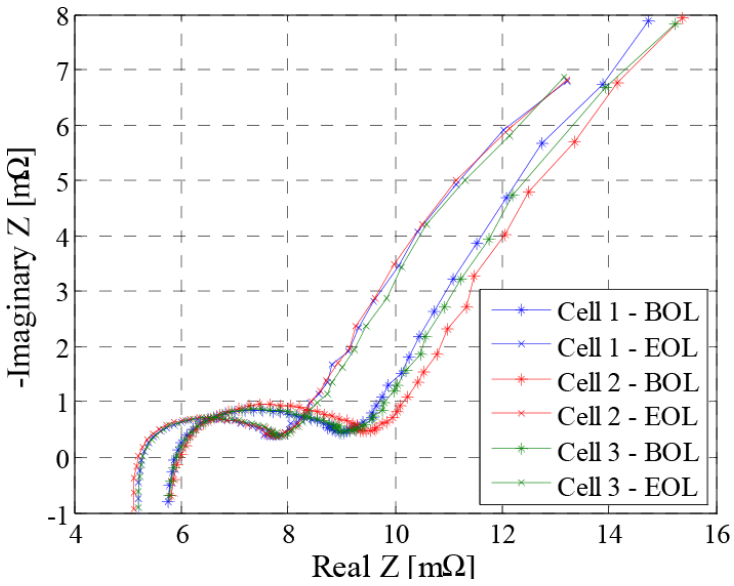

Figure 2-14: Nyquist Plot of EIS Data [33]

2.2.3. Coulombic Efficiency

Coulombic efficiency can be described as the ratio between the total charge put into a battery vs. the total charge extracted from a battery, [34]. In the perfect cell, free of defects or any side reactions, the coulombic efficiency would be 1. However as side reactions occur over time, the coulombic efficiency of a particular cell starts to lower along with its capacity indicating an aging cell, [19, 35]. This test is done by charging and discharging the battery at very low currents, usually C/20 or C/40, to charge or discharge the battery without having an effect on the voltage of the battery.

2.2.4. Drive Cycles

Drive cycle profiles are a set of speed profiles often used to asses a vehicle's fuel consumption as well as emissions. Likewise, these drive profiles are also utilized to assess the performance of electric vehicles and a vehicle's battery pack. Drive cycle

speed profiles are converted into current profiles using a vehicle model similar to the one in Figure 2-15. Running this current profile on a battery simulates the loads it would typically endure by the average driver.

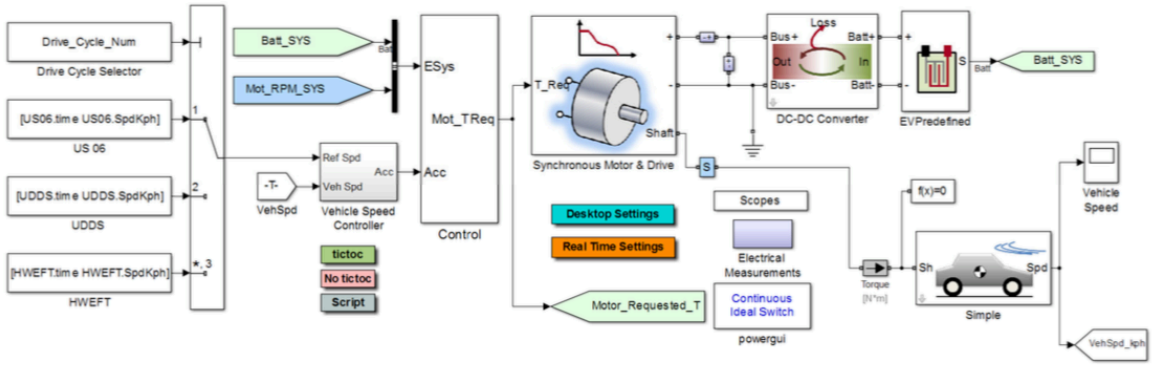

Figure 2-15: Electric Vehicle Model[19]

Two drive cycles in particular are used as benchmarks, namely the Urban Dynamometer Driving Schedule (UDDS) and the Highway Fuel Economy Driving Schedule (HWFET). The UDDS driving profile, as seen in Figure 2-16, represents the average vehicles city drive, [36]. Whereas the HWFET driving profile, as seen in Figure 2-17, represents highway driving conditions, [36].

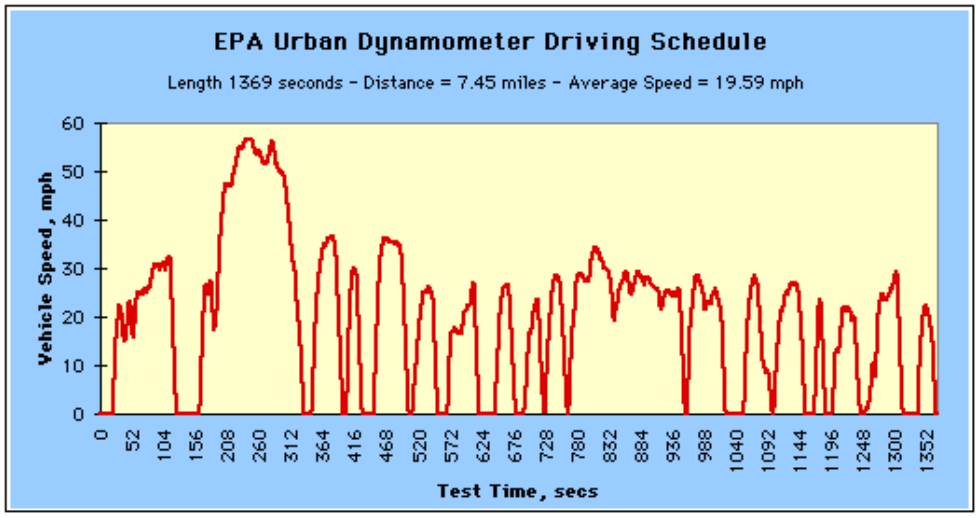

Figure 2-16: EPA Urban Dynamometer Driving Schedule(UDDS)[36]

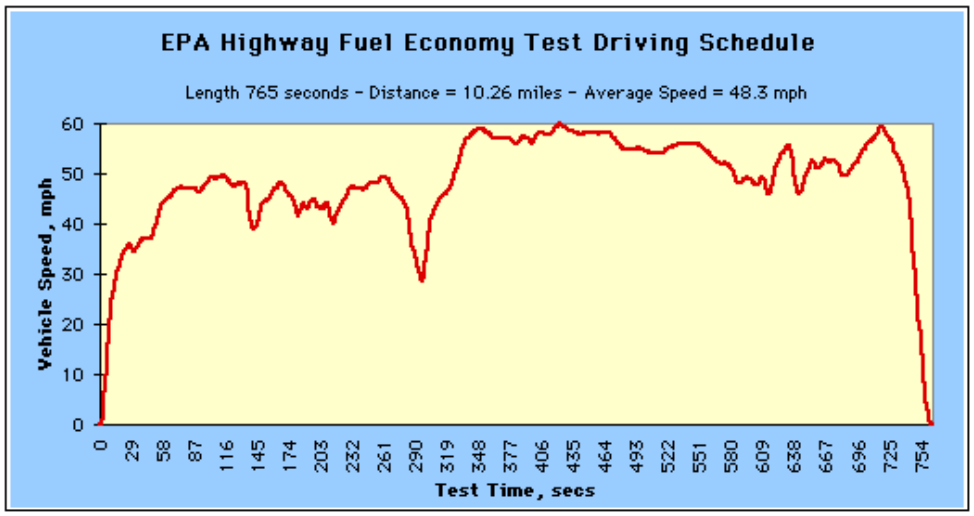

Figure 2-17: EPA Highway Fuel Economy Test Driving Schedule (HWFET) [36]

3. The D&V/McMaster Tester

Many of the models described in Section 2.1 and their parameterization depends solely on the testing strategies detailed in Section 2.2. A large number of the new testing strategies require specialized hardware and software that is capable of executing tests as per the user's request. The Center of Mechatronics and Hybrid Technologies (CMHT) at McMaster is leading the way for new state of charge and state of health estimation strategies. This resulted in a collaboration between D&V Electronics Ltd and this center for the production of an advanced battery cell tester here on referred to as the D&V Cell Tester. Under this collaboration, D&V Electronics produced the hardware for the tester according to the specification of CMHT and, CMHT provided its library of battery models, parameterization strategies, software algorithm expertise, and the software framework for its use within this tester. The aim of the tester is to create a commercial product capable of implementing new testing strategies that push the boundaries of research in the automotive industry.

3.1. Hardware

All hardware described in this section was designed and developed by D&V Electronics with functional specifications defined by The Center for Mechatronics and Hybrid Research (CMHT). The goal of the tester was to provide a complete testing solution for battery cells, providing both performance and automation currently not available on the market. The most basic requirements for the tester were to implement charge/discharge cycles, run drive profiles, and perform Electro-Impedance

Spectroscopy tests and coulombic efficiency tests without user intervention. Each of these tests requires dedicated hardware making it a challenge to include into a single module. This resulted in the creation of 3 separate modules referred to as: High Power Module, High Frequency Module and Ultra Precision Module, Combined, these modules are capable of fulfilling all the tests as outlined in Chapter 2.

3.1.1. High Power module

Drive profiles and charge/discharge cycles require very high-power outputs from the tester which lead to the creation of the 'High-Power' module, seen in Figure 3-1. Drive profiles are speed profiles that are meant to represent the driving conditions a vehicle would usually undergo. While there are a number of different tests meant to represent different parts of the world, in North America the two most significant drive profiles are the Urban Dynamometer Driving Schedule (UDDS) and the Highway Fuel Economy Driving Schedule (HWFET). It is important to simulate these tests to understand the dynamics that a regular driver would put the battery cells through. These profiles consist of large charge and discharge sections, as a result of heavy acceleration and heavy braking. When these profiles are run through a vehicle simulation model, the equivalent of the speed profile converted to a current profile is used to simulate the current drawn from a battery. These heavy accelerations and braking translate to large current draws and charge currents, something which the High-Power module must be able to emulate. Many modern batteries are capable of charge and discharge rates of more than 200Amp pulses. In order to keep the design

simple and allow for flexibility, the 'High-Power' Module is designed for loads of up to 100A as well as the ability to be stacked in parallel to allow for a larger range of currents. This allows the user to customize their tester unit based on the number of simultaneous tests they would like to run along with the amount of current each battery will be tested for. Due to restrictions with data handling, the total number of units per computer is restricted to 12 units. That means a maximum of up to 1200Amps or 12 simultaneous tests with a maximum of 100Amps each.

Figure 3-1: High Power Module

3.1.2. High-Frequency Module

Research surrounding Electro impedance spectroscopy requires current signals to be imposed at a very wide range of frequencies, ranging from 0.1Hz to 50Khz, which meant a separate High-Frequency Module was required for these tests. These tests involve injecting a current signal at varying frequencies and then calculating the

impedance at each frequency step. As the types of signals used for EIS test vary it was important that the module be capable of simulating customizable signals generated through third party software such as Matlab. By allowing its set point to be queued through USB communication, the signal is fully customizable through user formulae. It is capable of producing excitation signals up to 50Khz, with currents ranging from -5Amps to 5Amps. The module samples data at 200Khz, all streamed back to the computer for impedance calculations and post-processing. It is designed to connect in parallel across multiple High-Power Modules, allowing sharing of this module for multiple batteries in a single test system.

3.1.3. High Precision Module

Unlike the other two modules, the High-Precision Module's primary focus was high-accuracy current measurement in order to determine capacity and to address tests for Coulombic Efficiency calculations. Coulombic Efficiency is the ratio of the current charge taken out from the battery to the total put into the battery. Even after thousands of cycles the difference between the two can be minor, making the requirements for the current measurement accuracy to be very high. The High Precision Module is designed to measure current accurately with greater than ten parts per million (0.001%) with currents ranging from -5 Amps to 5 Amps. Data is sampled at two hundred kilohertz and averaged to fifty kilohertz within the module before being transmitted back to the computer for post-processing. Like the High-Frequency Module, the High

Precision module is also designed to connect in parallel across multiple High-Power Modules, allowing sharing of this module for multiple batteries in a single test system.

Figure 3-2: High Frequency & Ultra Precision Module

3.1.4. Automatic Switching

Current battery testing solutions require user intervention in order to perform Electro-Impedance Spectroscopy or Coulombic Efficiency while running drive profiles or cycling. Having to switch between devices for a test that can run for hours, weeks or months can limit the extent to which researchers can collect data. Allowing for automatic sequencing between the devices for testing allows for more in-depth analysis. Automatic sequencing will allow the user to gather data much more frequently and at many more points through a battery test cycle. This not only saves the user time but allows for much more in-depth research of Electro-Impedance Spectroscopy and coulombic efficiency.

Figure 3-3: Complete BCT Tester

3.2. Software

A crucial part of the test system is the Software, it acts as the gateway between the user and the hardware. With the hardware developed by D&V Electronics, the development of the software architecture, implementation and testing were part of this contribution. While functional requirements were an essential part of the software, a user-friendly user interface was a requirement as well. In order to enable the testing of batteries, the software package was responsible for:

- A User-friendly approach for editing test procedures.
- Running test procedures on all 12 channels simultaneously.
- Displaying live readings and current test progress to the user.
- Collecting measurement data through USB communication from all 12 channels.
- Storing data in memory space conscious way for easy access.

- Displaying, analyzing and post-processing the data.
- Remote notifications to the user, in case of emergency or test completion.

In addition to the above requirements, it was important that the software was computationally fast, and be capable of running for as long as the longest test. Due to a large number of requirements and variety of workflows, the software was broken up into a few different entities, namely: Drivers, Application, Data Acquisition and Post Processing as seen in Figure 3-4.

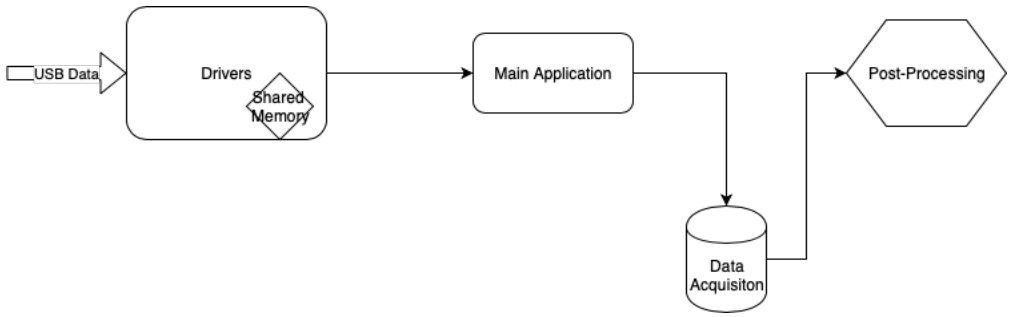

Figure 3-4: High Level View of Software Architecture

3.2.1. Drivers

Driver's in software are created to control low level interfaces and are designed to be simply, efficient programs. In this case, the drivers played the role of moving data from the hardware via USB and buffering it to shared memory. This played a crucial role in the software architecture and were one of the primary reasons for concern. Due to the demands of the specialized tests the system is capable of, this meant that each

module would need to transfer up to six hundred thousand samples per second to the computer and the computer to transfer three hundred and fifty thousand Voltage and Current set points per second to the device. The large number of set points is what opens the possibilities of user defined current signals during tests and the high sampling rate of the hardware results in the large amount of data sent back. However, this posed a major obstacle due to Windows Operating System being an inherently non-real time system. A separate application was required in order to handle data traveling to and from the modules.

In order for each module to execute set points as expected, they needed to be delivered to the module within eight milliseconds due to the hardware's onboard memory constraints. Limitation of the USB drivers in use meant a very short window for reading data from USB before it was overwritten. Ensuring that no data loss would occur within these limitations plays a crucial role in the battery testing system. Figure 3-5 shows a flow chart of the processing loop developed to accomplish the task of transferring data between the hardware and the software. In order to accomplish all the tasks seen in the workflow an understanding of Thread Priority and Shared Memory were required, as described in further detail below.

