

INTERROGATING BULK AND INTERFACIAL DECAY PATHWAYS IN
ELECTROACTIVE BATTERY MATERIALS

BY
KENNETH EARL MADSEN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Chemistry
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Doctoral Committee:

Professor Andrew A. Gewirth, Chair
Professor Ralph G. Nuzzo
Professor Nancy R. Sottos
Associate Professor Joaquín Rodríguez-López

ABSTRACT

The objectives of this thesis are twofold. Firstly, this thesis aims to interrogate the chemical, electrochemical, and mechanical pathways affecting the performance of battery materials on repeated charging and discharging. Secondly, it aims to develop approaches to suppress or circumvent these deleterious processes in the interest of developing redox active materials with adequate safety and longevity for practical application. The work contained herein addresses a range of materials including inorganic, organic, and organometallic solids and discusses their efficacies as electrodes and electrolytes for lithium-based energy storage. Both the bulk electrochemical behaviors and the interfacial dynamics of these materials are studied, with the goal of fully understanding the myriad factors that dictate battery stability. For clarity this thesis is divided into chapters based on the chemistry under investigation. A brief description of each chapter's contents is provided below.

Origin of Enhanced Cyclability in Covalently Modified $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ Cathodes. High-voltage lithium-ion cathode materials exhibit exceptional energy densities; however, rapid capacity fade during cell cycling prohibits their widespread utilization. Surface modification of cathode-active materials by organic self-assembled monolayers (SAMs) has emerged as an approach to improve the longevity of high-voltage electrodes; however, the surface chemistry at the electrode/electrolyte interphase and its dependence on monolayer structure remains unclear. Herein, we investigate the interplay between monolayer structure, electrochemical performance, and surface chemistry of high-voltage $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ (LMNO) electrodes by the application of silane-based SAMs of variable length and chemical composition. We demonstrate that the application of both hydrophobic and hydrophilic monolayers results in improved galvanostatic capacity retention relative to unmodified LMNO. The extent of this improvement is tied to the

structure of the monolayer with fluorinated alkyl-silanes exhibiting the greatest overall capacity retention, above 96% after 100 charge/discharge cycles. Postmortem surface analysis reveals that the presence of the monolayer enhances the deposition of LiF at the electrode surface during cell cycling and that the total surface concentration correlates with the overall improvements in capacity retention. We propose that the enhanced deposition of highly insulating LiF increases the anodic stability of the interphase, contributing to the improved galvanostatic performance of modified electrodes. Moreover, this work demonstrates that the modification of the electrode surface by the selection of an appropriate monolayer is an effective approach to tune the properties and behavior of the electrode/electrolyte interphase formed during battery operation.

Direct Observation of Interfacial Mechanical Failure in Thiophosphate Solid Electrolytes with Operando X-Ray Tomography. Herein, the mechanical behaviors of $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$ (LGPS) solid electrolytes during electrochemical cycling using operando X-ray tomography are investigated. It is demonstrated that the bulk mechanical decomposition of LGPS when cycled against lithium is a direct result of electrochemical reduction of the solid electrolyte at the LGPS/ Li_0 interface. The reductive decomposition of LGPS during lithium plating results in the formation of low-density domains at the electrode/electrolyte interface, which impose sufficient mechanical stress on the underlying LGPS to crack the SE pellet. The critical stress developed prior to pellet fracture is significantly lower than the bulk shear modulus of LGPS, suggesting that the electrochemical instability of LGPS dramatically worsens the mechanical stability of the material near the LGPS/ Li_0 interface. It is also shown that the application of a highly concentrated liquid electrolyte to the LGPS surface suppresses the reductive decomposition of LGPS, improving both the electrochemical performance and mechanical stability of the bulk LGPS solid electrolyte.

Pressure Dependent Electrochemical Behavior of Di-Lithium Rhodizonate Cathodes.

Herein, we investigate the electrochemical properties of the high-capacity organic cathode material di-lithium rhodizonate ($\text{Li}_2\text{C}_6\text{O}_6$) under different applied mechanical loads. We demonstrate, through a combination of pressure-dependent voltammetry and electrochemical impedance spectroscopy, that the charge-transfer kinetics at the cathode/electrolyte interface is strongly impacted by the magnitude of the load applied to the cathode. At low pressures, lithium rhodizonate displays untenably high charge-transfer impedances toward lithiation and delithiation. As the load applied to the cathode material is increased, the charge-transfer impedance decreases, reflecting a reduction in the overpotential associated with lithiation and delithiation. Furthermore, pressure-dependent galvanostatic cycling reveals that cells cycled at high pressures exhibit improvements in their overall capacity retention when compared with their low-pressure counterparts. Using a combination of postmortem X-ray diffraction and first-principles calculations, we show that, in the absence of sufficient external load, lithium rhodizonate converts from its redox-active structure to a redox-inactive structure, resulting in the observed rapid capacity fade. As the pressure applied to the electrode is increased, this phase transition is suppressed, resulting in improvements in both long-term stability and electrochemical kinetics.

