STATISTICALLY SIGNIFICANT CORRELATIONS OF QUICK DIAGNOSTIC TESTS AND STATES-OF-HEALTH

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

CHEMISTRY

By

Rezan Ezgi Sevgen

August 2025

STATISTICALLY SIGNIFICANT CORRELATIONS of QUICK DIAGNOSTIC TESTS and STATES-of-HEALTH

By Rezan Ezgi Sevgen

August 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Burak Ülgüt(Advisor)

Mehmet Kadri Aydınol

Fahri Alkan

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

STATISTICALLY SIGNIFICANT CORRELATIONS OF QUICK DIAGNOSTIC TESTS AND STATES-OF-HEALTH

Rezan Ezgi Sevgen
M.S. in Chemistry
Advisor: Burak Ülgüt
August 2025

With the popularity of electrical vehicles rising, research on Lithium ion batteries have the utmost importance. For an efficient and safe usage of these batteries, their performance and its decline have to be closely monitored. In this work, two techniques were used for predicting State of Health (SoH) for 8 Aspilsan INR18650 batteries: Electrochemical Impedance Spectroscopy (EIS) and Intermittent Current Interruption (ICI).

In this thesis, firstly Lithium ion batteries and their working principle is discussed. Afterwards, background information for both EIS and ICI were given for understanding why they are suitable for application; moreover, how they can be utilized to understand the battery chemistry that causes the SoH decline. Following, how EIS and ICI were combined and used to correlate to battery SoH were presented. All 8 batteries were tested with a sequence made up of EIS, ICI, charging and discharging procedures in different States of Charge every 50 cycles. Then, the data acquired from these tests were checked against the capacities known from the cycling. For correlating these parameters, Distance Correlation (dCor) was applied as dCor reveals any type of association between two parameters, not limited to linearity.

Afterwards, to verify the data acquired, the errors for ICI parameters were investigated. Moreover, Kramers-Kronig test was applied to all the EIS data acquired to validate the quality of the data. To add, binning using the ICI parameters were used to showcase the prediction power of using ICI parameters acquired. In the end, it is explained how and why capacity of this set of batteries can be predicted with 1.8% error range using ICI test at specific conditions in a relatively short amount of time.

Keywords: Li-ion batteries, State of Health, Electrochemical Impedance Spectroscopy, Intermittent Current Interruption.

ÖZET

HIZLI TANI TESTLERİ İLE SAĞLIK DURUMLARI ARASINDAKİ İSTATİSTİKSEL OLARAK ANLAMLI KORELASYONLAR

Rezan Ezgi Sevgen
Kimya, Yüksek Lisans
Tez Danışmanı: Burak Ülgüt
Ağustos 2025

Elektrikli araçların popülaritesinin artmasından dolayı, Lityum iyon piller üzerine yapılan araştırmalar büyük önem taşımaktadır. Bu bataryaların verimli ve güvenli bir şekilde kullanılabilmesi için performanslarının ve düşüşlerinin yakından takip edilmesi gerekmektedir. Bu çalışmada, 8 adet Aspilsan INR18650 pilinin Sağlık Durumunu (SoH) tahmin etmek için iki teknik kullanılmıştır: Elektrokimyasal Empedans Spektroskopisi (EIS) ve Aralıklı Akım Kesimi (ICI).

Bu tezde, öncelikle Lityum iyon piller ve çalışma prensipleri ele alınmıştır. Daha sonra, neden uygulamaya uygun olduklarını anlamak için hem EIS hem de ICI için arka plan bilgileri verilmiştir; ek olarak, SoH düşüşüne neden olan pil kimyasını anlamak için nasıl kullanılabilecekleri anlatılmıştır. Ardından, EIS ve ICI'nin nasıl birleştirildiği ve pil SoH'si ile ilişkilendirmek için nasıl kullanıldığı sunulmuştur. Her bir 8 pil, her 50 döngüde bir farklı Şarj Durumlarında (SoC) EIS, ICI, şarj ve deşarj prosedürlerinden oluşan bir sekans ile test edilmiştir. Daha sonra, bu testlerden elde edilen veriler, çevrimden bilinen kapasitelerle karşılaştırılarak kontrol edilmiştir. Bu parametreleri ilişkilendirmek için Mesafe Korelasyonu (dCor) uygulanmıştır, çünkü dCor iki parametre arasında doğrusallıkla sınırlı olmayan her türlü ilişkiyi ortaya koymaktadır.

Daha sonra, elde edilen verileri doğrulamak için ICI parametrelerindeki hata payları incelenmiştir. Ayrıca, verilerin kalitesini doğrulamak için elde edilen tüm EIS verilerine Kramers-Kronig testi uygulanmıştır. Ek olarak, elde edilen ICI parametrelerini kullanmanın tahmin gücünü göstermek için ICI doneleri kullanılarak gruplama tekniği kullanılmıştır. Sonuç olarak, bu pil setinin kapasitesinin ICI testi kullanılarak belirli koşullarda nispeten kısa bir sürede nasıl ve neden %1,8 hata aralığında tahmin edilebileceği açıklanmıştır.

Anahtar sözcükler: Lityum iyon pil, batarya sağlık durumu, Elektrokimyasal Empedans Spektroskopisi, Aralıklı Akım Kesimi.

Acknowledgement

First and foremost, my deepest thanks go to my dear mother, *Şahver Aslı Selçuk*, who shaped the person I am today. You have been my greatest supporter, the source of my strength, and my silver lining in all hardships. You are the best mom to have ever walked this earth. All of my gratitude begins and ends with you. *Sedat abi,* you are my second biggest supporter. I'm so glad our paths crossed, and that I have both you and a wonderful older sister in my life.

Next, I would like to express my thanks to Assoc. Prof. Dr. Burak Ülgüt. Before this period, I wasn't even considering pursuing academia further. However, thanks to you, I learned how supported, acknowledged, and lucky I could feel in an unfamiliar territory. I know that this experience was a once-in-a-lifetime opportunity. Thank you for this chance and for your guidance.

I'd also like to thank the rest of the professors from the Chemistry Department for teaching me that it's okay to say "I don't know" and encouraging me to be more curious. Also my labmates, especially Fatma Altuğ, Gözde Karaoğlu, Gökberk Katırcı, and Fazlı Eren Civan have helped me in one way or another. Thank you for your academic and emotional support. This has been a fun journey with you all. Moreover, for the past seven months, I have had another place to call home within the department. To my labmates downstairs, I wish the best for you. Special thanks go to Ruby Phul, the homemaker; thank you for your patience and understanding. You have shown me a new horizon in chemistry, and I feel lucky to work with you.

Additionally, to the rest of my extended family, you've always been there for

me, and I wouldn't be where I am without any of you. In my eyes, my extended family includes a number of close friends. Special thanks to Betül Sarıtepe, for being the perfect roommate and lifetime company; to add, Ece Özgür and Çağla Aydın for always being there to listen and for your understanding, exactly when I needed it. I hope we all continue to find ourselves wherever we wish to be and share the same joy whenever we get together.

Lastly, my thanks go to Murat Esen. Throughout all these years, you've been my wings when I felt like I was falling, my rainbow on rainy days; and you made everything even happier and funnier when it was already sunny. I can't wait to share a surname with you, but I'm even more excited to share experiences and a lifetime together. You've given me an amazing second family, which I didn't even know I could appreciate this much this early on. I wish all of us nothing but the best in the years to come.

Contents

List of Figures

List of Tables

List of Abbreviations

LIB lithium-ion battery

NMC lithium nickel manganese cobalt oxide

SoH State-of-Health

SoC State-of-Charge

EIS Electrochemical Impedance Spectroscopy

ICI Intermittent Current Interruption

dCor distance correlation

SEI solid electrolyte interphase

KK Kramers-Kronig

Chapter 1

Introduction

1.1 Lithium-Ion Batteries

The lithium-ion battery (LIB) market is an ever-growing one especially with the transition from cars with combustion engines to the electrical cars, to reduce carbon footprint. Manufacturing capacity of LIBs are expected to be nearly four times of 2023 data in 2030, reaching 7.6 terawatt-hours per year [1]. Among all the electrical energy storage materials, LIBs are used mainly due to the energy density. Energy storage is higher per kilogram than other materials that serve the same purpose. To add, their charge retention and cycle life is excellent, they offer a higher cell voltage than their counterparts and they can operate in various temperatures [2]. From their first production around 1970s [3], they have been a popular choice for many electronic devices' power supply such as mobile phones and portable computers for their robustness and versatility.

1.1.1 Working Principle and Materials Used in LIBs

Energy is supplied to or drawn from the battery through the movement of lithium ions' movement. For any electrochemical change to occur, all of the inside components of the battery has to be ionically conductive. LIBs are comprised of a cathode, an anode, an electrolyte, current collectors for each terminal and a separator, represented in Figure 1.2. Lithium-ion battery materials have been in development, with notable breakthrough of graphite anode [4]. Usually, aluminum is used for negative electrode side current collector, and copper for positive electrode side current collector. Graphitic carbon are the most popular choice for the anode due to its high surface area, but there are many options used in the cathode side. Lithium Manganese Oxides, Lithium Nickel Cobalt Oxides, Lithium Nickel Manganese Oxides, Lithium Nickel Manganese Cobalt Oxides and Lithium Iron Phosphate Oxides are some of the options with various compositions. The electrolyte that provides ionic conductivity between the electrodes generally comprise of salt with Li^+, generally lithium hexafluorophosphate and lithium perchlorate, with an organic solvent that can be a mixture of organic solvents such as polyethylene glycol/diethylene glycol and propylene carbonate/ethylene carbonate.

NMC cathodes vary in terms of their elemental compositions. They are a popular choice for cathode materials even though usage of cobalt is a environmentally and ethically challenging choice. However, as they offer both stability and high capacity, they are still being used. Moreover, to tune the durability and energy density, the content of the cathodes are varied. Three of the most popular NMC types are $LiNi_{0.33}Mn_{0.33}Co_{0.33}O_2$ or NMC111, $LiNi_{0.6}Mn_{0.2}Co_{0.2}O_2$ or NMC622 and $LiNi_{0.8}Mn_{0.1}Co_{0.1}O_2$ or NMC811. The optimum choice should be made for

Feature	Comparison
Environmental Safety	Mn > Ni > Co
Structural Stability	Co > Ni > Mn
Chemical Stability	Mn > Ni > Co
Abundance	Mn > Ni > Co
Electrical Conductivity	Co > Ni > Mn

Figure 1.1: Nickel, Cobalt and Manganese comparisons for NMC cathode tuning.

production and usage area; as Nickel improves the capacity and Cobalt and Manganese are typically used for chemical stability. The other aspects related to these materials are reported by Yerkinbekova, 2024 [5] and are reproduced in Figure 1.1.

During the charging process, lithium in the positive electrode is oxidized, ending up as Li^+ and released to the electrolyte (Figure 1.2(b)). Whereas on the negative electrode, graphite anode, reduction occurs as the Li^+ enter the matrix and gain electron from the current supplier. In the discharging process, the opposite happens; as ions are released from the negative electrode, they intercalate into the cathode, where Li^+ is reduced back to lithium atoms (Figure 1.2(a)). On the positive electrode side, commercialized secondary (rechargeable) batteries utilize lithium metal phosphate (e.g lithium iron phosphate, LFP) or lithium metal oxides (e.g lithium nickel manganese cobalt oxides, NMC). The overall reactions that take place in the positive electrode and negative electrode are given in Figure 1.3. For the positive electrode, $LiMO_2$ represents lithium (mixed) metal oxide, and C represents carbon based negative electrode material [2].

During initial manufacturing, a layer of reaction products form. This layer

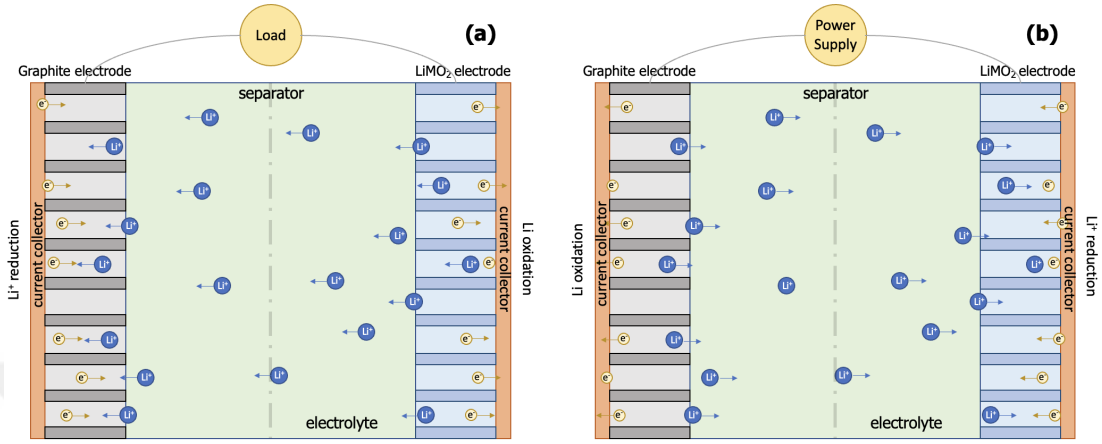

Figure 1.2: A schematic of secondary lithium ion batteries during discharge (a) and charge (b).

is named as Solid Electrolyte Interphase (SEI), and the reactions occur between the electrodes and the electrolyte. This layer inhibits some of the activity of the electrodes. SEI formation is inescapable, and always take part in performance drop, due to causing irreversible Lithium inactivation and surface area loss in the anode side. SEI, after its first formation, goes through changes in size with the usage of the battery. It keeps actively forming and dissolving and hence, is a heterogeneous, complex buildup [6]. Though its contribution to capacity fade, SEI buildup is also crucial for the battery as it allows the movement of ions but not the electrons. If it does not form, the electron transfer to/from the electrodes can cause parasitic reactions, electrolyte decomposition, Li plating and consequently inhibition of the electrodes, short circuits within the batteries and even thermal runaways [7].

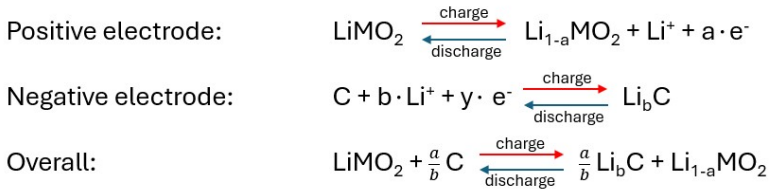

Figure 1.3: Reactions that take place in the electrodes of a secondary lithium ion battery.

1.2 Charging and Discharging Protocols

For both charging and discharging, current is either supplied to or drawn from the battery while the voltage response is followed. The main two protocols for charging and discharging are constant-current (CC) and constant-current constant-voltage (CCCV) protocols. As the name suggests, in CC mode, the negative current is applied throughout the discharging and positive current is applied for charging processes. In the CCCV's case, constant current is applied until a set point of voltage is reached, and then the current is gradually increased until a specific current reading is obtained from the battery while discharging. The charging CCCV is done vice versa where the current is gradually lowered until an aimed current reading is obtained. In Figure 1.4, a plot for CCCV charging (a) and CC discharging (b) are showcased, which include both voltage-time and current-time data. One important definition used for the charging and discharging processes is C-rate. C-rate defines the applied or withdrawn current through the battery's capacity. The current that will fully charge or discharge a battery in one hour is 1C, whereas if the process is to be completed in 4 hours, it corresponds to C/4. In one case, for a battery that has a capacity of 2.8Ah, application of 1.4A for 2 hours for charging corresponds to application of positive C/2 current. Consequently, discharging with 0.56A is the same as withdrawn current of C/5. One

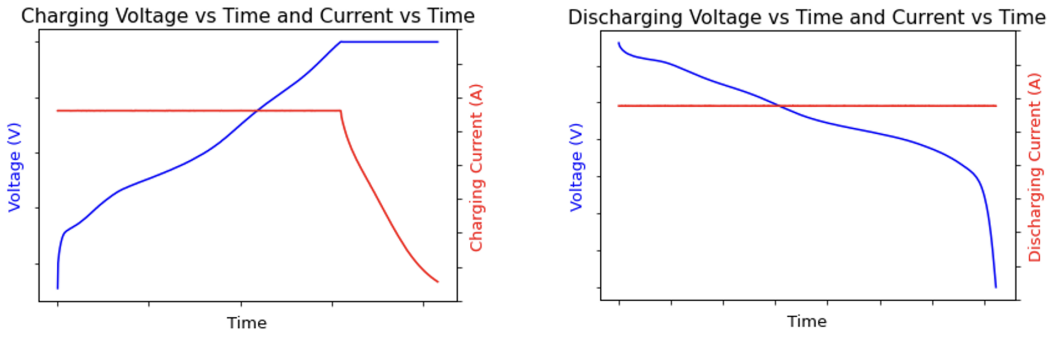

Figure 1.4: CCCV charging and CC discharging voltage-time and current-time curves.

consecutive full charge and discharge session for a battery is called a 'cycle'.

Another largely popular parameter for charging/discharging process is State of Charge (SoC). This parameter is the remaining capacity of the battery in the specific cycle of usage. The calculation of it varies from one source to another as the retainable capacity of the specific cycle can be determined in different manners. However, for this thesis, simple Coulomb counting [8] shown in Equation 2 is used for the purpose where the efficiency is taken as 1. Coulomb counting works by the current state of charge the battery has before the time of charging/discharging, t_0, and adding that SoC(t_0) to the ratio of applied current to the C_0.

$$SoC(t) = SoC(t_0) - \int_{t_0}^{t} \frac{I(t)}{C_0} dt \tag{1.1}$$

1.3 State of Health (SoH)

As the market grows, LIBs' performance gains more importance for a safe usage. For tracking the performance drop, a key aspect of the battery is used, namely State-of-Health (SoH). SoH relates to the battery's remaining useful life. In this work, its calculation is done through Equation 1.2; the capacity the battery retains after some number of charge and discharge cycles C_t divided by the capacity of the battery in pristine condition C_0. However, there are other works that define SoH using key parameters other than capacity, such as internal resistance [9] and energy [10]. A rule of thumb for the battery's end of life is when the SoH reaches the value of 80% [11].

$$SoH = \frac{C_t}{C_0} \tag{1.2}$$

Batteries, being used or not, go through changes in terms of chemical content. Those changes are mostly detrimental in nature, and cause degradation of health. If the battery is in use and SoH decline is observed, this is called cycle aging; whereas the decline during the storage is named as calendar aging. This work is exclusively done for cycle aging; however, the research on calendar aging is also an important topic for the internal understanding and proper storage of the batteries [12]. The degradation during storage and usage can be due to many different reasons that can be grouped under solid electrolyte interphase formation, active site loss of cathode and/or anode and crack growth on the electrodes, represented in Figure 1.5 [13]. Although some of the degradation modes contribute significantly more, any of those changes in the battery chemistry

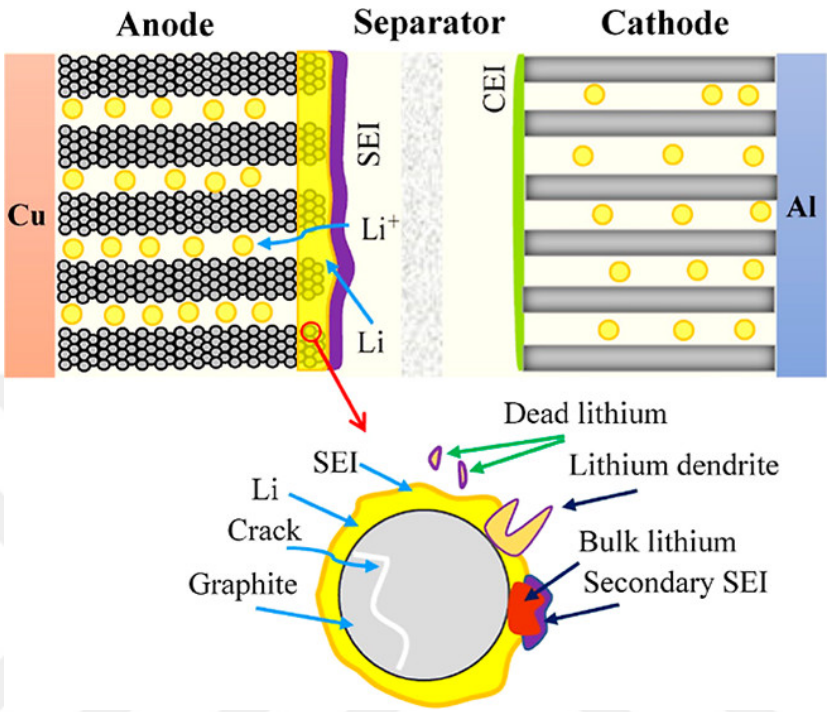

Figure 1.5: A representation of degradation modes. Reprinted with permission from ACS Appl. Energy Mater. 2022, 5, 5, 6462–6471. Copyright 2022 American Chemical Society.

results in capacity fade, which manifests itself through SoH.

The health decline can occur suddenly and drastically, making the battery unsuitable for operation. To obtain SoH, C_t data is gathered from Coulomb counting while fully discharging the battery. However, this may not be possible due to requiring a long time while the battery is in use. To add, SoH shows a different trend for each specific battery even from the same batch. Hence, the estimation, and prediction of SoH is critical. There are numerous studies about aging where different models are developed for this exact purpose. The need for all the different models stem from the nature of a battery: only voltage, current and time data can be collected without destructive methods. This information has to be thoroughly analyzed to be meaningful, hence the need for the models.

The models fall into two distinct categories: Experimental approach and model-based approach [14]. Experimental methods use voltage, current and time data directly, or with little manipulation of these data, to relate to SoH. Relatively more popular experimental models can be listed as Electrochemical Impedance Spectroscopy [15], Pulse Techniques [16–18], Incremental Capacity Analysis/Differential Voltage Analysis [19] and usage of Open Circuit Voltage [20]. Combinations of these models can also be found throughout the literature [21–24].

In this research, the aim is to develop a useful model for the degradation of Aspilsan INR18650. The sample choice was done for numerous reasons such as providing reproducible data, ease of access, and being local product to name a few. As our methods, we use Electrochemical Impedance Spectroscopy (EIS) and Intermittent Current Interruption (ICI). These methods are chosen as they are non-invasive, informative and practical. They offer information without disturbing the chemistry and in a relatively short time period. Thus, they fulfill the usefulness sought by being quick, simple tests should lead to SoH and more importantly, the degradation mode.

1.4 Electrochemical Impedance Spectroscopy

Impedance is the resistance towards any transport and charge transfer throughout an electrical system. For transient systems that complex processes occur in, Electrochemical Impedance Spectroscopy (EIS) is one of the most popular investigation methods. The main reasons behind this are due to EIS's non-invasive and informative nature. EIS is employed for different systems ranging from biological

environments to corrosion. Though it was first used in late 1800s, the adaptation to electrochemical systems took place around 1940 [25]. For batteries' case, EIS can distinguish various processes. Occurrences in SEIs for both of the positive and negative electrodes, charge transfers in the electrodes themselves; as well as diffusion and overall resistance. For all these occurrences the information can be collected because they occur in distinct timelines that are well documented through years. Moreover, EIS is actively being used to conclude many physical parameters of batteries other than SoH; such as internal temperature [26], state of charge [27], diffusion parameters [28] and also inspection of the electrodes [29], SEI layers [30] and more.

Application of EIS boils down to sending a signal of sinusoidal wave and following the response. For electrochemical systems, mainly voltage or current signal is sent and corresponding response of current or voltage is followed. Specifically for batteries, Galvanostatic EIS is utilized where the excitation signal is in current form (Equation 1.3) and voltage response is recorded (Equation 1.4). The reason why Galvanostatic mode gets used for secondary batteries is that the resistance is relatively lower when compared to other electrochemical systems for commerical batteries; hence, small voltage applications can yield in bigger current responses. Since the current applied can be fine tuned, this mode is preferred over Potentiostatic EIS [31].

The excitation signal of current (I_e) is sent with a frequency (f) and an amplitude (I_a) to the system. The voltage response will have a lag compared to the current signal which is the phase shift (ϕ), and a different amplitude (V_a), also seen in Figure 1.6. After collecting the voltage response (V_r), another excitation is sent with a different frequency. The frequency range is chosen according to

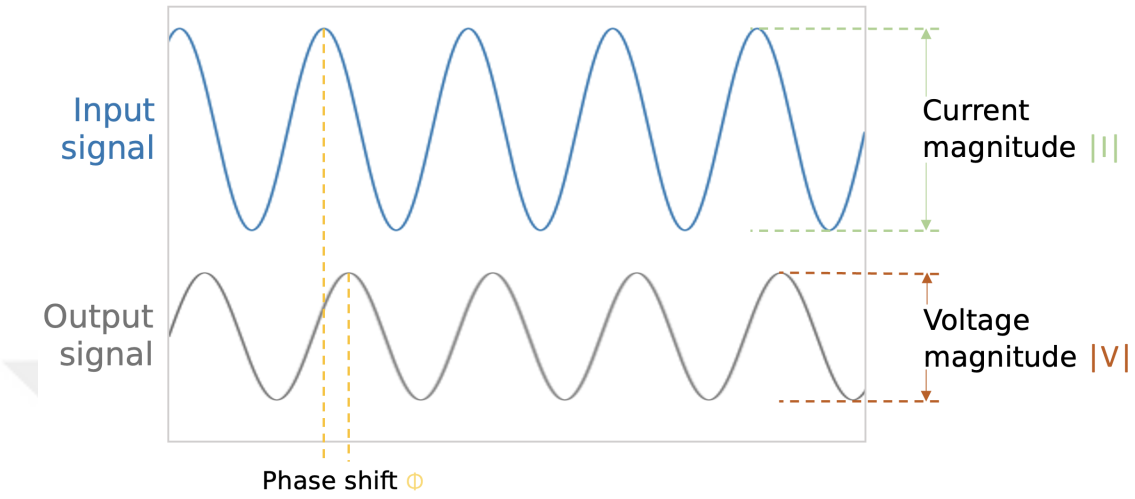

Figure 1.6: Galvanostatic EIS schematic representation of the input and output signals.

the system to be investigated, the time limitations on the low frequency side and the instrument limitations on the high frequency side. Typically, for batteries, a range of mHz to kHz is used. The number of data points that are gathered is specified under 'points per decade' setting.

$$I_e(f) = I_a sin(2\pi f) \tag{1.3}$$

$$V_r(f) = V_a sin(2\pi f + \phi) \tag{1.4}$$

The impedance (Z) of a system is calculated by dividing the voltage signal to the current response. Z has two components: a real (Z_r) and an imaginary part (Z_i), also seen in Equation 1.5. Nyquist plot of data can be constructed by plotting $+Z_r$ (x-axis) against $-Z_i$ (y-axis). Though it does not include information about frequencies explicitly, this type of plotting is a popular choice due to its visualization power. As the shapes visible to naked eye can reveal the processes

in the system, Nyquist plots are a valuable tool for electrochemical analysis. As another option, using the modulus ($\|Z\|$) and ϕ, Bode plots can be obtained. Bode plots are combinations of log Modulus (Mod) versus log frequency and -Phase Angle (-Phz) versus log frequency curves, represented with twin y axes. Similar to the Nyquist plots, Bode plots visualize the phase shifts and number of time constants of an electrochemical system.

$$Z = \frac{V_a sin(2\pi f + \phi)}{I_a sin(2\pi f)} = Z_r + iZ_i = \|Z\| * (cos(\phi) + isin(\phi)) \tag{1.5}$$

For inspecting batteries, EIS is generally dissected into three parts as high frequency range, mid frequency range and low frequency range. The low frequency range at the right hand side of the Nyquist plots mostly covers the ion transport and diffusion processes; as these take the longest time to occur. Conversely, the high frequency range showcases the Li^+ insertion to/dissolution from the SEI layer, observed in the left hand side of the Nyquist plots, which also includes the inductance that mainly arise from the cabling of the electrochemical testing instrument. The mid frequency range, 100 Hz to 100 mHz, typically covers the anodic charge transfer processes; namely Li^+ oxidation [32,33]. Hence, the differentiation of these processes throughout aging progress can be derived with EIS to conclude which part has the most contribution to the performance decline.

Though EIS is a thorough method to shed light on almost each process, battery chemistry is equally complex and compelling to deconvolute. Moreover, even the same batch of batteries can yield varying results for the exact same experiment setup. Consequently, to ensure the validity and robustness of the model, 8 batteries are employed.

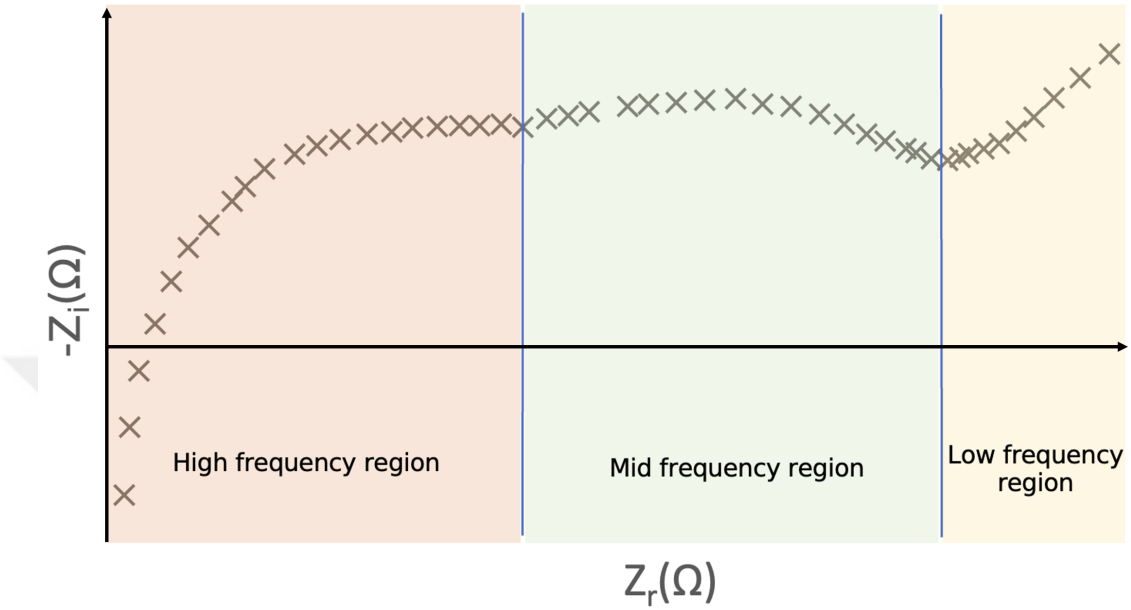

Figure 1.7: A representation of LIB EIS data with indicated regions.