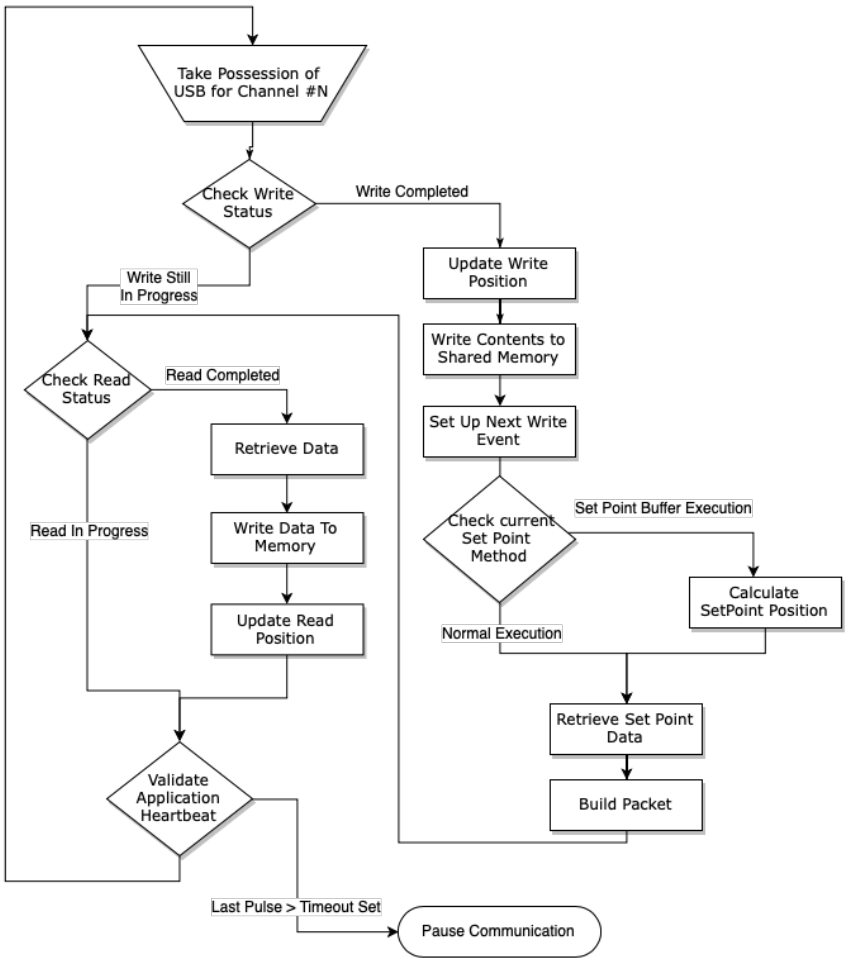

Figure 3-5: Driver Application Processing Loop

Thread Priority

In computer science, a thread is referred to as a sequence of instructions, [37]. Modern Operating Systems, such as Windows 10, are able to handle large number of parallel threads through the use of complex scheduling algorithms. The scheduler provides each thread a set amount of time to run on the processor based on a thread's

priority, [38]. A thread with a high priority will be allowed more time to execute before being interrupted. Thread priorities fall into either one of the following priority groups:

- THREAD_PRIORITY_ABOVE_NORMAL
- THREAD_PRIORITY_BELOW_NORMAL
- THREAD_PRIORITY_HIGHEST
- THREAD_PRIORITY_IDLE
- THREAD_PRIORITY_LOWEST
- THREAD_PRIORITY_NORMAL
- THREAD_PRIORITY_TIME_CRITICAL

The default priority for all application is 'Normal', [39], and for applications with a graphical user interface the highest thread privilege allowed is 'Above Normal'. Using this thread priority for the driver, would still make it very susceptible to the thread being preempted to allow higher privileged threads to execute. Once preempted, a thread is then pushed into a thread pool queue where it would be rescheduled, [38]. An overview of the thread states can be seen in Figure 3-6.

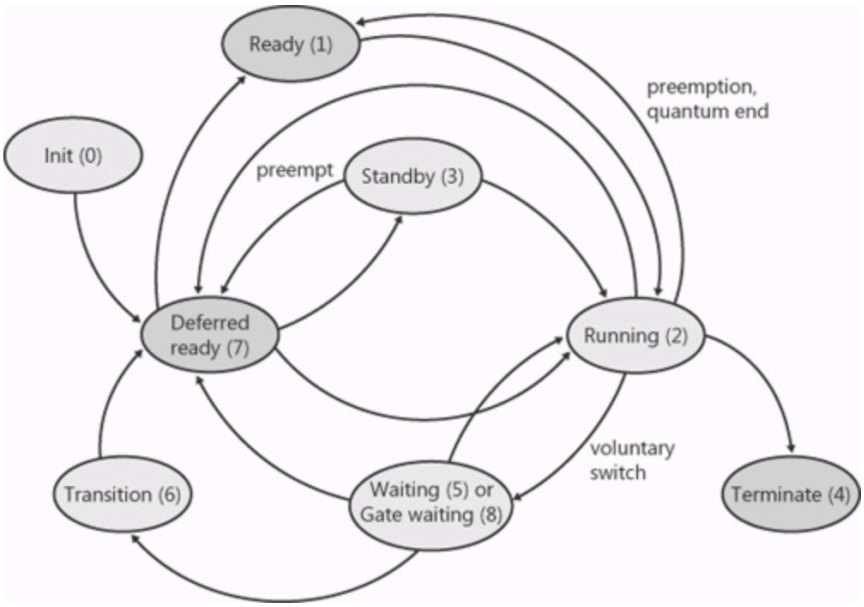

Figure 3-6: Thread States and Transitions, [38]

In Figure 3-7 an example of what the driver application demand might be on the processor. Right below that is the demand from a process with a higher thread privilege. An operating system's scheduler would give priority to the application resulting in the driver execution being preempted.

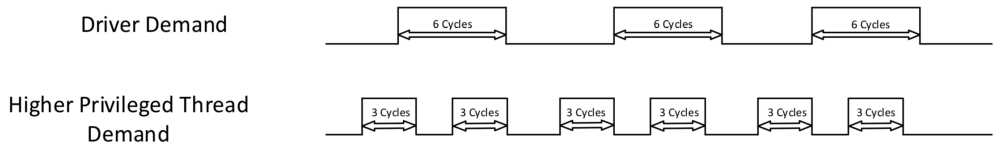

Figure 3-7: Driver Demand and Higher Thread Demand

In most consumer applications this is not a problem due to no timing requirement, meaning that there is no requirement for a particular set code to be run within a given

time. However, with the addition of the 8-millisecond time constraint due to limited memory in the hardware, it can be seen, in Figure 3-8, the shortfall of not being able to run at a higher thread privilege. The first line shows the ideal situation where the USB Driver is the only process running. The Driver required 6 cycles to meet its deadline and complete its work much before it's deadline. With the addition of higher privileged threads on the processor like on line two, the Driver thread gets interrupted. This results in a work load similar to line three where the USB Driver only gets five processor cycles before hitting its deadline. At this point data loss is experienced, resulting in a delay in set point execution. This was a frequent occurrence during early development of the communication between hardware and software.

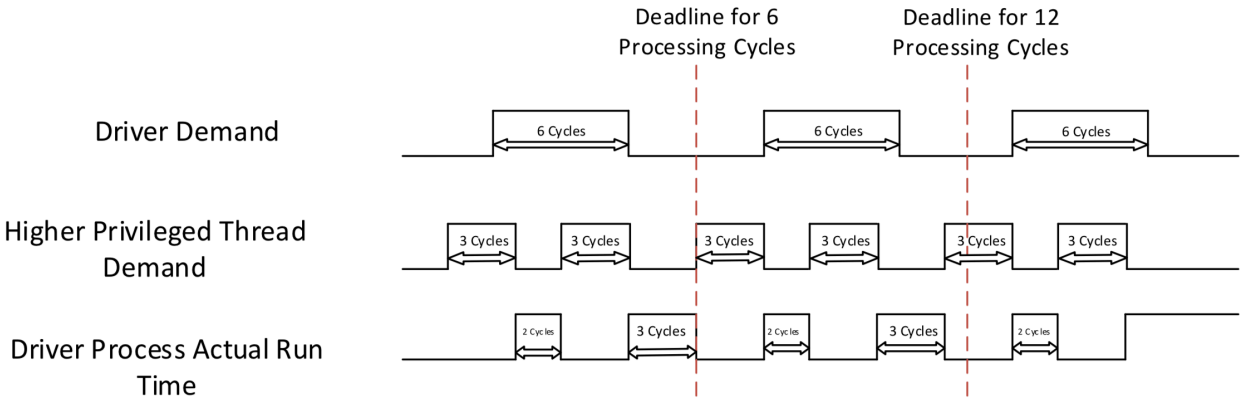

Figure 3-8: Missed Deadlines Due to Thread Priority

Figure 3-9 shows the output of a Scope that is monitoring the voltage setpoint on the hardware. The voltage is being set to 2 Volts, and the sudden drop to 0 Volts indicates that a set point was not delivered in time. A missed set point was observed on average once every 1.5 seconds, the missed set points were even more prominent when the processor was underload as seen in the left side.

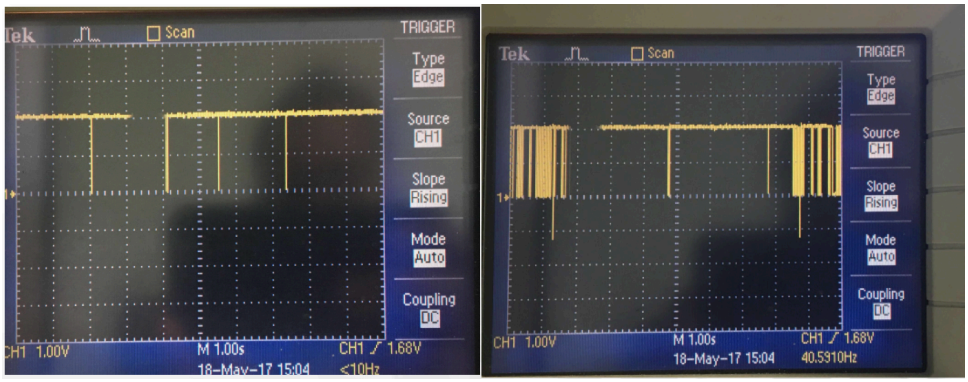

Figure 3-9: Voltage Set Point Data, Left: Processor Under No Load, Right: Processor Under Load

This interruption in set point execution resulted in inaccurate Electro Impedance Spectroscopy results as seen in Figure 3-10.

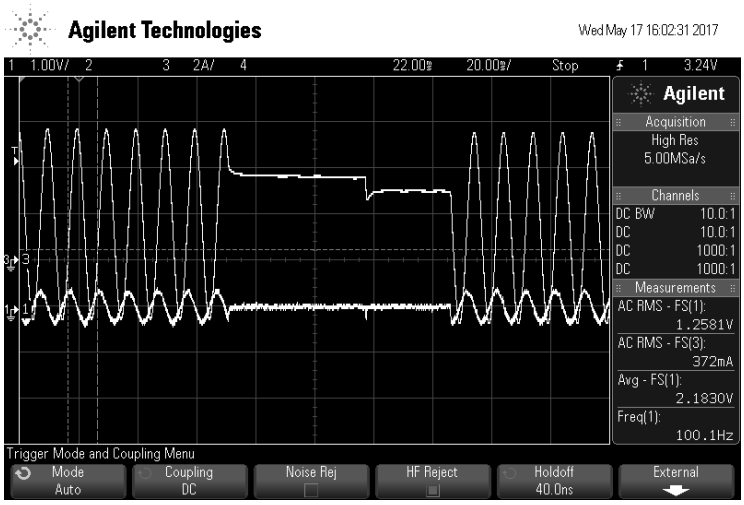

Figure 3-10: Current and Voltage Measurements during EIS test with missed set points

Elevating the privileges of the application thread to 'THREAD_PRIORITY_TIME_CRITICAL' allowed set point execution to operate as expected resulting in continuous sinusoidal profiles as seen in Figure 3-11.

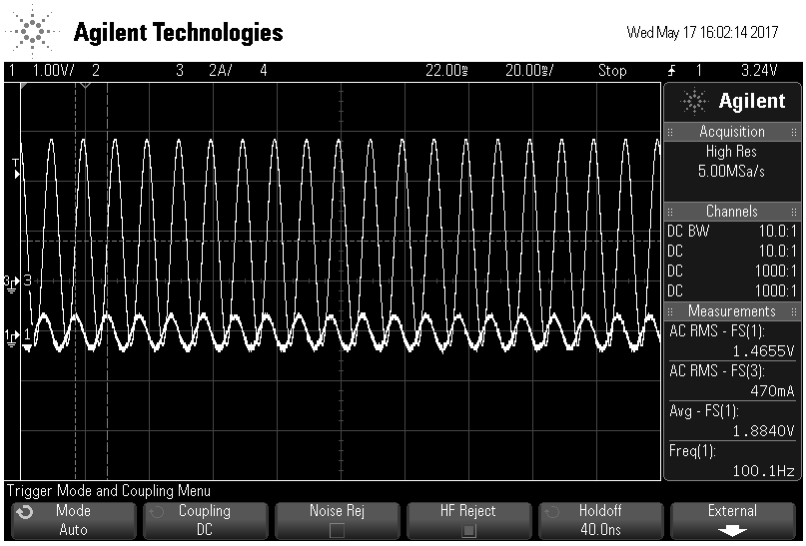

Figure 3-11: Current and Voltage Measurements during EIS test with no missed set points

Elevation of the thread allowed near perfect set point execution, however considerable care has to be taken with code executing on this thread to avoid system lock up. Other critical tasks are operated at this same thread privilege, include mouse input, keyboard input and disk flushing, [40]. Blocking these threads or any other critical ones below the driver could result in system failures or an unresponsive user interface. This played an important role in coding of the driver and careful thought in how it communicated with the main application. The driver required that there be no blocking calls or concurrent task in order to achieve this. Communication protocols such as TCP/IP or UDP were ruled out due to the nature of their two-way communication or no guarantee of data transmission. Instead the option of shared memory was decided on.

Shared memory

Shared memory refers to a memory space within the system that can be accessed by multiple threads or processes, [41]. In this case shared memory is used to communicate between the USB driver application and the main user interface. Windows' implementation of shared memory is referred to as memory mapped files, where an application is able to create a named file and other applications can access the same files with the right privileges, [42]. In order to maintain structure, five separate memory mapped files are created to facilitate data transfer as seen in Figure 3-12. Data flow through shared memory is indicated by arrows along with description of each files purpose below.

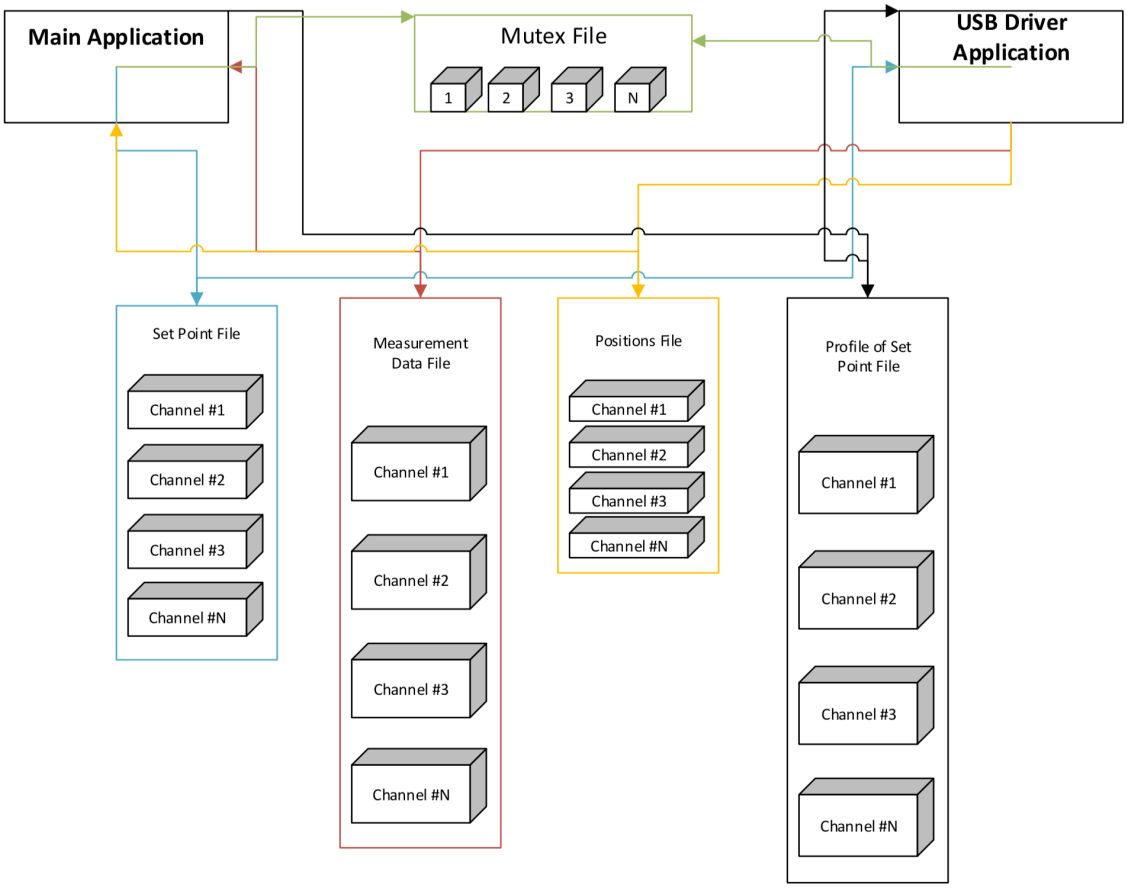

Figure 3-12: Memory Mapped File Overview

In addition to memory mapped files, Windows framework includes the ability to create memory mapped 'view accessors' which acts as a window into a section of the memory mapped file, as seen in Figure 3-13. The 'view accessors' provided an excellent approach that allowed each of the memory mapped files to be sectioned according to battery channel.