Demonstration of a Stable Redox-Active Metal Rhodizonate Cathode Material Through Crystal Structure Modification. In this report we describe the electrochemical behavior of several divalent metal rhodizonate complexes ($\text{M(II)C}_6\text{O}_6$, where M is either a first-row transition metal [Mn, Co, or Ni] or a heavy metal [Pb]) and discuss their efficacy as electrode materials for rechargeable lithium-ion batteries. We also compare the electrochemical responses of these materials to those of the better studied $\text{Li}_2\text{C}_6\text{O}_6$ to gain insight into the origins of and methods to circumvent the known electrochemical instability. We demonstrate that the different rhodizonate

complexes display similar structures, with hexagonally close packed layers of rhodizonate ions forming the framework into which ions can intercalate and deintercalated. We likewise demonstrate that structures which coordinate M(II) within these planes of rhodizonate ions universally demonstrate poor capacity retention, likely due to Li^+ occupation of low energy tetrahedral sites between the rhodizonate layers during discharge. In contrast, structures which occlude these sites with the metal chelate, as is the case with PbC_6O_6 , suppress this deleterious Li^+ coordination to some extent, thereby facilitating improved capacity retention on repeated lithiation and delithiation. This work suggests that careful control over the available Li^+ -binding sites within oxocarbon structures may lead to reversible high-capacity electrochemical energy storage.

ACKNOWLEDGEMENTS

In compiling this thesis, I am struck by how many people contributed, both directly and indirectly, to the ideas, concepts, and motivations that eventually led to its completion. It is impossible to list everyone, let alone to give them the recognition they deserve, and trying to do so is an exercise in futility; regardless, I have to put something in this section, so here we go. First and foremost, I will acknowledge the support and myriad contributions of my advisor Andy Gewirth without whom none of this would be possible (obviously). His continued advice direction and ideas in navigating the quagmire of experimental electrochemistry were invaluable, and the whole process would have been far more difficult (and a lot less fun) without Andy keeping the lights on and the wheels turning. Likewise, I extend my sincerest gratitude towards my thesis committee, Professors Ralph Nuzzo, Joaquín Rodríguez-López, and Nancy Sottos. Your insightful comments and kind support made this work possible. I also need to thank the entire Gewirth group, past and present, for their support both in and out of the lab. In 5 years, I've seen many group members come and go and can only say that working with all of you has been wonderful. I wish I could give each of you the mention you deserve in these acknowledgements, however in the interest of brevity I will just list a few.

To my contemporaries, Eric Thornburg, Chris Bandas, and Annie Esposito, working with you has been tremendous. You made the drudgery of daily lab work (and the interminable glovebox maintenance) bearable; and your ideas, comments, and thoughts were invaluable when the research seemed so crushingly confusing. To Dr. Kimberly Bassett, I cannot thank you enough for the role you played in training me in the lab and in providing advice and suggestions when tackling research related problems. It also goes without saying that the tomographic imaging work would not have happened without your persistence and hard work. Likewise,

without your “motivating,” commentary I doubt I would have made as much progress in the lab or on the climbing wall as I did. To Dr. Bruno Nicolau, I am in your debt for both your expertise on self-assembled monolayer chemistry and your positive attitude. I believe any lab is more fun and more interesting with you around. To Drs. Ryan Rooney, Minjeong Shin, Kim Ta, Ruixian Zhang, Stephanie Chen, and Lingzi Sang, thank you all for your constant help, support, and friendship. It was great to know that if I ever had a question, research related or otherwise, I could consult with any of you for advice.

Outside the Gewirth Group, I want to thank all of my friends and coworkers in the Chemistry Department at UIUC. To the past and present members of the Girolomi group, Chris Caroff, Joe Lastowski, and Brian Trinh, thank you for reminding me how little I know about inorganic chemistry, for the constant helpful recommendations about synthesis, and for enduring the burden of my company for many years. To Sean Meyer, thank you for bringing an air of levity to the fifth floor of CLSL, and for commiserating about the inequities of academia. Additionally, I must ardently thank all of my collaborators and co-authors whose contributions elevated the research contained within this thesis far beyond what I alone could accomplish. In particular, I would like to thank Dr. Timothy Fister for supervising my work at Argonne National Laboratory and showing some of the coolest science I have ever seen. Working with you at the beamline was great fun (even at 2 am). To Alan Kastengren, I had a blast learning about X-Ray imaging at your beamline, and hope that my general incompetence with Python and Linux wasn't too much of a hinderance.