1.5 Intermittent Current Interruption

Current interruption techniques have been around since 1970's [34], and the correlations behind all pulse techniques were mathematically explained around the same decade [35]. Current's interruption, as well as pulse application of it give clues about the mass transport inside the system.

Intermittent Current Interruption (ICI) is a straightforward, powerful technique developed by Matthew J. Lacey in 2015. It helps to learn about the material transport processes inside the battery, specifically overall resistance of the battery (R, $\Omega \cdot \mathrm{cm}^2$) and diffusion resistance coefficient (k, $\Omega \cdot \mathrm{s}^{-\frac{1}{2}}$).

The implementation of the technique aforementioned involves slow charging or discharging process, to ensure the reactions in the battery are diffusion controlled, followed by interruption of the current for a short time, hence the name. The

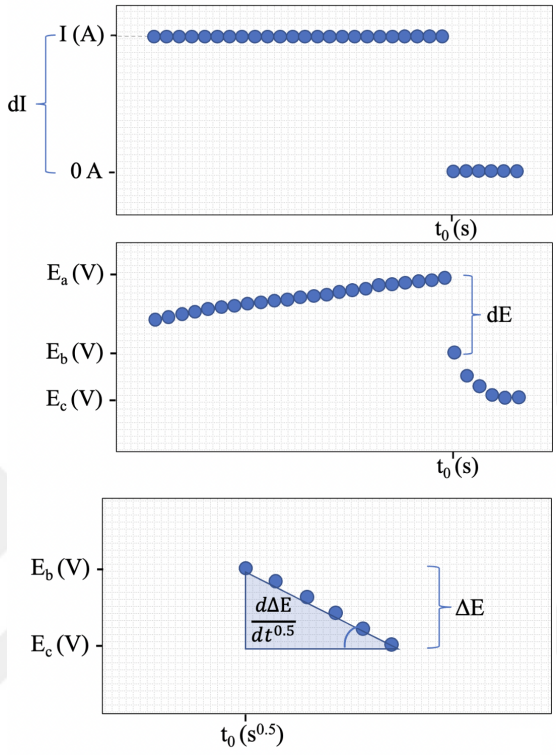

Figure 1.8: Intermittent Current Interruption technique visualized.

voltage response at t_0, which corresponds to the time just before the current is set as zero, is a sudden decline. Subsequently, a steady decay in the voltage occurs. The voltage data after t_0 is plotted against $t^{\frac{1}{2}}$. By doing linear extrapolation towards t_0 and subtracting the actual voltage at t_0, dE is obtained. Division of dE by the current applied gives R, whereas the slope found from the linearization divided by -dI, the current applied, gives k [36]. A schematic representation of the calculations are given in Figure 1.8.

1.6 Structure of the Thesis

In this work, the Introduction chapter covers the general information about Lithium ion batteries; mainly their importance, how they work, what they consist of, the reactions that take place inside them, and important parameters that are prevalently used. One of them, State of Health (SoH), is central to this work. This is a metric to observe the performance of a battery, and is the ratio of current capacity to nominal capacity. Following, the background information about the methods used for SoH detection are given, specifically experimental and data driven methods. In this work, two of the experimental methods are chosen to help reveal the chemical reasons behind why performance drop occurs and how important it is to have a robust model that can explain the occurrences beyond cell-to-cell variation.

In Chapter 2, the experimental setup and the methods followed to utilize the methods, namely EIS and ICI, are revealed. Moreover, this chapter explains in detail how these two methods were validated. Lastly, the statistics used to correlate the parameters that come from EIS and ICI with SoH is described.

In the next chapter, Results, the data acquired through the methods are visualized with plots and tables. Both the raw data and the processed data are represented.

In Chapter 4, the inferences are drawn from the said data. The comments are made for experimental details, statistics, validity, and the best correlations. To add, the practicality of application of the methods used, the possible culprit of the SoH decline(s) and how these conclusions are reached are discussed.

The Conclusions chapter includes a brief summary of the whole work. Lastly, in Appendices, the data fed to the codes, as well as codes themselves, and the material eliminated from the main body for clarity are provided.

Chapter 2

Experimental Setup and Methods

The batteries used were rechargeable ASPİLSAN INR18650. Their chemistry is nickel-rich lithium-nickel-manganese-cobalt oxide and its nominal capacity is 2800mAh with nominal voltage of 3.65 V [37]. To see the effect of SoH decline, the batteries were aged by cycling. The cycler was NEWARE 8 Channels 5V6A. Temperature was kept constant throughout all experiments at 25 °C. The charging and discharging were done according to the specification sheet of the said batteries: 0.5 C, 1400 mA was supplied until the battery's voltage reached to 4.2 V and voltage finish-off was used until the current dropped to 0.05 C, 140 mA for continuous current-continuous voltage (CC-CV) charging process. For continuous current discharging, 0.2 C was used. 2 minutes of rest was set in between the charging and discharging processes. This process was done in 50 cycle intervals. The setup is given in 2.1.

At the end of each 50 cycles, the batteries were connected to Gamry Instruments REF3000 potentiostat. A sequence of electrochemical processes were

Step name	StepTime(hh:mm:ss:ms)	Volt(V)	Curr(A)	Cap(Ah)	Eng(Wh)	-ΔV(mV)	Power(W)	R(Ω)	Stop Cur(A)
Rest	00:02:00:000								
CCCV_Chg		4.2000	1.4000						0.1400
Rest	00:02:00:000								
CC_DChg		2.5000	0.5600						
Cycle	Begin ID: 1	Times: 50							
End									

Figure 2.1: Setup screen for the cycler.

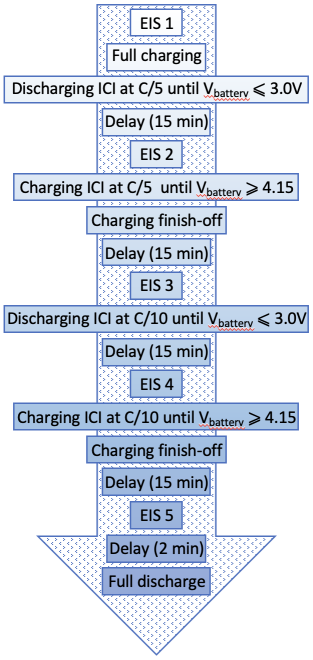

Figure 2.2: The sequence for EIS and ICI.

specifically designed to collect ICI and EIS data and is represented in Figure 2.2.

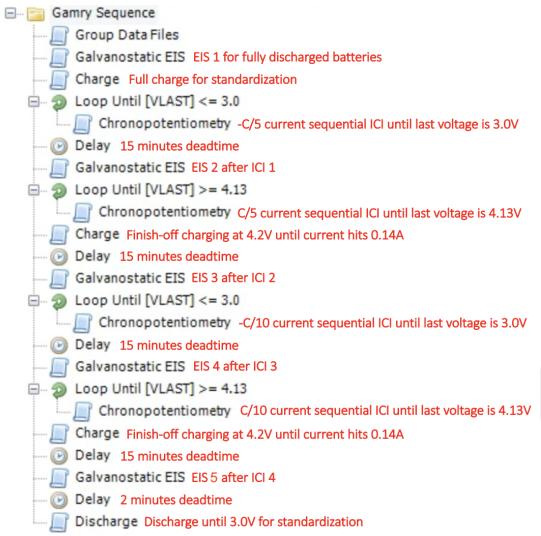

Figure 2.3: Sequence for ICI & EIS processes done with potentiostat with explanations added in red.

The battery to be tested was connected to the potentiostat. For this purpose, strip electrodes using a spot welder were connected to both terminals of each battery. In the battery's current discharged state right after cycling, the first Galvanostatic EIS, EIS 1 was done. Following, the battery was fully charged as described above. Then, ICI 1 was done with C/5 current, discharging the battery until the voltage hits 3.0 V. The negative current was applied for 10 minutes, followed by 10 second of current interruption to collect the data of the voltage decay. Before each EIS, except the first one, a delay of 15 minutes was placed to gather the data from the stabilized battery. Going forward, the second EIS (EIS 2) was done and the battery was charged with C/5 ICI procedure with the same parameters mentioned above as ICI 2. Next, all of the same experiments are repeated with C/10 rate for ICI, discharging with C/10 as ICI 3, followed by EIS 4 and charging with C/10 as ICI 4, followed by EIS 5. Both ICI's with positive current is followed by a charge function. This is done to mimic the standard CC-CV charging process. The voltage limit for positive ICI is 4.18 V,

whereas for the negative ICI it is 3.0 V. The reason why the limits are taken this way is to not exceed the working voltages of the batteries. In time, as the capacity value decreases, the set point voltage was gradually lowered until 4.14 V. The reasoning is the battery's voltage would hit 4.20 or higher before the last voltage of interruption would be less than the set point. In those cases, the current would be interrupted before 10 minutes and the battery could not be charged adequately to give an end point high enough to stop the ICI processing. By lowering the set point, the battery gets charged properly and voltage of 4.20 is never observed; consequently, battery is protected. After the end of all EIS and ICI, 2 minutes of deadtime was set to stabilize the battery and the battery is discharged with 0.2 C rate for maximum of 4 hours. All of the EIS experiments have the same parameters, frequency ranging from 5000 Hz to 0.1 Hz, AC Current of 0.01 A(rms) and DC Current of 0 A.

Firstly, this sequence was optimized with a set of 3 batteries before switching to then, to fill all slots of the cycler, as well as to make sure of reproducibility and robustness of the methods, another set of 5 batteries were employed. As the previous sets were being used for ICI, it was found out that currents higher than C/5 did not produce valid data. During interruption, it was observed that the linearization of voltage against square root of time did not produce a curve of first order and raw data of voltage versus time gave an exponential curve. To add, the deadtimes were added before EIS acquisition as the data showed significant instability throughout the sets. Moreover, during the transportation between the cycler and the potentiostats, some batteries' electrodes were fused and thus, batteries were deemed unusable. Consequently, the electrodes were prepared much shorter. Afterwards, a fresh set of 8 batteries were employed. All

the findings shared in the document belong to the set of 8 batteries prepared and tested with optimized parameters.

After the data acquisition sequence mentioned above, data extraction and processing was done. All data was processed using Python. In the case of EIS, raw data were edited and put in formats of .xlsx and .csv to be fed to codes. Nyquist data after acquisition were processed by cycle number. Each .csv was named with the cycle number, 8 batteries were listed and for each, a real (Zreal) and imaginary (Zimag) parts were taken from raw .DTA file, named as r (X-axis) or i (Y-axis) respectively in .csv data. Then, all data from the same cycle number were placed in one .csv file to be processed by a Python code, for each EIS. One part of .csv file was provided in Appendix A.1, and the code to plot Nyquist plot for EIS 1 is in Appendix B.1. In the case of ICI, data were extracted, processed and represented in plots and values through Python codes. Both R and k were calculated using Python where k comes from the slope of E vs $\sqrt{t}$ and R comes from the intercept of the linearized E versus square root of t. Entirety of the code for processing averaged results and is provided in Appendix B.3.

Throughout all the dataset, instrumental error is deemed small and neglected since the fitting errors and the cell-to-cell variation will dominate. Hence, no error was calculated for EIS or cycling data. In the case of ICI, errors vary widely; R values' percent fitting errors range from 0.02% to 11600%, whereas k values' errors are between 0.14%-6% as produced by the fit algorithm. For k, the slope of the line of V vs. $\sqrt{t}$ is needed; and for R, intercept of the said line is needed. These are obtained from the fit. The error for slope, given as slope_stderr, was calculated with Equation 2.2 and error for intercept, intercept_stderr, was calculated by Equation 2.3.

$$r = \frac{\frac{1}{N}\sum_{i=1}^{N}\left((\sqrt{t_i} - \frac{1}{N}\sum_{j=1}^{N}\sqrt{t_j}) * (V_i - \frac{1}{N}\sum_{k=1}^{N}V_k)\right)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\sqrt{t_i} - \frac{\sum_n^N \sqrt{t_n}}{N})^2 * \frac{1}{N}\sum_{i=1}^{N}(V_i - \frac{\sum_n^N V_n}{N})^2}} \tag{2.1}$$

$$slope\_stderr = \sqrt{1 - r^2} * \frac{\sum_{i=1}^{N}\frac{1}{N}(V_i - \frac{\sum_n^N V_n}{N})^2}{\sum_{i=1}^{N}\frac{1}{N}(\sqrt{t_i} - \frac{\sum_n^N \sqrt{t_n}}{N})^2 * (N-1)} \tag{2.2}$$

$$intercept\_stderr = slope\_stderr * \sqrt{\sum_{i=1}^{N}\frac{1}{N}(\sqrt{t_i} - \frac{\sum_n^N \sqrt{t_n}}{N})^2 + (\frac{\sum_n^N \sqrt{t_n}}{N})^2} \tag{2.3}$$

Equations given above hold true, given a line with form of V=m($\sqrt{t}$)+n with total of N points and $\sqrt{t_n}$ and V_n being the nth data point. Following, the actual errors for k (Equation 2.4) and R (Equation 2.5) are found by;

$$error_k = \frac{slope\_stderr}{dI} \tag{2.4}$$

$$error_R = \frac{intercept\_stderr}{dI} \tag{2.5}$$

More statistics were utilized for the verification of any possible correlation between data and capacities. For this purpose, distance correlation (dCor) was employed. Though Pearson Coefficient Analysis is a popular method for statistics, however; any relationship beyond linear is also sought after in this manuscript. Hence, dCor was utilized. dCor is implemented as following: two vectors with

same length of n, X and Y, is processed. For this work, one vector is the capacity values with one dimension; whereas the other is the test values with one dimension which are grouped. As the comparison is done symmetrically, the naming of X and Y is interchangeable for these values. Using X and Y, A and B matrices are defined as Equation 2.6 and 2.7 respectively:

$$A_{kl} = a_{kl} - \frac{1}{n}\sum_{l=1}^{n} a_{kl} - \frac{1}{n}\sum_{k=1}^{n} a_{kl} + \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl} \tag{2.6}$$

$$B_{kl} = b_{ij} - \frac{1}{N}\sum_{l=1}^{N} b_{kl} - \frac{1}{N}\sum_{k=1}^{N} b_{kl} + \frac{1}{N^2}\sum_{k,l=1}^{N} b_{kl} \tag{2.7}$$

where a_{kl} and b_{kl} is defined as

$$a_{kl} = \sqrt{\sum_{k,l=1}^{n} (X_k - X_l)^2} \tag{2.8}$$

$$b_{kl} = \sqrt{\sum_{k,l=1}^{n} (Y_k - Y_l)^2} \tag{2.9}$$

Obtaining A_{kl} and B_{kl}, one can calculate sample distance covariance, the unnormalized measure of dependency $\mathcal{V}$, using the Equation 2.10:

$$\mathcal{V}_n^2(x, y) = \frac{1}{n^2}\sum_{i,j=1}^{n} A_{i,j} B_{i,j} \tag{2.10}$$

The squared distance coefficient which is the normalized and emphasized coefficient that showcases the correlation between the two dataset, denoted as $\mathcal{R}^2$, is

calculated with Equation 2.11. It should be noted that $\mathcal{R}^2$, after the normalization, is a value between 0-1; representing a better connection with values closer to 1.

$$\mathcal{R}_n^2(x,y) = \begin{cases} \frac{\mathcal{V}_n^2(x,y)}{\sqrt{\mathcal{V}_n^2(x,x)\mathcal{V}_n^2(y,y)}} & \text{if } \mathcal{V}_n^2(x,x)\mathcal{V}_n^2(y,y) > 0, \\ 0 & \text{if } \mathcal{V}_n^2(x,x)\mathcal{V}_n^2(y,y) = 0. \end{cases} \tag{2.11}$$

It is crucial for the data acquired to be inspected for validity, especially for the works that follow experimental methods. For the systems investigated by EIS, this inspection is done with by Kramers-Kronig (KK) test. For an EIS to be claimed valid, the system under investigation has to have 3 important characteristics: linearity, stability and causality [38]. Linearity is when the system's response to a sinusoidal excitation is followed by another sinusoidal excitation with the same frequency. Stability represents a system where changes do not occur in time or after halting the excitation signal. Causality is achieved if the perturbation of the system happens only because of the signal sent. KK analysis ensures all 3 features by constructing Modulus data from Phase Angle data and vice versa. However, for KK to be implemented, data has to be continuous and have data in frequencies 0 to infinity. As this is not possible, Boukamp approach can be used. All of the data gets expressed in serial circuits of parallelly constructed resistance and capacitance (RC). In Boukamp's method, R·C values are used to calculate either Modulus' resistance values from Phase Angle data and vice versa [39]. The Python code [40] to implement this approach can be found in Appendix B.6.

The ultimate aim for this work is to use the selective parameters to predict the SoH. Hence, a way to achieve this goal has to be also presented. For this purpose,

a type of binning was employed. After each individual errors were calculated for k and R, their averages were taken for each test (σ). Onto the k or R versus capacity plots, using this value, boxes that have the height of the errors were drawn. The boxes' vertical span was constructed with two data that fall in the height of the box, but are the furthest away from each other in capacity values. This was done for both σ and 3σ. This methodology not only showcases the accuracy to predict the SoH, but also reveals the standard deviations for the k and R parameters under specific conditions.

Chapter 3

Results

This chapter is exclusively dedicated to all visual representations of the data and its interpretations, including correlations and statistics. The observations derived from the findings shared here are given in Discussion chapter.

As explained in the previous section, raw data for both EIS and ICI were processed, and correlated with capacity values using visual means and distance correlations. The capacity values versus cycle numbers are given in Figure 3.1.

3.1 EIS Data & Correlation Results

Full Nyquist plots for 5 EIS, 1150 cycles and all 8 batteries are given in Figure 3.2. Though Nyquist and full Bode plots are used for the representation of EIS, as explained in the Experimental Methods section, 1 frequency values were processed to obtain the correlations with capacity values. Some of the plots for

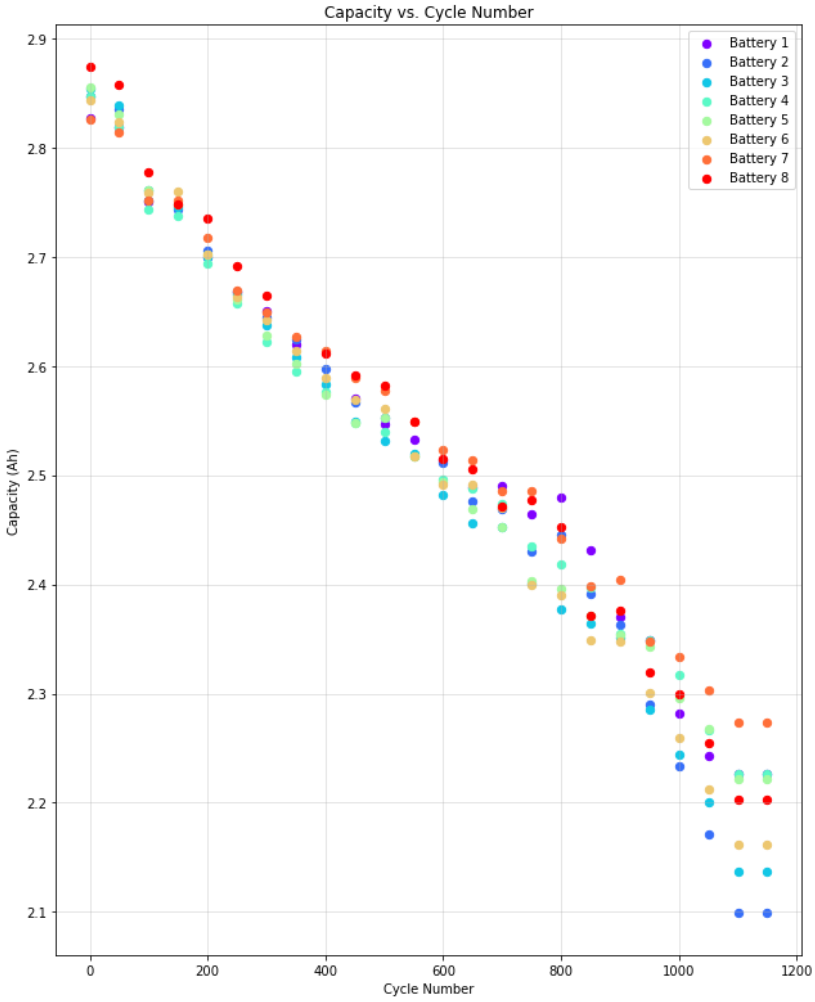

Figure 3.1: Capacities for 8 Batteries vs. Cycle Numbers.

Modulus/Phase Angle vs. Cycle plots are given in 3.3, used to observe if any trend is visible.

To be used for dCor, Modulus vs. Capacity and Phase Angle vs. Capacity plots are constructed. Examples of the former are given in Appendix A.1 and plots of the latter are given in Appendix A.2. Afterwards, distance correlation was implemented to the data. Representative numerical findings can be found in Table 3.1 and full data is given in Appendix A.3. The highest coefficients, hence the best correlations belong to EIS 5 at 0.100 Hz for Modulus and 0.397 Hz for Phase Angle. The plots of the aforementioned data is given in Figure 3.4 for visual representation. All of the results can be obtained through the raw data provided in Chapter 6 and running the corresponding code, given in Appendix B.2.

To add, all the dCor results from the data described above were plotted against frequencies. This was done to reveal any high correlation at a specific frequency, to better understand the mechanism for SoH degradation. The results are given in Figure 3.5(a) for Modulus dCor values and (b) for Phase Angle dCor values.

The residuals of KK analysis done on every EIS are provided in the Appendix A.4. In the residuals as well as the graphs plotted from the data and the code, it can be observed that the difference between the constructed data from the calculations and the experimental data are relatively higher in the first and last cycles.

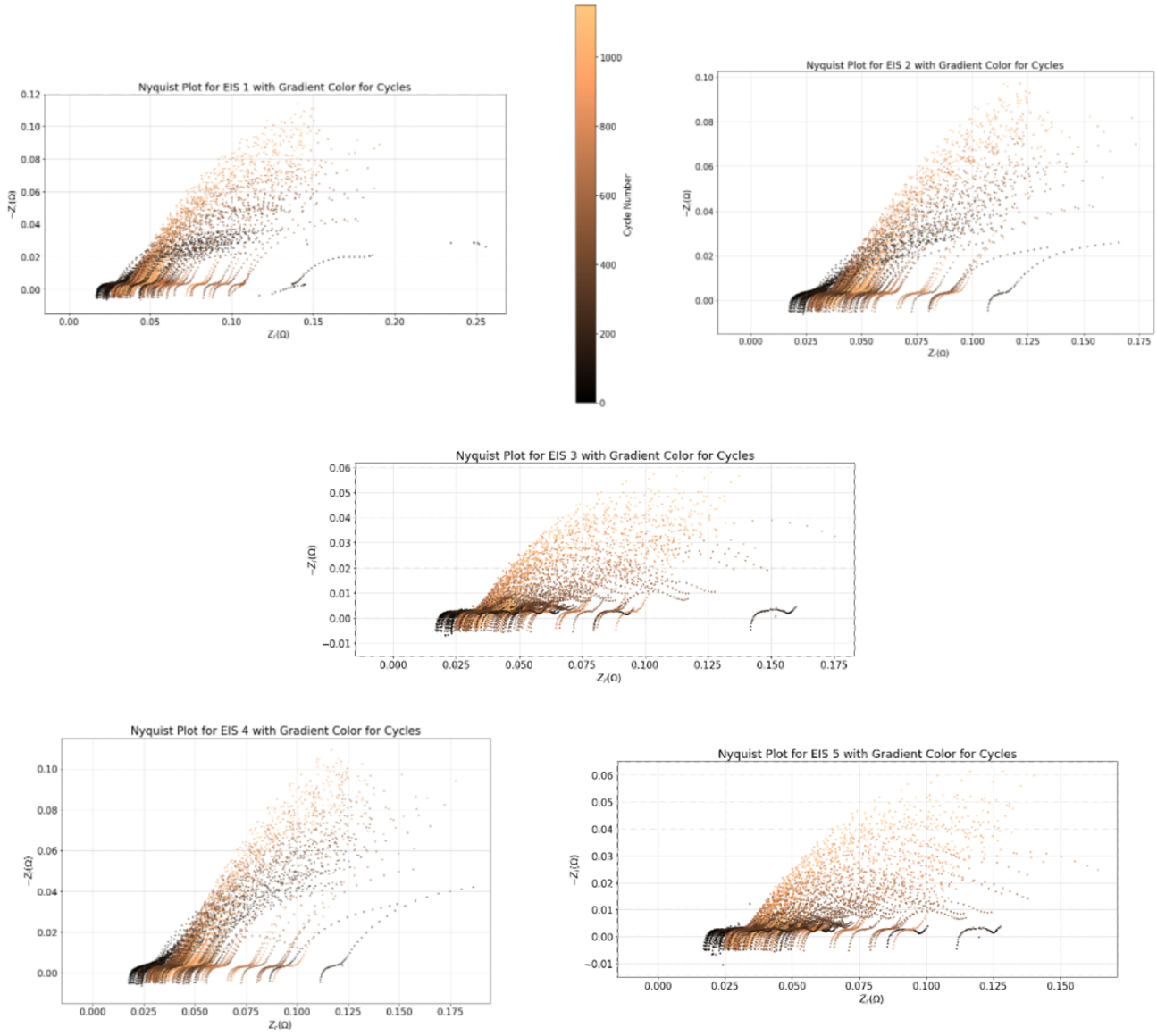

Figure 3.2: All of the Nyquist plots for EIS 1-5, color coded by cycle number.

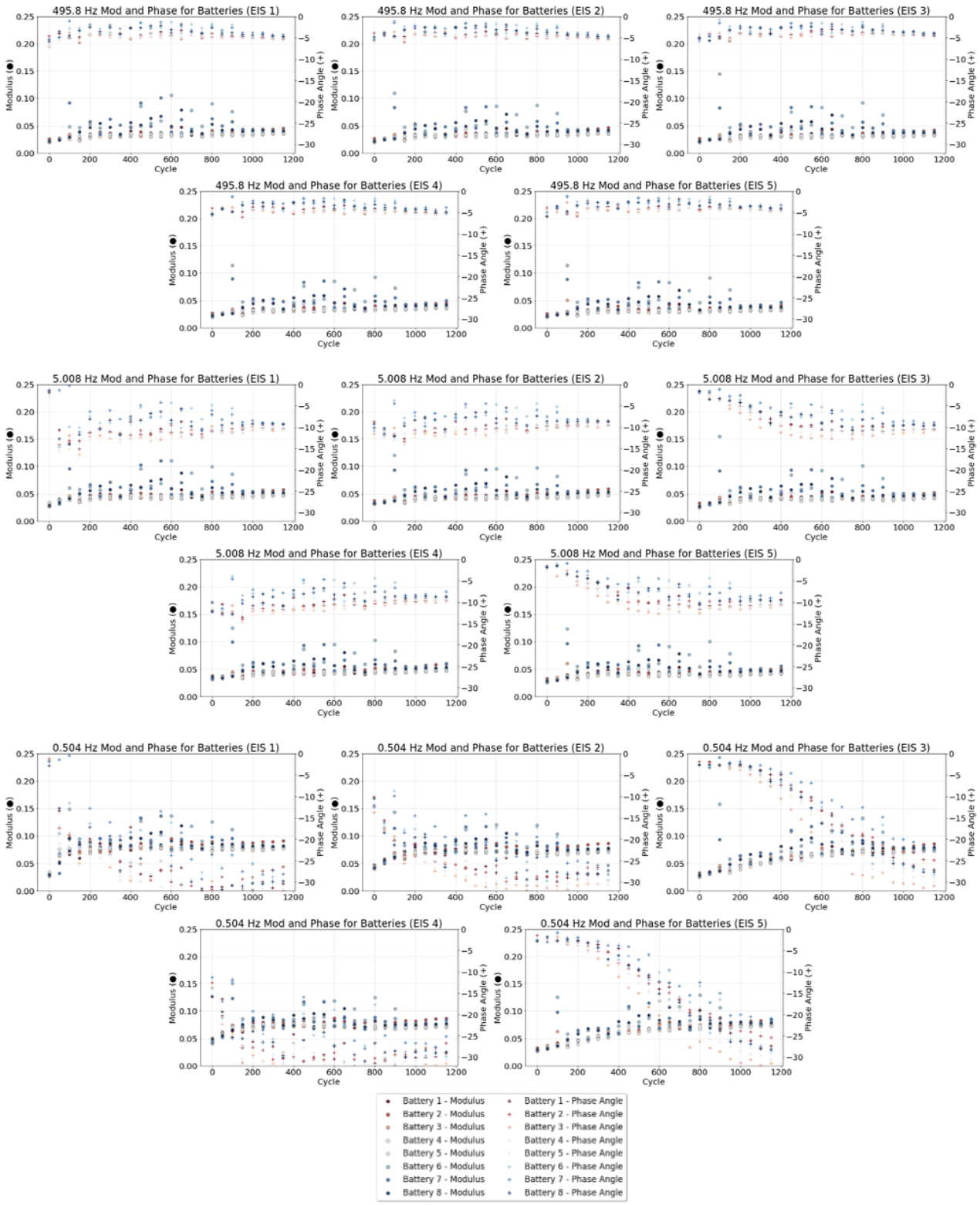

Figure 3.3: Bode plot representations of data for specific frequencies vs. cycle number.