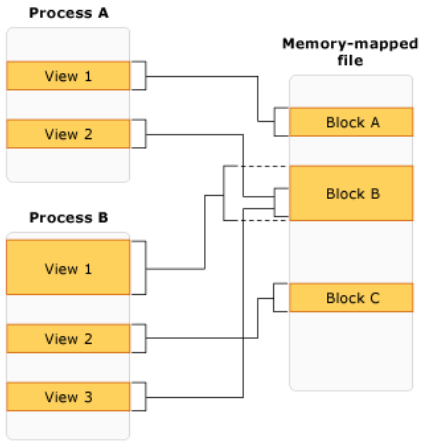

Figure 3-13: Memory Mapped File with multiple views, [42]

These view accessors allow for simpler implementation with a fewer number of memory mapped files. The accessor limits the access provided to a particular section of code, allowing for additional protection from programming errors where data is written to the wrong part of a shared memory file.

The first memory mapped file is responsible for sending individual set point information from the main user interface to the driver application. Its size is dictated by the set point structure which includes set point Voltage, Current, and Digital Outputs. When the user changes the voltage or current set point, the value would be written to this file. The driver application would then read the updated structure from the file prior to building a USB packet to send.

The second memory mapped file passes measurement data received from the hardware to the main application. This shared memory is treated as a circular array,

rewriting data at the beginning of the file once the end is reached as seen in Figure 3-14.

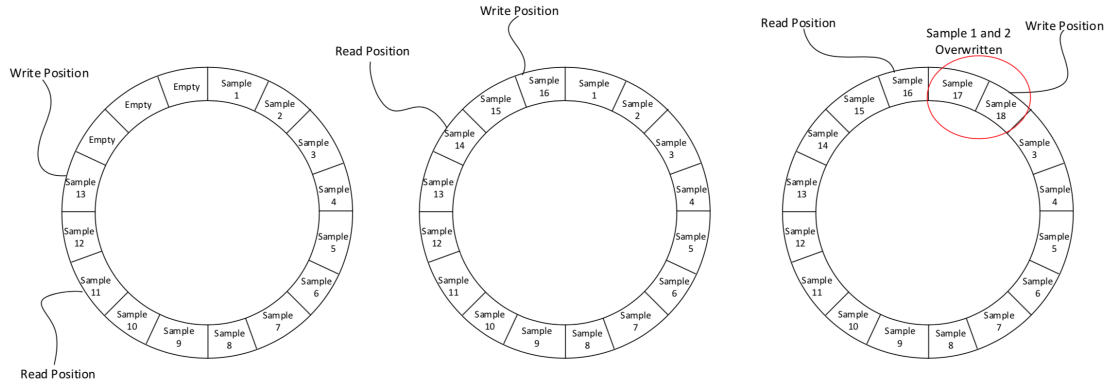

Figure 3-14: Circular Array Population Over Time

As the length of the memory mapped file reaches the end, the write position moves back to the first byte in the file. The driver always keeps track of the current position of data in the memory mapped file and works hand in hand with the third file which holds the read positions and write positions. The main application monitors this third memory mapped file for changes. Upon a write position increment, it knows where to look in the second memory mapped file for new measurement data. It subsequently updates it's read position and continues to wait for new data.

The fourth memory mapped file is used to facilitate the need for high frequency current profiles. As described above, the main application is unable to deliver set points fast enough to the hardware nor to the driver application. This memory mapped file acts as a large buffer, relieving the need for real time performance from the main application. The application is able to take user defined high-speed profiles and places

them in shared memory all at once before commanding the driver application to start sending the profile set points.

Lastly the mutual exclusion (Mutex) memory mapped file plays one of the most crucial roles when using inter-process communication by acting as a gate keeper. Its role is to ensure that each of the files are not accessed simultaneously by different threads preventing a thread from reading or writing data as that same data is being updated, [43]. Once a particular thread locks on to this object, every other thread is placed in a queue to access it. Each thread is only able to access all of the shared memory files described above by first taking ownership of the mutex.

3.2.2. Main Application

The Main Application is the primary User Interface that the user interacts with. It is responsible for providing the tools required to control up to sixteen battery channels to the user. These include:

- Editing test procedures
- Starting test
- Viewing current progress and
- general system settings related to hardware and software of the tester.

The main application posed many challenges due to the constant changing nature of the application. The user interface had to keep track of sixteen channels of data and update information based on the selected channel while still being lightweight and

responsive. As each channel was independent of each other, classes were developed with flexibility in mind to allow reuse across all channels. Four major developments as part of main application are the development of the state machine, the battery channel class, data driver class and the test functions, seen in Figure 3-15.

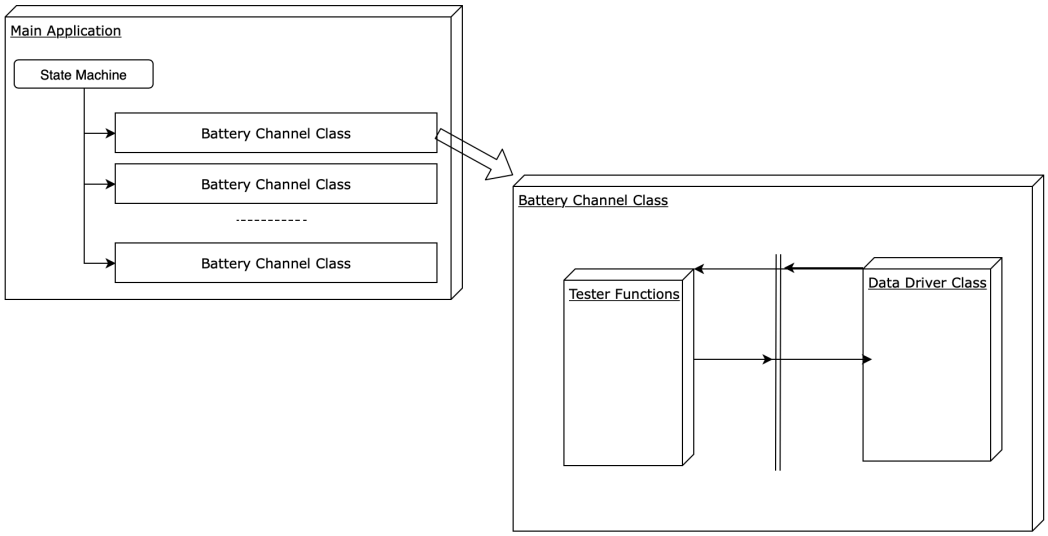

Figure 3-15: Main Class' Overview

The state machine controls what the users see based on their inputs. A user selecting Channel #1 would cause the state machine to display the measurement screen related to Channel #1. The battery class holds all information related to individual battery channels, including measurements information, the test being executed and safety information. The test functions refer to predefined functions to supply the users test sequences. These include basic functions such as Charge, Discharge, EIS and Coulombic Efficiency. Lastly, the data driver portion of the application as described in Chapter 3.2.1, transfers data to and from the application and the hardware.

State Machine

A State machine, in computer science, can be described as a simple model where an input from the user is mapped to a transition action followed by the output, [44]. State machines play an important role in large applications such as the battery tester as they simplify the management of screen transitions, handling of views and presenting of data.

For the battery tester a hierarchical state machine is in use, which allows for channels to operate independently of each other while still allowing the main state machine to control the user interface. Each state dictates what changes on the user interface, the data shown to the user and enables or disables buttons based on user input. Sub-states provide the user with alternate views available within a given state. A user in a 'Testing' state, might choose a sub-state for viewing the measurement screen, charts or logs.

States are usually transparent to the user as there can be multiple states and transition for a single user view. In the battery tester software there are four main views presented to the user and one supporting view. These states as seen in Figure 3-16 are:

1. Home;
2. Channel Selection;
3. Test Selection; and

4. Measurement.

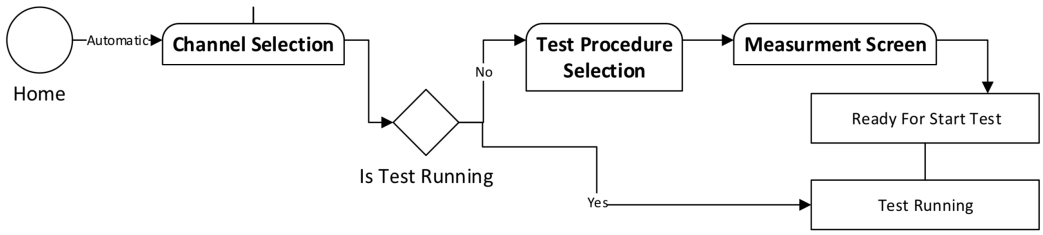

Figure 3-16: Software Screen Workflow

On the first load of the application, the user is taken to the home screen, pictured in Figure 3-17, where the home state machine is active. Selecting any of the given options will result in the state machine transition to the corresponding state.

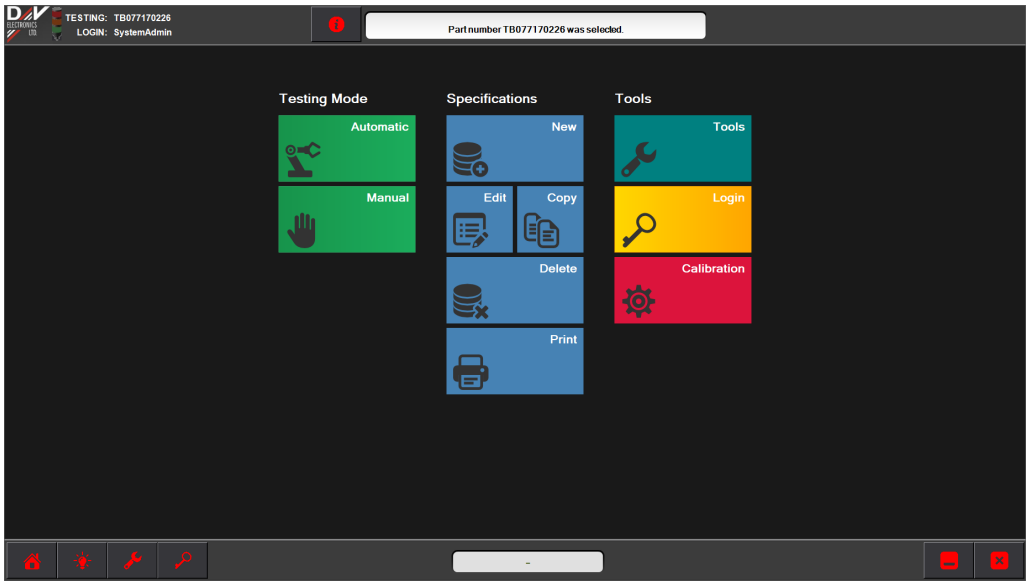

Figure 3-17: Application Home Screen

Choosing **Channel Selection**, brings the user to a new screen, seen in Figure 3-18, where all available channels are displayed in the form of 'Channel Tiles'. These Channels are predefined in software settings by D&V Electronics based on the number

of channels purchased. Each 'Channel Tile' as seen in Figure 3-18, represents a physical High-Power module connected to a battery for testing. The tile contains a basic overview of the battery state of charge, voltage, current and which test is running.

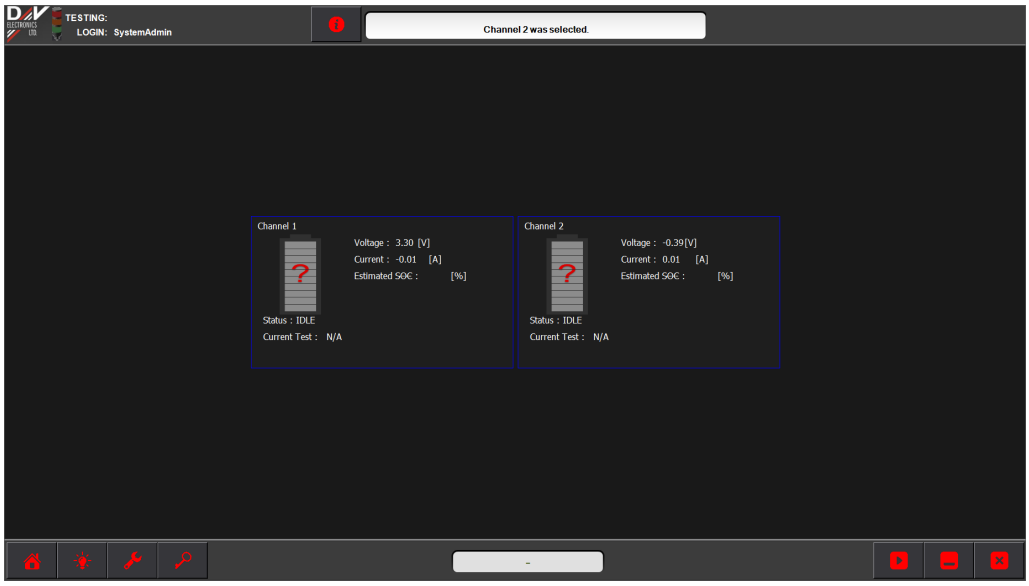

Figure 3-18: Channel Selection

Upon double-clicking a channel, the state machine transitions based on that channel's current internal state. The current state of the selected channel defines what happens next:

- State: Idle
    - In this case, the user is taken to the Part Number and Test Selection Screen.
- State: Test in Progress
    - If a test is already running, the user is transitioned to the Runtime screen.

The **Part Number** screen contains a list of all part numbers as defined by the user. Each part number contains a battery's respective settings, specifications, and associated safety limits. Once a part number is selected, the user is then presented with a number of standard tests available for the selected part number. Selecting a test, then transitions to the Runtime screen.

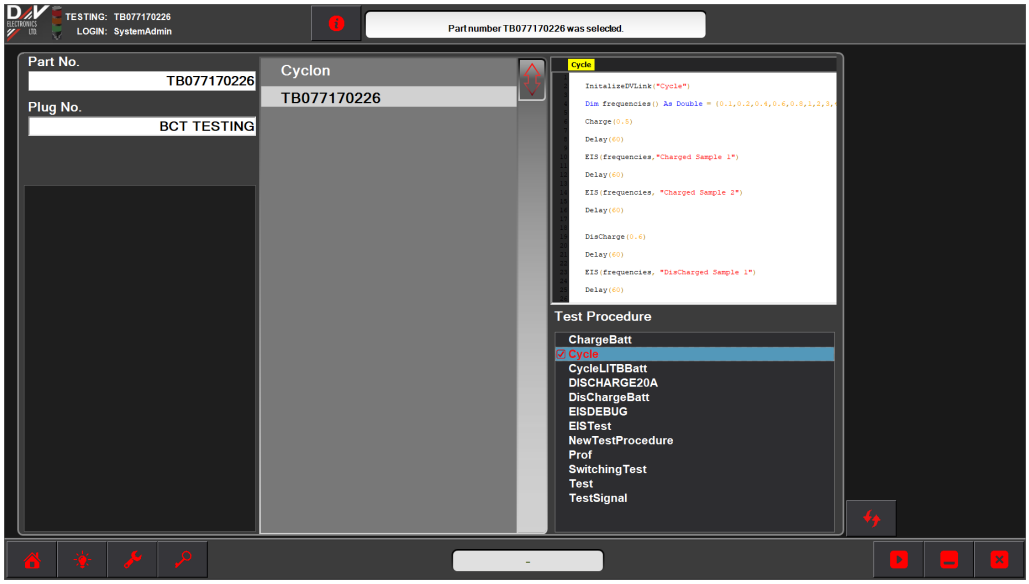

Figure 3-19: Part Number Screen

The **Runtime Screen** is the final state in the sub-state machine. The user is able to start and stop the test here as well as providing an indication of the current step in the test. The rest of the screen is initially blank and allows for a customizable area through the use of editable panels. The panels are able to draw from a number of available data buffers:

- Current
- Voltage
- Temperature
- Power
- Set Points
- Average of all measurement's

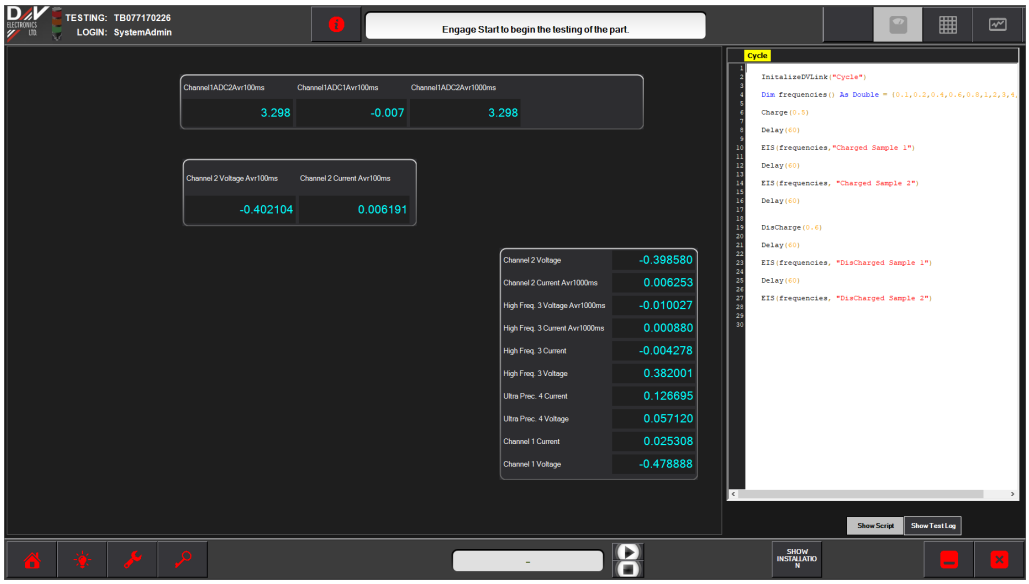

Figure 3-20: Runtime Screen

Clicking any other buttons on this screen will hand over control to the main state machine and maintain the state in the channel's sub-state machine. By doing this, the user is able to go to another channel and define a test while the current channel continues to operate in the Measurement State.