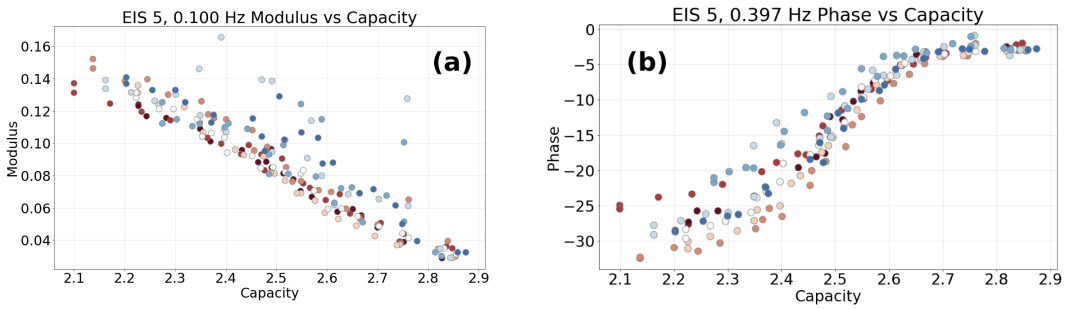

Figure 3.4: Best EIS and Capacity correlations found with dCor. Results for Modulus vs. Capacities (a) and Phase Angle vs Capacities (b)

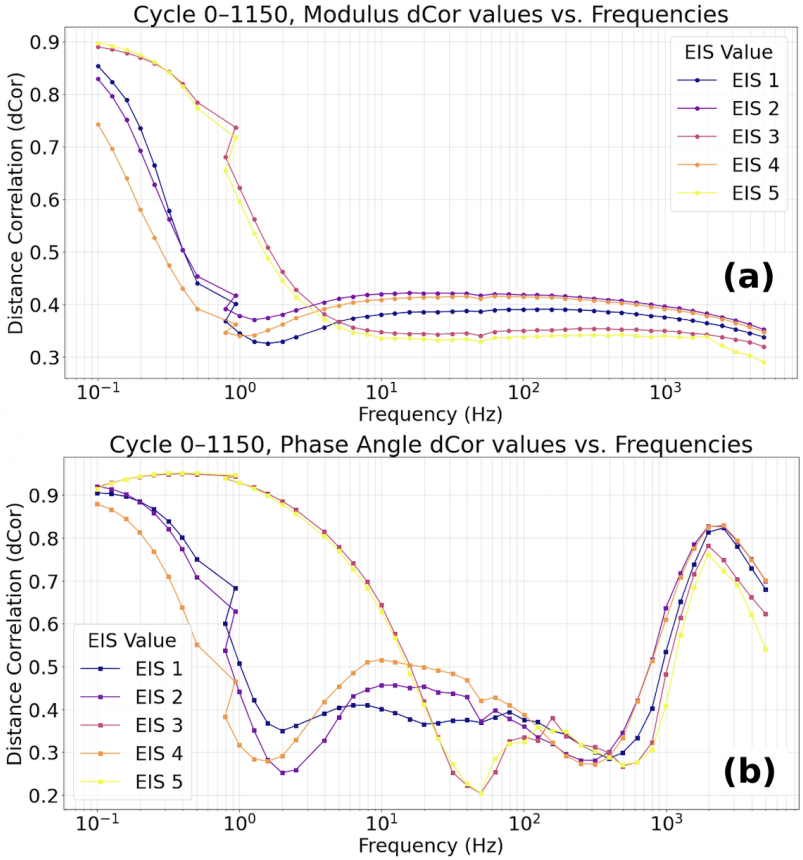

Figure 3.5: dCor values for Modulus (a) and Phase Angle (b) against frequencies.

Table 3.1: Some of the EIS Modulus and Phase Angle Data versus Capacity Value Distance Coefficient results.

EIS #	Frequency	Mod. Distance Coeff.	Phz. Distance Coeff.
1	495.8	0.384	0.296
	5.008	0.367	0.408
	0.504	0.440	0.748
2	495.8	0.407	0.349
	5.008	0.410	0.381
	0.504	0.452	0.710
3	495.8	0.352	0.266
	5.008	0.367	0.767
	0.504	0.784	0.943
4	495.8	0.403	0.334
	5.008	0.398	0.454
	0.504	0.391	0.551
5	495.8	0.342	0.271
	5.008	0.356	0.769
	0.504	0.772	0.951
		Best coefficient results	
5	0.100	0.898	—
5	0.397	—	0.952

3.2 ICI Data & Correlation Results

All of the data for both R and k, acquired from data with qualifying length and battery voltage intervals, are presented in Figure 3.6 for k and Figure 3.7.

Clearly seen in the figures for all ICI values, not all plots have the same amount of data. For a better visualization, averages were taken as mentioned in Experimental Methods section. Averaged k and R values are given in Figures 3.8 and 3.9 respectively. To add, distance correlation was sought between the averaged values and their corresponding capacity values. The results are given in Table

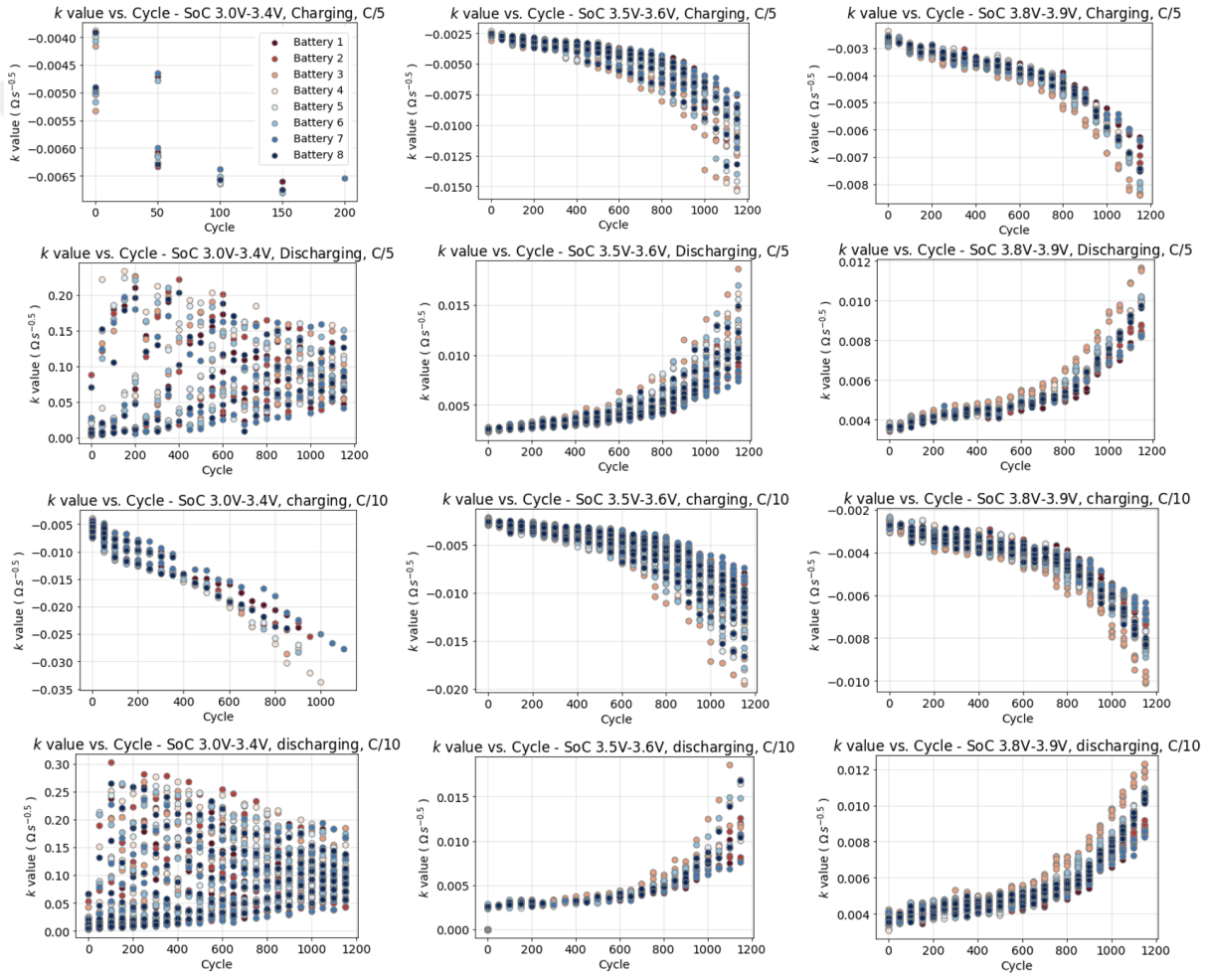

Figure 3.6: All of the k Values; classified by voltage, current sign and current rate. Each of these plots includes all of the dataset for R.

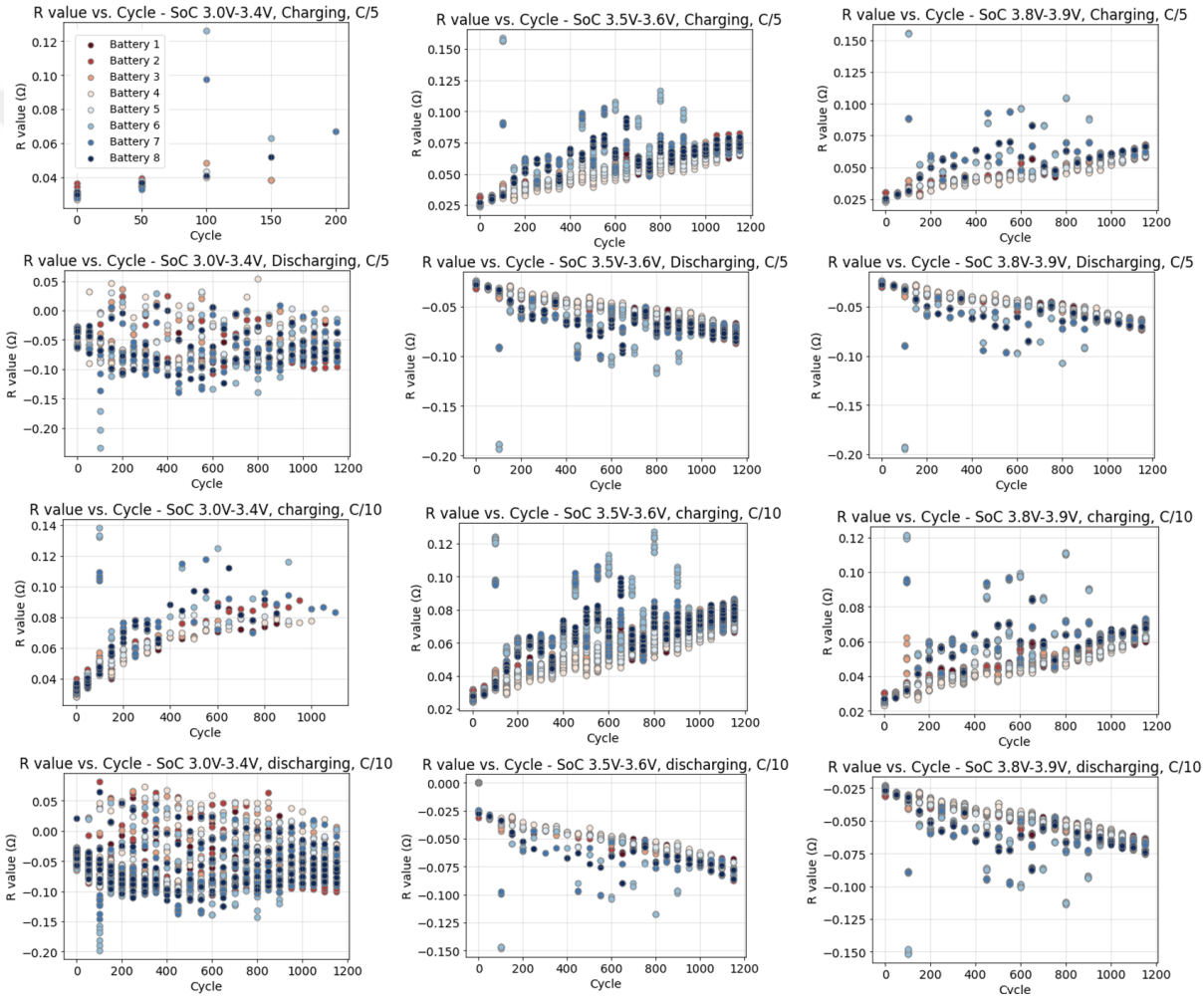

Figure 3.7: All of the R Values; classified by voltage, current sign and current rate. Each of these plots includes all of the dataset for R.

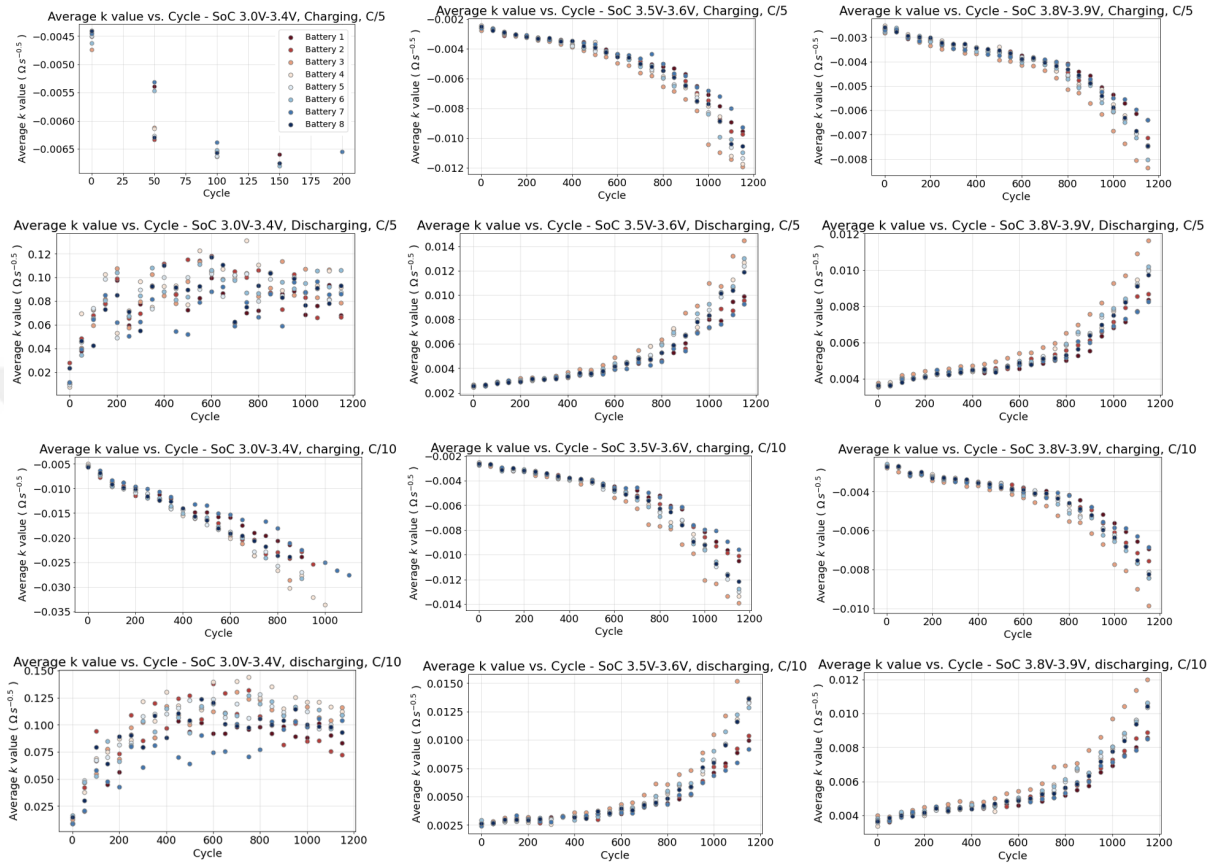

Figure 3.8: All averaged k values vs. cycle, voltage intervals and positive/negative charging rates.

3.2.

As the ultimate goal is set to have a reliable and quick testing for detecting SoH, taking average is counter-intuitive. EIS takes 15 seconds and a single ICI test has a duration of 10 minutes and 10 seconds. These durations are short enough to qualify for the goal, hence single values have been checked. Figures 3.10, 3.11 and 3.12 includes the plots for the first, mid and last data for the datasets of k respectively; whereas Figures 3.13, 3.14 and 3.15 have the data for k & R values vs. cycle. It is important to note that for some datasets, there is only 1 data to represent and thus, those data in plots stay the same. However,

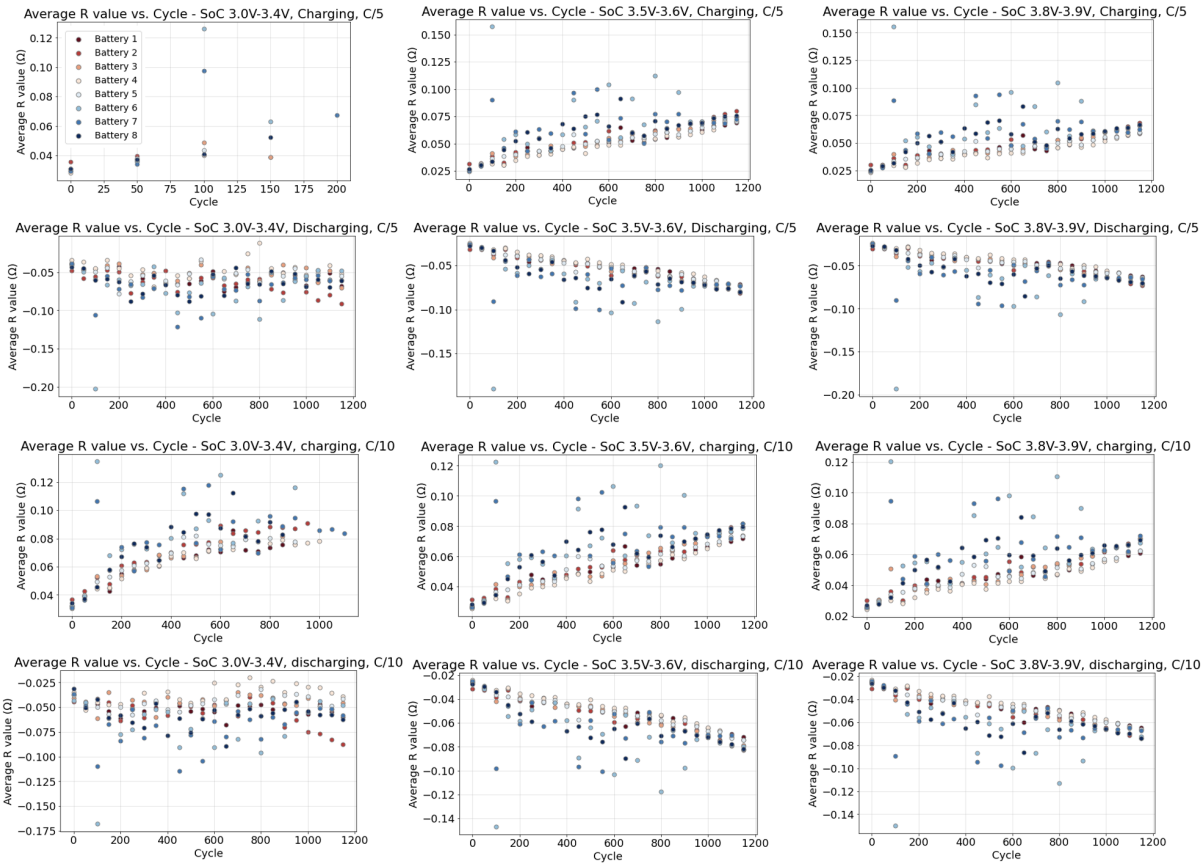

Figure 3.9: All averaged R values vs. cycle, voltage intervals and positive/negative charging rates.

Table 3.2: Averaged k&R Distance Coefficient results.

k/R	**Voltage**	**C rate**	**Current(+)/(-)**	**Dist. Coeff.**
k	high	C/5	+	0.957
			-	0.932
		C/10	+	0.952
			-	0.929
	mid	C/5	+	0.959
			-	0.939
		C/10	+	0.950
			-	0.910
	low	C/5	+	0.772
			-	0.567
		C/10	+	**0.972**
			-	0.694
R	high	C/5	+	0.660
			-	0.696
		C/10	+	0.669
			-	0.715
	mid	C/5	+	0.740
			-	0.757
		C/10	+	0.759
			-	0.746
	low	C/5	+	0.688
			-	0.243
		C/10	+	**0.782**
			-	0.162

they are kept for consistent display.

Distance correlation results were also obtained for the data presented for the single values of R and k. Findings can be seen in Table 3.3 and 3.4 for k and R respectively.

The results for binning, which visualizes the standard deviation and the certainty in predictions done by k or R, are given in Figure 3.16 and 3.17 for σ and 3.18 and 3.19 for 3σ respectively. To add, the average spread on the capacities

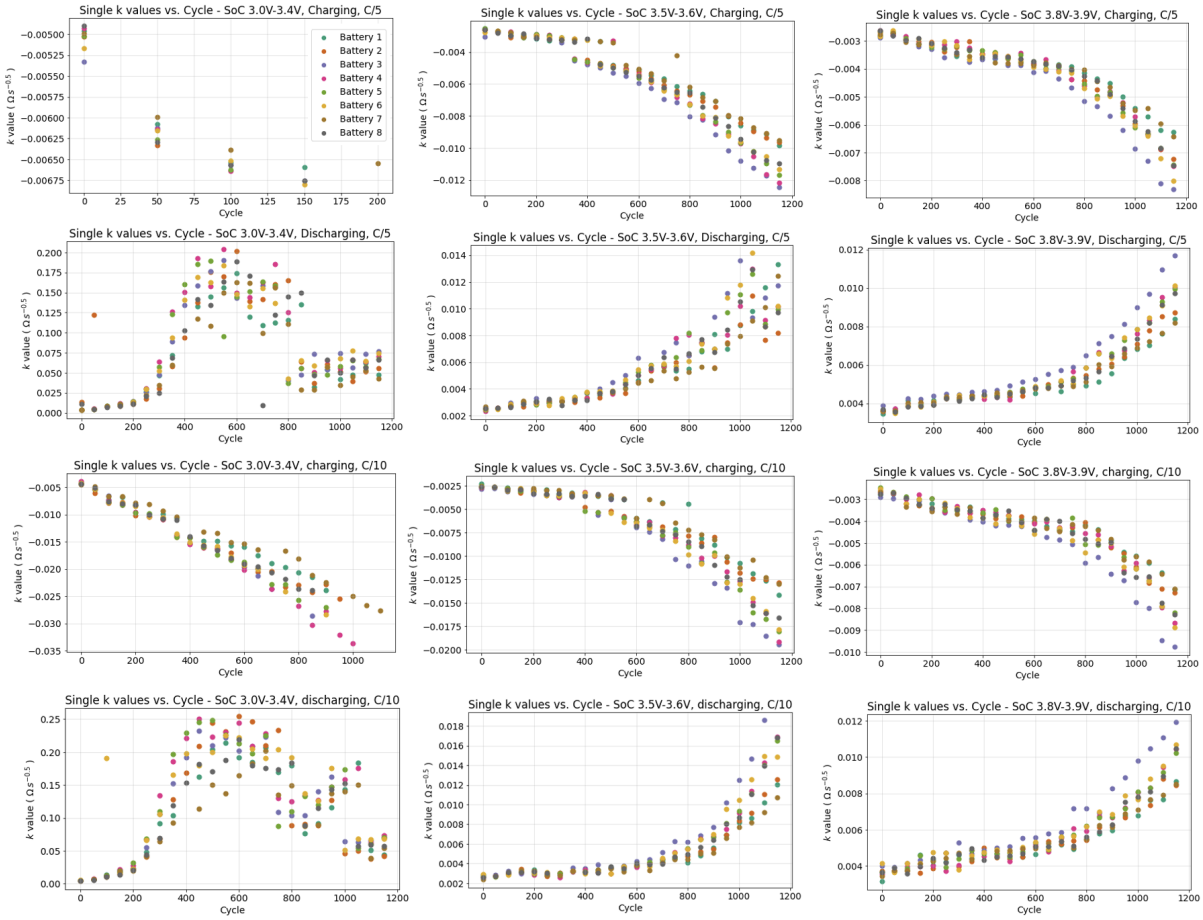

Figure 3.10: First data for each dataset, k versus capacity.

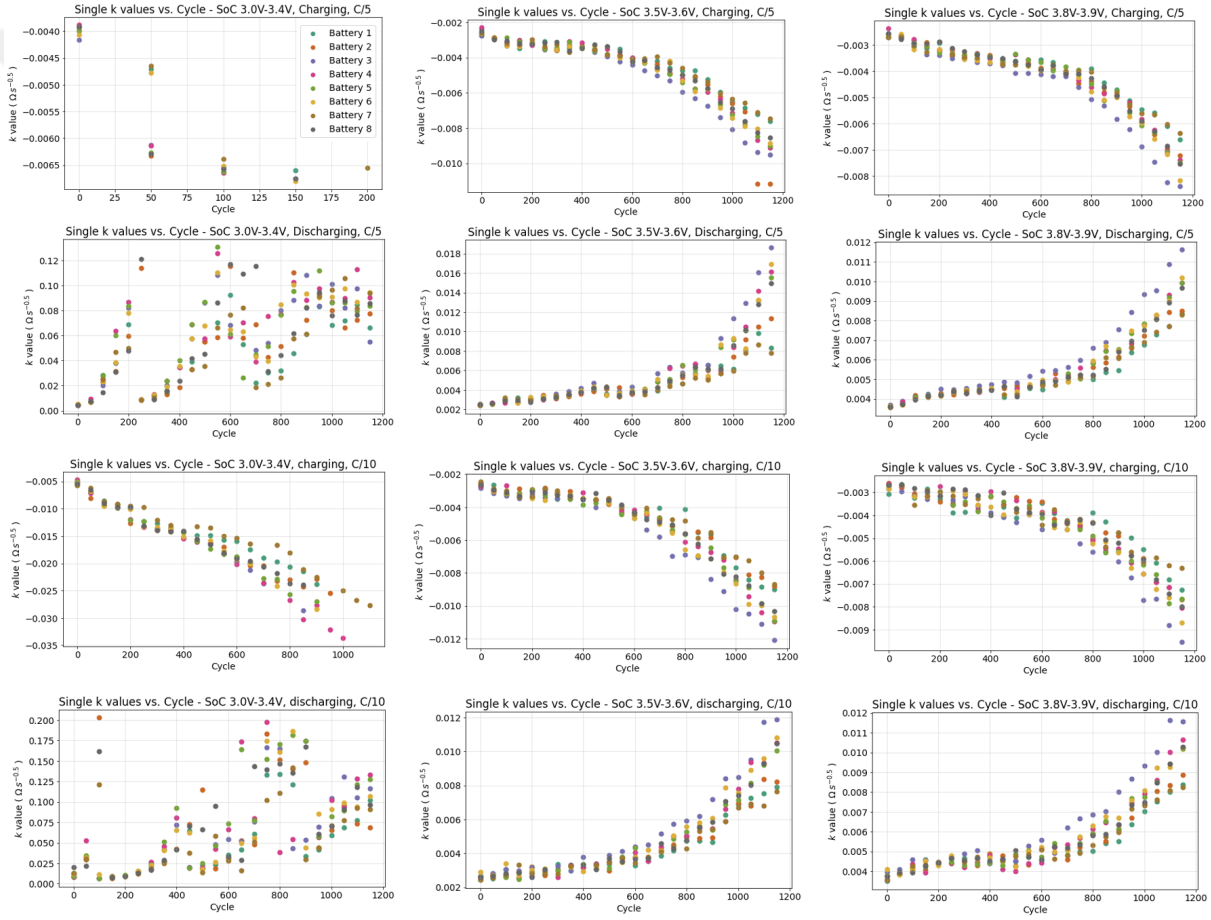

Figure 3.11: Mid data for each dataset, k versus capacity.

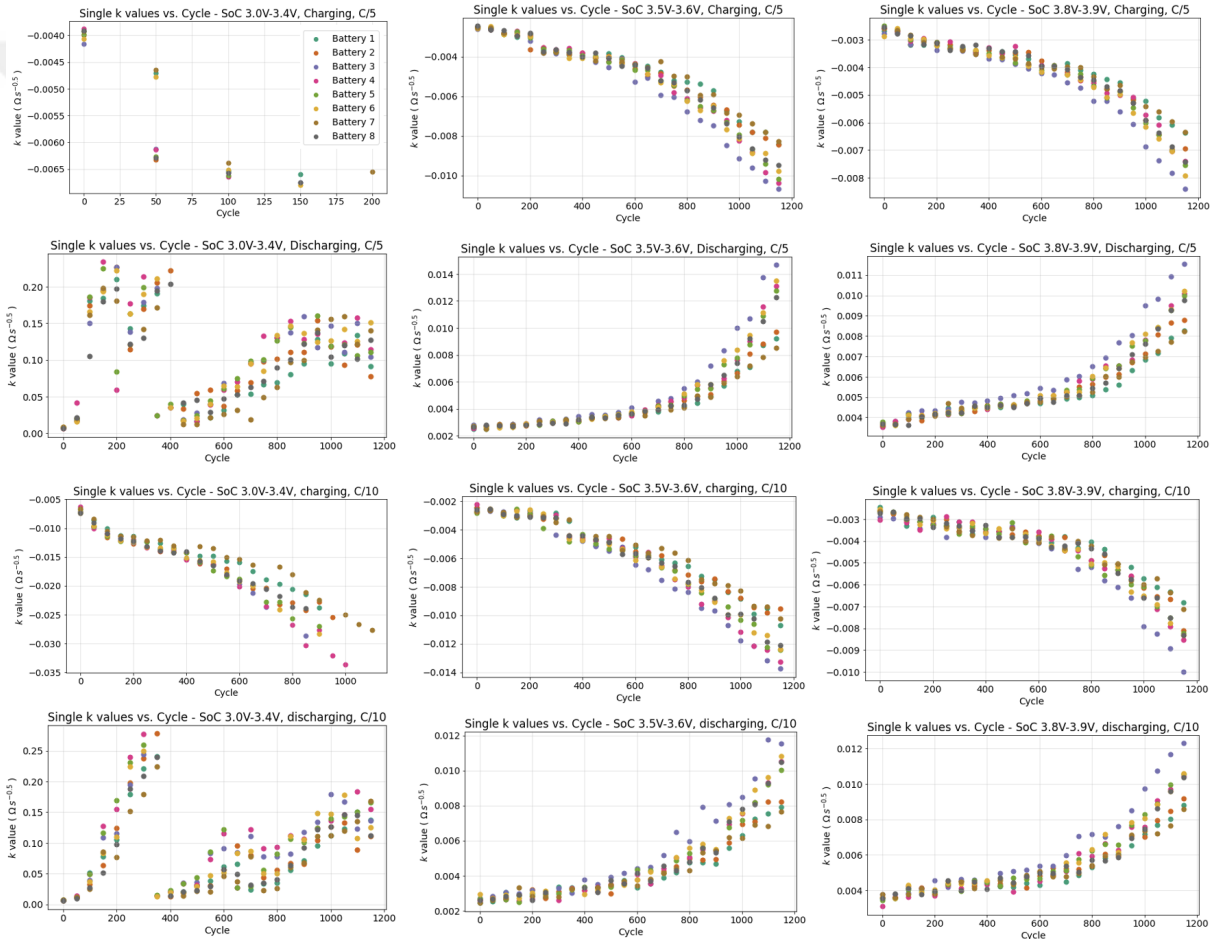

Figure 3.12: Last data for each dataset, k versus capacity.

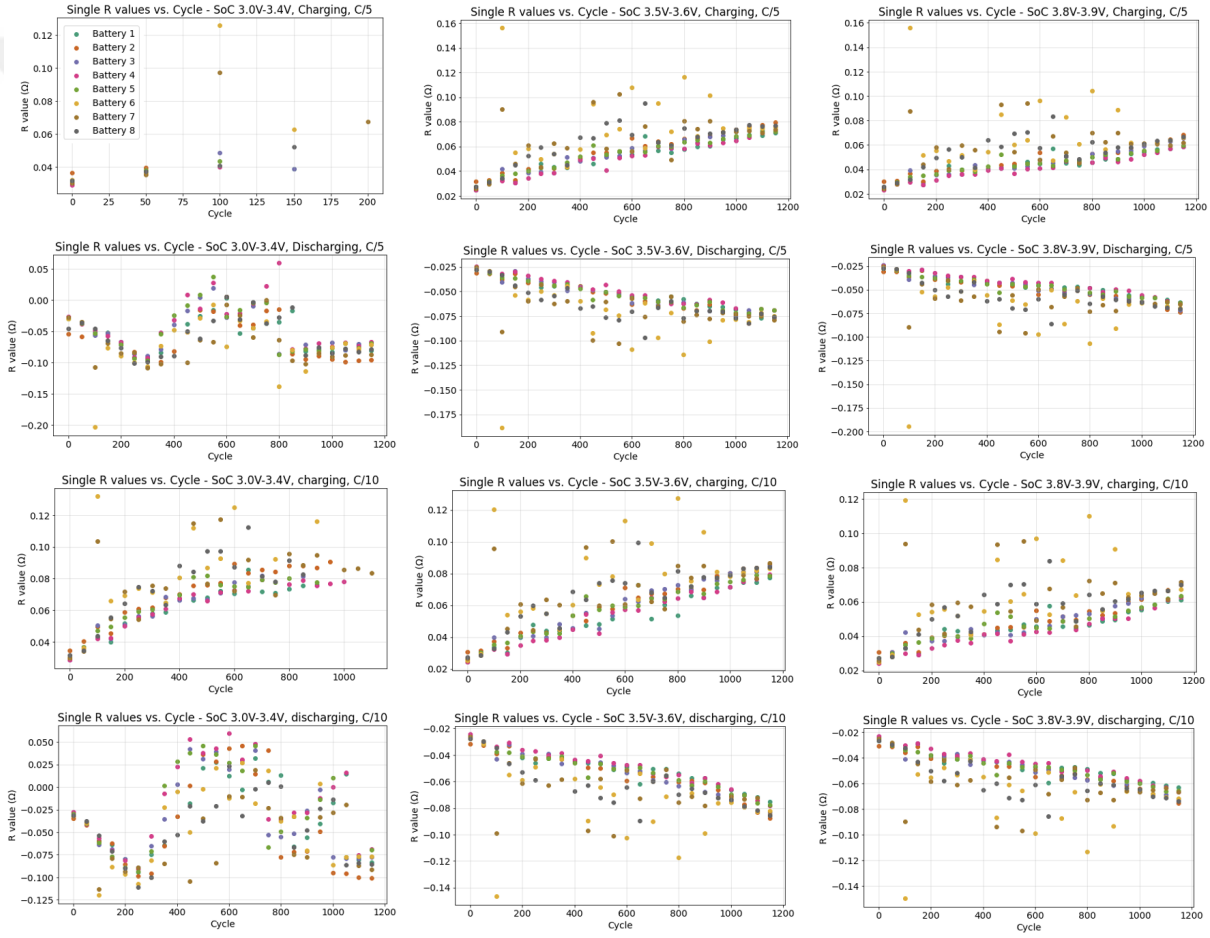

Figure 3.13: First data for each dataset, R versus capacity.

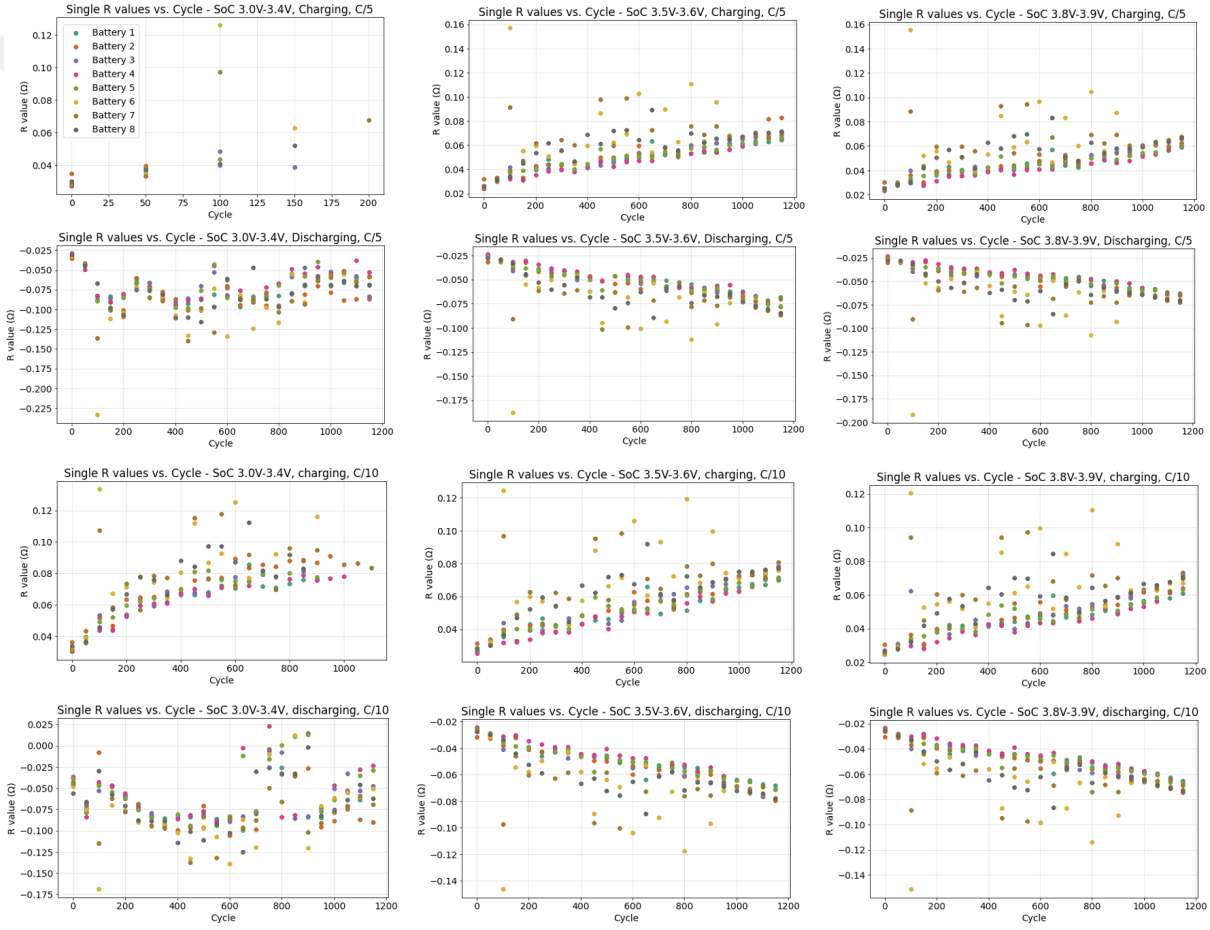

Figure 3.14: Mid data for each dataset, R versus capacity.

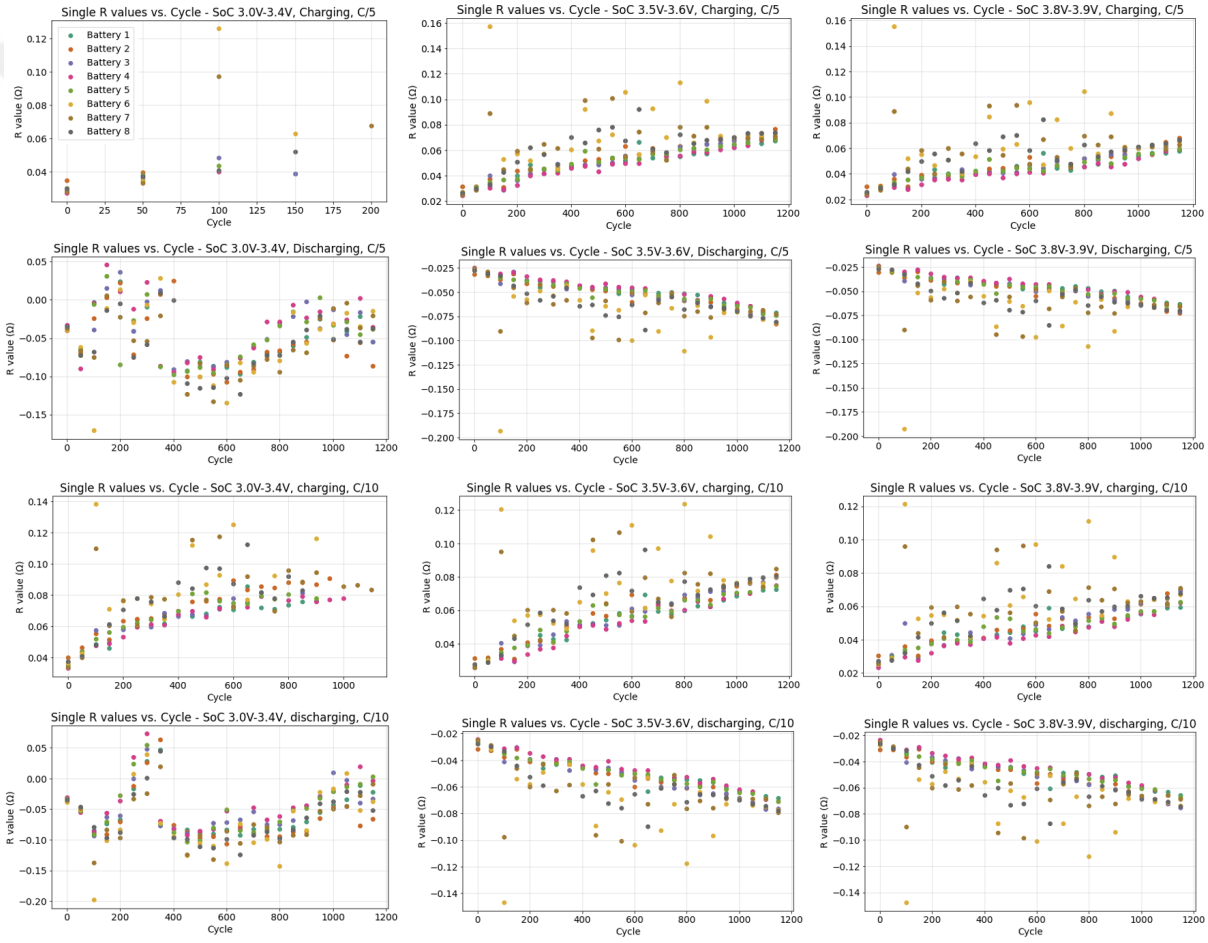

Figure 3.15: Last data for each dataset, R versus capacity.

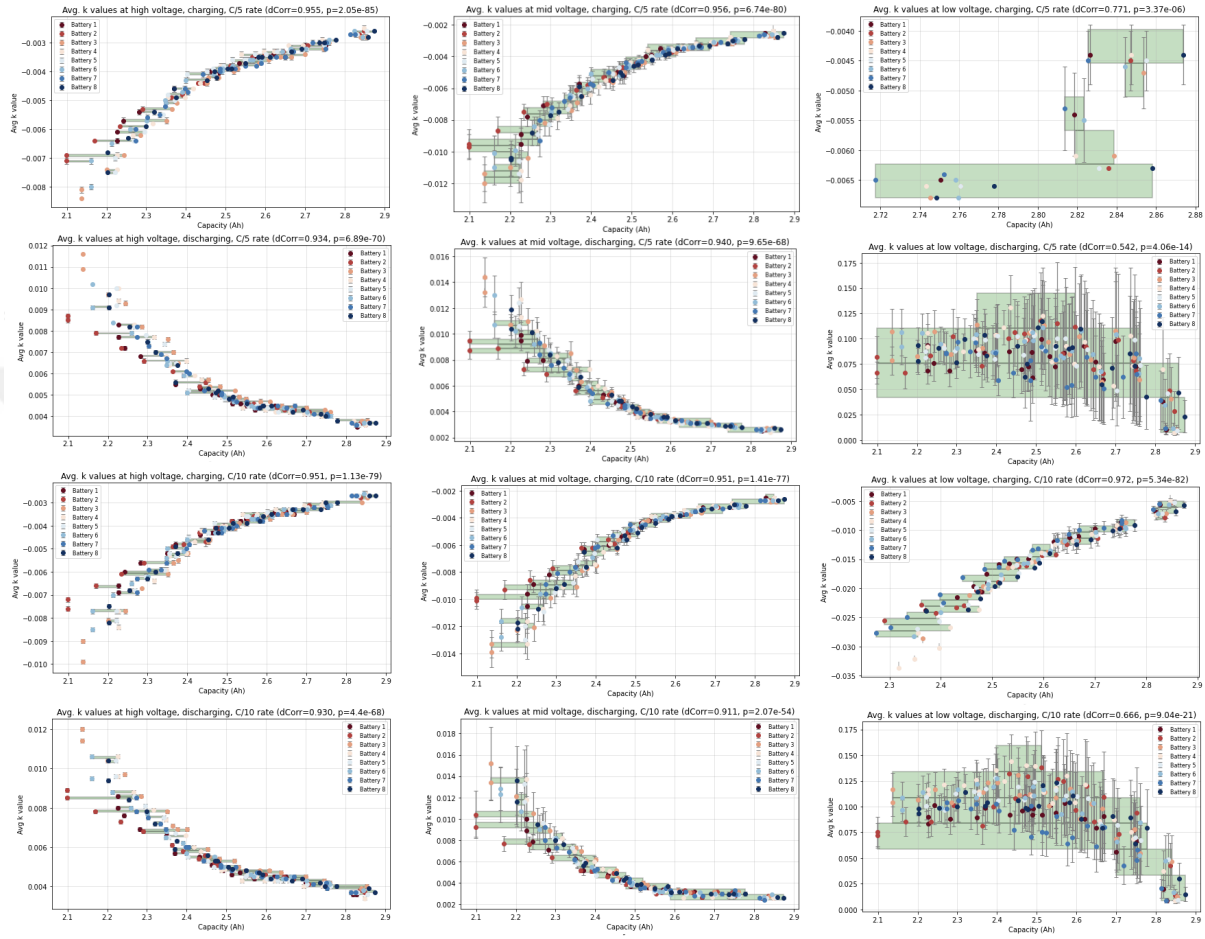

Figure 3.16: Results for bins with height of σ for k.

taken from the bins' width, are given in Table 3.5.

3.3 Other Results

Because the R taken from ICI and overall resistance obtainable from EIS correspond to the same parameter, they were compared to each other to see if the values were the same. The R's from EIS (R_1) are extracted from the first Z_{real} values. R values from ICI (R_2) are taken from the ones that are closest to corresponding EIS in time, e.g the first R data from ICI 1 for EIS 1, the last R from

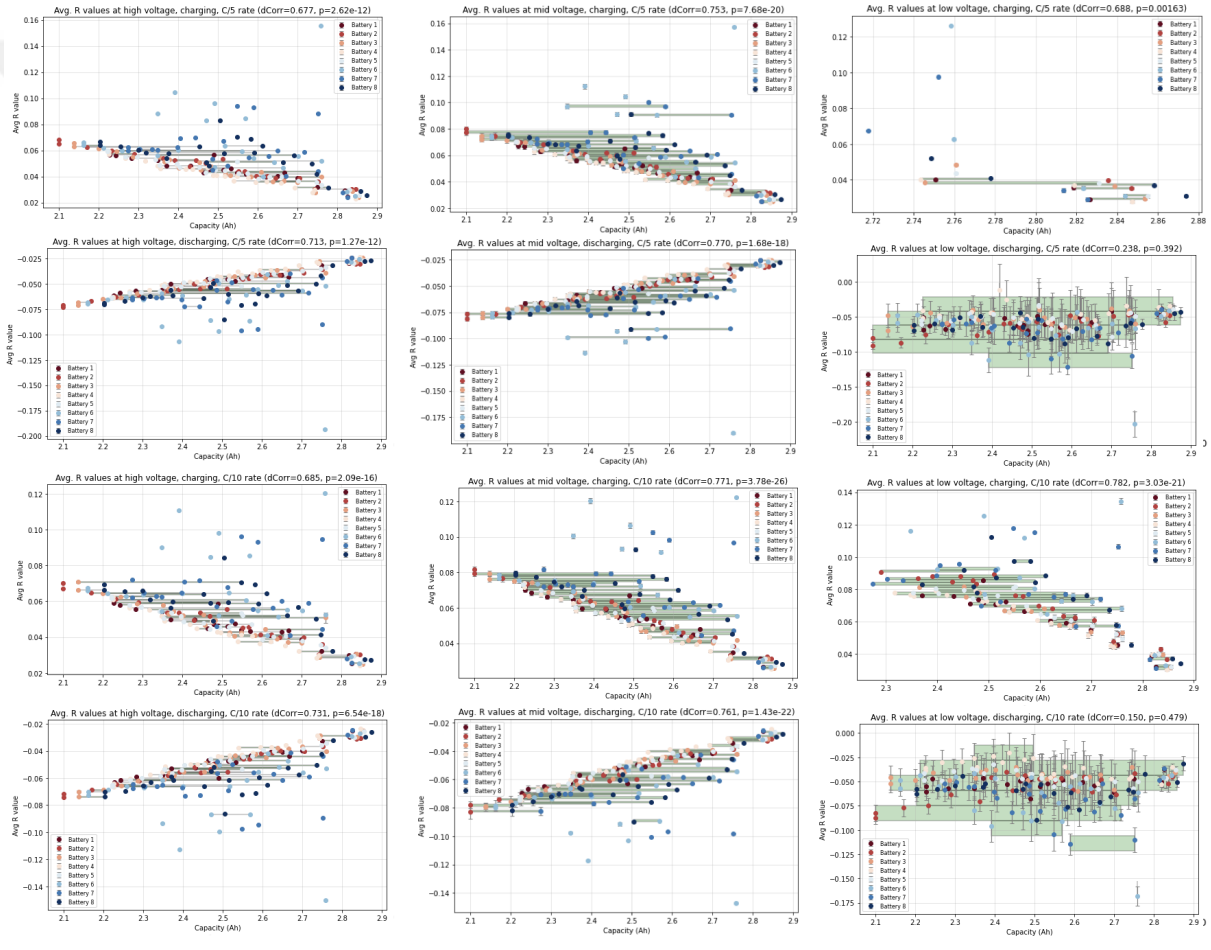

Figure 3.17: Results for bins with height of σ for R.

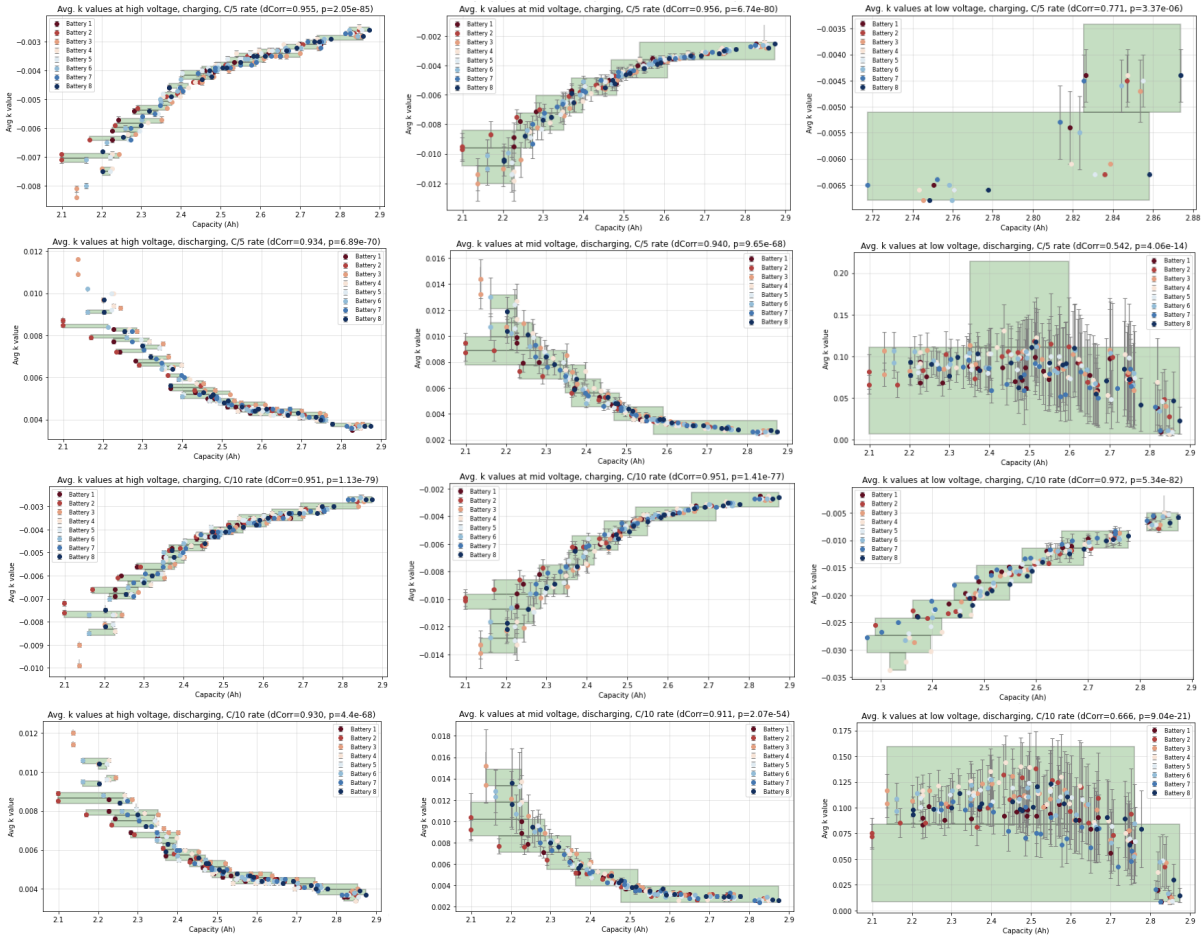

Figure 3.18: Results for bins with height of 3σ for k.

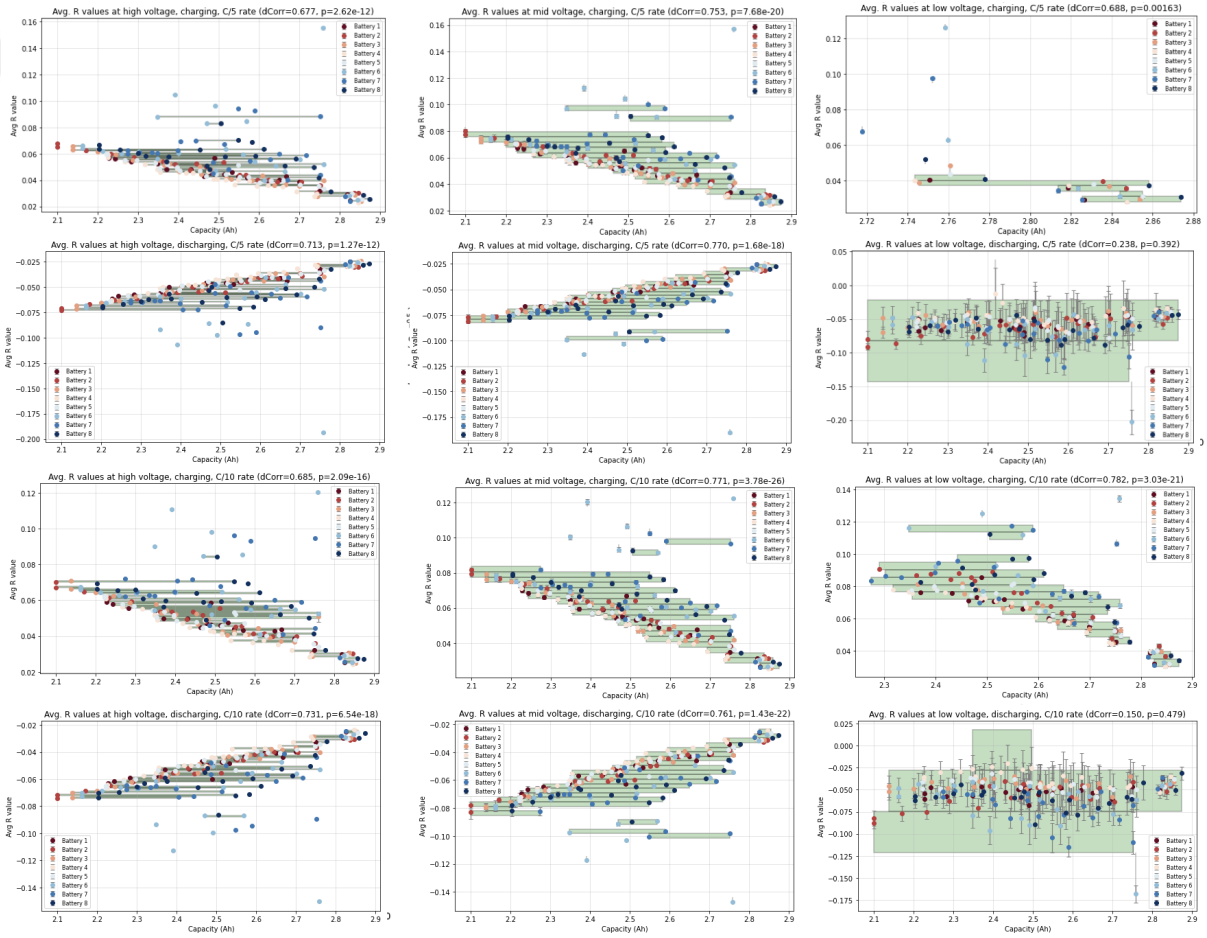

Figure 3.19: Results for bins with height of 3σ for R.

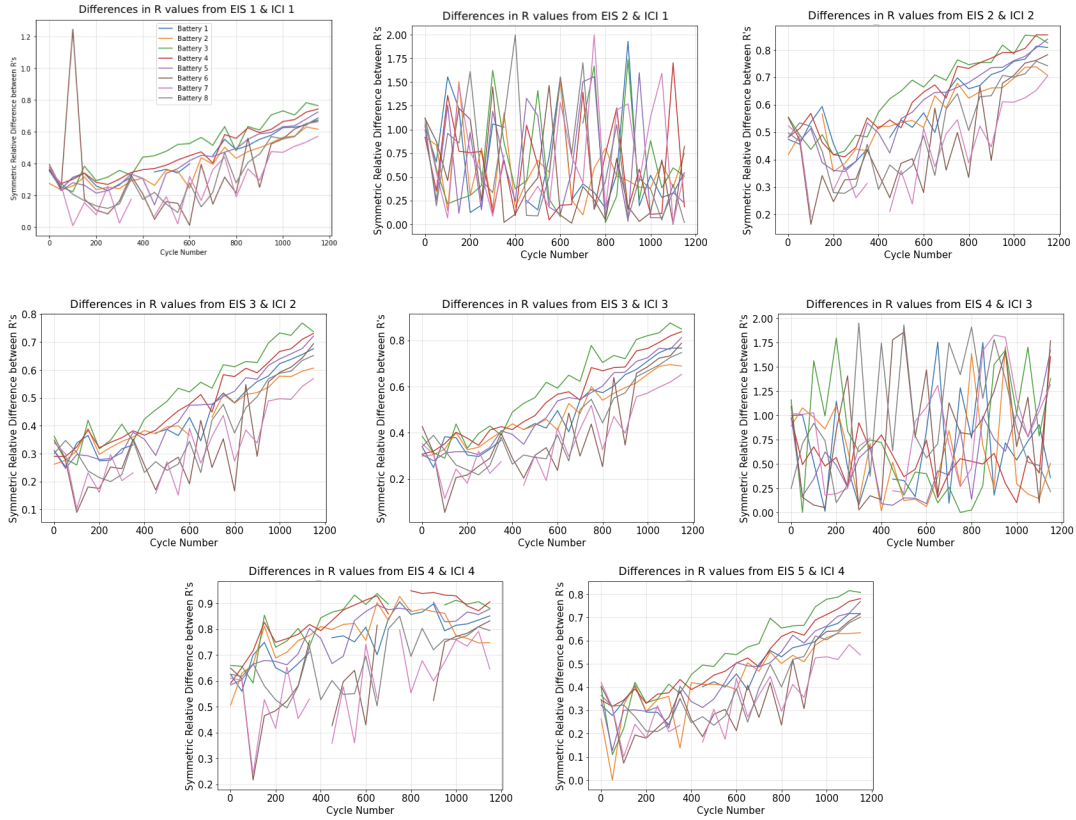

Figure 3.20: Symmetric relative differences in R values from EIS and ICI.