Tools

The tools button takes the user to the tools and settings screen, where they are presented with settings for the tester, modules and other application extensions. In order for the machine to be customizable for each user, a large number of settings are required. However, settings are only shown to the user based on their login credentials as incorrect settings can have severe outcomes.

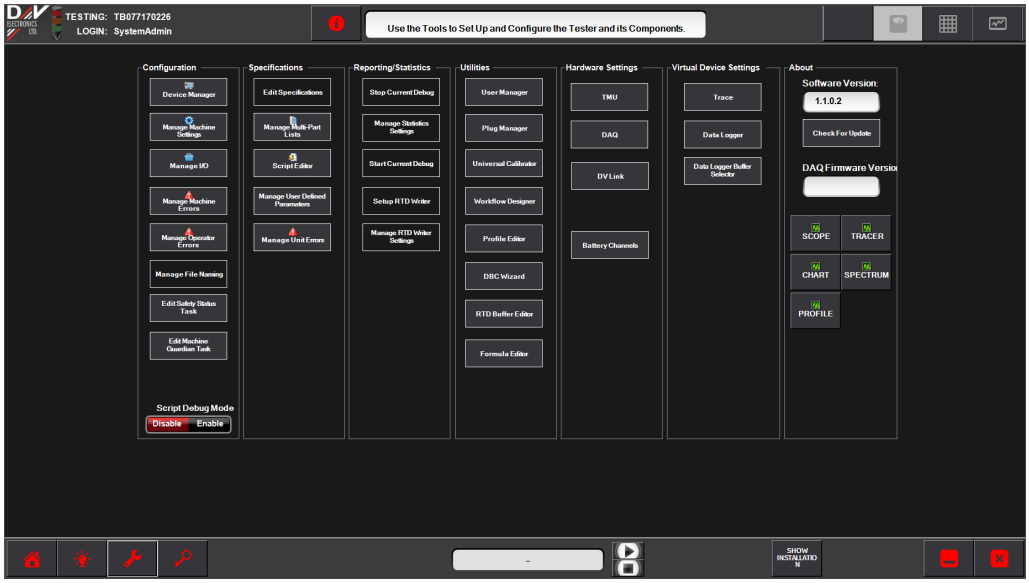

Figure 3-21: Tools Screen

The Device Manager allows the customization of the hardware. This includes setting such as:

- Number of Modules Connected;
- Thermal Monitoring Unit Type; and
- Thermal Chamber Communication Model.

These settings are usually one time set up and are reserved for the highest login levels.

Each module also has its respective settings. These settings define, the serial number, read and write sampling rate and whether it is in parallel with any other module.

This screen also provides the user access to tools to edit the state machine flow, the scripts for each test as well as allow them to define new buffer with custom calculations.

Battery Channel Class

In object-oriented programming, a class is defined as a blueprint or specification for a particular object, [45]. In this case the class created was a blueprint that defines the battery channel in the software. The main goals of the channel class are to keep track of the current state, as well as hold important information related to the battery under test such as safety limits. The channel class also contains instances of the two other major classes, the data driver and test functions described in later chapters.

3.2.3. Data Acquisition

Data Acquisition is a critical aspect for battery testing as the amount of data processed from tests can be very large. All measurements received from the modules and read from the driver are placed into circular arrays within the application. The arrays act as a form of transportation for all the measurements by assisting in

distribution to other areas of the software such as the run-time screen. The arrays also play an important role in storing of the measurements.

The high sampling rates of the measurements meant saving of all data was not practical. The rate at which these measurements were saved, was left up to the user of the software. The software allows the user to pick from a range of frequencies for which to save the data being gathered. Based on the selected option, throughout the test, the software will do the following to store the measurements:

1. get data from the last collection interval to the current point in time;
2. average the measurements;
3. attach the measurement to current test info; and
4. compress the data send to the server.

The Server is an always running a service on the computer that is connected to a database. It is separated from the main application, and allows for a central storage location of data from a number of testers. It also plays an integral part in maintaining data integrity. Upon receiving new data, its processing includes verifying that data was not corrupted, in the right format, and has the correct test associated with it. Once the data has been vetted, the server then proceeds to commit the data into the database for storage. An overview of the process can be seen in Figure 3-22.

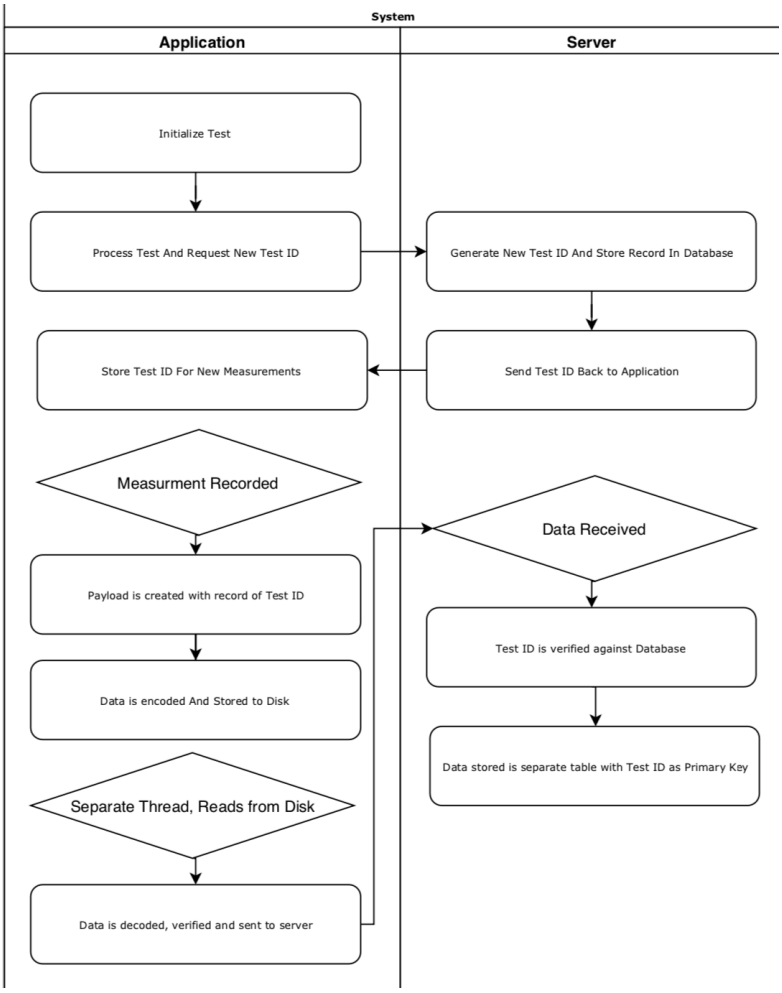

Figure 3-22: Data Storage Process

3.2.4. Post Processing

The last application part of the suite is dedicated to post-processing, providing a way for the user to graphically view the data in a meaningful way. Data is retrieved from the database through the windows service. The user has access to a list of all tests (started or completed) to choose from. After selecting one of the test results, there are a number of different options with which to process the data.

The default method is plotting of the measurements. All signals gathered from the test are available to the user to use in the creation of plots. If advanced testing was done with the built-in EIS test, then specialized graphs are automatically created to visualize this data. In Figure 3-23, we can see an example of a Nyquist plot based on an EIS test sequence. If needed, there is also the option to export the raw data to CSV.

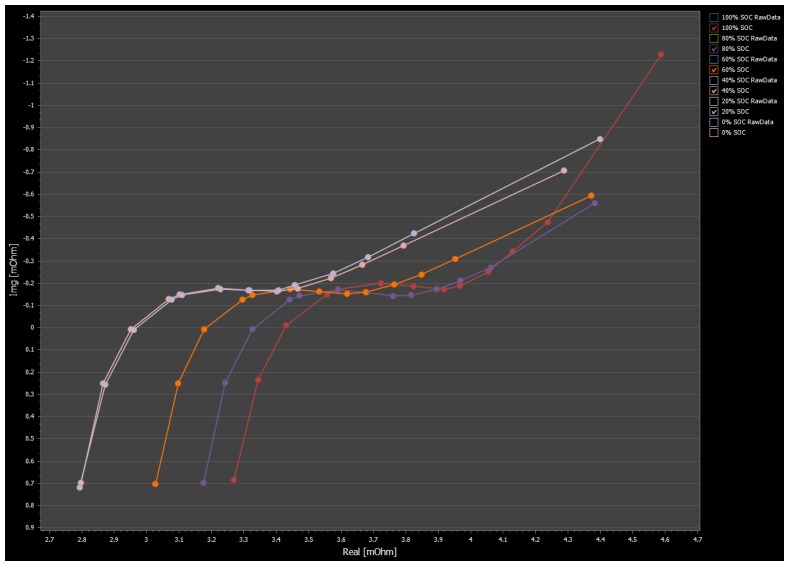

Figure 3-23: Nyquist Plot Generated by Post Processing Suite

The addition of a model parameterization routine was also an essential feature of the post-processing application. Data from each test can be utilized to parameterize any of the battery models included in the library. Utilizing the modeling framework, described in the next chapter, user-created models can also be imported and parameterized.

4. Model Library

The model library is one of the contributions and has a framework that is constructed to provide an environment for further battery model development, validation and testing of new battery chemistries. A software framework can be defined as a conceptual platform with generic functionality that can be used and overridden by developers or users, [46]. Frameworks often act as a foundation for a development platform and consist of a library of API, Interfaces and functions, [47].

For the development of this framework a library of function calls, interfaces and parameterization functions have been created to act as the foundation for further model development. It allows users to integrate new models with a minimal understanding of the software or programming languages. The framework is constructed around the idea of software interfaces and abstraction so that new models developed in Matlab or any other way can be used in the software without any modification of the original source code.

Software abstraction is an important concept in object-oriented programming which aims to remove any physical, spatial or temporal details of a particular object, [48]. Two common techniques used to abstract software from the details of implementation are Interfaces and Inheritance.

4.1. Interfaces

A software interface is an outline of what a particular piece of code should do and is often used in order to abstract code. In this case, it Is used to allow models created outside of the library to be dynamically loaded by the software. Most models consist of functions that serve a similar purpose, which allows for easily fitting an interface to them. The interfaces were developed into two separate entities in order to further expand the capabilities in the future, the Model Interface and Battery Model Interface.

4.1.1. Model Interface

The Model Interface acts as a base for all models and consist of definitions common to them. This denotes that it can act as an interface for all models and not just battery models. The properties of the interface were limited to:

- Function: Initialization
    - Loads the parametrized model with default values
- Property: Properties
    - Provides access to all the models parameters available for modification
- Property: Tunable Properties
    - Provides access to parameters that are to be parameterized during model optimization
- Function: Step
    - Steps the model by one time step and updates outputs of the model

- Property: Outputs
    - Provides access to all outputs of the model
- Function: Dispose
    - Provides a way to properly release resources no longer required in memory.

When an application uses this interface, it calls the generic functions defined above such as 'Step'. The code that executes when this step function is called, is completely dependent on the user and on the model that is selected.

Models can be imported from the Matlab battery model library developed specifically for the tester as described earlier. Each model imported from Matlab will contain a Step function that defines the mathematical operations required to simulate a model step. Since the model declares itself as adhering to the Model Interface, the application is able to execute code specific to that model. The interface acts as the link from the application to the generated model. Figure 4-1 shows an example use of the interface. In the example, an application uses the model interface to simulate a battery model but is also applicable in generic terms to others such as a vehicle model or a blood flow model. However, the application never sees the detailed implementation of each model. To the application, it only is aware that it has to call the 'Step' function in the interface to run the model. The user develops the details of that 'Step' function in Matlab and exports it to a dynamic link library (.dll), that the application can access. The defined interface links the application and the specifics of the model together. It

is this approach that allows for expansion of the model library without the need for changes to the main application and allows the model library to grow.

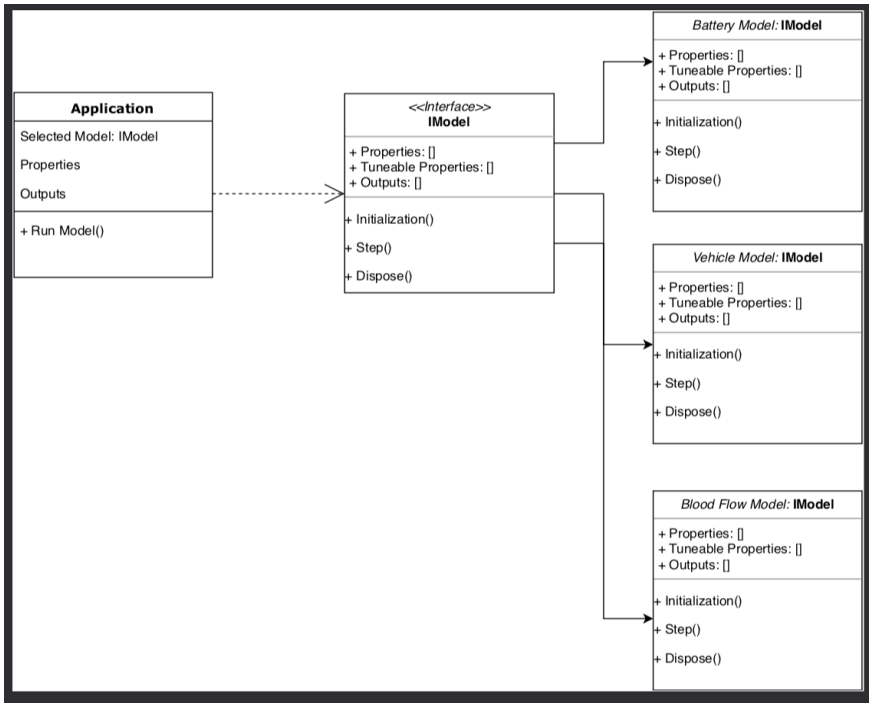

Figure 4-1: Example Use of the Model Interface

4.1.2. Battery Model Interface

The second interface created is the Battery Model interface, which uses the Model Interface as a starting point. This means that all parameters that are part of the model interface also become part of the battery model interface. The framework is created for the development of battery models and thus specifics of a battery were added to the model interface, including:

- Property: Current

- This sets the current value to be used during the next simulation step.

- Function: Step (Current)
  - This function overloads the previous step function and sets the current of the next step prior.

All new models would have to reference this interface and specify the functions that implement it. This would identify to the application that it can be used as a battery model. All function calls to the interface would then be rerouted to the particular battery model's functions. Figure 4-2 shows a similar example to that of the Model Interface. In this case the battery model interface can be adapted to any type of model including R-RC, Electrochemical, or the Simple model as seen below.

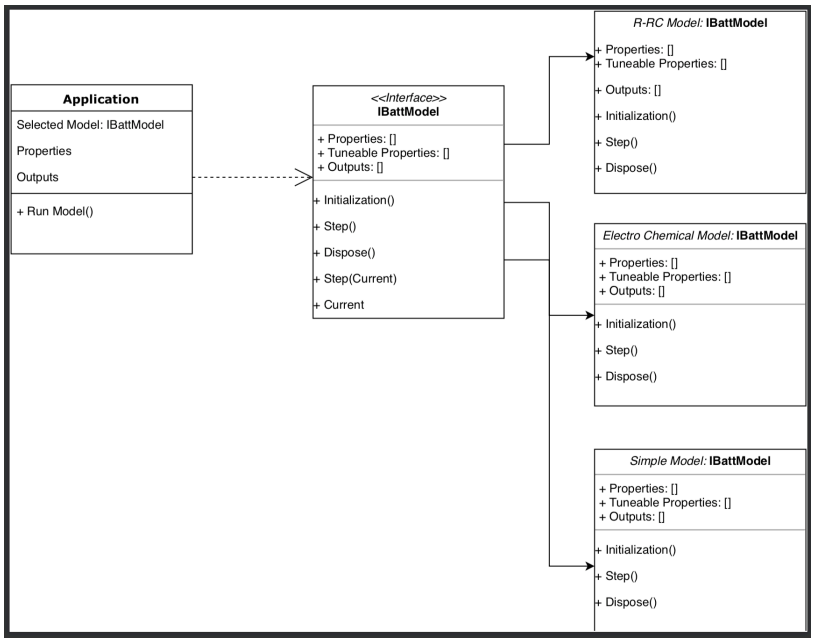

Figure 4-2: Example usage of Battery Model Interface

4.2. Inheritance

In order to ensure expandability as may be required for the optimization strategy, a very different approach was required to abstract code from the application. Optimization strategies differ vastly from one another in terms of input, and parameters, leading to the use of inheritance in order to allow dynamic loading of new optimization strategies. Inheritance in computer science is defined simply as the process where in characteristics, functions, properties are inherited from a base class, [49]. While Interfaces hide all details of the implementation to the application, Inheritance is able to provide the application some of the details of implementation required.