ICI 1 and first data from ICI 2 for EIS 2. Then, symmetric relative differences are calculated. The findings can be found in Figure 3.20.

$$\text{Symmetric Relative Difference} = \frac{|R_1 - R_2|}{(R_1 + R_2)/2}$$

Table 3.3: Single k Data Distance Coefficient results.

Data #	**ICI #**	**Voltage (V)**	**Distance Coeff.**
First	ICI 1	3.0-3.4	0.597
		3.5-3.6	0.940
		3.8-3.9	0.930
	ICI 2	3.0-3.4	0.781
		3.5-3.6	0.965
		3.8-3.9	0.947
	ICI 3	3.0-3.4	0.619
		3.5-3.6	0.887
		3.8-3.9	0.929
	ICI 4	3.0-3.4	0.977
		3.5-3.6	0.940
		3.8-3.9	0.940
Mid	ICI 1	3.0-3.4	0.688
		3.5-3.6	0.902
		3.8-3.9	0.924
	ICI 2	3.0-3.4	0.752
		3.5-3.6	0.939
		3.8-3.9	0.949
	ICI 3	3.0-3.4	0.647
		3.5-3.6	0.923
		3.8-3.9	0.898
	ICI 4	3.0-3.4	0.970
		3.5-3.6	0.933
		3.8-3.9	0.934
Last	ICI 1	3.0-3.4	0.477
		3.5-3.6	0.915
		3.8-3.9	0.935
	ICI 2	3.0-3.4	0.752
		3.5-3.6	0.965
		3.8-3.9	0.962
	ICI 3	3.0-3.4	0.472
		3.5-3.6	0.924
		3.8-3.9	0.921
	ICI 4	3.0-3.4	0.953
		3.5-3.6	0.970
		3.8-3.9	0.933

Table 3.4: Single R Data Distance Coefficient results.

Data #	**ICI #**	**Voltage (V)**	**Distance Coeff.**
First	ICI 1	3.0-3.4	0.385
		3.5-3.6	0.783
		3.8-3.9	0.694
	ICI 2	3.0-3.4	0.680
		3.5-3.6	0.778
		3.8-3.9	0.663
	ICI 3	3.0-3.4	0.394
		3.5-3.6	0.750
		3.8-3.9	0.711
	ICI 4	3.0-3.4	0.799
		3.5-3.6	0.811
		3.8-3.9	0.672
Mid	ICI 1	3.0-3.4	0.392
		3.5-3.6	0.739
		3.8-3.9	0.696
	ICI 2	3.0-3.4	0.696
		3.5-3.6	0.707
		3.8-3.9	0.658
	ICI 3	3.0-3.4	0.328
		3.5-3.6	0.742
		3.8-3.9	0.720
	ICI 4	3.0-3.4	0.778
		3.5-3.6	0.744
		3.8-3.9	0.658
Last	ICI 1	3.0-3.4	0.378
		3.5-3.6	0.728
		3.8-3.9	0.695
	ICI 2	3.0-3.4	0.696
		3.5-3.6	0.751
		3.8-3.9	0.657
	ICI 3	3.0-3.4	0.348
		3.5-3.6	0.740
		3.8-3.9	0.710
	ICI 4	3.0-3.4	0.756
		3.5-3.6	0.791
		3.8-3.9	0.667

Table 3.5: Averaged Capacity Uncertainty Ranges and Corresponding Percent Capacity Ranges Obtained from Binning.

k/R	**Voltage**	**C rate**	**Current(+)/(-)**	**Capacity Range (Ah)**	**Percent Range (Ah)**
k	high	C/5	+	0.125	4.4
			-	0.156	5.6
		C/10	+	0.134	4.8
			-	0.158	5.6
	mid	C/5	+	0.132	4.7
			-	0.156	5.6
		C/10	+	0.136	4.9
			-	0.151	5.4
	low	C/5	+	0.049	1.8
			-	0.388	14
		C/10	+	0.104	3.7
			-	0.350	13
R	high	C/5	+	0.383	14
			-	0.394	14
		C/10	+	0.337	12
			-	0.388	14
	mid	C/5	+	0.346	12
			-	0.418	15
		C/10	+	0.300	11
			-	0.321	11
	low	C/5	+	0.055	1.9
			-	0.576	21
		C/10	+	0.225	8.0
			-	0.510	18

Chapter 4

Discussion

The decline in the capacities are not in the same rate for each battery, even though they start out with almost same capacity with standard deviation of 0.016Ah. This clearly shows how unpredictable and complex the battery chemistry is; though they are cycled and tested in the same conditions the differences can grow over time. To be more specific, the standard deviation between the capacities of 8 batteries has grown more than 3 times of its initial value, reaching 0.057Ah after 1150 charging and discharging cycles. The literature is overpopulated with publications that develop experimental models using only a few batteries as these models require thorough statistics analysis and a sound model that fits all [41] [42]; however, this conclusion showcases how it can be a pitfall for utilizing those models in the field. This is also the main reason experimental methods are usually not used by themselves but in combination [43].

In the Appendix section, Table A.3 includes all of the distance correlations between EIS data and capacities. This is done by correlating the frequency specific

data point and the capacity value for the same cycle number. The correlation results vary greatly, between 0.290 and 0.898 for modulus data; and between 0.204 and 0.952 for the phase angle data. The best result belongs to the phase angle data for EIS 5 and 0.398 Hz. EIS 5 data are acquired after slow charging; but it should be noted that the low frequency data are acquired the last for every EIS. Although the best correlation belongs to the EIS 3 data for modulus, the coefficient is lower than the EIS 5 low frequency phase angle data. As EIS investigates all the data that have different timelines, every datapoint signifies an occurrence. The frequency that the data with the highest correlations help relate to the reason behind the capacity decline. In this case, as the 0.398 Hz corresponds to the lower frequency region, this signifies that the capacity decline is mostly due to the problems with the mass transfer and cathodic processes.

EIS actually includes information about both R and k. In the Nyquist plots, the distance between the first positive data point on the left hand side and y-axis is (-Z_i) the overall resistance (Figure 3.2), which is R. Moreover, k and modulus of the Warburg impedance (σ), which gives information about the diffusion coefficient, are linked with a simple formula [44]:

$$k = \sqrt{\frac{8}{\pi}} \cdot \sigma \tag{4.1}$$

However, these parameters were not extracted from the EIS data in this work for two main reasons. Obtaining Warburg impedance requires low frequency data points as the mass transport processes take longer times. Though the instruments don't have any limitations on the lower frequency side, voltage drifts and time constraints can create complications. For R values at the left hand side of the

Nyquist plot, which also correspond to high frequencies, the inductances can create problems for the correct evaluation of this parameter. To add, as data for all frequencies cannot be obtained, the preciseness when acquiring R data gets hindered. Nonetheless, EIS and ICI R's were compared with the data already obtained. Seen in Figure 3.20, some of these values are incomparable, especially for EIS 2 & ICI 1 and EIS 4 & ICI 3. However, an upwards trend can be observed for the rest of the plots, which means the differences between the R values grow with the cycle number.

For the Kramers-Kronig tests, the residuals show significance in the first and last cycles of batteries' lifetimes. This is expected due to batteries' chemistries stabilizing inbetween the first usage and near end of life. The first SEI creation occurs in the first cycles, hence, unpredictbility in the performance arises. Near end of life, due to possible changes in the morphologies of electrodes and/or loss of active material decrease; the performance and thus, stability drops. This can be observed from the rapidly decreasing capacity near the 80% mark.

Throughout the cycles, the end voltage for charging ICI had to be reduced for several times, as also stated in the experimental section. This was because during the current interruption, the voltage was retained throughout 10 seconds instead of dropping to the set voltage; however, interruption took place because the upper limit for the voltage was defined as 4.20V. The retaining of the said voltage can be due to slower rearrangement of the ions and electrons that are near the electrodes, or the changes in the charge transfer reactions. These may be due to numerous reasons; explained previously.

Figures 3.6 and 3.7 include all of the single data for k and R values versus cycle

respectively. This data was used for the distance correlation in the beginning. Though, it was quickly realized that this data produced skewed multiple results that were hard to draw conclusion from. The plots also indicate why the data had to be reduced; either using single data or averaged data. Hence, these were abandoned.

To increase the applicability on the field, all of the methods that are published have to be quick and straightforward. This is why single value R and k's were tested in the same manner with the averaged values. It can be seen by comparing 3.2 with 3.3 and 3.4 which include the numerical results for R and k averaged and single data, the best results belong to the ICI 4 for each of the datasets and most of the values are relatively close to the dCor values that belong to the averaged data. However, the single last data shows the best correlations for the mid voltage ICI 4 data, whereas all the other data as well as the averages have the best correlation for low voltage ICI 4. As the average is the most reliable dataset and the singles show different results, the quicker method could not be deemed as applicable. Though it can be said that first and mid data seems usable, this claim may not be viable for a different set of Aspilsan INR18650's when it produces different outcomes for this set with controlled environment. However, it should be noted that acquiring data can be integrated into a standard charging procedure, for both a seamless usage of the model and practicality.

The results from binning coincides with the distance coefficients. It should be noted that the bins constructed, or in other words, the standard deviation of the data used to signify the prediction power, is not connected to the distance coefficients inherently. How they are related is due to the data quality acquired from an aging battery being lower compared to the otherwise. Hence, higher

dCor observed with lower uncertainty in the capacity intervals fortify the validity of the models used.

In the end, the best correlations show up to be the diffusion resistance coefficient for the ICI 4. Also the best dCor coming from the EIS 5 is a supporting factor, which follows the same ICI. At lower voltages and under positive current loads, the main transportation process is Li^+ ions' movement from positive electrode (NMC side) to the negative electrode (graphite side). As the low frequency EIS correlation also suggests, the decline in the efficiency of mass transport is the limiting factor for the capacity decline. Though ICI 2 also inspects the same mass transportation, ICI 4's lower rate has made it the best test among others. The slow charging may have influenced a better stability for the ions to rearrange, increasing the data quality. Which, in the end, both showcased the phenomenon better and helped drawing conclusion from.

Moreover, apart from transportation, there are several other things happening in the battery's chemistry. As the cathode highly lithiated at low SoC and beginning of the charging, SEI is also in a built up state at this stage. Hence, it can be said that relatively slow SEI dissolution on the cathode side can be occurring, which means less Li^+ dissolved in the electrolyte to do work. Conversely, if the Lithium dissolution is too fast at the cathode, SEI poisoning may be occurring. As cycling takes place, cracking on both the electrodes is inevitable. These cracks may get activated during low SoC, charging state; which may cause dissolution of Ni, Co, and mostly Mn because of small scale electrolyte decomposition [13]. On the anode side, SEI reaarangement may not be as rapid or efficient at low SoC's, which may be causing the SoH decline. Moreover, Lithium plating might be occurring at the same electrode in relatively small, partially recoverable

amounts, which would be a contributing factor. However, it should be noted that as k parameter only deals with the transport process, these discussions cannot be confidently supported.

Chapter 5

Conclusions

In this work, mainly EIS and ICI were used for investigation of battery chemistry that causes SoH drop. The specific methodology was presented in the Experimental Setup and Methods section which consisted of sequential EIS, ICI, charging and discharging procedures. the Modulus, Phase Angle, k and r parameters that are acquired at each 50 cycles were used o see if any correlation between them and the capacity, ultimately SoH, existed. 8 batteries were employed for this purpose to ensure a sound statistical correlation that takes battery-to-battery variation into account.

For this investigation, the data was processed in several ways to find the highest correlations. EIS data was probed separately; as Modulus versus capacities and Phase Angles versus capacities. For ICI, firstly no grouping was done to see the whole data represented. Then, it was seen that the data were either too bulky or too sparse between different voltage ranges. Hence, averages were taken according to the specific voltage ranges which were set up to have balanced data population.

Furthermore, single data were also used to see if a strong correlation existed; as it would be more practical when data is acquired.

Also, for all the data mentioned, their validity were examined. For EIS's case, Kramers-Kronig test was applied and it was seen that most of the data was sound. For ICI, errors were found for each data point, as these errors dominate the errors that come from the cycler and potentiostat.

Lastly, after finding out that ICI at low SoC and slow charging produced the best results with the highest distance correlation, which is ICI 4, the usability of the method was tested. This was done with binning; the errors from the aforementioned ICI data were averaged and used for the height of the bins. The x-axis span of these bins were adjusted to coincide with the distance between two furthest data in the bins. In the end, it was seen that the prediction using data from ICI 4 produces a capacity error margin of 3.7%.

5.1 Future Work

In light of this work and these conclusions, studies with temperature and calendar aging can be done to see if any other mechanism is behind the SoH decline. Moreover, other battery chemistries and geometries can be tested with the same methodology to further validate this work and shed light on their aging processes. Furthermore, though 80% SoH left is deemed as end of life, studies can be continued with further reduced SoH values. This can indicate different occurrences in battery chemistries that take over as the culprits for performance drop.

Chapter 6

Data Availability

The raw data produced and analyzed throughout the thesis is available upon request, as well as at Zenodo database, under Creative Commons Attribution 4.0 International license with DOI: 10.5281/zenodo.14606633

Bibliography

[1] IRENA, *Critical materials: Batteries for electric vehicles*. Abu Dhabi, ae: International Renewable Energy Agency, 2024.

[2] K. W. Beard, *Linden's Handbook of Batteries, Fifth Edition*. McGraw Hill Professional, 5 2019.

[3] M. S. Whittingham, "Electrical energy storage and intercalation chemistry," *Science*, vol. 192, pp. 1126–1127, 6 1976.

[4] K. Mizushima, P. Jones, P. Wiseman, and J. Goodenough, "LixCoO2 (0<x<-1): A new cathode material for batteries of high energy density," *Materials Research Bulletin*, vol. 15, pp. 783–789, 6 1980.

[5] Y. Yerkinbekova, A. Kumarov, B. Tatykayev, A. Mentbayeva, E. Repo, and E. Laakso, "Ni-rich cathode materials with concentration gradients for high-energy and safe lithium-ion batteries: A comprehensive review," *Journal of Power Sources*, vol. 626, p. 235686, 11 2024.

[6] M. Winter, "The solid electrolyte interphase – the most important and the least understood solid electrolyte in rechargeable LI batteries," *Zeitschrift für Physikalische Chemie*, vol. 223, pp. 1395–1406, 11 2009.

[7] A. Wang, S. Kadam, H. Li, S. Shi, and Y. Qi, "Review on modeling of the anode solid electrolyte interphase (SEI) for lithium-ion batteries," *npj Computational Materials*, vol. 4, 3 2018.

[8] A. Stefanopoulou and Y. Kim, "10 - system-level management of rechargeable lithium-ion batteries," *Rechargeable Lithium Batteries*, pp. 281–302, 2015.

[9] Q. Qin, X. Li, Z. Wang, J. Wang, G. Yan, W. Peng, and H. Guo, "Experimental and simulation study of direct current resistance decomposition in large size cylindrical lithium-ion battery," *Electrochimica Acta*, vol. 465, p. 142947, 8 2023.

[10] X. Zhang, Y. Wang, C. Liu, and Z. Chen, "A novel approach of battery pack state of health estimation using artificial intelligence optimization algorithm," *Journal of Power Sources*, vol. 376, pp. 191–199, 12 2017.

[11] K. Song, D. Hu, Y. Tong, and X. Yue, "Remaining life prediction of lithium-ion batteries based on health management: A review," *Journal of Energy Storage*, vol. 57, p. 106193, 2023.

[12] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications," *Journal of Power Sources*, vol. 241, pp. 680–689, 2013.

[13] J. M. Reniers, G. Mulder, and D. A. Howey, "Review and performance comparison of mechanical-chemical degradation models for lithium-ion batteries," *Journal of The Electrochemical Society*, vol. 166, p. A3189, sep 2019.

[14] R. Xiong, L. Li, and J. Tian, "Towards a smarter battery management system: A critical review on battery state of health monitoring methods," *Journal of Power Sources*, vol. 405, pp. 18–29, 2018.

[15] Z. Pang, K. Yang, Z. Song, P. Niu, G. Chen, and J. Meng, "A new method for determining SOH of lithium batteries using the real-part ratio of EIS specific frequency impedance," *Journal of Energy Storage*, vol. 72, p. 108693, 8 2023.

[16] A. Nickol, T. Schied, C. Heubner, M. Schneider, A. Michaelis, M. Bobeth, and G. Cuniberti, "Gitt analysis of lithium insertion cathodes for determining the lithium diffusion coefficient at low temperature: Challenges and pitfalls," *Journal of The Electrochemical Society*, vol. 167, p. 090546, jun 2020.

[17] Y.-C. Chien, H. Liu, A. S. Menon, W. R. Brant, D. Brandell, and M. J. Lacey, "Rapid determination of solid-state diffusion coefficients in li-based batteries via intermittent current interruption method," *Nature communications*, vol. 14, no. 1, p. 2289, 2023.

[18] M. D. Kolamkar, L. Losinski, P. G. Lacroix, IV, A. H. D. Pham, and M. Inc, "US20230358818A1 - Electronic battery tester ," 5 2022.

[19] M. Dubarry, V. Svoboda, R. Hwu, and B. Y. Liaw, "Incremental capacity analysis and close-to-equilibrium ocv measurements to quantify capacity fade in commercial rechargeable lithium batteries," *Electrochemical and Solid-State Letters*, vol. 9, p. A454, jul 2006.

[20] X. Bian, L. Liu, J. Yan, Z. Zou, and R. Zhao, "An open circuit voltage-based model for state-of-health estimation of lithium-ion batteries: Model development and validation," *Journal of Power Sources*, vol. 448, p. 227401, 11 2019.

[21] H.-G. Schweiger, O. Obeidi, O. Komesker, A. Raschke, M. Schiemann, C. Zehner, M. Gehnen, M. Keller, and P. Birke, "Comparison of several methods for determining the internal resistance of lithium ion cells," *Sensors*, vol. 10, pp. 5604–5625, 6 2010.

[22] C. Weng, J. Sun, and H. Peng, "A unified open-circuit-voltage model of lithium-ion batteries for state-of-charge estimation and state-of-health monitoring," *Journal of Power Sources*, vol. 258, pp. 228–237, 2 2014.

[23] M. Ouyang, Z. Chu, L. Lu, J. Li, X. Han, X. Feng, and G. Liu, "Low temperature aging mechanism identification and lithium deposition in a large format lithium iron phosphate battery for different charge profiles," *Journal of Power Sources*, vol. 286, pp. 309–320, 4 2015.

[24] W. Waag, S. Käbitz, and D. U. Sauer, "Experimental investigation of the lithium-ion battery impedance characteristic at various conditions and aging states and its influence on the application," *Applied Energy*, vol. 102, pp. 885–897, 10 2012.

[25] M. E. Orazem and B. Tribollet, *Electrochemical Impedance spectroscopy*. John Wiley & Sons, Inc, 2 2008.

[26] R. Koch and A. Jossen, "Temperature measurement of large format pouch cells with impededance spectroscopy," in *The 28th International Electric Vehicle Symposium and Exhibition*, 2015.

[27] A. Cuadras and O. Kanoun, "Soc li-ion battery monitoring with impedance spectroscopy," in *2009 6th International Multi-Conference on Systems, Signals and Devices*, pp. 1–5, IEEE, 2009.

[28] J. Song and M. Z. Bazant, "Effects of nanoparticle geometry and size distribution on diffusion impedance of battery electrodes," *Journal of The Electrochemical Society*, vol. 160, no. 1, p. A15, 2012.

[29] M. Klett, J. A. Gilbert, S. E. Trask, B. J. Polzin, A. N. Jansen, D. W. Dees, and D. P. Abraham, "Electrode Behavior RE-Visited: Monitoring potential windows, capacity loss, and impedance changes in

LI1.03(NI0.5CO0.2MN0.3)0.97O2/Silicon-Graphite full cells," *Journal of The Electrochemical Society*, vol. 163, pp. A875–A887, 1 2016.

[30] P. S. Sabet, A. J. Warnecke, F. Meier, H. Witzenhausen, E. Martinez-Laserna, and D. U. Sauer, "Non-invasive yet separate investigation of anode/cathode degradation of lithium-ion batteries (nickel–cobalt–manganese vs. graphite) due to accelerated aging," *Journal of Power Sources*, vol. 449, p. 227369, 12 2019.

[31] N. Meddings, M. Heinrich, F. Overney, J.-S. Lee, V. Ruiz, E. Napolitano, S. Seitz, G. Hinds, R. Raccichini, M. Gaberšček, and J. Park, "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review," *Journal of Power Sources*, vol. 480, p. 228742, 9 2020.

[32] J. Huang, Y. Gao, J. Luo, S. Wang, C. Li, S. Chen, and J. Zhang, "Editors' Choice—Review—Impedance Response of Porous Electrodes: Theoretical Framework, Physical Models and Applications," *Journal of The Electrochemical Society*, vol. 167, p. 166503, 10 2020.

[33] M. A. Zabara, G. Katırcı, and B. Ülgüt, "Operando investigations of the interfacial electrochemical kinetics of metallic lithium anodes via Temperature-Dependent electrochemical impedance spectroscopy," *The Journal of Physical Chemistry C*, vol. 126, pp. 10968–10976, 6 2022.

[34] S. Bruckenstein and B. Miller, "Circuit for Transient-Free Current-Potential Control Conversion," *Journal of The Electrochemical Society*, vol. 117, p. 1040, 1 1970.

[35] W. Weppner and R. A. Huggins, "Determination of the kinetic parameters of Mixed-Conducting electrodes and application to the system LI3SB," *Journal of The Electrochemical Society*, vol. 124, pp. 1569–1578, 10 1977.

[36] M. J. Lacey, "Influence of the electrolyte on the internal resistance of lithium-sulfur batteries studied with an intermittent current interruption method," *ChemElectroChem*, vol. 4, no. 8, pp. 1997–2004, 2017.

[37] S. Yaman, "ASPİLSAN INR18650A28 - ASPİLSAN," 12 2023.

[38] B. Boukamp, "Practical application of the Kramers-Kronig transformation on impedance measurements in solid state electrochemistry," *Solid State Ionics*, vol. 62, pp. 131–141, 7 1993.

[39] B. A. Boukamp, "A linear Kronig-Kramers Transform test for immittance data validation," *Journal of The Electrochemical Society*, vol. 142, pp. 1885–1894, 6 1995.

[40] B. Ülgüt, "On DRT, measurement model and Kramers-Kronig compatibility of EIS," 1 2024.

[41] M. Dubarry and B. Y. Liaw, "Development of a universal modeling tool for rechargeable lithium batteries," *Journal of Power Sources*, vol. 174, pp. 856–860, 6 2007.

[42] X. Li, Z. Wang, L. Zhang, C. Zou, and D. D. Dorrell, "State-of-health estimation for Li-ion batteries by combing the incremental capacity analysis method with grey relational analysis," *Journal of Power Sources*, vol. 410-411, pp. 106–114, 11 2018.

[43] S. K. Pradhan and B. Chakraborty, "Battery management strategies: An essential review for battery state of health monitoring techniques," *Journal of Energy Storage*, vol. 51, p. 104427, 3 2022.

[44] Y.-C. Chien, A. S. Menon, W. R. Brant, D. Brandell, and M. J. Lacey, "Simultaneous monitoring of crystalline active materials and resistance evolution in Lithium–Sulfur batteries," *Journal of the American Chemical Society*, vol. 142, pp. 1449–1456, 12 2019.

Appendix A

Data

A.1 Part of n_cyc300.csv for Plotting Related Nyquist

cycle	battery	r_i													
350	1	r	0.029945	0.0300967	0.0303292	0.030659	0.0309469	0.0313239	0.0316652	0.0320794	0.0324869	0.0328185	0.0333266	0.0337653	0.034262
300	1	i	0.0042613	0.0029612	0.0018327	0.0009256	0.0001089	-0.0005027	-0.0009621	-0.0015081	-0.0018031	-0.0020718	-0.0024274	-0.0026171	-0.0028077
300	2	r	0.030086	0.0303353	0.0306799	0.0310366	0.0314212	0.0318925	0.0323712	0.0327084	0.0333124	0.0337509	0.0343556	0.0348911	0.0354213
300	2	i	0.0042093	0.0027956	0.0015752	0.0005549	-0.0002524	-0.0009323	-0.0014061	-0.0018947	-0.0023335	-0.0026232	-0.002942	-0.0031449	-0.0033461
300	3	r	0.0319969	0.032147	0.0324221	0.0327126	0.0329672	0.0333908	0.0337239	0.034118	0.034477	0.0348903	0.0353885	0.0358515	0.0363214
300	3	i	0.0047144	0.0032578	0.0020519	0.0010862	0.0002187	-0.000402	-0.0009367	-0.0013123	-0.0017499	-0.002177	-0.0023966	-0.0026164	-0.0028215
300	4	r	0.0258247	0.0260561	0.0263362	0.026638	0.0269863	0.0273708	0.0277333	0.0281719	0.0285764	0.028943	0.0295359	0.0299996	0.030524
300	4	i	0.0046621	0.0031052	0.0019445	0.0009729	0.0001261	-0.0005735	-0.001044	-0.0015	-0.0019504	-0.0022812	-0.0025811	-0.0028125	-0.0029649
300	5	r	0.027767	0.0280044	0.0283134	0.0285725	0.0289828	0.029323	0.0296684	0.030076	0.0305609	0.0309604	0.03153	0.0320166	0.0325459
300	5	i	0.0038494	0.0025698	0.0014057	0.0006095	-0.0002309	-0.0008145	-0.0012879	-0.0016951	-0.0021199	-0.0024359	-0.0026871	-0.0029243	-0.0030909
300	6	r	0.0294023	0.0296555	0.0298024	0.0301492	0.0304881	0.0307474	0.0310902	0.0317542	0.0317951	0.032325	0.0328046	0.0332332	0.0337241
300	6	i	0.003442	0.0022367	0.0011841	0.0004526	-0.0001864	-0.000794	-0.0013496	-0.001589	-0.0017919	-0.0022429	-0.0025202	-0.0026892	-0.0028944
300	7	r	0.0478808	0.0479827	0.0481425	0.0484492	0.0487515	0.0490095	0.0493223	0.0497447	0.0500829	0.0503722	0.0509282	0.05136	0.0518557
300	7	i	0.0033371	0.002102	0.0011363	0.0003591	-0.0003356	-0.0008975	-0.0012801	-0.0017876	-0.0019879	-0.0022511	-0.0025369	-0.0027225	-0.0028901
300	8	r	0.0310386	0.0311904	0.0314868	0.0317229	0.0320476	0.0323163	0.0327421	0.0331054	0.03351	0.0340728	0.0343802	0.0348226	0.0353268
300	8	i	0.0036585	0.0024455	0.0014747	0.000578	-0.0001339	-0.0006857	-0.0011682	-0.001564	-0.0019302	-0.0022352	-0.0024798	-0.0026917	-0.0028389