Inheritance is often implemented in the form of a base class in which other classes can inherit it to comply with an application set of functionalities. As an example, in the building of a vehicle the base class would define that the vehicle has four tires, a chassis, and an engine. A manufacturer would build its own car on top of this by inheriting the details of the base car and implementing the details with respect to the size of the tires, the look of the chassis, mechanics of the engine and so forth. With the implementation for Optimization, the base class would contain the model, 'Optimize' and 'Initalize functions allowing for the details of them to be implemented by the Optimization strategy chosen as detailed below.

4.2.1. Optimization Base Class

The base class outlines the basic requirements for the optimization, as well as a suggestion for what other functions should accomplish. When adding a new optimization strategy, the 'Base' class is inherited, and the new strategy is filled in by the user. When a battery is loaded, the associated Optimization Strategy with it is also loaded.

The Optimization class is built to function directly with the battery model interface and contains two functions 'InIt' and 'Optimize'. Initialization is done as part of the 'InIt' function, where the model object is passed in and is assigned to a local variable useable by the other functions within the class. The Optimize function is a function that must be overridden by the implementer. This function is meant to contain all the code associated with parameterization of a given battery model. Its inclusion of the battery model interface also means it can be used across the model library as well as with new model additions.

4.3. Genetic Algorithm

In order to showcase the use of the parameterization class a genetic algorithm was implemented using the framework. An open-source genetic algorithm, Genetic Sharp,

which is created to work with the .NET framework[1]. The use of a third-party solution showcases the use of the framework for expanding the parameterization options.

To start the implementation, a new class was created that inherits the properties, parameters, and functions from the parameterization base class. The code related to the genetic sharp was then added to the Optimize function. Here the properties related to the Genetic Algorithm are defined such as population, mutation, crossover, termination as well as the fitness function. Once the properties are defined, the function is then responsible for starting the process of parameterizing the model. A fitness function is required in order to help determine the accuracy, the current parameters provide. In the case of this implementation, a root mean square error was used as defined in equation 4.1.

$$RMSE = \sqrt{\sum_{i=0}^{n} (Actual\ Voltage - Simulated\ Voltage)^2} \qquad (4.1)$$

4.4. Model Library

A number of existing battery models developed in Matlab were modified and converted to dynamic link libraries for use with the framework and the tester. Any model making use of Simulink's advanced blocks such as the State-Machine blocks

[1] A software framework developed by Microsoft, with a large library of function to simply development on the Windows platform.

were not compatible with exporting to C++ and posed a challenge. In order to overcome this, the non-compatible blocks were replaced with S-Functions replicating the state machine functionality. The code was then modified to accept initialization parameters for the state of charge and time step.

The following models are part of the model library:

- Equivalent Circuit Models
- First Order R-RC Model
- Second Order R-RC-RC Model
- Third Order R-RC-RC-RC Model
- First Order R-RC Model with Hysteresis
- Second Order R-RC-RC Model with Hysteresis
- Third Order R-RC-RC-RC Model with Hysteresis
- Empirical Models
- Simple Model
- One State Hysteresis Model
- Enhanced Self-Correcting Model Two State

4.5. Demo Application

In order to showcase the framework's functionality, its ease of use and expandability, a demo application has been created. The application is created with no knowledge of the types of battery models, nor the properties required to be

parameterized for each model. In addition to that, its functionality was based around the following:

- Be able to load battery models dynamically
- Parameterize models based on drive profiles
- Save and load battery parameter.
- Plot simulated drive profiles and compare against tester data

4.5.1. Battery Model Loading

On application load, a search is performed in the root folder for dynamic link libraries (DLL). The application then uses reflection, a software technique used to learn more about the code in the library, to see if the dynamic link library uses the model interface (IBATTModel). If the model does implement the model interface, then the application continues to create an instance of the model. Once more using reflection the application gets the model name and proceeds to add it to the list of available models in the user interface with other models found. These models are then selectable by the user as seen in Figure 4-3.

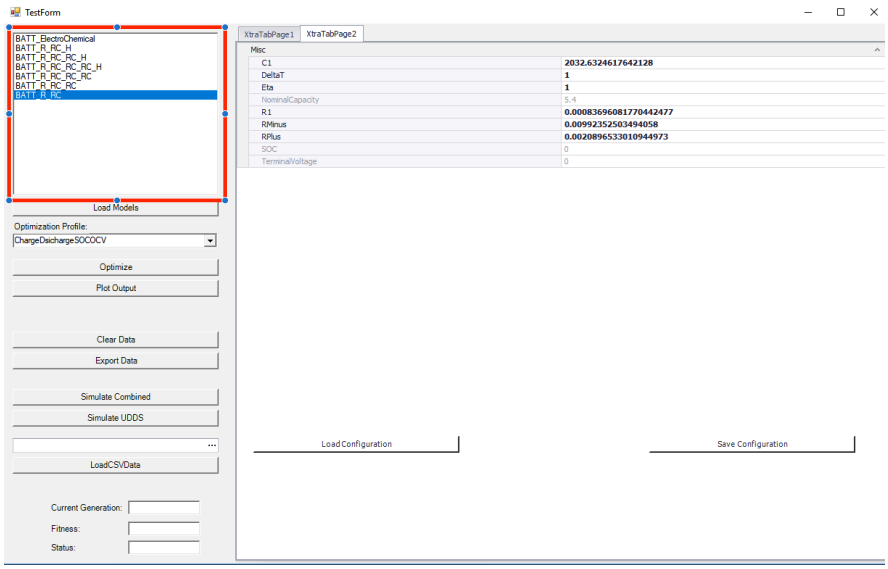

Figure 4-3: Screenshot of Demo Application

4.5.2. Optimization Profiles

Upon selecting a model, the user is then able to select a drive or current profile that will be used for optimization of the model. There are currently four profiles that can be loaded, all of which use data gathered from the D&V Battery Tester. These include:

- SOC-OCV (State of Charge vs. Open Circuit Voltage)
    - A single SOC vs. OCV curve, generated by charging the battery at a constant current.
- UDDS Profile
    - Current, Voltage and SOC data collected by running a UDDS current profile.
- Combined Profile
    - Current, Voltage and SOC data collected by running a combination of a UDDS and US06 drive profiles.

- Charge Discharge SOC-OCV
    - Two SOC vs. OCV curves, for charge and discharge with currents at +- C/20.

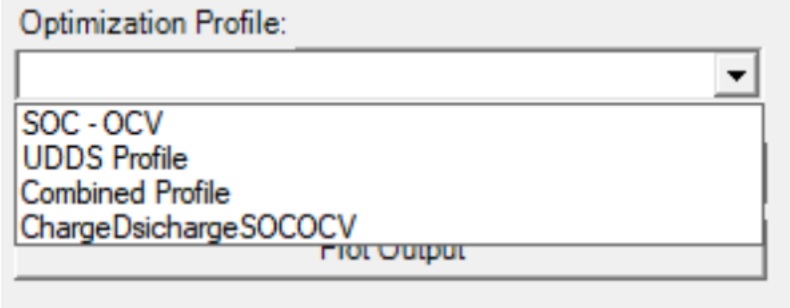

Figure 4-4: Optimization Profiles

4.5.3. Optimization

Once a profile has been selected, the user is able to start the optimization of the model. At this point, a new instance of the selected battery model is created, and its respective optimization strategy is invoked. The application has no knowledge of the model's optimization strategy and makes use of the optimization base class in order to begin the process.

All models built thus far make use of the GeneticSharp Library[2] with the same properties used for all models. The population is user configurable, however the default value is set to 500. The default selection is 'Elite' and mutation is Uniform. The optimization is considered when the fitness is stagnant for 60 or more generations. The model optimization is also halted if the optimization runs for longer than 1000 minutes or 16 hours.

[2] A Genetic Algorithm Library/Framework for implementing Genetic Algorithm as part of the model framework

4.5.4. Best Model Finder

In addition to allowing for single model optimization, the framework also allows the application to do parallel processing in order to optimize multiple models simultaneously. The application is then able to provide the user with a real-time view of which models are best suited for the data provided. An example of this process working can be seen in Figure 4-5.

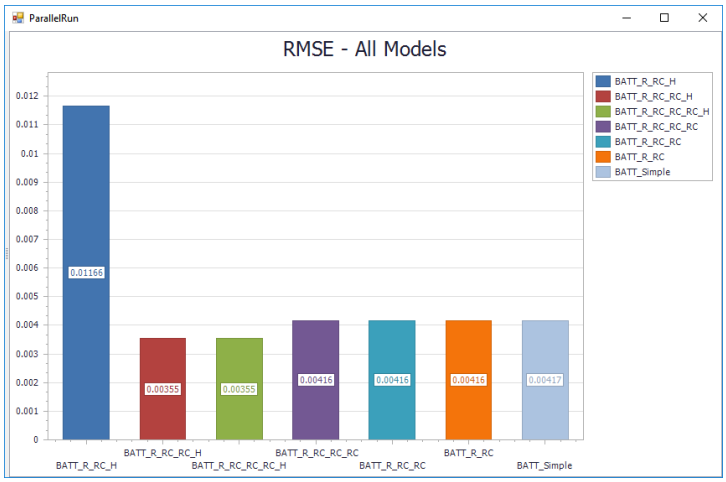

Figure 4-5: Best Model Finder Screen

4.5.5. Analysis and Post Processing

Once optimization has been completed, the user is able to see the final fitness value as well as the number of generations run, as well as the respective values for each parameter and saved for later use. The application can then simulate the other profiles using the new parameters and plot it against data obtained from the battery

tester, as seen in , to provide an overview of the model's performance. Lastly, all data can be exported for further analysis by third-party programs.

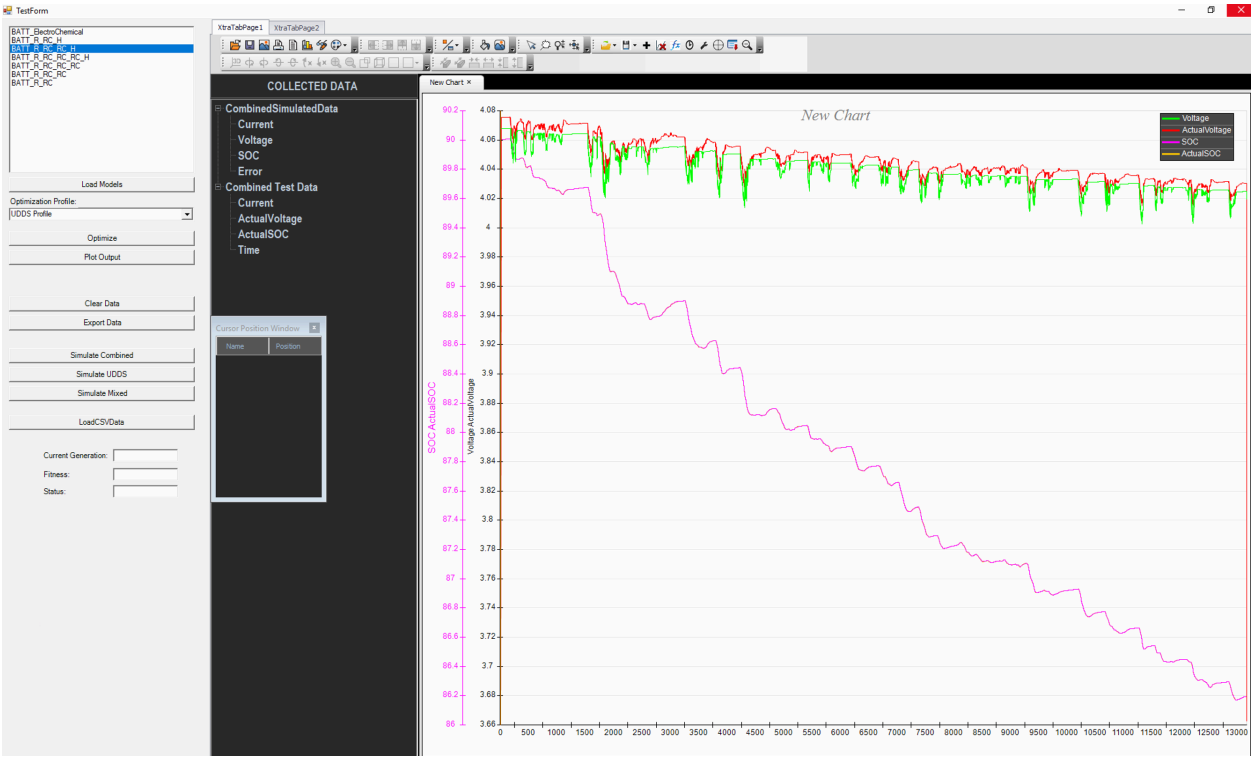

Figure 4-6: Application Plot of Simulated data vs Tester Data

5. Application & Analysis

This chapter aims to do a complete battery characterization of a prismatic automotive battery cell from SB Libomotive. This required the development of a number of characterization test functions in order to provide a better understanding of the battery under test such as:

- Charge/Discharge Test
- SOC-OCV Curve Generation
- Electro Impedance Spectroscopy
- Capacity Test
- Model Fitting

Lastly, the ageing test, will be a combination of all of the above tests combined over the life of the battery. This is done through accelerated ageing of the battery by constant charging and discharging of the battery. This will provide a deeper understanding of the battery's performance over time resulting on the characterization we aim to achieve.

5.1. Charge/Discharge

The Charge and Discharge functions will act as a basis for a large number of tests. Apart from the charge or discharge functionality, this function also executes two other important tasks; a safety monitoring task as well as a data recording task. These two tasks run in parallel to the main function once the battery connection is made. Figure

5-1 Provides a flowchart of each step involved in the Charge/Discharge function, this is followed by further details of the steps.

The tester hardware provides two ways of providing current to the battery through constant current and constant voltage. For constant current, the voltage is set to a value higher than the desired voltage and a limit is put on the current. Constant voltage is done by setting the voltage to the desired value and allowing the current to decay as it approaches the voltage limit. An understanding of these charge and discharging methods were crucial in the development of this function.

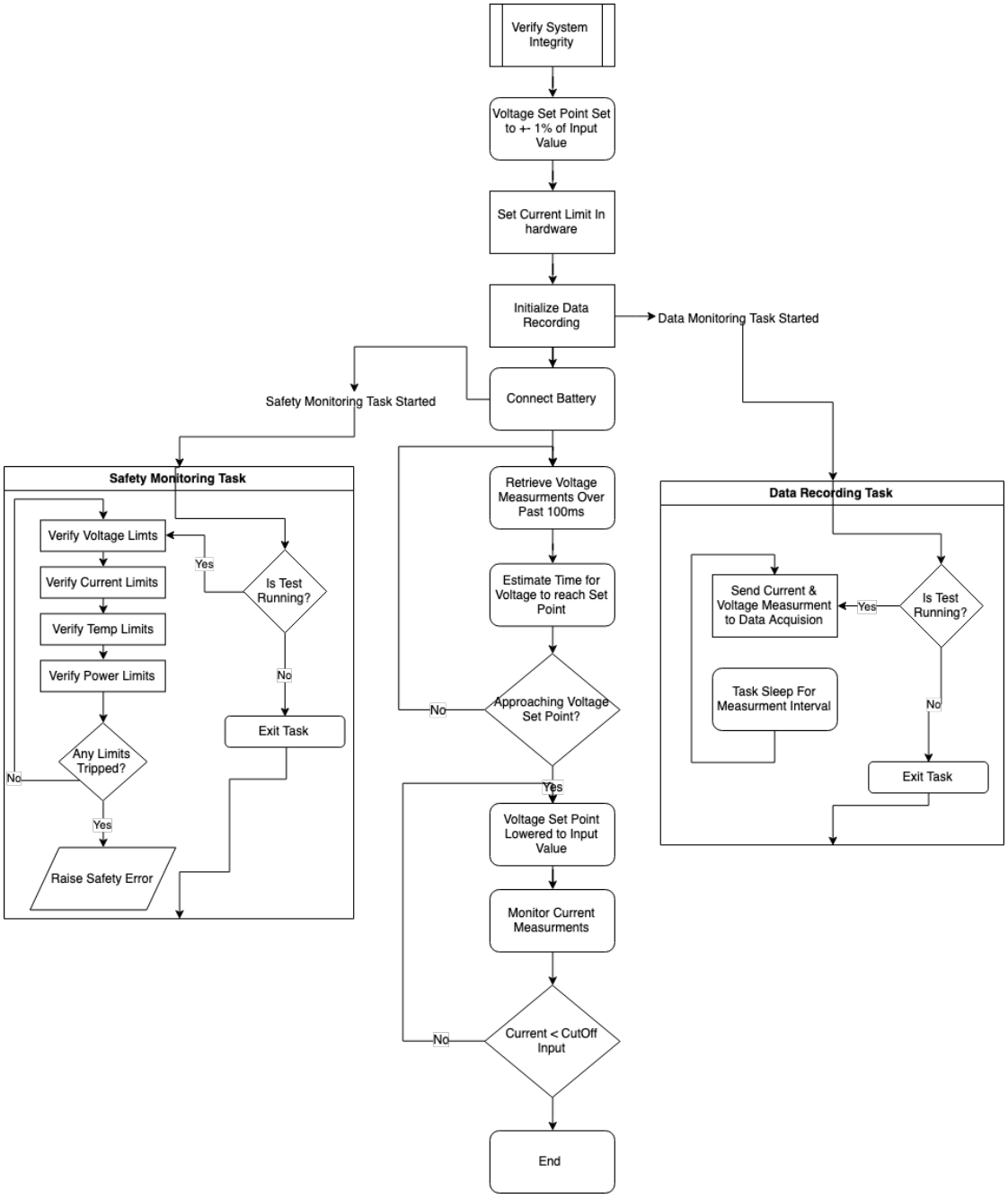

Figure 5-1: Flowchart of Charge/Discharge Function

The function starts by setting the current limit, upper voltage and lower voltage limit in the hardware. On connection, the voltage is set to a value slightly higher than its set point, followed by the commencement of constant current charging. The function will monitor the current and voltage during this time, calculating the estimated time for the voltage to hit its set value. This is important in order to gracefully transition the charging or discharging from constant current to constant voltage. Once in constant voltage mode, the current start to slowly lower towards zero as the battery reaches maximum capacity. An additional parameter is available to allow the user to define at what current threshold should the software consider the charge/discharge completed.