A.2 EIS Data (to_be_fed)

eis	cycle	battery	data	5015hz	3984hz	3170hz	2537hz	1976hz	1577hz	1265hz	998p2hz	796p9hz	627p8hz	495p6hz	398p0hz	315p5hz	252p4hz	198p6hz	158p4hz	125p6hz
1	0	1	mod	0.0184621	0.0182896	0.0182721	0.0183942	0.0186325	0.018932	0.0192641	0.0195697	0.0199614	0.0203966	0.0208494	0.0212698	0.0217382	0.0222083	0.0226232	0.0230912	0.0235107
	0	1	phz	14.60928	10.35064	6.954238	3.980114	1.475405	-0.3786434	-1.1806364	-2.815349	-3.790097	-4.923872	-5.470247	-5.870347	-6.049805	-6.089847	-6.040298	-5.918214	-5.684683
	0	2	mod	0.0239479	0.0236618	0.0236138	0.0236788	0.0238513	0.0241796	0.024446	0.0247922	0.0253085	0.0256854	0.0261962	0.0267175	0.0272038	0.0276636	0.0282604	0.028733	0.0293214
	0	2	phz	15.12652	11.17885	7.740435	5.054743	2.647315	0.7074993	-0.848963	-2.1211112	-3.306367	-4.233421	-4.674968	-5.2903	-5.63654	-5.649876	-5.87621	-6.009129	-5.799212
	1	3	mod	0.0173203	0.017106	0.0171338	0.0172843	0.0175011	0.0177772	0.0181335	0.0185431	0.0189428	0.0193032	0.0199053	0.0203614	0.0209717	0.0214882	0.0221199	0.0227238	0.0233584
	1	3	phz	15.03367	10.71486	6.848897	3.725431	0.8047458	-1.357702	-2.646918	-4.10849	-5.322567	-6.310657	-7.171999	-7.791751	-8.204563	-8.550146	-8.823853	-8.91437	-8.785625
	0	4	mod	0.0176669	0.0174894	0.01751	0.0175621	0.0177864	0.0180418	0.0183766	0.0187034	0.0190951	0.0195269	0.0199773	0.0204481	0.0209604	0.0214591	0.022047	0.0226364	0.0232552
	0	4	phz	15.51008	10.89377	7.330505	4.154675	1.399716	-0.5908133	-2.081753	-3.522385	-4.710846	-5.735905	-6.574673	-7.16772	-7.64942	-7.948725	-8.286341	-8.358963	-8.350158
	1	5	mod	0.0192142	0.0190807	0.0190768	0.0191797	0.019417	0.0197226	0.0200089	0.0203224	0.0206601	0.0212146	0.0218848	0.0223769	0.0229778	0.023552	0.0242111	0.0248841	0.0256168
	1	5	phz	13.99685	9.92029	6.378541	3.416611	1.067964	-1.031192	-2.550393	-3.980176	-5.172081	-6.348989	-6.924277	-7.618945	-8.156419	-8.64511	-9.127722	-9.280689	-9.481181
	1	6	mod	0.0182324	0.0180023	0.0179056	0.0179885	0.0182215	0.0185009	0.0187705	0.019013	0.0193366	0.0198546	0.0202477	0.0206352	0.0211202	0.0217571	0.0221009	0.0225939	0.0231031
	1	6	phz	16.29581	11.79244	7.964775	5.081534	2.31128	0.2907342	-1.179997	-2.620796	-3.486557	-4.899338	-5.488739	-6.055484	-6.471472	-6.700424	-6.79447	-6.904258	-6.888648
	1	7	mod	0.0180549	0.0177683	0.0177152	0.0178375	0.0180133	0.0182518	0.0185416	0.0189354	0.0193725	0.0197751	0.02016	0.0205832	0.0210607	0.0215276	0.0220321	0.0224616	0.0229767
	1	7	phz	16.75028	12.06888	8.28167	5.02685	2.360598	-0.0111722	-1.1655806	-2.861849	-4.08932	-4.770007	-5.544829	-6.11994	-6.479477	-6.661454	-6.707451	-6.750147	-6.612656
	0	8	mod	0.0194533	0.019415	0.0193015	0.0194187	0.01981	0.0203544	0.0207413	0.0212954	0.0217938	0.0221954	0.0226898	0.0229291	0.0233786	0.0241382	0.0245663	0.0251144	
	0	8	phz	14.70758	10.70142	6.709581	4.129317	1.582363	-0.3580924	-2.126004	-4.232436	-4.366227	-5.551648	-5.889059	-5.125588	-6.191232	-7.145552	-6.34288	-5.798032	-6.247684
	1	1	mod	0.0237841	0.0240536	0.0240519	0.0241348	0.0243773	0.0246	0.0249655	0.0252376	0.0256218	0.0260337	0.0264015	0.0268819	0.0272325	0.0276069	0.0280424	0.0284631	0.0288411		
	1	1	phz	12.21631	8.643904	6.11174	3.42875	1.64895	0.1012437	-1.1706676	-2.016275	-2.801331	-3.670191	-4.0547	-4.412168	-4.687763	-5.05968	-5.185149	-5.143515	-5.424298	
	1	2	mod	0.0246686	0.0246868	0.0246906	0.0248526	0.0251071	0.0255731	0.0259641	0.0263483	0.0267581	0.0272468	0.0277519	0.0281992	0.0286382	0.029276	0.0297162	0.0302082	0.0306499		
	1	2	phz	11.74268	8.288804	5.243297	2.773601	0.8374147	-0.6558417	-1.938794	-2.929026	-3.71762	-4.411065	-4.82337	-5.248338	-5.654922	-5.753187	-6.240646	-6.190409	-6.320377		
	1	3	mod	0.0227803	0.0225936	0.0226075	0.0227401	0.0229741	0.0233337	0.0237332	0.024065	0.0245528	0.0249416	0.0252846	0.025769	0.0261418	0.0267826	0.0271736	0.0276139	0.0282231		
	1	3	phz	12.21207	8.779825	5.645443	3.291325	1.232036	-0.3162648	-1.576621	-2.926787	-3.767365	-4.215645	-4.858071	-5.37361	-5.783427	-5.939106	-6.269702	-6.355084	-6.811739		
	1	4	mod	0.0229234	0.0228662	0.0229292	0.0231341	0.023422	0.0237582	0.0241218	0.0245338	0.0249842	0.0254825	0.0258816	0.0264648	0.0269212	0.0274941	0.0280099	0.0286166	0.029201		
	1	4	phz	11.85668	8.357678	5.120523	2.715544	0.671234	-1.050623	-2.284107	-3.340943	-4.246172	-5.054134	-5.725822	-6.232088	-6.503387	-6.931212	-7.20788	-7.491655	-8.022656		
	1	5	mod	0.0240834	0.0239568	0.0240647	0.0241285	0.0243893	0.0246938	0.0250959	0.0254821	0.0258546	0.0262149	0.0266695	0.0271801	0.0276098	0.0280777	0.0286248	0.0289883	0.0294653		
	1	5	phz	12.46066	9.113966	6.017587	3.682295	1.48467	-0.2750524	-1.474658	-2.560182	-3.40642	-4.031915	-4.62287	-4.971053	-5.369999	-5.683422	-6.009064	-6.217749	-6.384532		
	1	6	mod	0.0229074	0.0230992	0.0230704	0.0230905	0.0234153	0.0236684	0.0240373	0.0243829	0.0248045	0.0252579	0.0256471	0.0261887	0.0266666	0.0271551	0.0275661	0.028172	0.0287668		
	1	6	phz	12.41999	8.664888	5.901621	3.274266	1.305647	-0.5088949	-1.691771	-2.963732	-3.752687	-4.404996	-5.07797	-5.663385	-5.980019	-6.470895	-6.598042	-6.844257	-7.192149		
	1	7	mod	0.0220236	0.0218177	0.021854	0.0219587	0.0222713	0.0225332	0.0228592	0.0233505	0.0237195	0.0241043	0.0245356	0.0250077	0.0254438	0.0259726	0.0264158	0.0269393	0.0272859		
	1	7	phz	13.92584	9.96749	6.635557	3.733314	1.475457	-0.1100712	-1.626265	-2.805145	-3.702493	-4.408672	-4.928493	-5.173723	-5.377183	-5.61047	-5.810016	-5.606383	-5.520851		
	1	8	mod	0.0219358	0.0216193	0.021588	0.0216877	0.0218994	0.0221652	0.0223851	0.0227087	0.0230816	0.0233876	0.0237918	0.0240913	0.0245117	0.0248625	0.0251497	0.0255368	0.0259029		
	1	8	phz	14.0779	10.40718	7.074105	4.589421	2.576961	0.7036932	-0.6387489	-1.774015	-2.555539	-3.270768	-3.594364	-4.069037	-4.559237	-4.753221	-4.845456	-4.997428	-5.024853		
	1	1	mod	0.0235638	0.0234977	0.0235834	0.023782	0.0240192	0.0244287	0.0247633	0.0250374	0.025398	0.0258795	0.0263172	0.0267867	0.0272119	0.0277601	0.0282287	0.0286715	0.0292882		
	1	1	phz	9.197209	6.24339	3.853196	1.910966	0.0835657	-1.257084	-2.312918	-3.542109	-4.050861	-4.502684	-5.134111	-5.609427	-5.934734	-6.202456	-6.430807	-6.714423	-6.850223		
	1	2	mod	0.0283831	0.0284208	0.0285789	0.0288152	0.0291464	0.0294257	0.0298193	0.0303619	0.0307004	0.0312128	0.0317008	0.0321753	0.0327118	0.0332882	0.0338439	0.0344121			
	1	2	phz	7.264877	4.835175	2.701995	1.135326	-0.2820108	-1.582533	-2.416076	-3.342379	-3.951472	-4.423917	-4.883462	-5.196157	-5.495555	-5.607492	-5.871602	-5.942885	-6.059797		
	1	3	mod	0.031708	0.0317197	0.0318387	0.0319555	0.0322979	0.0325132	0.0328673	0.0332712	0.033476	0.0340167	0.0344449	0.0348787	0.0353095	0.0357505	0.0362818	0.0368362	0.0372968		
	1	3	phz	6.866864	4.690712	2.955609	1.395316	0.070284	-0.9050968	-1.560594	-2.305316	-2.929484	-3.431331	-3.818281	-4.115673	-4.372952	-4.502715	-4.808624	-4.977474	-5.06976		
	1	4	mod	0.02248	0.0224541	0.0225813	0.0227711	0.0231163	0.0234567	0.0239091	0.0242109	0.0244558	0.0247792	0.0252697	0.0257061	0.0262705	0.0268424	0.027007	0.0275061	0.0279901	0.028529	
	1	4	phz	9.94729	6.85824	4.19242	2.043388	0.0777946	-1.608532	-2.512104	-3.180672	-4.08628	-4.881907	-5.431758	-5.708298	-6.100485	-6.374303	-6.5659	-6.89767	-6.986156		
	1	5	mod	0.0265309	0.0266454	0.0266073	0.0267811	0.0271777	0.0274775	0.0278163	0.0281152	0.0286118	0.0290264	0.0294927	0.0300422	0.0308925	0.0315117	0.0319541	0.0326138			
	1	5	phz	7.976658	5.339759	3.136355	1.29872	-0.1360706	-1.361511	-2.318552	-3.250958	-3.807224	-4.493448	-4.942493	-5.333283	-5.667027	-5.742941	-6.139468	-6.38429	-6.554049		
	1	6	mod	0.0454952	0.0454858	0.0456033	0.0457547	0.046046	0.046387	0.0467111	0.0470608	0.0473181	0.0477413	0.0482201	0.0486541	0.0491365	0.049585	0.0501436	0.0506918	0.0512093		

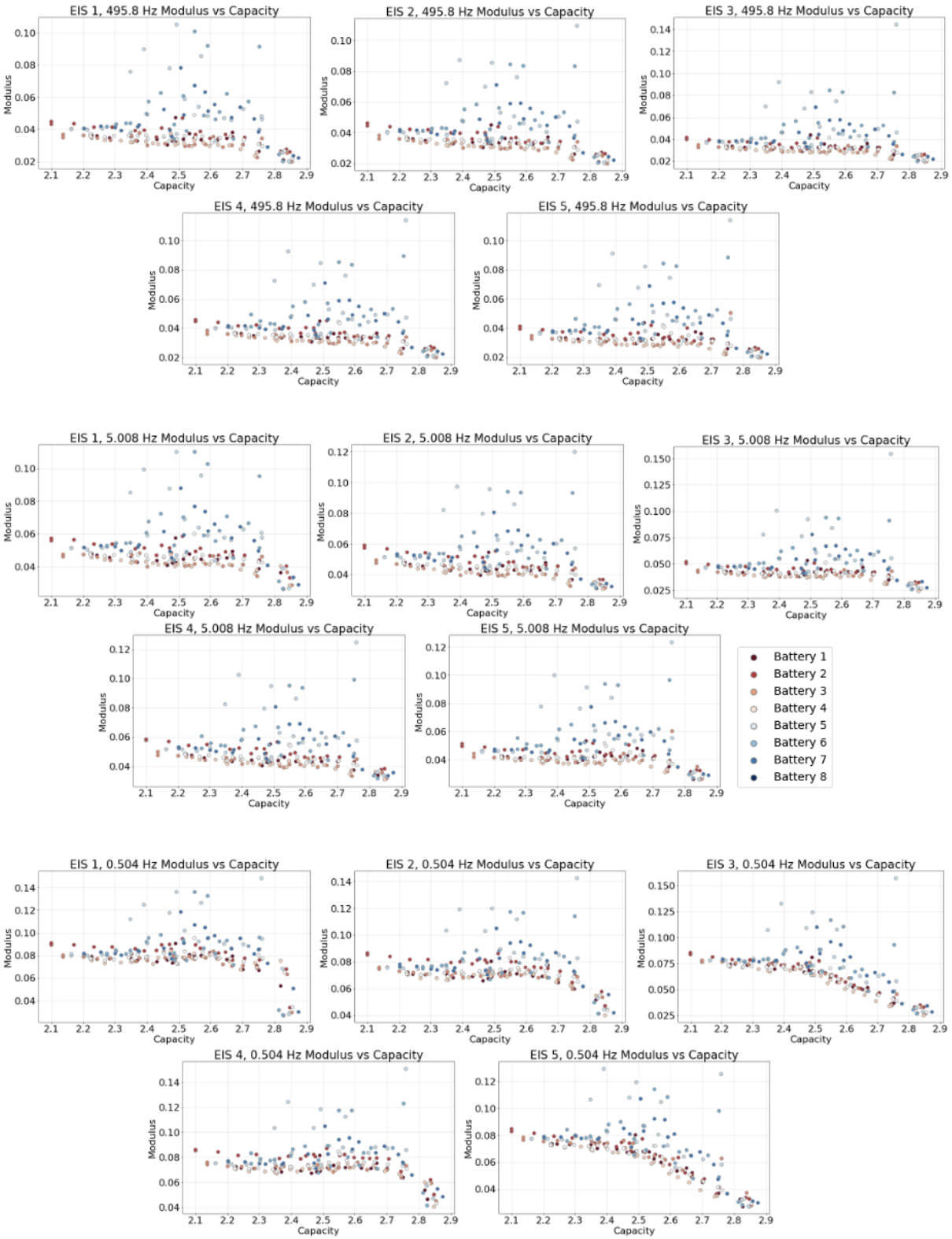

Figure A.1: Some of the EIS plots, Modulus at specific frequencies.

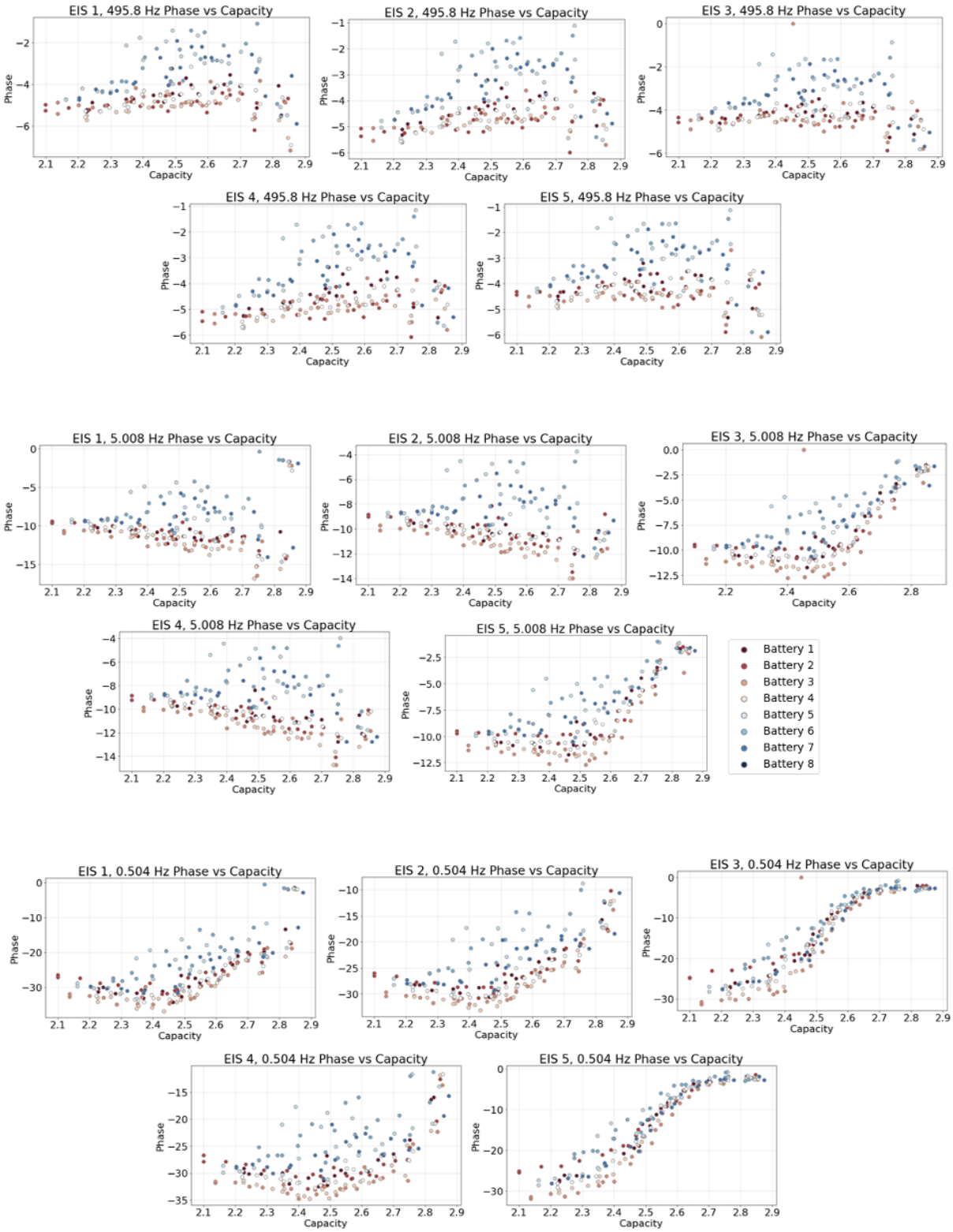

Figure A.2: Some of the EIS plots, Phase angles at specific frequencies.

A.3 EIS Distance Coefficients for each Frequency

Table A.1: EIS Distance Coefficients for each Frequency

— Modulus Data —								
EIS#	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.
1	5015	0.338	2	2.504	0.389	4	125.6	0.414
1	3984	0.346	2	1.998	0.381	4	100.4	0.414
1	3170	0.352	2	1.585	0.375	4	79	0.415
1	2527	0.359	2	1.267	0.370	4	62.78	0.416
1	1976	0.364	2	0.999	0.378	4	49.87	0.410
1	1577	0.369	2	0.794	0.392	4	39.72	0.415
1	1265	0.372	2	0.933	0.417	4	31.67	0.415
1	998.2	0.376	2	0.504	0.453	4	24.93	0.414
1	796.9	0.378	2	0.397	0.504	4	19.86	0.414
1	627.8	0.381	2	0.317	0.562	4	15.84	0.412
1	495.9	0.384	2	0.251	0.628	4	12.40	0.411
1	398.0	0.386	2	0.199	0.693	4	9.931	0.409
1	315.5	0.388	2	0.159	0.751	4	7.945	0.407
1	252.4	0.389	2	0.126	0.795	4	6.317	0.404
1	198.6	0.390	2	0.100	0.829	4	5.008	0.398
1	158.4	0.391	3	5015	0.320	4	3.946	0.391
1	125.6	0.391	3	3984	0.328	4	2.504	0.374
1	100.4	0.390	3	3170	0.333	4	1.998	0.362
1	79.00	0.390	3	2527	0.338	4	1.585	0.351
1	62.78	0.389	3	1976	0.343	4	1.267	0.341
1	49.87	0.386	3	1577	0.344	4	0.999	0.340
1	39.72	0.387	3	1265	0.348	4	0.794	0.346
1	31.67	0.387	3	998.2	0.349	4	0.933	0.363
1	24.93	0.386	3	796.9	0.350	4	0.504	0.391
1	19.86	0.386	3	627.8	0.352	4	0.397	0.430
1	15.84	0.385	3	495.9	0.352	4	0.317	0.474
1	12.40	0.383	3	398.0	0.353	4	0.251	0.527
1	9.931	0.380	3	315.5	0.354	4	0.199	0.581
1	7.945	0.377	3	252.4	0.354	4	0.159	0.639
1	6.317	0.373	3	198.6	0.352	4	0.126	0.696
1	5.008	0.367	3	158.4	0.351	4	0.100	0.743
1	3.946	0.356	3	125.6	0.351	5	5015	0.290
1	2.504	0.338	3	100.4	0.350	5	3984	0.302
1	1.998	0.329	3	79.00	0.350	5	3170	0.310
1	1.585	0.326	3	62.78	0.348	5	2527	0.321
1	1.267	0.329	3	49.87	0.340	5	1976	0.339
1	0.999	0.345	3	39.72	0.346	5	1577	0.337
1	0.794	0.368	3	31.67	0.344	5	1265	0.338
1	0.933	0.401	3	24.93	0.343	5	998.2	0.340
1	0.504	0.440	3	19.86	0.344	5	796.9	0.338
1	0.397	0.504	3	15.84	0.344	5	627.8	0.340
1	0.317	0.578	3	12.40	0.345	5	495.9	0.342
1	0.251	0.664	3	9.931	0.347	5	398.0	0.340
1	0.199	0.735	3	7.945	0.351	5	315.5	0.342
1	0.159	0.788	3	6.317	0.356	5	252.4	0.341
1	0.126	0.823	3	5.008	0.367	5	198.6	0.340

Continued on next page

Table A.1 – continued from previous page

EIS#	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.
1	0.100	0.853	3	3.946	0.381	5	158.4	0.340
2	5015	0.352	3	2.504	0.428	5	125.6	0.338
2	3984	0.362	3	1.998	0.462	5	100.4	0.338
2	3170	0.370	3	1.585	0.508	5	79.00	0.337
2	2527	0.376	3	1.267	0.562	5	62.78	0.337
2	1976	0.382	3	0.999	0.622	5	49.87	0.329
2	1577	0.388	3	0.794	0.680	5	39.72	0.334
2	1265	0.392	3	0.933	0.737	5	31.67	0.333
2	998.2	0.396	3	0.504	0.784	5	24.93	0.332
2	796.9	0.400	3	0.397	0.819	5	19.86	0.334
2	627.8	0.403	3	0.317	0.842	5	15.84	0.335
2	495.9	0.407	3	0.251	0.858	5	12.40	0.336
2	398.0	0.409	3	0.199	0.870	5	9.931	0.336
2	315.5	0.411	3	0.159	0.878	5	7.945	0.342
2	252.4	0.413	3	0.126	0.884	5	6.317	0.347
2	198.6	0.415	3	0.100	0.889	5	5.008	0.356
2	158.4	0.417	4	5015	0.348	5	3.946	0.372
2	125.6	0.418	4	3984	0.357	5	2.504	0.414
2	100.4	0.418	4	3170	0.365	5	1.998	0.445
2	79.00	0.419	4	2527	0.372	5	1.585	0.487
2	62.78	0.420	4	1976	0.378	5	1.267	0.535
2	49.87	0.416	4	1577	0.384	5	0.999	0.595
2	39.72	0.421	4	1265	0.388	5	0.794	0.655
2	31.67	0.421	4	998.2	0.392	5	0.933	0.717
2	24.93	0.421	4	796.9	0.395	5	0.504	0.772
2	19.86	0.421	4	627.8	0.400	5	0.397	0.815
2	15.84	0.422	4	495.9	0.403	5	0.317	0.841
2	12.40	0.420	4	398.0	0.405	5	0.251	0.861
2	9.931	0.420	4	315.5	0.407	5	0.199	0.874
2	7.945	0.418	4	252.4	0.409	5	0.159	0.884
2	6.317	0.415	4	198.6	0.411	5	0.126	0.891
2	5.008	0.411	4	158.4	0.413	5	0.100	0.898
2	3.946	0.404						
				— Phase Angle Data —				
EIS#	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.
1	5015	0.338	2	2.504	0.389	4	125.6	0.414
1	3984	0.346	2	1.998	0.381	4	100.4	0.414
1	3170	0.352	2	1.585	0.375	4	79.00	0.415
1	2527	0.359	2	1.267	0.370	4	62.78	0.416
1	1976	0.364	2	0.999	0.378	4	49.87	0.410
1	1577	0.369	2	0.794	0.392	4	39.72	0.415
1	1265	0.372	2	0.933	0.417	4	31.67	0.415
1	998.2	0.376	2	0.504	0.453	4	24.93	0.414
1	796.9	0.378	2	0.397	0.504	4	19.86	0.414
1	627.8	0.381	2	0.317	0.562	4	15.84	0.412
1	495.9	0.384	2	0.251	0.628	4	12.40	0.411
1	398.0	0.386	2	0.199	0.693	4	9.931	0.409
1	315.5	0.388	2	0.159	0.751	4	7.945	0.407
1	252.4	0.389	2	0.126	0.795	4	6.317	0.404
1	198.6	0.390	2	0.100	0.829	4	5.008	0.398
1	158.4	0.391	3	5015	0.320	4	3.946	0.391
1	125.6	0.391	3	3984	0.328	4	2.504	0.374
1	100.4	0.390	3	3170	0.333	4	1.998	0.362
1	79.00	0.390	3	2527	0.338	4	1.585	0.351
1	62.78	0.389	3	1976	0.343	4	1.267	0.341

Continued on next page

Table A.1 – continued from previous page

EIS#	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.
1	49.87	0.386	3	1577	0.344	4	0.999	0.340
1	39.72	0.387	3	1265	0.348	4	0.794	0.346
1	31.67	0.387	3	998.2	0.349	4	0.933	0.363
1	24.93	0.386	3	796.9	0.350	4	0.504	0.391
1	19.86	0.386	3	627.8	0.352	4	0.397	0.430
1	15.84	0.385	3	495.9	0.352	4	0.317	0.474
1	12.40	0.383	3	398.0	0.353	4	0.251	0.527
1	9.931	0.380	3	315.5	0.354	4	0.199	0.581
1	7.945	0.377	3	252.4	0.354	4	0.159	0.639
1	6.317	0.373	3	198.6	0.352	4	0.126	0.696
1	5.008	0.367	3	158.4	0.351	4	0.100	0.743
1	3.946	0.356	3	125.6	0.351	5	5015	0.290
1	2.504	0.338	3	100.4	0.350	5	3984	0.302
1	1.998	0.329	3	79.00	0.350	5	3170	0.310
1	1.585	0.326	3	62.78	0.348	5	2527	0.321
1	1.267	0.329	3	49.87	0.340	5	1976	0.339
1	0.999	0.345	3	39.72	0.346	5	1577	0.337
1	0.794	0.368	3	31.67	0.344	5	1265	0.338
1	0.933	0.401	3	24.93	0.343	5	998.2	0.340
1	0.504	0.440	3	19.86	0.344	5	796.9	0.338
1	0.397	0.504	3	15.84	0.344	5	627.8	0.340
1	0.317	0.578	3	12.40	0.345	5	495.9	0.342
1	0.251	0.664	3	9.931	0.347	5	398.0	0.340
1	0.199	0.735	3	7.945	0.351	5	315.5	0.342
1	0.159	0.788	3	6.317	0.356	5	252.4	0.341
1	0.126	0.823	3	5.008	0.367	5	198.6	0.340
1	0.100	0.853	3	3.946	0.381	5	158.4	0.340
2	5015	0.352	3	2.504	0.428	5	125.6	0.338
2	3984	0.362	3	1.998	0.462	5	100.4	0.338
2	3170	0.370	3	1.585	0.508	5	79.00	0.337
2	2527	0.376	3	1.267	0.562	5	62.78	0.337
2	1976	0.382	3	0.999	0.622	5	49.87	0.329
2	1577	0.388	3	0.794	0.680	5	39.72	0.334
2	1265	0.392	3	0.933	0.737	5	31.67	0.333
2	998.2	0.396	3	0.504	0.784	5	24.93	0.332
2	796.9	0.400	3	0.397	0.819	5	19.86	0.334
2	627.8	0.403	3	0.317	0.842	5	15.84	0.335
2	495.9	0.407	3	0.251	0.858	5	12.40	0.336
2	398.0	0.409	3	0.199	0.870	5	9.931	0.336
2	315.5	0.411	3	0.159	0.878	5	7.945	0.342
2	252.4	0.413	3	0.126	0.884	5	6.317	0.347
2	198.6	0.415	3	0.100	0.889	5	5.008	0.356
2	158.4	0.417	4	5015	0.348	5	3.946	0.372
2	125.6	0.418	4	3984	0.357	5	2.504	0.414
2	100.4	0.418	4	3170	0.365	5	1.998	0.445
2	79.00	0.419	4	2527	0.372	5	1.585	0.487
2	62.78	0.420	4	1976	0.378	5	1.267	0.535
2	49.87	0.416	4	1577	0.384	5	0.999	0.595
2	39.72	0.421	4	1265	0.388	5	0.794	0.655
2	31.67	0.421	4	998.2	0.392	5	0.933	0.717
2	24.93	0.421	4	796.9	0.395	5	0.504	0.772
2	19.86	0.421	4	627.8	0.400	5	0.397	0.815
2	15.84	0.422	4	495.9	0.403	5	0.317	0.841
2	12.40	0.420	4	398.0	0.405	5	0.251	0.861
2	9.931	0.420	4	315.5	0.407	5	0.199	0.874

Continued on next page

Table A.1 – continued from previous page

EIS#	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.	**EIS#**	Frequency	Dist. Coeff.
2	7.945	0.418	4	252.4	0.409	5	0.159	0.884
2	6.317	0.415	4	198.6	0.411	5	0.126	0.891
2	5.008	0.411	4	158.4	0.413	5	0.100	0.898
2	3.946	0.404						

A.4 K-K Residuals

Table A.2: Kramers-Kronig Residuals for Each Battery at Various Cycles

EIS#	**Cycle**	Battery 1	Battery 2	Battery 3	Battery 4	Battery 5	Battery 6	Battery 7	Battery 8
1	0	0.0035	0.0078	0.0041	0.0036	0.0073	0.0137	0.0148	–
1	50	0.0088	0.0091	0.0177	0.0063	0.0128	0.0127	0.0208	0.0192
1	100	0.0048	0.0078	0.0058	0.0077	0.0039	0.3481	0.0185	0.0140
1	150	0.0075	0.0105	0.0149	0.0118	0.0070	0.0037	0.0042	0.0036
1	200	0.0111	0.0047	0.0028	0.0087	0.0058	0.0055	0.0043	0.0074
1	250	0.0032	0.0129	0.0115	0.0092	0.0122	0.0082	0.0054	0.0021
1	300	0.0029	0.0110	0.0078	0.0085	0.0070	0.0033	0.0167	0.0034
1	350	0.0070	0.0035	0.0071	0.0057	0.0113	0.0082	0.0053	0.0046
1	400	0.0077	0.0094	0.0110	0.0093	0.0082	0.0050	0.0096	0.0029
1	450	0.0115	0.0071	0.0117	0.0091	0.0036	0.0020	0.0063	0.0033
1	500	0.0051	0.0090	0.0099	0.0105	0.0061	0.0044	0.0065	0.0040
1	550	0.0060	0.0081	0.0144	0.0093	0.0069	0.0041	0.0043	0.0035
1	600	0.0078	0.0052	0.0114	0.0074	0.0061	0.0218	0.0045	0.0074
1	650	0.0079	0.0118	0.0074	0.0105	0.0065	0.0073	0.0038	0.0054
1	700	0.0096	0.0093	0.0098	0.0043	0.0175	0.0034	0.0061	0.0314
1	750	0.0064	0.0089	0.0072	0.0138	0.0103	0.0113	0.0082	0.0090
1	800	0.0139	0.0073	0.0047	0.0113	0.0098	0.0046	0.0043	0.0027
1	850	0.0139	0.0106	0.0117	0.0183	0.0108	0.0111	0.0079	0.0054
1	900	0.0109	0.0089	0.0099	0.0113	0.0076	0.0042	0.0039	0.0082
1	950	0.0079	0.0087	0.0164	0.0108	0.0101	0.0071	0.0107	0.0105
1	1000	0.0110	0.0051	0.0192	0.0086	0.0127	0.0082	0.0071	0.0078
1	1050	0.0081	0.0107	0.0116	0.0147	0.0092	0.0124	0.0053	0.0111
1	1100	0.0119	0.0111	0.0139	0.0167	0.0143	0.0108	0.0085	0.0078
1	1150	0.0082	0.0090	0.0139	0.0112	0.0137	0.0128	0.0104	0.0074
2	0	0.0310	0.0151	0.0176	0.0347	0.0241	0.0152	0.0274	0.0170
2	50	0.0104	0.0383	0.0092	0.0145	0.0122	0.0478	0.0231	0.0218
2	100	0.0147	0.0114	0.0127	0.0053	0.0123	0.0033	0.0054	0.0113
2	150	0.0075	0.0172	0.0114	0.0181	0.0051	0.0048	0.0078	0.0073
2	200	0.0056	0.0143	0.0101	0.0122	0.0057	0.0057	0.0051	0.0058
2	250	0.0093	0.0118	0.0103	0.0123	0.0092	0.0064	0.0121	0.0078
2	300	0.0097	0.0033	0.0088	0.0082	0.0052	0.0048	0.0111	0.0067
2	350	0.0050	0.0058	0.0058	0.0034	0.0034	0.0038	0.0117	0.0056
2	400	0.0170	0.0092	0.0045	0.0063	0.0037	0.0034	0.0196	0.0025
2	450	0.0032	0.0025	0.0035	0.0073	0.0063	0.0014	0.0067	0.0030
2	500	0.0099	0.0031	0.0058	0.0079	0.0038	0.0050	0.0123	0.0050

Continued on next page

Table A.2 – continued from previous page

EIS#	Cycle	Battery 1	Battery 2	Battery 3	Battery 4	Battery 5	Battery 6	Battery 7	Battery 8
2	550	0.0053	0.0062	0.0086	0.0064	0.0066	0.0043	0.0044	0.0025
2	600	0.0069	0.0079	0.0044	0.0048	0.0116	0.0031	0.0075	0.0045
2	650	0.0093	0.0061	0.0029	0.0078	0.0068	0.0101	0.0104	0.0027
2	700	0.0077	0.0040	0.0052	0.0054	0.0145	0.0042	0.0132	0.0058
2	750	0.0074	0.0092	0.0049	0.0064	0.0093	0.0058	0.0144	0.0043
2	800	0.0133	0.0030	0.0028	0.0045	0.0052	0.0014	0.0120	0.0053
2	850	0.0034	0.0066	0.0078	0.0037	0.0032	0.0065	0.0094	0.0028
2	900	0.0112	0.0151	0.0035	0.0046	0.0082	0.0065	0.0075	0.0099
2	950	0.0065	0.0174	0.0074	0.0038	0.0040	0.0135	0.0077	0.0102
2	1000	0.0189	0.0148	0.0068	0.0096	0.0084	0.0083	0.0109	0.0087
2	1050	0.0130	0.0176	0.0153	0.0079	0.0085	0.0121	0.0114	0.0151
2	1100	0.0091	0.0157	0.0090	0.0111	0.0131	0.0056	0.0143	0.0129
2	1150	0.0187	0.0188	0.0113	0.0124	0.0148	0.0146	0.0083	0.0114
3	0	0.0133	0.0254	0.0080	0.0128	0.0114	0.0059	0.0096	0.0434
3	50	0.0356	0.0669	0.0984	0.0222	0.0321	0.0319	0.0311	0.0320
3	100	0.0107	0.0132	0.0042	0.0129	0.0091	0.0027	0.0027	0.0063
3	150	0.0127	0.0095	0.0094	0.0152	0.0088	0.0068	0.0069	0.0065
3	200	0.0068	0.0087	0.0134	0.0092	0.0063	0.0069	0.0059	0.0080
3	250	0.0055	0.0142	0.0097	0.0091	0.0061	0.0055	0.0049	0.0038
3	300	0.0166	0.0064	0.0086	0.0058	0.0058	0.0083	0.0041	0.0049
3	350	0.0068	0.0090	0.0112	0.0060	0.0095	0.0047	0.0108	0.0075
3	400	0.0084	0.0101	0.0107	0.0095	0.0068	0.0047	0.0072	0.0079
3	450	0.0082	0.0099	0.0121	0.0112	0.0085	0.0040	0.0035	0.0066
3	500	0.0094	0.0067	0.0092	0.0062	0.0053	0.0076	0.0040	0.0084
3	550	0.0068	0.0069	0.0080	0.0154	0.0083	0.0068	0.0031	0.0025
3	600	0.0082	0.0100	0.0058	0.0152	0.0099	0.0020	0.0093	0.0052
3	650	0.0086	0.0107	0.0090	0.0182	0.0108	0.0047	0.0039	0.0033
3	700	0.0099	0.0081	0.0083	0.0042	0.0082	0.0027	0.0063	0.0090
3	750	0.0076	0.0084	0.0131	0.0118	0.0127	0.0033	0.0135	0.0066
3	800	0.0115	0.0046	0.0080	0.0100	0.0092	0.0042	0.0051	0.0079
3	850	0.0095	0.0055	0.0080	0.0070	0.0143	0.0127	0.0036	0.0056
3	900	0.0108	0.0083	0.0124	0.0114	0.0107	0.0083	0.0056	0.0085
3	950	0.0138	0.0121	0.0097	0.0125	0.0096	0.0117	0.0077	0.0130
3	1000	0.0080	0.0068	0.0122	0.0163	0.0097	0.0137	0.0106	0.0059
3	1050	0.0128	0.0074	0.0174	0.0181	0.0161	0.0111	0.0089	0.0100
3	1100	0.0176	0.0083	0.0160	0.0121	0.0122	0.0152	0.0087	0.0105
3	1150	0.0116	0.0113	0.0144	0.0137	0.0204	0.0158	0.0069	0.0132
4	0	0.0534	0.0461	0.0170	0.0186	0.0232	0.0210	0.0175	0.0681
4	50	0.0263	0.0104	0.0095	0.0236	0.0408	0.0151	0.0090	0.0299
4	100	0.0177	0.0130	0.0122	0.0126	0.0134	0.0042	0.0065	0.0153
4	150	0.0305	0.0138	0.0200	0.0152	0.0083	0.0087	0.0180	0.0186
4	200	0.0227	0.0139	0.0229	0.0115	0.0139	0.0100	0.0154	0.0098
4	250	0.0166	0.0121	0.0197	0.0190	0.0129	0.0111	0.0264	0.0111
4	300	0.0228	0.0207	0.0205	0.0150	0.0110	0.0118	0.0153	0.0170
4	350	0.0247	0.0146	0.0115	0.0161	0.0137	0.0088	0.0231	0.0181
4	400	0.0170	0.0128	0.0178	0.0193	0.0045	0.0114	0.0231	0.0105
4	450	0.0296	0.0181	0.0148	0.0093	0.0075	0.0091	0.0095	0.0144
4	500	0.0192	0.0105	0.0177	0.0160	0.0148	0.0088	0.0173	0.0051
4	550	0.0188	0.0099	0.0168	0.0094	0.0134	0.0084	0.0161	0.0152
4	600	0.0136	0.0110	0.0074	0.0259	0.0051	0.0067	0.0183	0.0122
4	650	0.0099	0.0204	0.0194	0.0103	0.0114	0.0169	0.0224	0.0044
4	700	0.0281	0.0113	0.0188	0.0149	0.0189	0.0045	0.0328	0.0127
4	750	0.0155	0.0225	0.0049	0.0221	0.0113	0.0118	0.0222	0.0194
4	800	0.0243	0.0152	0.0164	0.0213	0.0210	0.0068	0.0167	0.0131

Continued on next page

Table A.2 – continued from previous page

EIS#	**Cycle**	Battery 1	Battery 2	Battery 3	Battery 4	Battery 5	Battery 6	Battery 7	Battery 8
4	850	0.0186	0.0225	0.0261	0.0176	0.0033	0.0097	0.0161	0.0124
4	900	0.0257	0.0140	0.0084	0.0121	0.0122	0.0121	0.0179	0.0132
4	950	0.0202	0.0251	0.0155	0.0258	0.0128	0.0099	0.0131	0.0147
4	1000	0.0163	0.0161	0.0160	0.0296	0.0254	0.0115	0.0243	0.0172
4	1050	0.0318	0.0259	0.0188	0.0220	0.0141	0.0287	0.0231	0.0167
4	1100	0.0268	0.0246	0.0164	0.0271	0.0205	0.0172	0.0241	0.0224
4	1150	0.0210	0.0219	0.0289	0.0206	0.0293	0.0384	0.0171	0.0235
5	0	0.0062	0.0503	0.0101	0.0043	0.0106	0.0163	0.0042	0.0642
5	50	0.0415	0.1387	0.4202	0.0550	0.0810	0.0393	0.0368	0.0436
5	100	0.0078	0.0103	0.0051	0.0153	0.0143	0.0037	0.0013	0.0108
5	150	0.0109	0.0059	0.0126	0.0095	0.0088	0.0062	0.0095	0.0077
5	200	0.0055	0.0085	0.0107	0.0076	0.0064	0.0063	0.0033	0.0088
5	250	0.0049	0.0036	0.0090	0.0149	0.0058	0.0062	0.0095	0.0042
5	300	0.0092	0.0045	0.0041	0.0149	0.0079	0.0064	0.0029	0.0085
5	350	0.0070	0.0106	0.0106	0.0117	0.0062	0.0071	0.0089	0.0112
5	400	0.0066	0.0050	0.0056	0.0096	0.0072	0.0033	0.0075	0.0057
5	450	0.0086	0.0132	0.0095	0.0128	0.0077	0.0029	0.0043	0.0059
5	500	0.0085	0.0086	0.0076	0.0080	0.0088	0.0069	0.0073	0.0031
5	550	0.0040	0.0069	0.0081	0.0058	0.0072	0.0084	0.0045	0.0053
5	600	0.0066	0.0063	0.0107	0.0142	0.0103	0.0032	0.0063	0.0064
5	650	0.0061	0.0071	0.0075	0.0167	0.0098	0.0137	0.0039	0.0026
5	700	0.0119	0.0071	0.0104	0.0072	0.0080	0.0034	0.0064	0.0050
5	750	0.0122	0.0130	0.0092	0.0081	0.0092	0.0093	0.0130	0.0112
5	800	0.0119	0.0068	0.0143	0.0078	0.0156	0.0059	0.0074	0.0044
5	850	0.0084	0.0103	0.0114	0.0081	0.0150	0.0066	0.0077	0.0094
5	900	0.0099	0.0114	0.0133	0.0148	0.0113	0.0080	0.0051	0.0074
5	950	0.0093	0.0111	0.0176	0.0074	0.0142	0.0097	0.0122	0.0121
5	1000	0.0099	0.0126	0.0188	0.0113	0.0172	0.0149	0.0078	0.0128
5	1050	0.0085	0.0055	0.0207	0.0164	0.0143	0.0146	0.0134	0.0140
5	1100	0.0121	0.0083	0.0147	0.0115	0.0112	0.0118	0.0073	0.0134
5	1150	0.0097	0.0076	0.0136	0.0152	0.0165	0.0135	0.0071	0.0154

Appendix B

Code

B.1 Code for Plotting Nyquist for EIS1

```
1
2   import numpy as np
3   import matplotlib.pyplot as plt
4   import glob
5   from matplotlib.cm import get_cmap
6   from matplotlib.colors import Normalize
7   from mpl_toolkits.mplot3d import Axes3D
8
9   plt.rcParams.update({'font.size': 15})
10
11
12  csv_files = glob.glob('n_cyc*.csv')
13
14
15  data_by_cycle_battery = {}
16
17  # Load data from CSV files
18  for file in csv_files:
19      data = np.genfromtxt(file, delimiter=',', skip_header=1)
20      for i in range(0, data.shape[0], 2):
21          cycle_number = int(data[i, 0])  # Cycle number
22          battery_number = int(data[i, 1])  # Battery number
23          r_values = data[i, 3:]
24          i_values = data[i + 1, 3:]
25
26          if np.isnan(r_values).all() or np.isnan(i_values).all():
27              continue
28
29          if cycle_number not in data_by_cycle_battery:
```

```
30              data_by_cycle_battery[cycle_number] = {}
31
32          data_by_cycle_battery[cycle_number][battery_number] = {
33              'r_values': r_values,
34              'i_values': i_values
35          }
36
37  # Function to create a colormap based on cycle numbers
38  def get_colormap(cycle_numbers):
39      cmap = get_cmap('copper')  # Choose a colormap, e.g., 'viridis', 'plasma', 'coolwarm'
40      norm = Normalize(vmin=min(cycle_numbers), vmax=max(cycle_numbers))
41      return cmap, norm
42
43  # Function to ensure 1:1 aspect ratio in 2D plots
44  def set_aspect_equal(ax):
45      """Set equal scaling for the axes."""
46      ax.set_aspect('equal', adjustable='box')
47
48  def plot_with_gradient(data_dict):
49      cycle_numbers = sorted(data_dict.keys())
50      cmap, norm = get_colormap(cycle_numbers)
51
52      plt.figure(figsize=(15, 10))
53      ax = plt.gca()
54      for cycle_number, battery_data in data_dict.items():
55          color = cmap(norm(cycle_number))  # Get color based on cycle number
56          for battery_number, values in battery_data.items():
57              r_values = values['r_values']
58              i_values = values['i_values']
59              ax.scatter(r_values, -i_values, color=color, s=3, alpha=0.7)
60
61      sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
62      sm.set_array([])
63      plt.colorbar(sm, label='Cycle Number')
64      plt.xlabel(r'$-Z_{r}$()')
65      plt.ylabel(r'$-Z_{i}$()')
66      plt.title('Nyquist Plot for EIS 1 with Gradient Color for Cycles')
67      set_aspect_equal(ax)
68
69
70      ax.set_xlim(left=-0.015)
71      ax.set_ylim(bottom=-0.015)
72
73      plt.tight_layout()
74      plt.grid(True, alpha=0.4)
75      plt.show()
76
77
78  plot_with_gradient(data_by_cycle_battery)
79
80
```

B.2 Code for EIS versus Capacities

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from dcor import distance_correlation
from matplotlib import cm

plt.rcParams.update({'font.size': 30})

# Function to read CSV files
def read_csv(file_path):
    data = np.genfromtxt(file_path, delimiter=',', names=True, dtype=None, encoding=None)
    return data

# File paths
eis_csv_file = 'to_be_fed2.csv'
capacity_csv_file = 'for_pca.csv'

# Load data
results_array = read_csv(eis_csv_file)
capacity_data = pd.read_csv(capacity_csv_file)
capacity_data = capacity_data.rename(columns={'Unnamed: 0': 'Cycle'}).set_index('Cycle')
capacity_data.columns = capacity_data.columns.astype(int)  # Convert battery indices to integers

# Distinct EIS values and frequencies
distinct_hz_values = ['5015hz','3984hz','3170hz','2527hz','1976hz','1577hz','1265hz','998p2hz','796p9hz','627p8hz','495p8hz','398p0hz','315p5hz',
                      '252p4hz','198p6hz','158p4hz','125p6hz','100p4hz','79p00hz','62p78hz','49p87hz','39p72hz','31p67hz','24p93hz','19p86hz','15p84hz',
                      '12p40hz','9p931hz','7p945hz','6p317hz','5p008hz','3p946hz','2p504hz','1p998hz','1p585hz','1p267hz',
                      '0p999hz','0p794hz','0p633hz','0p504hz','0p397hz','0p317hz','0p251hz','0p199hz','0p159hz','0p126hz','0p100hz']
hz_value_mapping = {
    '5015hz': '5015 Hz',
    '3984hz': '3984 Hz',
    '3170hz': '3170 Hz',
    '2527hz': '2527 Hz',
    '1976hz': '1976 Hz',
    '1577hz': '1577 Hz',
    '1265hz': '1265 Hz',
    '998p2hz': '998.2 Hz',
    '796p9hz': '796.9 Hz',
    '627p8hz': '627.8 Hz',
    '495p8hz': '495.9 Hz',
    '398p0hz': '398.0 Hz',
    '315p5hz': '315.5 Hz',
    '252p4hz': '252.4 Hz',
    '198p6hz': '198.6 Hz',
    '158p4hz': '158.4 Hz',
    '125p6hz': '125.6 Hz',
    '100p4hz': '100.4 Hz',
    '79p00hz': '79.00 Hz',
    '62p78hz': '62.78 Hz',
    '49p87hz': '49.87 Hz',
    '39p72hz': '39.72 Hz',
```

```
    '31p67hz': '31.67 Hz',
    '24p93hz': '24.93 Hz',
    '19p86hz': '19.86 Hz',
    '15p84hz': '15.84 Hz',
    '12p40hz': '12.40 Hz',
    '9p931hz': '9.931 Hz',
    '7p945hz': '7.945 Hz',
    '6p317hz': '6.317 Hz',
    '5p008hz': '5.008 Hz',
    '3p946hz': '3.946 Hz',
    '3p159hz': '3.159 Hz',
    '2p504hz': '2.504 Hz',
    '1p998hz': '1.998 Hz',
    '1p585hz': '1.585 Hz',
    '1p267hz': '1.267 Hz',
    '0p999hz': '0.999 Hz',
    '0p794hz': '0.794 Hz',
    '0p633hz': '0.933 Hz',
    '0p504hz': '0.504 Hz',
    '0p397hz': '0.397 Hz',
    '0p317hz': '0.317 Hz',
    '0p251hz': '0.251 Hz',
    '0p199hz': '0.199 Hz',
    '0p159hz': '0.159 Hz',
    '0p126hz': '0.126 Hz',
    '0p100hz': '0.100 Hz'
}
distinct_eis_values = np.unique(results_array['eis'])

# Colors for distinct batteries
distinct_colors = [cm.RdBu(i / len(capacity_data.columns)) for i in range(len(capacity_data.columns))]

# Function to get capacity value for a specific cycle and battery
def get_capacity_value(cycle, battery_index):
    try:
        return capacity_data.loc[cycle, battery_index]
    except KeyError:
        return None

# Prepare data for plotting
plot_data = {
    eis: {
        hz: {
            'modulus_vs_capacity': [],
            'phase_vs_capacity': []
        } for hz in distinct_hz_values
    } for eis in distinct_eis_values
}

# Populate plot data
for eis_value in distinct_eis_values:
    filtered_eis = results_array[results_array['eis'] == eis_value]

    for hz_value in distinct_hz_values:
        modulus_data = filtered_eis[filtered_eis['data'] == 'mod']
        phase_data = filtered_eis[filtered_eis['data'] == 'phz']

        if modulus_data.size == 0 or phase_data.size == 0:
            continue
```

```
112
113         for battery_index in range(len(distinct_colors)):
114             battery_modulus_data = modulus_data[modulus_data['battery'] == battery_index + 1]
115             battery_phase_data = phase_data[phase_data['battery'] == battery_index + 1]
116
117             for cycle, mod_val in zip(battery_modulus_data['cycle'], battery_modulus_data[hz_value]):
118                 capacity = get_capacity_value(cycle, battery_index + 1)
119                 if capacity is not None:
120                     plot_data[eis_value][hz_value]['modulus_vs_capacity'].append((battery_index + 1, capacity, mod_val))
121
122             for cycle, phz_val in zip(battery_phase_data['cycle'], battery_phase_data[hz_value]):
123                 capacity = get_capacity_value(cycle, battery_index + 1)
124                 if capacity is not None:
125                     plot_data[eis_value][hz_value]['phase_vs_capacity'].append((battery_index + 1, capacity, phz_val))
126
127 # Compute distance correlation coefficients
128 all_modulus_corr = []
129 all_phase_corr = []
130
131 for eis_value in distinct_eis_values:
132     for hz_value in distinct_hz_values:
133         modulus_data = plot_data[eis_value][hz_value]['modulus_vs_capacity']
134         if modulus_data:
135             valid_data = [(cap, mod) for cap, mod in [(entry[1], entry[2]) for entry in modulus_data] if not (np.isnan(cap) or
                ↪  np.isnan(mod))]
136             if valid_data:
137                 capacities, moduli = map(np.array, zip(*valid_data))
138                 coefficient = distance_correlation(capacities, moduli)
139                 all_modulus_corr.append({'eis_value': eis_value, 'hz_value': hz_value, 'coefficient': coefficient})
140
141         phase_data = plot_data[eis_value][hz_value]['phase_vs_capacity']
142         if phase_data:
143             valid_data = [(cap, phz) for cap, phz in [(entry[1], entry[2]) for entry in phase_data] if not (np.isnan(cap) or
                ↪  np.isnan(phz))]
144             if valid_data:
145                 capacities_arr, phases_arr = map(np.array, zip(*valid_data))
146                 coefficient = distance_correlation(capacities_arr, phases_arr)
147                 all_phase_corr.append({
148                     'eis_value': eis_value,
149                     'hz_value': hz_value,
150                     'coefficient': coefficient,
151                     'capacities': capacities_arr,
152                     'phases': phases_arr
153                     })
154
155 # Print all coefficients
156 print("All Modulus Correlations:")
157 for entry in all_modulus_corr:
158     print(f"EIS {entry['eis_value']}, {hz_value_mapping[entry['hz_value']]}: {entry['coefficient']:.3f}")
159
160 print("\nAll Phase Correlations:")
161 for entry in all_phase_corr:
162     print(f"EIS {entry['eis_value']}, {hz_value_mapping[entry['hz_value']]}: {entry['coefficient']:.3f}")
163
164 # Plot data
165 for eis_value in distinct_eis_values:
166     for hz_value in distinct_hz_values:
167         # Find corresponding correlation coefficients for modulus and phase
168         modulus_corr = next(entry['coefficient'] for entry in all_modulus_corr
```

```
                        if entry['eis_value'] == eis_value and entry['hz_value'] == hz_value)
    phase_corr = next(entry['coefficient'] for entry in all_phase_corr
                      if entry['eis_value'] == eis_value and entry['hz_value'] == hz_value)

    fig_mod, ax_mod = plt.subplots(figsize=(15, 9), constrained_layout=True)
    fig_phase, ax_phase = plt.subplots(figsize=(15, 9), constrained_layout=True)


    for battery_index in range(len(distinct_colors)):
        mod_data = [(cap, mod) for b, cap, mod in plot_data[eis_value][hz_value]['modulus_vs_capacity'] if b == battery_index
        ↪  + 1]
        phase_data = [(cap, phz) for b, cap, phz in plot_data[eis_value][hz_value]['phase_vs_capacity'] if b == battery_index
        ↪  + 1]

        if mod_data:
            caps, mods = zip(*mod_data)
            ax_mod.scatter(caps, mods, color=distinct_colors[battery_index], label=f'Battery {battery_index + 1}' ,s=180,
            ↪  edgecolor='gray', linewidth=1)

        if phase_data:
            caps, phases = zip(*phase_data)
            ax_phase.scatter(caps, phases, color=distinct_colors[battery_index], label=f'Battery {battery_index + 1}' ,s=180,
            ↪  edgecolor='gray', linewidth=1)

    # Update plot titles to include correlation coefficients
    ax_mod.set_title(f"EIS {eis_value}, {hz_value_mapping[hz_value]} Modulus vs Capacity") #f"dCor Coeff.:
    ↪  {modulus_corr:.4f}")
    ax_mod.set_xlabel('Capacity')
    ax_mod.set_ylabel('Modulus')
    #ax_mod.legend()
    ax_mod.grid(True, alpha = 0.4)

    ax_phase.set_title(f"EIS {eis_value}, {hz_value_mapping[hz_value]} Phase vs Capacity") #f"dCor Coeff.: {phase_corr:.4f}")
    ax_phase.set_xlabel('Capacity')
    ax_phase.set_ylabel('Phase')
    #ax_phase.legend()
    ax_phase.grid(True, alpha = 0.4)

    plt.tight_layout()
    plt.show()
```

B.3 Code for Averaged ICI Results

```

import numpy as np
from scipy.optimize import curve_fit
from scipy.stats import linregress
import glob
import matplotlib.pyplot as plt
import json

results_data = []
```

```
plt.rcParams.update({'font.size': 18})

def CheckSoCDesired(SoC, rate_c_over):

    if rate_c_over == 10:
            return True

    elif rate_c_over == 5:
            return True

    elif rate_c_over == 20:
        return False

    else:
        print(datafile)
        print("Unknown Rate")
        return False


def filter_data(data, interval):
        min_val, max_val = interval
        min_val = float(min_val)
        max_val = float(max_val)
        filtered_data = {}
        for key, values in data.items():
            filtered_values = {SoC: value for SoC, value in values.items() if  min_val <= float(SoC) <= max_val}
            if filtered_values:
                filtered_data[key] = filtered_values
        return filtered_data

for datafile in glob.glob("**/*CHRONOP*.DTA", recursive=True):

    with open(datafile, 'r') as file:
        for line in file:
            if line.startswith("CURVE"):
                break
        else:
            print(datafile)
            print("No 'CURVE' row found in the file.")
            continue


        data = np.genfromtxt(file, dtype=[('Pt', 'i8'), ('T', 'f8'), ('Vf', 'f8'), ('Im', 'f8'), ('Vu', 'f8'), ('Sig', 'f8'),
                                          ('Ach', 'f8'), ('IERange', 'i8'), ('Over', 'S12')])

        SoC = data['Vf'][-1]
        C_over = int(datafile.split("cover")[1].split("_")[0])
        if data['Pt'][-1] >= 6001 and CheckSoCDesired(SoC, C_over):
            cycle = int(datafile.split("th")[0])
            battery = int(datafile.split("#")[1].split("_")[0])
            ICIData_t = data['T'][6002:] - data['T'][6002]
            ICIData_V = data['Vf'][6002:]
            result = linregress(ICIData_t**(1/2), ICIData_V)
            ICIcalc=np.zeros(len(ICIData_V))
            ICIcalc=result.slope*ICIData_t**(1/2)+result.intercept
            if(result.rvalue**2 < 0.2):
                plt.scatter(ICIData_t,ICIData_V,label='exp')
                plt.scatter(ICIData_t,ICIcalc,label='calc')
                plt.legend()
```

```
69                  plt.show()
70              delE = data['Vf'][6001] - result.intercept
71
72              if result.slope > 0:
73                  c_d = 'dch'
74              else:
75                  c_d = 'chg'
76
77              if C_over == 5:
78                  R = (delE / 0.560)
79                  k = (result.slope / 0.560)
80              elif C_over == 10:
81                  R = (2 * delE / 0.560)
82                  k = (2 * result.slope / 0.560)
83              else:
84                  continue
85
86              results_data.append((c_d, C_over, battery, cycle, SoC,
87                                   result.slope, result.intercept,
88                                   result.rvalue**2, delE, R, k))
89
90
91  dtype = [('c_d', 'U3'), ('C_over', 'int'), ('battery', 'int'),
92           ('cycle', 'int'), ('SoC', 'float'), ('slope', 'float'),
93           ('intercept', 'float'), ('R^2', 'float'), ('delE', 'float'),
94           ('R_value', 'float'), ('k_value', 'float')]
95
96  results_array = np.array(results_data, dtype=dtype)
97
98  soc1_c_5=np.empty(shape=(len(results_data)),dtype=dtype)
99  soc1_d_5=np.empty(shape=(len(results_data)),dtype=dtype)
100 soc2_c_5=np.empty(shape=(len(results_data)),dtype=dtype)
101 soc2_d_5=np.empty(shape=(len(results_data)),dtype=dtype)
102 soc3_c_5=np.empty(shape=(len(results_data)),dtype=dtype)
103 soc3_d_5=np.empty(shape=(len(results_data)),dtype=dtype)
104 soc1_c_10=np.empty(shape=(len(results_data)),dtype=dtype)
105 soc1_d_10=np.empty(shape=(len(results_data)),dtype=dtype)
106 soc2_c_10=np.empty(shape=(len(results_data)),dtype=dtype)
107 soc2_d_10=np.empty(shape=(len(results_data)),dtype=dtype)
108 soc3_c_10=np.empty(shape=(len(results_data)),dtype=dtype)
109 soc3_d_10=np.empty(shape=(len(results_data)),dtype=dtype)
110
111 n_SoC1=n_SoC2=n_SoC3=n_SoC4=n_SoC5=n_SoC6=n_SoC7=n_SoC8=n_SoC9=n_SoC10=n_SoC11=n_SoC12=0
112
113 for row in results_array:
114     if row['SoC'] < 3.4:
115         if row['c_d'] == 'chg' and row['C_over'] == 5:
116             soc1_c_5[n_SoC1] = row.copy()
117             n_SoC1=n_SoC1+1
118         elif row['c_d'] == 'dch' and row['C_over'] == 5:
119             soc1_d_5[n_SoC2] = row.copy()
120             n_SoC2=n_SoC2+1
121         elif row['c_d'] == 'chg' and row['C_over'] == 10:
122             soc1_c_10[n_SoC3] = row.copy()
123             n_SoC3=n_SoC3+1
124         elif row['c_d'] == 'dch' and row['C_over'] == 10:
125             soc1_d_10[n_SoC4] = row.copy()
126             n_SoC4=n_SoC4+1
127         else: pass
```