As the voltage approaches its set point, the function will start to lower the voltage back to the desired set point. Failure to do this correctly, will result in voltage overshoot and a sudden change in current. Figure 5-2 shows the ideal transition from constant current to constant voltage, whereas Figure 5-3 shows cases where overshoot was present prior to switching to constant voltage.

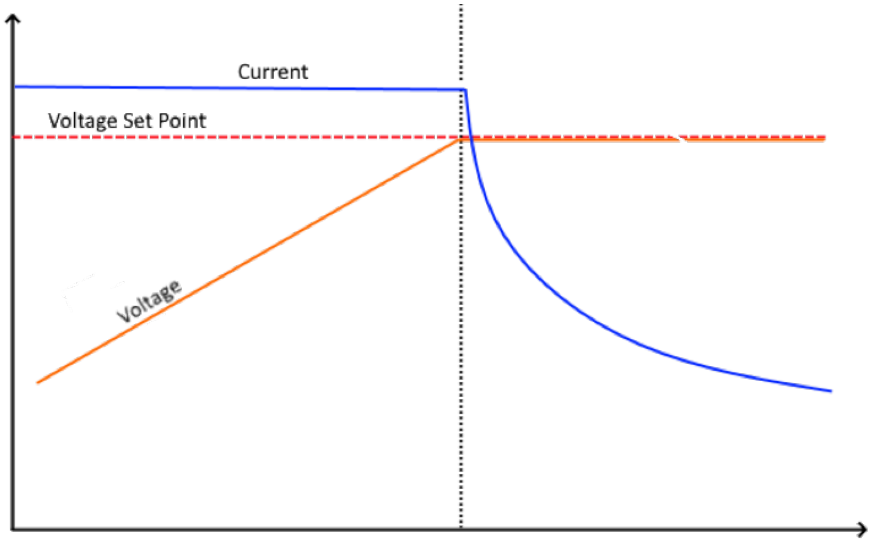

Figure 5-2: Current and Voltage during a good transition from Constant Current to Constant Voltage

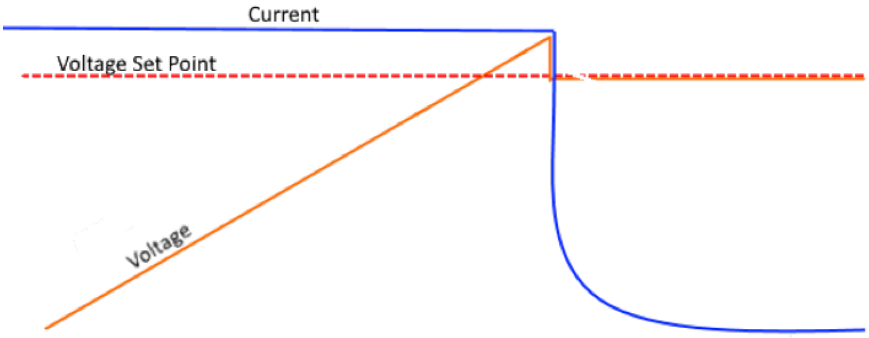

Figure 5-3: Current and Voltage during a non-ideal transition from Constant Current to Constant Voltage

5.2. SOC-OCV Curve Generation

Generating a state of charge to open circuit voltage curve is the first step in battery characterization as the state of charge is a base for many of the other tests. It maps the battery's open circuit voltage to the battery's state of charge. The procedure relies heavily on the charge/discharge function described in the previous section with the curve being generated in post-processing out of the test procedure. An overview of the test procedure and the post-processing can be seen in Figure 5-4, followed by more details below.

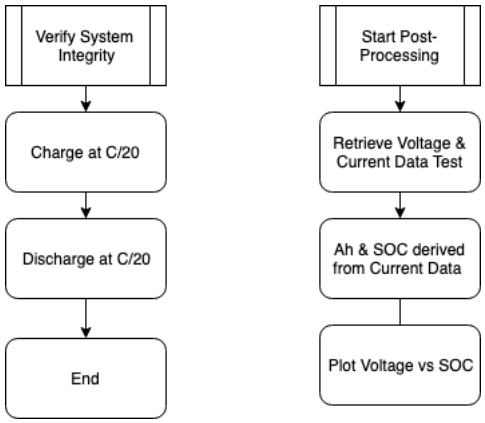

Figure 5-4: Flowchart of SOC-OCV Generation Procedure

5.2.1. Test Procedures

The state of charge vs. open circuit voltage generation is a simple yet a lengthy test procedure. The first part of the test script involves ensuring the battery is completely discharged. This is then followed by the generation of the curve by charging the battery at a very low rate of C/20. Once fully charged, the battery is then discharged at the same rate of -C/20. The voltage is collected throughout the test along with the

amount of current. The amp-hour values are then used to calculate the approximate state of charge. Lastly, plotting the voltage on the y-axis with the state of charge values on the x-axis provides the user the state of charge vs. open circuit voltage curve.

5.2.2. Test Results

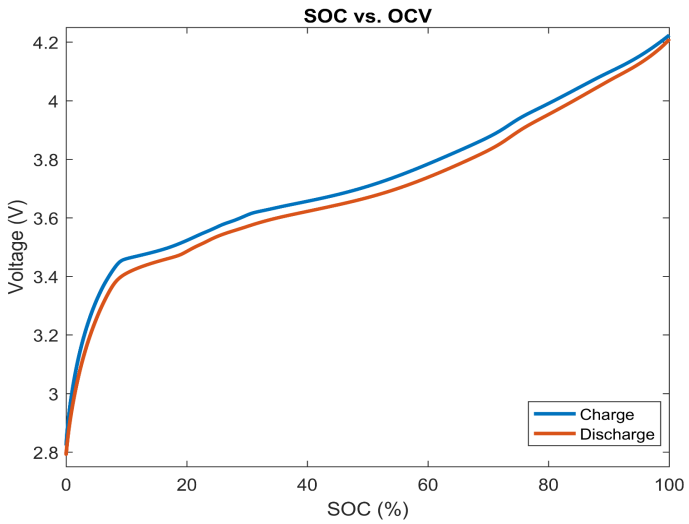

Figure 5-5: State of Charge vs. Open Circuit Voltage for Charge and Discharge

The hysteresis effect between charge and discharge is visible and plays an important part in battery modelling. The relationship is modeled by a 10^{th} order polynomial, equation 7.1, for estimating the open circuit voltage based on the state of charge of the battery.

$$\begin{aligned} &2.357 \times 10^{-18}x^{10} - 4.684 \times 10^{-16}x^{9} - 1.070 \times 10^{-13}x^{8} \\ &+ 4.788 \times 10^{-11}x^{7} - 7.279 \times 10^{-9}x^{6} \\ &+ 6.021 \times 10^{-7}x^{5} - 2.971 \times 10^{-5}x^{4} \\ &+ 8.832 \times 10^{-4}x^{3} - 0.0153x^{2} + 0.145x^{2} \\ &+ 2.81 \end{aligned} \qquad (5.1)$$

5.3. Electro-Impedance Spectroscopy

Electro-Impedance Spectroscopy test involves imposing an AC current signal and calculating the Impedance of the current and voltage measurements. In order to ensure consistent results are obtained, a number of steps are taken to exclude outliers and ensure accurate results. Figure 5-6 provides an overview of the steps involved in doing an Electro-Impedance Spectroscopy, followed by a more in-depth review of the details below.

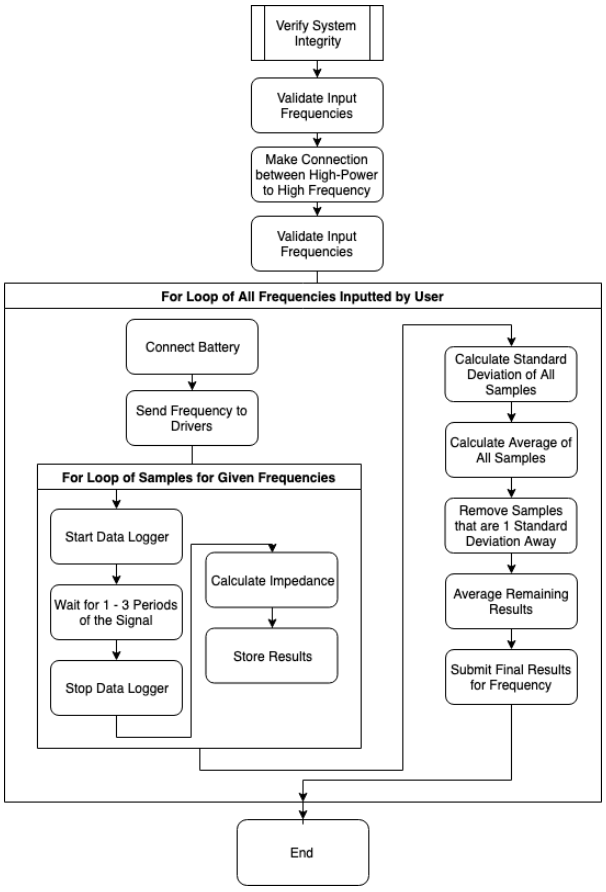

Figure 5-6: Flowchart of the EIS Test Procedure

5.3.1. Test Procedure

A number of precautions were needed to be taken into consideration to obtain accurate results from an Electro-Impedance Spectroscopy test. Voltage and current measurements need to be perfectly synchronized in order to obtain accurate results, making the data driver described in Section 3.2.1 an important part of this test. In addition to ensuring hardware measurements were accurate and transferred to the computer in a synchronized form, further analysis was done to ensure better results.

The first step in the process is to set the frequency of the current signal applied to the battery. Measurements are then collected over a number of periods as defined in Table 2. A fast Fourier transform is then done on both signals to determine the dominant frequency of the current and voltage signal. If this frequency matches that which is expected, then the impedance is calculated by dividing the voltage by the current.

This process is then repeated a number of times to obtain a few samples per frequency. The number of samples is dictated by Table 2 as follows.

Table 2: Sample Count per Frequency

Frequency Range	*Number Of Samples*
Greater than 1,000Hz	10
100 Hz to 1,000Hz	8
1Hz to 100 Hz	4
0.01Hz to 1Hz	2
Below 0.01Hz	1

At higher frequencies, more noise is present which results in inaccurate impedance values. A higher number of samples are collected in order to segregate these incorrect values. Once the samples are collected, a standard deviation is calculated for all measurements. Any impedance values found outside of one standard deviation is disregarded. The average of the rest of the measurements is then used as a final value. The entire process is then run again for every frequency value.

5.3.2. Results

To demonstrate the function in use, a test was done for frequencies from thirty thousand hertz down to ten milli-hertz. An example of the measurements can be seen in Figure 5-7, where 3 periods were captured of a one hundred hertz sinusoidal signal.

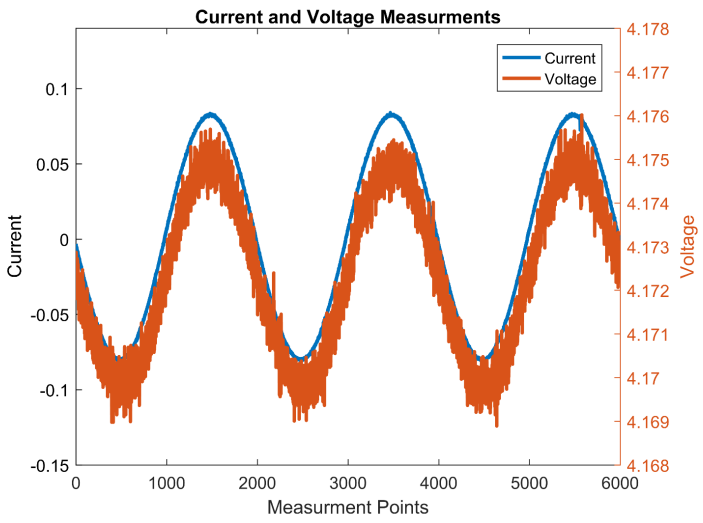

Figure 5-7: Voltage and Current measurements during a 100 Hertz EIS test

The Fast Fourier transform is done on both signals with results displayed in Figure 5-8. This is followed by dividing the current magnitude by the voltage as reviewed in Chapter 2.2.

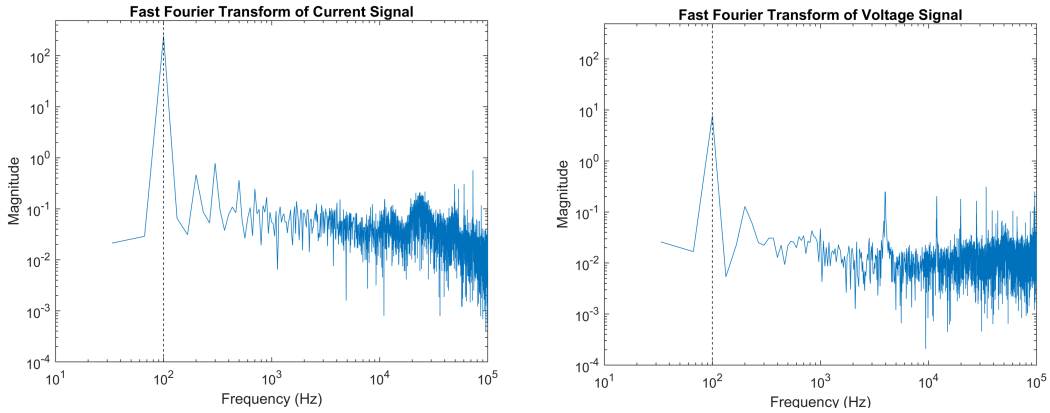

Figure 5-8: Fast Fourier Transform of Current and Voltage Signals

Figure 5-9a shows impedance results of ten different samples of measurements done at five kilohertz. Figure 5-9b then displays the same set of samples with any outside of

one standard deviation removed. Lastly in Figure 5-9c, the average of the remaining results can be seen and is what is used in the final Nyquist Plot.

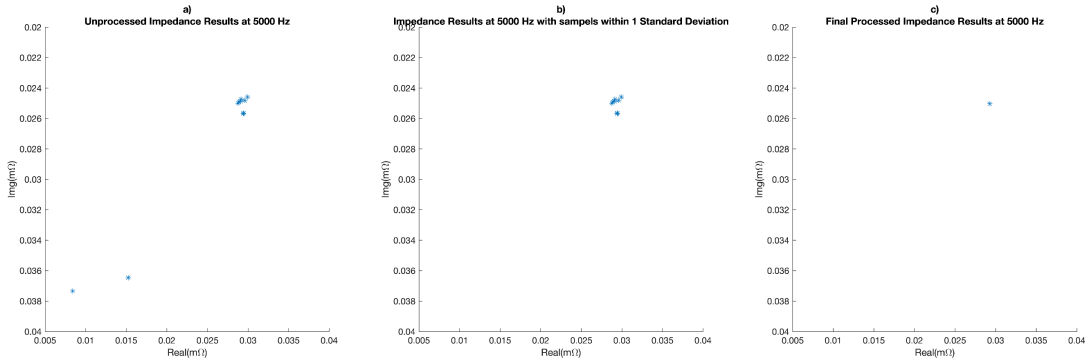

Figure 5-9: Impedance Sample Filtering at 5000Hz

A complete Electro-Impedance Spectroscopy was done to showcase the performance of the tester and the procedure. The test covered frequencies from ten kilohertz, down to ten millihertz seen in Figure 5-10.