```
128         elif 3.5 < row['SoC'] < 3.6:
129             if row['c_d'] == 'chg' and row['C_over'] == 5:
130                 soc2_c_5[n_SoC5] = row.copy()
131                 n_SoC5=n_SoC5+1
132             elif row['c_d'] == 'dch' and row['C_over'] == 5:
133                 soc2_d_5[n_SoC6] = row.copy()
134                 n_SoC6=n_SoC6+1
135             elif row['c_d'] == 'chg' and row['C_over'] == 10:
136                 soc2_c_10[n_SoC7] = row.copy()
137                 n_SoC7=n_SoC7+1
138             elif row['c_d'] == 'dch' and row['C_over'] == 10:
139                 soc2_d_10[n_SoC5] = row.copy()
140                 n_SoC8=n_SoC8+1
141             else: pass
142         elif 3.8 < row['SoC'] < 3.9:
143             if row['c_d'] == 'chg' and row['C_over'] == 5:
144                 soc3_c_5[n_SoC9] = row.copy()
145                 n_SoC9=n_SoC9+1
146             elif row['c_d'] == 'dch' and row['C_over'] == 5:
147                 soc3_d_5[n_SoC10] = row.copy()
148                 n_SoC10=n_SoC10+1
149             elif row['c_d'] == 'chg' and row['C_over'] == 10:
150                 soc3_c_10[n_SoC11] = row.copy()
151                 n_SoC11=n_SoC11+1
152             elif row['c_d'] == 'dch' and row['C_over'] == 10:
153                 soc3_d_10[n_SoC12] = row.copy()
154                 n_SoC12=n_SoC12+1
155             else: pass
156 
157         else: pass
158 
159     soc1_c_5 = soc1_c_5[:n_SoC1 - 1]
160     soc1_d_5 = soc1_d_5[:n_SoC2 - 1]
161     soc2_c_5 = soc2_c_5[:n_SoC5 - 1]
162     soc2_d_5 = soc2_d_5[:n_SoC6 - 1]
163     soc3_c_5 = soc3_c_5[:n_SoC9 - 1]
164     soc3_d_5 = soc3_d_5[:n_SoC10 - 1]
165     soc1_c_10 = soc1_c_10[:n_SoC3 - 1]
166     soc1_d_10 = soc1_d_10[:n_SoC4 - 1]
167     soc2_c_10 = soc2_c_10[:n_SoC7 - 1]
168     soc2_d_10 = soc2_d_10[:n_SoC8 - 1]
169     soc3_c_10 = soc3_c_10[:n_SoC11 - 1]
170     soc3_d_10 = soc3_d_10[:n_SoC12 - 1]
171 
172         # Define colors for different batteries
173     battery_colors = {
174             1: '#65001b',
175             2: '#c33034',
176             3: '#f49b78',
177             4: '#fde5d6',
178             5: '#ddebf4',
179             6: '#85bfdc',
180             7: '#2d77b4',
181             8: '#002a5e'
182         }
183 
184     from collections import defaultdict
185     avg_R_values = defaultdict(list)
186     avg_k_values = defaultdict(list)
```

```
arrays = {'soc1_c_5': soc1_c_5, 'soc1_d_5': soc1_d_5, 'soc2_c_5': soc2_c_5, 'soc2_d_5': soc2_d_5,
          'soc3_c_5': soc3_c_5, 'soc3_d_5': soc3_d_5, 'soc1_c_10': soc1_c_10, 'soc1_d_10': soc1_d_10,
          'soc2_c_10': soc2_c_10, 'soc2_d_10': soc2_d_10, 'soc3_c_10': soc3_c_10, 'soc3_d_10': soc3_d_10}

for array_name, array in arrays.items():


    max_cycle = max(row['cycle'] for row in array) + 1
    for cycle in range(max_cycle):
        for battery in range(1, max(row['battery'] for row in array) + 1):
            R_values = [row['R_value'] for row in array if row['cycle'] == cycle and row['battery'] == battery and row['R_value']
            ↪  != 0]
            k_values = [row['k_value'] for row in array if row['cycle'] == cycle and row['battery'] == battery and row['k_value']
            ↪  != 0]
            avg_R_value = sum(R_values) / len(R_values) if R_values and None not in R_values else None
            avg_k_value = sum(k_values) / len(k_values) if k_values and None not in k_values else None
            if avg_R_value is not None:
                avg_R_values[(array_name, cycle, battery)] = avg_R_value
            if avg_k_value is not None:
                avg_k_values[(array_name, cycle, battery)] = avg_k_value


def plot_array_R_data(array_name, graph_label, avg_R_values):
    cycles_R = []
    avg_R_values_list = []
    batteries_R = []

    for key, avg_R_value in avg_R_values.items():
        if key[0] == array_name:
            cycles_R.append(key[1])
            batteries_R.append(key[2])
            avg_R_values_list.append(avg_R_value)
            plt.figure(figsize=(9, 5))

    for battery in set(batteries_R):
        indices = [i for i, b in enumerate(batteries_R) if b == battery]
        plt.scatter([cycles_R[i] for i in indices], [avg_R_values_list[i] for i in indices], label=f'Battery {battery}',
                    color=battery_colors[battery], s=50, edgecolor='gray', linewidth=1)

    plt.xlabel('Cycle')
    plt.ylabel('Average R value ($\Omega)')
    plt.title(f'Average R value vs. Cycle - {graph_label}')
    #plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.grid(True, alpha=0.4)
    plt.show()


def plot_array_k_data(array_name, graph_label, avg_k_values):
    cycles_k = []
    avg_k_values_list = []
    batteries_k = []


    for key, avg_k_value in avg_k_values.items():
        if key[0] == array_name:
            cycles_k.append(key[1])
            batteries_k.append(key[2])
            avg_k_values_list.append(avg_k_value)
            plt.figure(figsize=(9, 5))
```

```
244
245
246         for battery in set(batteries_k):
247             indices = [i for i, b in enumerate(batteries_k) if b == battery]
248             plt.scatter([cycles_k[i] for i in indices], [avg_k_values_list[i] for i in indices], label=f'Battery {battery}',
                ↪  color=battery_colors[battery], s=50, edgecolor='gray', linewidth=1)
249
250         plt.xlabel('Cycle')
251         plt.ylabel(r'Average $k$ value ( $\Omega \, s^{-0.5}$ )')
252         plt.title(f'Average k value vs. Cycle - {graph_label}')
253         #plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
254         plt.grid(True, alpha=0.4)
255         plt.show()
256
257
258     R_results = {}
259
260     for (group, index, sub_index), value in avg_R_values.items():
261         key = f"('{group}', {sub_index})"
262         if key not in R_results:
263             R_results[key] = []
264         R_results[key].append([index, value])
265
266     with open('R_results.json', 'w') as json_file:
267         json.dump(R_results, json_file, separators=(',', ':'), indent=None)
268
269
270     k_results = {}
271
272     for (group, index, sub_index), value in avg_k_values.items():
273         key = f"('{group}', {sub_index})"
274         if key not in k_results:
275             k_results[key] = []
276         k_results[key].append([index, value])
277
278
279     with open('k_results.json', 'w') as json_file:
280         json.dump(k_results, json_file, separators=(',', ':'), indent=None)
281
282
283     plot_array_R_data('soc1_c_5', 'SoC 3.0V-3.4V, Charging, C/5', avg_R_values)
284     plot_array_R_data('soc1_d_5', 'SoC 3.0V-3.4V, Discharging, C/5', avg_R_values)
285     plot_array_R_data('soc2_c_5', 'SoC 3.5V-3.6V, Charging, C/5', avg_R_values)
286     plot_array_R_data('soc2_d_5', 'SoC 3.5V-3.6V, Discharging, C/5', avg_R_values)
287     plot_array_R_data('soc3_c_5', 'SoC 3.8V-3.9V, Charging, C/5', avg_R_values)
288     plot_array_R_data('soc3_d_5', 'SoC 3.8V-3.9V, Discharging, C/5', avg_R_values)
289     plot_array_R_data('soc1_c_10', 'SoC 3.0V-3.4V, charging, C/10', avg_R_values)
290     plot_array_R_data('soc1_d_10', 'SoC 3.0V-3.4V, discharging, C/10', avg_R_values)
291     plot_array_R_data('soc2_c_10', 'SoC 3.5V-3.6V, charging, C/10', avg_R_values)
292     plot_array_R_data('soc2_d_10', 'SoC 3.5V-3.6V, discharging, C/10', avg_R_values)
293     plot_array_R_data('soc3_c_10', 'SoC 3.8V-3.9V, charging, C/10', avg_R_values)
294     plot_array_R_data('soc3_d_10', 'SoC 3.8V-3.9V, discharging, C/10', avg_R_values)
295
296     plot_array_k_data('soc1_c_5', 'SoC 3.0V-3.4V, Charging, C/5', avg_k_values)
297     plot_array_k_data('soc1_d_5', 'SoC 3.0V-3.4V, Discharging, C/5', avg_k_values)
298     plot_array_k_data('soc2_c_5', 'SoC 3.5V-3.6V, Charging, C/5', avg_k_values)
299     plot_array_k_data('soc2_d_5', 'SoC 3.5V-3.6V, Discharging, C/5', avg_k_values)
300     plot_array_k_data('soc3_c_5', 'SoC 3.8V-3.9V, Charging, C/5', avg_k_values)
301     plot_array_k_data('soc3_d_5', 'SoC 3.8V-3.9V, Discharging, C/5', avg_k_values)
```

```
plot_array_k_data('soc1_c_10', 'SoC 3.0V-3.4V, charging, C/10', avg_k_values)
plot_array_k_data('soc1_d_10', 'SoC 3.0V-3.4V, discharging, C/10', avg_k_values)
plot_array_k_data('soc2_c_10', 'SoC 3.5V-3.6V, charging, C/10', avg_k_values)
plot_array_k_data('soc2_d_10', 'SoC 3.5V-3.6V, discharging, C/10', avg_k_values)
plot_array_k_data('soc3_c_10', 'SoC 3.8V-3.9V, charging, C/10', avg_k_values)
plot_array_k_data('soc3_d_10', 'SoC 3.8V-3.9V, discharging, C/10', avg_k_values)
```

B.4 Code for Implementation of dCor

```
import csv
import json
import matplotlib.pyplot as plt
from scipy.stats import linregress
import dcor
import numpy as np


aa_capacity_data = {}
plt.rcParams.update({'font.size': 13})

with open('for_pca.csv', 'r') as file:
    reader = csv.reader(file)

    header = next(reader)
    batteries = header[1:]

    for battery in batteries:
        aa_capacity_data[battery] = []

    for row in reader:
        cycle_number = int(row[0])
        values = row[1:]

        for i, value in enumerate(values):
            if value.strip():
                aa_capacity_data[batteries[i]].append((cycle_number, float(value)))
            else:
                aa_capacity_data[batteries[i]].append((cycle_number, None))

def process_json_file(json_filename, prefix):
    data_array = {}
    with open(json_filename, 'r') as file:
        json_data = json.load(file)

    for key, value in json_data.items():
        data_type, battery_number = eval(key)
        prefixed_data_type = f"{prefix}{data_type}"
        if prefixed_data_type not in data_array:
            data_array[prefixed_data_type] = {battery: [] for battery in range(1, 9)}
        for cycle, val in value:
            data_array[prefixed_data_type][battery_number].append((int(cycle), float(val)))
    return data_array

```

```
aa_k_value = process_json_file('k_results.json', 'k_')
aa_R_value = process_json_file('R_results.json', 'R_')

grouped_data = {}
cycle_range = range(0, 1151, 50)

for battery, capacities in aa_capacity_data.items():
    for test_name, k_results in aa_k_value.items():
        if test_name not in grouped_data:
            grouped_data[test_name] = []
        for cycle in cycle_range:
            cap_value = next((cap for (cyc, cap) in capacities if cyc == cycle), None)
            k_result = next((val for (cyc, val) in k_results[int(battery)] if cyc == cycle), None)
            if cap_value is not None and k_result is not None:
                grouped_data[test_name].append((int(battery), cycle, cap_value, k_result))
    for test_name, R_results in aa_R_value.items():
        if test_name not in grouped_data:
            grouped_data[test_name] = []
        for cycle in cycle_range:
            cap_value = next((cap for (cyc, cap) in capacities if cyc == cycle), None)
            R_result = next((val for (cyc, val) in R_results[int(battery)] if cyc == cycle), None)
            if cap_value is not None and R_result is not None:
                grouped_data[test_name].append((int(battery), cycle, cap_value, R_result))



def calculate_statistics(grouped_data, prefix):
    p_values = {}
    distance_correlations = {}

    for test_name, data in grouped_data.items():
        if test_name.startswith(prefix):
            combined_x = []
            combined_y = []
            for _, _, cap, val in data:
                combined_x.append(cap)
                combined_y.append(val)
            if len(combined_x) > 1:
                _, _, r_value, p_value, _ = linregress(combined_x, combined_y)
                p_values[test_name] = p_value

                combined_x = np.array(combined_x)
                combined_y = np.array(combined_y)

                distance_corr = dcor.distance_correlation(combined_x, combined_y)
                distance_correlations[test_name] = distance_corr
    return p_values, distance_correlations

k_p_values, k_distance_corrs = calculate_statistics(grouped_data, 'k_')
R_p_values, R_distance_corrs = calculate_statistics(grouped_data, 'R_')

sorted_k_tests = sorted(k_distance_corrs, key=k_distance_corrs.get, reverse=True)
sorted_R_tests = sorted(R_distance_corrs, key=R_distance_corrs.get, reverse=True)

top_k_tests = sorted_k_tests
top_R_tests = sorted_R_tests

print("k-tests:")
for test_name in sorted_k_tests:
```

```
    print(f"{test_name}: Distance Correlation = {k_distance_corrs[test_name]:.3f}, p-value = {k_p_values[test_name]:.2e}")

print("\nR-tests:")
for test_name in sorted_R_tests:
    print(f"{test_name}: Distance Correlation = {R_distance_corrs[test_name]:.3f}, p-value = {R_p_values[test_name]:.2e}")

top_tests = top_k_tests + top_R_tests
cmap = plt.cm.get_cmap('RdBu', 8)
battery_colors = [cmap(i) for i in range(8)]

fig, axs = plt.subplots(12, 2, figsize=(16, 56))

def plot_test_data(test_name, ax, p_values, distance_corrs):
    if test_name in grouped_data:
        data = grouped_data[test_name]
        for battery in range(1, 9):
            battery_data = [(cap, val) for bat, cyc, cap, val in data if bat == battery]
            if battery_data:
                capacities, test_values = zip(*battery_data)
                ax.scatter(
                    capacities, test_values, label=f'Battery {battery}',
                    color=battery_colors[battery - 1], s=60, edgecolor='gray', linewidth=1
                )

        plot_title = ''
        y_axis_label = ''

        if 'k' in test_name:
            plot_title += 'Avg. k values'
            y_axis_label = 'Avg k value'
        elif 'R' in test_name:
            plot_title += 'Avg. R values'
            y_axis_label = 'Avg R value'

        voltage_condition = ''
        charging_condition = ''
        c_rate_condition = ''

        if 'soc1' in test_name:
            voltage_condition = 'at low voltage'
        elif 'soc2' in test_name:
            voltage_condition = 'at mid voltage'
        elif 'soc3' in test_name:
            voltage_condition = 'at high voltage'

        if 'c' in test_name:
            charging_condition = 'charging'
        elif 'd' in test_name:
            charging_condition = 'discharging'

        c_rate = test_name.split('_')[-1]
        c_rate_condition = f'C/{c_rate}'

        full_title = f"{plot_title} {voltage_condition}, {charging_condition}, {c_rate_condition}"

        ax.set_title(f"{full_title} (dCorr={distance_corrs[test_name]:.3f}, p={p_values[test_name]:.3g})")
        ax.set_xlabel('Capacity (Ah)')
        ax.set_ylabel(y_axis_label)
        ax.grid(True, alpha=0.4)
```

```
163
164
165 for i, test_name in enumerate(top_k_tests):
166     plot_test_data(test_name, axs[i, 0], k_p_values, k_distance_corrs)
167
168 for i, test_name in enumerate(top_R_tests):
169     plot_test_data(test_name, axs[i, 1], R_p_values, R_distance_corrs)
170
171 plt.tight_layout()
172 plt.show()
```

B.5 Code for Binning Plots and Results

```
1  import csv
2  import matplotlib.pyplot as plt
3  from scipy.stats import linregress
4  import dcor
5  import numpy as np
6  import matplotlib.patches as patches
7
8  aa_capacity_data = {}
9  with open('for_pca.csv', 'r') as file:
10     reader = csv.reader(file)
11     header = next(reader)
12     batteries = header[1:]
13     for battery in batteries:
14         aa_capacity_data[battery] = []
15     for row in reader:
16         cycle_number = int(row[0])
17         values = row[1:]
18         for i, value in enumerate(values):
19             if value.strip():
20                 aa_capacity_data[batteries[i]].append((cycle_number, float(value)))
21             else:
22                 aa_capacity_data[batteries[i]].append((cycle_number, None))
23
24 k_values = {}
25 R_values = {}
26
27 def parse_row(row):
28     return {k.strip(): v.strip() for k, v in row.items()}
29
30 with open('avg_values_errors.txt', 'r') as f:
31     reader = csv.DictReader(f)
32     for row in reader:
33         row = parse_row(row)
34         arr = row['Array_Name']
35         cyc = int(row['Cycle'])
36         bat = int(row['Battery'])
37         key = (arr, cyc, bat)
38         # Use NaN if data is missing
39         k_val = float(row['Avg_k_Value']) if row['Avg_k_Value'] != "nan" else np.nan
40         k_sem = float(row['SEM_k']) if row['SEM_k'] != "nan" else np.nan
```

```
41          R_val = float(row['Avg_R_Value']) if row['Avg_R_Value'] != "nan" else np.nan
42          R_sem = float(row['SEM_R']) if row['SEM_R'] != "nan" else np.nan
43          k_values[key] = (k_val, k_sem)
44          R_values[key] = (R_val, R_sem)
45
46  grouped_data = {}
47  cycle_range = range(0, 1151, 50)
48  for battery, capacities in aa_capacity_data.items():
49      battery_num = int(battery)
50      for arr in set([k[0] for k in k_values] + [r[0] for r in R_values]):
51          k_test_name = f'k_{arr}'
52          R_test_name = f'R_{arr}'
53          grouped_data.setdefault(k_test_name, [])
54          grouped_data.setdefault(R_test_name, [])
55
56          for cycle in cycle_range:
57              key = (arr, cycle, battery_num)
58              cap_tuple = next((cap for (cyc, cap) in capacities if cyc == cycle), None)
59
60              if cap_tuple is None:
61                  continue
62
63              if key in k_values:
64                  k_val, k_sem = k_values[key]
65                  grouped_data[k_test_name].append((battery_num, cycle, cap_tuple, k_val, k_sem))
66              if key in R_values:
67                  r_val, r_sem = R_values[key]
68                  grouped_data[R_test_name].append((battery_num, cycle, cap_tuple, r_val, r_sem))
69
70  def calculate_statistics(grouped_data):
71      p_values = {}
72      distance_correlations = {}
73      for test_name, data in grouped_data.items():
74          x_vals = np.array([x[2] for x in data])
75          y_vals = np.array([x[3] for x in data])
76          mask = (~np.isnan(x_vals)) & (~np.isnan(y_vals))
77          x_vals = x_vals[mask]
78          y_vals = y_vals[mask]
79          if len(x_vals) > 1 and len(y_vals) > 1:
80              _, _, _, p_value, _ = linregress(x_vals, y_vals)
81              p_values[test_name] = p_value
82              distance_correlations[test_name] = dcor.distance_correlation(x_vals, y_vals)
83      return p_values, distance_correlations
84
85  p_values, distance_corrs = calculate_statistics(grouped_data)
86
87  def build_sorted_data(data):
88      valid_y_vals = [val for _, _, _, val, _ in data if not np.isnan(val)]
89      avg_y = np.mean(valid_y_vals)
90      avg_sem = np.mean([sem for _, _, _, val, sem in data if not np.isnan(val)])
91
92      return sorted([
93          {
94              'battery': bat,
95              'cycle': cyc,
96              'x': cap,
97              'y': val if not np.isnan(val) else avg_y,
98              '_y': sem if not np.isnan(sem) else avg_sem,
99              'is_estimated': np.isnan(val)
```

```
        }
        for bat, cyc, cap, val, sem in data
        if cap is not None and not np.isnan(cap)
    ], key=lambda d: d['y'])

def compute_highlight_boxes(sorted_data):
    if not sorted_data:
        return []

    valid_y_vals = [pt['y'] for pt in sorted_data if not pt['is_estimated']]
    if not valid_y_vals:
        return []

    y_min = min(valid_y_vals)
    y_max = max(valid_y_vals)

    valid_sems = [pt['_y'] for pt in sorted_data if not pt['is_estimated'] and not np.isnan(pt['_y'])]

    if valid_sems:
        avg_y_err = np.mean(valid_sems)
        bin_height = 1 * avg_y_err
    else:
        return []

    bins = np.arange(y_min, y_max + bin_height, bin_height)
    if len(bins) < 2:
        return []

    boxes = []

    for i in range(len(bins) - 1):
        bin_low = bins[i]
        bin_high = bins[i + 1]
        bin_group = [pt for pt in sorted_data if bin_low <= pt['y'] < bin_high]

        if not bin_group:
            continue

        x_min = min(pt['x'] for pt in bin_group)
        x_max = max(pt['x'] for pt in bin_group)
        boxes.append((x_min, x_max, bin_low, bin_high))

    return boxes


def draw_highlight_boxes(ax, boxes):
    for x_min, x_max, y_min, y_max in boxes:
        ax.add_patch(patches.Rectangle(
            (x_min, y_min),
            x_max - x_min,
            y_max - y_min,
            facecolor='green', alpha=0.25, edgecolor='black', lw=1.5
        ))

cmap = plt.cm.get_cmap('RdBu', 8)
battery_colors = [cmap(i) for i in range(8)]
fig, axs = plt.subplots(12, 2, figsize=(16, 56))
all_tests = list(distance_corrs.keys())

```

```
159 def plot_test_data(test_name, ax):
160     raw_data = grouped_data[test_name]
161     sorted_data = build_sorted_data(raw_data)
162     boxes = compute_highlight_boxes(sorted_data)
163     draw_highlight_boxes(ax, boxes)
164 
165     for battery in range(1, 9):
166         full_data = [d for d in sorted_data if d['battery'] == battery]
167 
168         if not full_data:
169             continue
170 
171         x_real, y_real, yerr_real = zip(*[
172             (d['x'], d['y'], d['_y']) for d in full_data if not d['is_estimated']
173         ]) if any(not d['is_estimated'] for d in full_data) else ([], [], [])
174 
175         x_est, y_est, yerr_est = zip(*[
176             (d['x'], d['y'], d['_y']) for d in full_data if d['is_estimated']
177         ]) if any(d['is_estimated'] for d in full_data) else ([], [], [])
178 
179         if x_real:
180             ax.errorbar(
181                 x_real, y_real, yerr=yerr_real,
182                 fmt='o', color=battery_colors[battery - 1], ecolor='gray',
183                 elinewidth=1.2, capsize=3, label=f'Battery {battery}'
184             )
185         if x_est:
186             ax.errorbar(
187                 x_est, y_est, yerr=yerr_est,
188                 fmt='o', markerfacecolor='none', markeredgecolor=battery_colors[battery - 1],
189                 ecolor='lightgray', elinewidth=1.2, capsize=3
190             )
191 
192     plot_title = ''
193     y_axis_label = 'Value'
194 
195     if 'k' in test_name:
196         plot_title += 'Avg. k values'
197         y_axis_label = 'Avg k value'
198     elif 'R' in test_name:
199         plot_title += 'Avg. R values'
200         y_axis_label = 'Avg R value'
201 
202     voltage_condition = ''
203     charging_condition = ''
204     c_rate_condition = ''
205 
206     if 'soc1' in test_name:
207         voltage_condition = 'at low voltage'
208     elif 'soc2' in test_name:
209         voltage_condition = 'at mid voltage'
210     elif 'soc3' in test_name:
211         voltage_condition = 'at high voltage'
212 
213     if '_c_' in test_name:
214         charging_condition = 'charging'
215     elif '_d_' in test_name:
216         charging_condition = 'discharging'
217 
```

```

    c_rate = test_name.split('_')[-1]
    c_rate_condition = f'C/{c_rate} rate'

    full_title = f"{plot_title} {voltage_condition}, {charging_condition}, {c_rate_condition}"

     # Set plot titles and labels
    ax.set_title(f"{full_title} (dCorr={distance_corrs[test_name]:.3f}, p={p_values[test_name]:.3g})")
    ax.set_xlabel('Capacity (Ah)')
    ax.set_ylabel(y_axis_label)
    ax.grid(True, alpha=0.4)
    ax.legend(fontsize=8)

for idx, test_name in enumerate(all_tests[:24]):
    plot_test_data(test_name, axs.flat[idx])

plt.tight_layout()
plt.show()

print("\n=== Average X-span of Highlight Boxes ===")
for test_name in all_tests[:24]:
    raw_data = grouped_data[test_name]
    sorted_data = build_sorted_data(raw_data)
    boxes = compute_highlight_boxes(sorted_data)

    if boxes:
        x_spans = [x_max - x_min for x_min, x_max, _, _ in boxes]
        avg_span = np.mean(x_spans)
        print(f"x_err span for test {test_name} = {avg_span:.4f}")
    else:
        print(f"x_err span for test {test_name} = No boxes")

```

B.6 Code for Implementation of KK

```

import numpy as np
import matplotlib.pyplot as mpl
import os

PI = 3.1415926

frequencies = np.array([
    4973.323, 3993.056, 3144.055, 2519.531, 2000.762, 1582.031, 1253.634,
    1000.702, 790.0281, 628.811, 502.2321, 395.3313, 315.5048, 249.3351,
    198.6229, 158.0056, 125.558, 99.734, 79.44915, 62.91946, 49.86702,
    38.42213, 31.25, 24.93351, 20.03205, 15.625, 12.46675, 9.9734, 7.971939,
    6.351626, 5.013369, 3.985969, 3.175813, 2.485419, 1.99553, 1.583615,
    1.258052, 0.997765, 0.7927448, 0.6300403, 0.5008013, 0.3985969, 0.316723,
    0.2514753, 0.1992985, 0.1583615, 0.1257377, 0.099819
])

downsample = 1
```

```
batteries_count = 8

result_lines = []

for eis_idx in range(1, 6):
    folder_name = f"eis{eis_idx} nyq"
    if not os.path.isdir(folder_name):
        continue

    for cyc in range(0, 1151):
        file_path = os.path.join(folder_name, f"n_cyc{cyc}.csv")
        if not os.path.isfile(file_path):
            continue

        try:
            data = np.genfromtxt(file_path, delimiter=',', skip_header=1)

            line = f"eis {eis_idx}, cycle {cyc} results"
            fig, axes = mpl.subplots(4, 2, figsize=(14, 14))
            fig.suptitle(f"EIS {eis_idx}, Cycle {cyc}", fontsize=16)

            for battery in range(batteries_count):
                Zreal = data[battery * 2][3:]
                Zimag = data[battery * 2 + 1][3:]

                Zreal_downsampled = Zreal[::downsample]
                Zimag_downsampled = Zimag[::downsample]
                freq_downsampled = frequencies[::downsample]

                Tcarray = 1.0 / (2 * PI * freq_downsampled)
                dfreqarray = 2 * PI * freq_downsampled
                dimarray = Zimag_downsampled

                matrix = np.zeros((len(Tcarray), len(dfreqarray)))
                for i in range(len(Tcarray)):
                    for j in range(len(dfreqarray)):
                        matrix[i, j] = dfreqarray[j] * Tcarray[i] / (1 + (dfreqarray[j] * Tcarray[i])**2)

                Rk = np.linalg.solve(matrix, dimarray)

                Zresol = np.zeros(len(frequencies))
                for i in range(len(frequencies)):
                    for k in range(len(Tcarray)):
                        Zresol[i] += Rk[k] / (1 + (Tcarray[k] * 2 * PI * frequencies[i])**2)

                residuals = (Zreal_downsampled + Zresol[:len(freq_downsampled)] - np.average(Zreal_downsampled +
                ↪  Zresol[:len(freq_downsampled)])) / Zreal_downsampled
                avg_residual = np.mean(np.abs(residuals))

                line += f" b{battery + 1}: {avg_residual:.4f}"

                # Plot for current battery in grouped figure
                ax = axes[battery // 2][battery % 2]
                ax.semilogx(freq_downsampled, Zreal_downsampled, 'x-', label="Measurement")
                ax.semilogx(freq_downsampled, -1 * Zresol[:len(freq_downsampled)] + np.average(Zreal_downsampled +
                ↪  Zresol[:len(freq_downsampled)]), 'o-', label="Fit")
                ax.set_title(f"Battery {battery + 1}")
                ax.set_ylabel("Z'")
                ax.legend()
```

```
76                  ax.grid(True)
77
78              mpl.tight_layout(rect=[0, 0, 1, 0.96])
79              mpl.show()
80
81              print(line)
82              result_lines.append(line)
83
84          except Exception as e:
85              print(f"Skipping {file_path} due to error: {e}")
86
87  # Write results to a text file
88  with open("results.txt", "w") as f:
89      f.write("\n".join(result_lines))
90
```