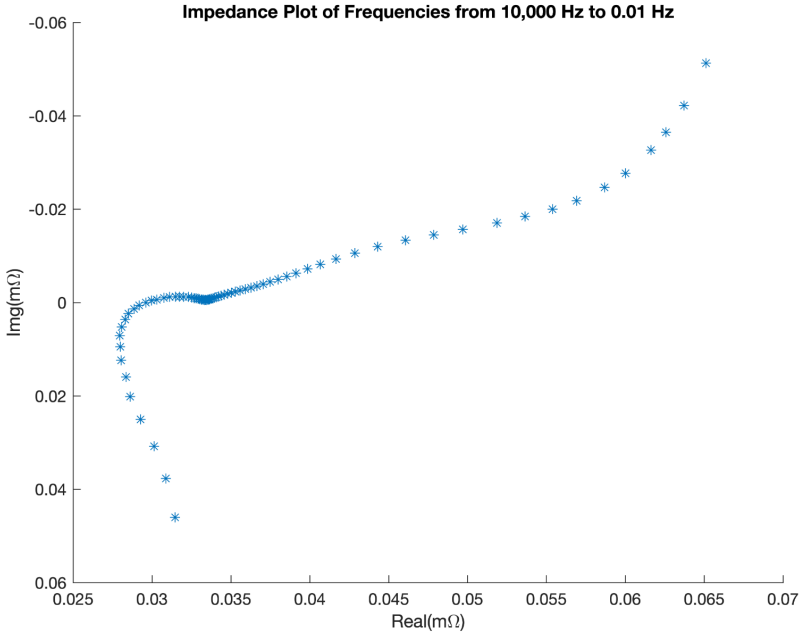

Figure 5-10: Nyquist Plot of Impedance of Frequencies from 10,000Hz to 0.01Hz

5.4. Automated Electro Impedance Spectroscopy

Similar to Electro-Impedance Spectroscopy discussed above, this test uses that procedure as the base along with additional profiles. This allows for calculating the Impedance not only at different frequencies but also at different state of charge, state of health, temperatures and different points of a drive cycle. In the case of this paper, a procedure was created to automate the Electro Spectroscopy test every ten cycles for two hundred cycles as seen in Figure 5-11 and detailed below.

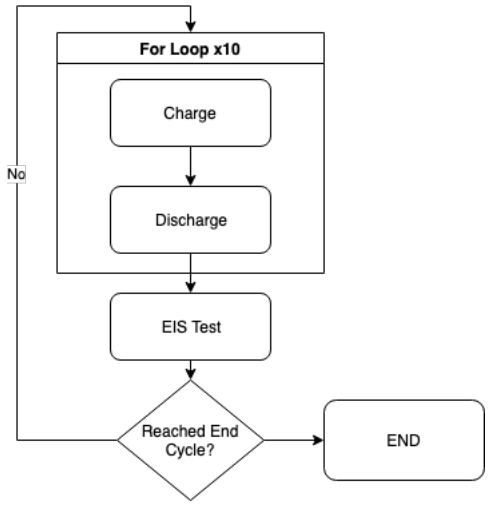

Figure 5-11: Flowchart for Automated EIS Test

5.4.1. Test Procedure

This script is broken in to two parts, where one controls the high-power unit and the other the high-frequency module. The first script developed for the automated test is running the EIS test script across the battery's state of charge. This involved charging the battery to one hundred percent state of charge, waiting for a user defined relaxation

time and then running the first Electro-Impedance Spectroscopy test function. The battery is then discharged by 10% before the EIS test function is rerun and repeated until the state of charge is 0%. The same method can be applied to the state of health temperature as well as drive cycle profiles. The user is able to choose the interval at which the EIS test is done.

5.4.2. Results

In order to demonstrate the functionality of the procedure, an ageing test was done. The test ran for thirty days and cycled the battery from 100% to 0%, with an EIS test completed every eleven cycles. The Nyquist Plot with EIS results from all seventeen measurements over one hundred and eighty-seven charge and discharge cycles can be seen in Figure 5-12.

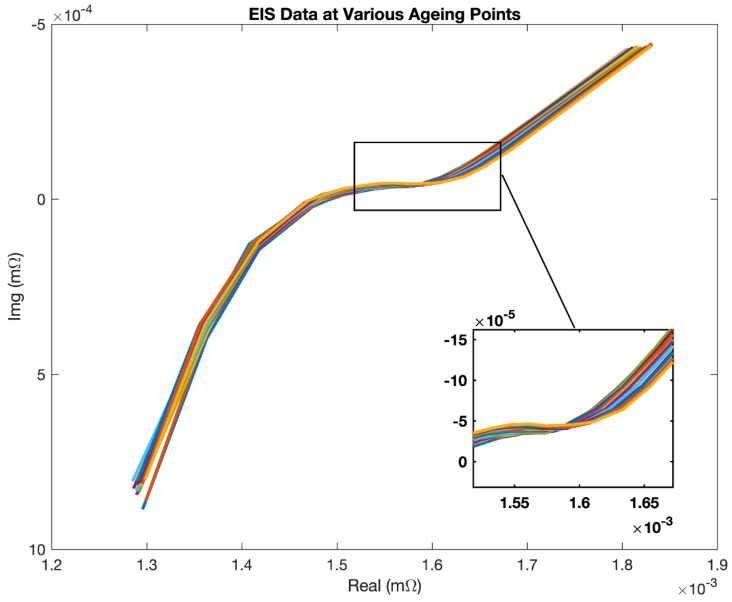

Figure 5-12: EIS Measurements over 187 Cycles

Removing some of the measurements from the above Nyquist plot, to include a fewer number (three) provides a better look into what is happening. The EIS results obtained at cycle 11, cycle 121 and cycle 187 can be seen in Figure 5-13. As the battery ages with the charge and discharge cycles, the EIS measurements start to show the increase in internal resistance though drift in the x-coordinates.

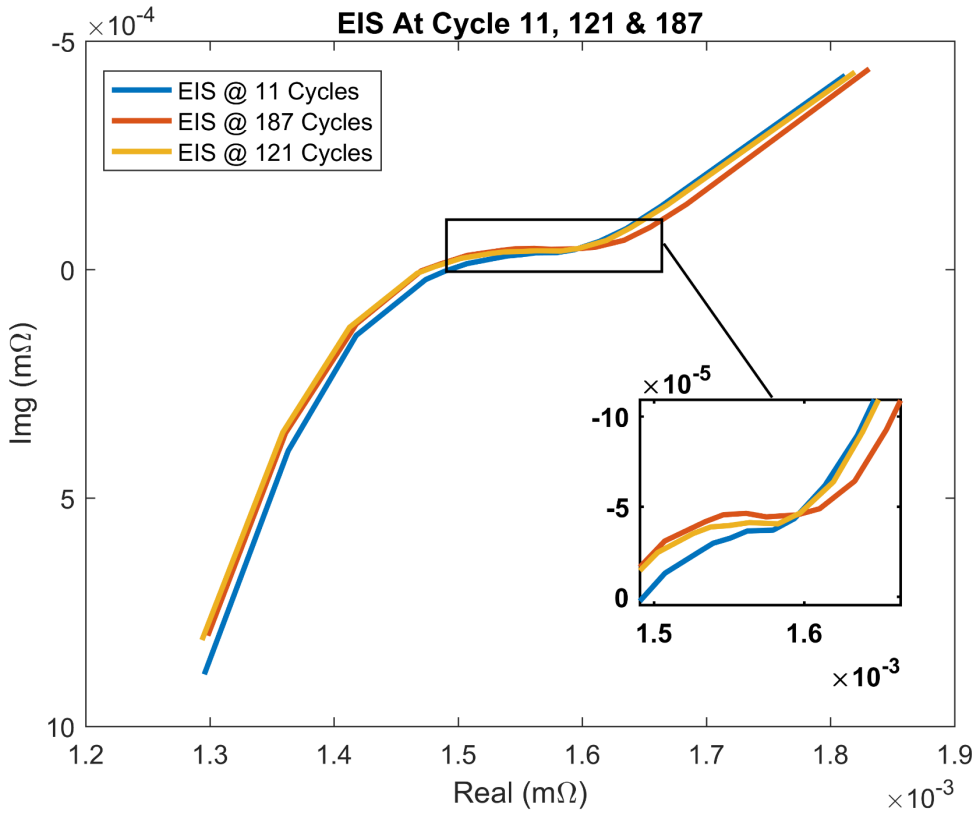

Figure 5-13: EIS Measurement's at Cycle 11, Cycle 121 and Cycle 187

5.5. Capacity Test

A capacity test follows a similar procedure to the SOC-OCV curve generation as seen in Figure 5-14; however, it differs in the hardware used along with the final results

of the test. The goal of the capacity test is to accurately determine the total amount of current a battery can hold and is accomplished through coulomb counting by measuring the current with extreme accuracy and integrating it.

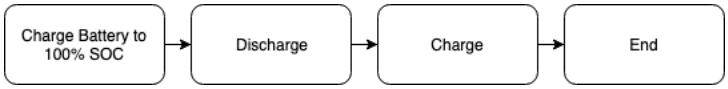

Figure 5-14: Flowchart of Capacity Test Procedure

5.5.1. Test Procedure

This procedure starts by fully charging the battery. The charge function defined is used to get the battery to maximum capacity. Depending on the level of accuracy required, the user is able to set a very low current threshold for constant voltage mode. The balance between time and accuracy is left up to the user. Once the battery is fully charged, the discharge process is then started and the amount of current drained from the battery monitored. The total amp-hours recorded at the end of the discharge is considered the capacity.

5.5.2. Results

A capacity test over one hundred and eighty-seven cycles was done in conjunction with the EIS test mentioned earlier. The testing was done to validate functionality of all the test functions as well as showcase the possibilities of the developed tester. Due to this, the test was done in a shared thermal chamber where temperature was not

controlled. However, data collected correlates with what is expected of a cell ageing over one hundred and eighty-seven cycles as seen in Figure 5-15.

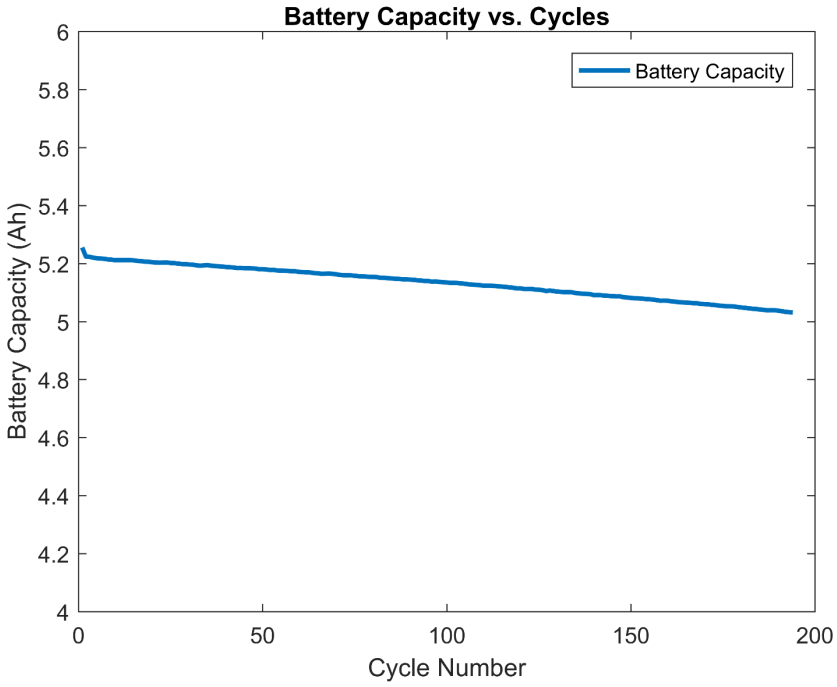

Figure 5-15: Battery Capacity Over 187 Cycles

5.6. Battery Model Parameterization

All of the functions defined above are then used to collect data to parameterize a number of models using the battery model framework described in Chapter 4. There are two profiles used to parameterize all the models below, the SOC-OCV curve as well as the Current & Voltage measurements obtained from running a drive profile. A number of other drive cycle profiles were run on the tester as well, including:

- EPS US06 Drive Cycle

- EPA Highway Fuel Economy Test (HWFET) Cycle
- A Mixed Drive cycle made up of US06, EPA and UDDS drive cycles.
- A Combined Drive Cycle of US06 and UDDS drive cycle.

The data collected from running these drive cycles are then used to parameterize and benchmark the models below. The UDDS measurement data was used as the profile for parameterizing all models. The US06 drive cycles was then run on the model with the results compared to the measurements collected from the battery cell tester as validation.

With the development of the Demo Application outlined in Chapter 4.5, the process of parameterizing all the models became straight forward. As the model Library was created to be part of the tester as mentioned in Chapter 4, all that was required was:

- selecting the models to be parameterized;
- selecting the Measurement Data to be used for parameterizing;
- selecting the Measurement Data to be used for the calculation of the fitness; and
- clicking the Start Parameterizing Button.

The end results are the application outputting a list of the selected model with their corresponding parameters which can be seen in Table 3.

Table 3: Optimized Parameters for All Models

Parameters	R+	R-	R1	C1	R2	C2	R3	C3	M-	M+
1st Order R-RC	0.0044	2.9E-5	0.00144	2005						
1st Order R-RC With Hysteresis	0.0036	2.8E-5	0.00149	2003					0.029	0.028
2nd Order R-RC	0.0044	0.00012	0.0015	2003	0.099	2016				
2nd Order R-RC With Hysteresis	0.0043	6.75E-5	0.00143	2004	0.099	2003			5.6E-5	4.5E5
3rd Order R-RC	0.0043	0.00013	0.0015	2004	0.099	2001	0.099	2000		
3rd Order R-RC With Hysteresis	0.0044	3.16E-6	0.00147	2007	0.099	2002	0.099	2001	2.8E-5	4.4E-5
Simple Model	0.0043	0.00019								

Using these parameters determined by the genetic algorithm, all models were run to simulate a US06 drive cycle profile. Their output was then compared to the measured results obtained from the tester and can be seen in Figure 5-16a-g.

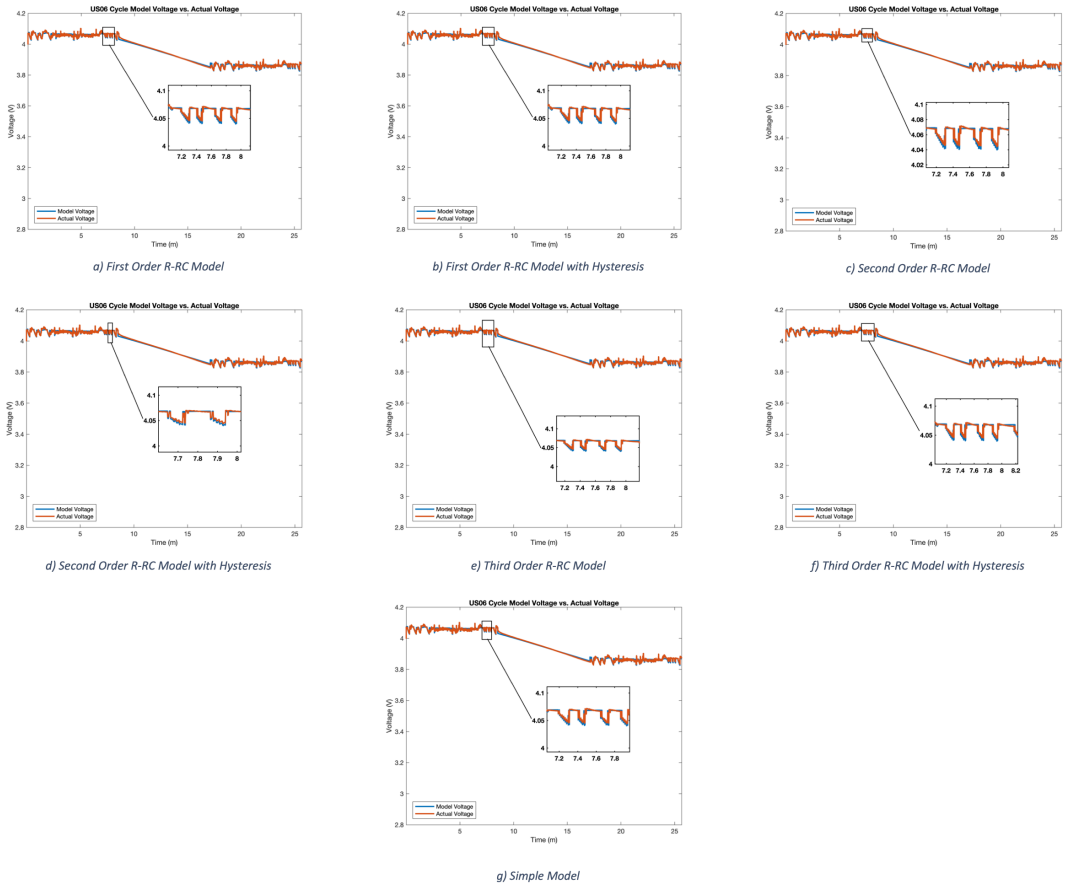

Figure 5-16: US06 Drive Profile, measured compared to model simulated

Lastly, the root mean square error was calculated for each set of simulated results, seen in Table 4, to gauge the accuracy of the models.

Table 4: Root Mean Square Error of all parameterized models in mV

Model	Root Mean Square Error (mV)
1st Order R-RC Model	8.3
1st Order R-RC Model with Hysteresis	8.26
2nd Order R-RC Model	8.28
2nd Order R-RC Model with Hysteresis	8.2
3rd Order R-RC Model	8.25
3rd Order R-RC Model with Hysteresis	8.16
Simple Model	40

These results correspond to what is expected, where we see an increase in accuracy when adding additional RC nodes. The accuracy is further improved with the addition of hysteresis from 1st to the 3rd order models. The results also agree with the best model finder application which both 3rd order R-RC models as the ones with the least root mean square error as seen in Figure 5-17.

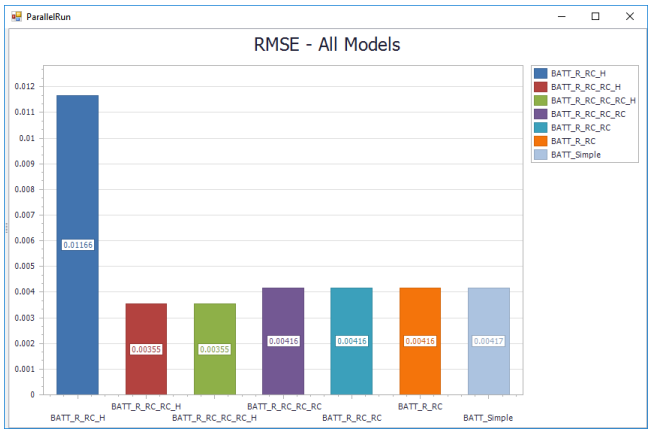

Figure 5-17: Root mean square error for all models during parameterization

5.7. Battery Characterization

The final test script is a combination of all test functions developed in this chapter to allow for users to do a complete battery characterization with the press of a button. The first step is to characterize the battery prior to ageing it. This involves the generation of the SOC-OCV curve, followed by a Capacity test and EIS test. The battery is then aged for ten cycles, followed by the characterization test done at the beginning of the test. This procedure is repeated a number of times until a desired cycle or battery age. In order to validate the process of complete validation the script was run on a SB LiMotive prismatic cell, pictured in Figure 5-18.

Figure 5-18: SB LiMotive Prismatic Battery Cell

This cell was chosen due to its durable characteristics, allowing for room for error when testing the newly developed functions and characterization scripts. The battery has a nominal voltage of 3.7 Volts, capable discharging up to 200Amps discharge for ten seconds and cycle life of two thousand cycles as seen in Table 5.

Table 5: Battery Specifications for SB LiMotive Battery Cell

5Ah SB LiMotive Cell	
Voltage (Upper, Nominal, Lower)	4.2 / 3.7 / 2.8
Peak Discharge Power (10s @ 35% SOC)	700W
Peak Charger Power (10s @ 65% SOC)	550W
Capacity (1C Rate @ 25° C)	5.2Ah
Cycle Life 80% SOC (1C @ 25° C)	2000 Cycles

The first part of the characterization was generation of the state of charge vs. open circuit voltage relation, also referred to as open circuit voltage characterization. This relation for the SB Limotive Cell can be seen in Figure 5-5. The results of this test are

also compared from cycle one to cycle 194 in Figure 5-19, below. We can see due to the battery ageing; the voltage reaches the lower limit at a much lower state of charge at cycle one hundred and eighty-seven as compared to cycle one.

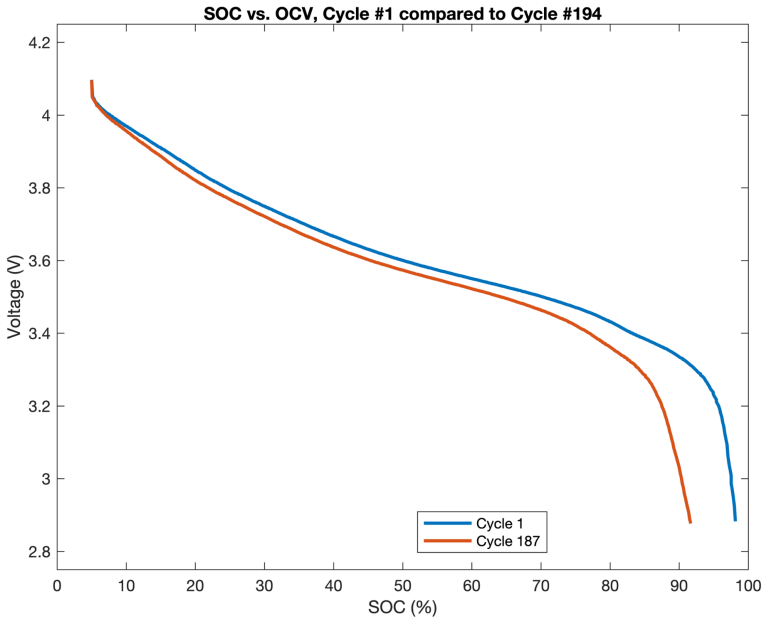

Figure 5-19: SOC-OCV Cycle 1 compared to Cycle 187

The next important characterization done during this test is capacity. During every charge and discharge cycle, the tester keeps track of the amount of current flowing into the battery and out. This provides the means to measure the battery capacity at each cycle and is expected to gradually decrease after each cycle, which can be seen in Figure 5-15. The data shows a three percent loss of capacity over one hundred and eighty-seven cycles as compared to a fresh cell. Although the battery is expected to only degrade one percent over 200 cycles[50], the lack of a temperature-controlled environment could be a result of the deviation.

The next characterization done was the electro-impedance spectroscopy. The length required to run a full Electro-Impedance Spectroscopy from ten kilohertz to ten milli-hertz is quite lengthy, meaning it was only possible to run every ten charge/discharge cycles. Due to the increase in impedance as the battery ages, the electro-impedance curve moves further down the x, real impedance, axis. This is an indication of battery health deteriorating and is seen during this characterization in Figure 5-12 and Figure 5-13.

The final part of characterization is the running of drive cycles profiles. Due to oversight at the beginning of the test, drive cycles were not run throughout the battery characterization and only run once at the end. This is noted for future work as it helps to characterize the battery's real-world performance as it starts to age. The data obtained from running of the drive cycle profiles can also be very helpful in the parameterization of a number of battery models. The data collected was used to parameterize using the model framework developed, with the parameters for each mode displayed in Table 3, a comparison of simulated to measure of the US06 drive cycle in Figure 5-16 and the root mean square error in Table 4.

6. Conclusion

As batteries continue to become a major component of the vehicle in the electrified automotive future, the tools required for the evaluation and analysis of cells will play an increasingly important role. Tools that simplify battery testing, analysis and model

parameterization, provide users the foundation for breakthroughs in this industry. The results of this research provide tools and strategies that enable:

1. the development of a robust, comprehensive and user-friendly software platform for battery testing;
2. development of new testing strategies to automate Electro-Impedance Spectroscopy test, ageing test, capacity test and SOC-OCV curve generations.
3. Creation of a battery modelling framework, allowing for ease of battery model integration into post-processing of data. The framework provides the foundation of generic model integration and optimization without the need for changes to an applications original code.
4. Integration of a battery model library and optimization strategy into the framework provides an example for future expansion of the system.

Lastly, all of the defined contributions were utilized to collect real world test data, integrate battery models and parameterize and validate battery models. These were applied for the characterization of the SB LiMotive battery cell as shown in Figure 5-18. The results obtained for the battery cell using the characterization script confirm that found in literature. All models in the library that were parameterized using the framework provided, provide results that are similar to what is expected. While the current library, framework and application can be excellent tools for battery research and testing, there is still room for improvement.

6.1. Future Work

The Software, Hardware and Framework developed provide an excellent foundation for battery testing and research. However, there is still a number of improvements that would further aid in the processes.

While the framework does close the gap between research tools such as MATLAB and the tester production software, there is still some manual work required to use Simulink Models as part of the Tester. This can be solved by further integrating a module in MATLAB. Mathworks provides a development environment to expand on C/C++ code generation, [51], allowing for complete automation. This could allow for direct communication between the two software and no interaction from the user for model preparation, optimization and evaluation.

The Genetic Sharp Library provides an excellent tool for optimization and serves as an example for model optimization. However, the framework could be further built to include more advanced, evolutionary parameterization strategies. Lastly, the types of models can be expanded to include a more comprehensive list of models including electro-chemical models and Electro-Impedance Spectroscopy models.

From a hardware perspective, there is still room for improvement as well. Moving some of the complexities of battery testing developed in the software to the hardware level will help increase overall reliability of the system while providing much more real time response of developed testing strategies.

7. References

1. Høyer, K.G., *The history of alternative fuels in transportation: The case of electric and hybrid cars.* Utilities Policy, **16**(2): p. 63-71.

2. Narins, T.P., *The battery business: Lithium availability and the growth of the global electric car industry.* The Extractive Industries and Society, **4**(2): p. 321-328.

3. Stubbe, R. *Global Demand for Batteries Multiplies*. Bloomberg [21 Dec 2018]; Available from: https://www.bloomberg.com/news/articles/2018-12-21/global-demand-for-batteries-multiplies. Last accessed on 30 Mar 2020

4. Curry, C. *Lithium-ion Battery Costs and Market*. [05 July 2017]; Available from: https://data.bloomberglp.com/bnef/sites/14/2017/07/BNEF-Lithium-ion-battery-costs-and-market.pdf. Last accessed on 30 Mar 2020

5. PILLOT, C., *Lithium ion battery raw material Supply & demand 2016-2025.* AABC Europe, (05 Dec 2018).

6. James Eddy, A.P., Jasper van de Staaij. *Recharging economies: The EV-battery manufacturing outlook for Europe*. [June 2019]; Available from: https://www.mckinsey.com/industries/oil-and-gas/our-insights/recharging-

economies-the-ev-battery-manufacturing-outlook-for-europe. Last accessed on 30 Mar 2020

7. *Electric Vehicle Battery: Materials, Cost, Lifespan*. [09 March 2018]; Available from: https://www.ucsusa.org/clean-vehicles/electric-vehicles/electric-cars-battery-life-materials-cost#! Last accessed on 30 Mar 2020

8. Rissman, J. *The Future Of Electric Vehicles In The U.S., Part 1: 65%-75% New Light-Duty Vehicle Sales By 2050*. [14 Sept 2017]; Available from: https://www.forbes.com/sites/energyinnovation/2017/09/14/the-future-of-electric-vehicles-in-the-u-s-part-1-65-75-new-light-duty-vehicle-sales-by-2050. Last accessed on 30 Mar 2020

9. Schmidt, E. *Electric Vehicle Sales In Canada, 2017*. [08 Feb 2018]; Available from: https://www.fleetcarma.com/electric-vehicle-sales-canada-2017/. Last accessed on 30 Mar 2020

10. Lenferna, G.A., *Can we equitably manage the end of the fossil fuel era?* Energy Research & Social Science, **35**: p. 217-223.

11. Orsi, F., et al., *A multi-dimensional well-to-wheels analysis of passenger vehicles in different regions: Primary energy consumption, CO2 emissions, and economic cost.* Applied Energy, **169**: p. 197-209.

12. Vancoillie, J., et al., *Comparison of the renewable transportation fuels, hydrogen and methanol formed from hydrogen, with gasoline – Engine efficiency study.* International Journal of Hydrogen Energy, **37**(12): p. 9914-9924.

13. Berndt, C. *Energy expert explains why Tesla and the electric car industry is here to stay*. [30 Mar 2017]; Available from: https://www.teslarati.com/energy-expert-says-tesla-electric-vehicle-industry-here-stay/. Last accessed on 30 Mar 2020

14. *How Do Gasoline & Electric Vehicles Compare?* ; Available from: https://avt.inl.gov/sites/default/files/pdf/fsev/compare.pdf. Last accessed on 30 Mar 2020

15. Chen, H., et al., *Progress in electrical energy storage system: A critical review.* Progress in Natural Science, **19**(3): p. 291-312.

16. Lu, L., et al., *A review on the key issues for lithium-ion battery management in electric vehicles.* Journal of Power Sources, **226**: p. 272-288.

17. Farag, M., *Thermal-Electrochemical Modeling and State of Charge Estimation for Lithium Ion Batteries in Real-Time Applications.* McMaster Macsphere.

18. Farag, M.S., et al. *A comparative study of Li-ion battery models and nonlinear dual estimation strategies*. in *2012 IEEE Transportation Electrification Conference and Expo (ITEC)*. 2012.

19. Ahmed, R., *Modeling and State of Charge Estimation of Electric Vehicle Batteries.* McMaster Macsphere.

20. Meng, J., et al., *Overview of Lithium-Ion Battery Modeling Methods for State-of-Charge Estimation in Electrical Vehicles*. Vol. 8. 2018. 659.

21. *Battery Chargers and Charging Methods*. Available from: https://www.mpoweruk.com/chargers.htm. Last accessed on 30 Mar 2020

22. Farag, M., M. Fleckenstein, and S.R. Habibi, *Li-Ion Battery SOC Estimation Using Non-Linear Estimation Strategies Based on Equivalent Circuit Models*, in *SAE Technical Paper Series*. 2014, SAE International.

23. Li, A., et al., *Fast Characterization Method for Modeling Battery Relaxation Voltage*. MDPI. Vol. Battery Modeling. 2016.

24. Buller, S., et al., *Impedance-based simulation models of supercapacitors and Li-ion batteries for power electronic applications.* IEEE Transactions on Industry Applications, **41**(3): p. 742-747.

25. Payyappilly, B. and V. John. *A generic in-circuit modeling approach for electrochemical batteries*. in *2016 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*. 2016.

26. Hu, X., S. Li, and H. Peng, *A comparative study of equivalent circuit models for Li-ion batteries.* Journal of Power Sources, **198**: p. 359-367.

27. Dai H, J.B., Wei X, *Impedance Characterization and Modeling of Lithium-Ion Batteries Considering the Internal Temperature Gradient.* Energies 2018.

28. Mallawaarachchi, V. *Introduction to Genetic Algorithms*. [07 Jul 2017]; Available from: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. Last accessed on 30 Mar 2020

29. Eugen Stripling, S.v.B., Bart Baesens. *Solving the Knapsack Problem with a Simple Genetic Algorithm*. [12 Mar 2017]; Available from: https://www.dataminingapps.com/2017/03/solving-the-knapsack-problem-with-a-simple-genetic-algorithm/. Last accessed on 30 Mar 2020

30. Joongpyo Shim, K.A.S., *Characterization of High-Power Lithium-Ion Cells During Constant Current Cycling*. 2003, Lawrence Berkeley National Laboratory: U.S. Department of Energy Office of Scientific and Technical Information.

31. Makinejad, K., *A Lumped Electro-Thermal Model for Li-Ion Cells in Electric Vehicle Application.* ResearchGate.

32. Bloom, I., *Battery Testing and Life Estimation in the US*. 2011, Argonne National Laboratory.

33. Stroe, D.I., et al. *Diagnosis of lithium-ion batteries state-of-health based on electrochemical impedance spectroscopy technique*. in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*. 2014.

34. *Coulombic and Energy Efficiency with the Battery*. [25 Oct 2017]; Available from: https://batteryuniversity.com/learn/article/bu_808c_coulombic_and_energy_efficiency_with_the_battery. Last accessed on 30 Mar 2020

35. Yang, F., et al., *A study of the relationship between coulombic efficiency and capacity degradation of commercial lithium-ion batteries.* Energy, **145**: p. 486-495.

36. *Dynamometer Drive Schedules*. [31 Jan 2017]; Available from: https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules. Last accessed on 30 Mar 2020

37. Lee, E.A., *The Problem with Threads.* Electrical Engineering and Computer Sciences University of California at Berkeley.

38. Mark E. Russinovich, D.A.S. *Processes, Threads, and Jobs in the Windows Operating System*. Microsoft Press Store [17 Jun 2009]; Available from: https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7. Last accessed on 30 Mar 2020

39. Microsoft. *Thread Priority*. .Net API; Available from: https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread.priority?view=netframework-4.7.2. Last accessed on 30 Mar 2020

40. Microsoft. *Scheduling Priorities*. .Net API [31 May 2018]; Available from: https://docs.microsoft.com/en-us/windows/desktop/procthread/scheduling-priorities. Last accessed on 30 Mar 2020

41. Ajay Kshemkalyani, M.S., *Distributed Shared Memory*, in *Distributed Computing*. 2008, Cambridge University: Cambridge University Press.

42. Open Source, M.C. *Memory-Mapped Files*. [03 Mar 2017]; Available from: https://docs.microsoft.com/en-us/dotnet/standard/io/memory-mapped-files. Last accessed on 30 Mar 2020

43. Open Source, M.C. *Mutexes*. .Net API [03 Mar 2017]; Available from: https://docs.microsoft.com/en-us/dotnet/standard/threading/mutexes. Last accessed on 30 Mar 2020

44. Shead, M. *Understanding State Machines*. An intro to Computer Science concepts [11 Feb 2018]; Available from: https://medium.freecodecamp.org/state-machines-basics-of-computer-science-d42855debc66. Last accessed on 30 Mar 2020

45. Stroustrup, B., *What is object-oriented programming?* IEEE Software, **5**(3): p. 10-20.

46. *Software Framework*, in *techopedia*.

47. *Framework*, in *TechTerms*.

48. Colburn, T.S., G, *Abstraction in Computer Science.* Minds and Machines.

49. Jumba, I. *Overview of Inheritance, Interfaces and Abstract Classes in Java*. [25 Sept 2015]; Available from: https://medium.com/@isaacjumba/overview-of-inheritance-interfaces-and-abstract-classes-in-java-3fe22404baf8. Last accessed on 30 Mar 2020

50. *TB077170226mFF1 Specification*, in *Batterist*. 2011.

51. *Build Integrated Code Outside the Simulink Environment*. Available from: https://www.mathworks.com/help/ecoder/examples/integrating-the-generated-code-into-the-external-environment.html. Last accessed on 30 Mar 2020