Finally, to my close friends and family. To Nader Emonjemeh, Kathleen and Julian Oolman-Wade, Owen Jennings, Eric “the lizard-man” Connelly, Noah Osmund, Caroline Hernandez and Tommy Zamiar, thank you all for the good times both on the climbing wall and

off, your friendships and support mean the world to me. To my parents and to my brother, there are no words that I can write that fully express how grateful I am for your patience, tutelage, and love. To say that this work could not have happened without you is an understatement. I hope you will consider this thesis as much yours as it is mine, and hope that you know that all that I have done was, at its core, because of you.

*“Above, where seated in his tower,
I saw Conquest depicted in his power
There was a sharpened sword above his head
That hung there by the thinnest simple thread.”
-Chaucer*

TABLE OF CONTENTS:

Chapter 1:	The Lithium-Ion Battery as an Energy Storage Medium	1
Chapter 2:	Origin of Enhanced Cyclability in Covalently Modified $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ Cathodes	38
Chapter 3:	Direct Observation of Interfacial Mechanical Failure in Thiophosphate Solid Electrolytes with <i>Operando</i> X-Ray Tomography	78
Chapter 4:	Pressure Dependent Electrochemical Behavior in Di-Lithium Rhodizonate Electrodes	122
Chapter 5:	Demonstration of a Stable Redox-Active Metal Rhodizonate Cathode Material Through Crystal Structure Modification.....	160
Appendix A:	Double Layer Model for Packing Density Approximation by XPS	200
Appendix B:	Critical Stress Approximation from Tomographic Data	207
Appendix C:	Pressure Sensing Electrochemical Cell (Design and Data Acquisition)	213
Appendix D:	Preliminary Results for <i>Operando</i> Neutron Depth Profiling on All-Solid-State and Interlayer Modified Cells	245
Appendix E:	VASP Utilities for Variable Pressure Simulations (Bash Scripts)	252
Appendix F:	Computational Utilities for Tomographic Reconstructions	254

Appendix G:	Charge Density Visualization from VASP CHGCAR File (MATLAB)	344
Appendix H:	General Peak Fitting Program for Multi-Spectrum Fitting (MATLAB)	368
Appendix I:	Program to Fit Heterogeneous Diffraction Data to a Sum of Lorentzians (MATLAB)	418
Appendix J:	Program to Process Galvanostatic Battery Cycling Data (MATLAB)	433

Chapter 1: The Lithium-Ion Battery as an Energy Storage Medium

1.1 The Lithium-Ion Battery

Since their commercialization by the Sony corporation in 1991 lithium ion batteries (LIBs) have revolutionized the portable electronics industry and have made transformative impacts on public and private transportation.¹⁻⁴ In addition to these major impacts, LIBs have proven invaluable in other niche fields including medical devices, telecommunication networks, space exploration, and emergency power systems to name a few.⁵⁻⁹ Such commercial success can be attributed to the beneficial combination of high gravimetric and volumetric energy densities accessible to LIBs relative to other battery chemistries such as lead acid, nickel cadmium, or nickel metal hydride (Figure 1.1).¹⁰ Indeed, the small, lightweight form-factors and high capacities of existing LIBs have made them the energy storage medium of choice for small and medium scale energy storage with a projected market value of ca. \$100 billion by 2025.¹¹ Likewise, the societal and technological impacts of LIBs were recently recognized with a (perhaps long overdue) Nobel prize honoring John Goodenough, Stanley Whittingham and Akira Yoshino for their work developing the chemistries that made these batteries a reality.¹²

Despite the technological significance and commercial success of LIBs their utility has been largely restricted to small-scale electronics (cellular telephones, laptop computers, handheld power tools, etc.) and, only very recently, to commercial electric vehicles (EV's). While much effort has gone towards adapting existing LIB architectures to accommodate high capacity and high energy density applications, including intermittent power storage for renewable energy or grid-scale load leveling, the fundamental limitations associated with existing battery chemistries have prevented significant progress in this area.^{3, 13} To address these deficiencies, and to meet societies growing energy needs, new chemistries must be deployed which can accommodate high

density electrochemical energy storage without sacrificing safety or longevity.^{14, 15} Many candidate chemistries are currently under investigation which exhibit sufficiently high energy densities for such large-scale applications; however, to date none of these so-called “next-generation” battery materials have seen widespread utilization as they, almost universally, exhibit poor long-term stabilities relative to well-established technologies.¹⁶⁻¹⁹

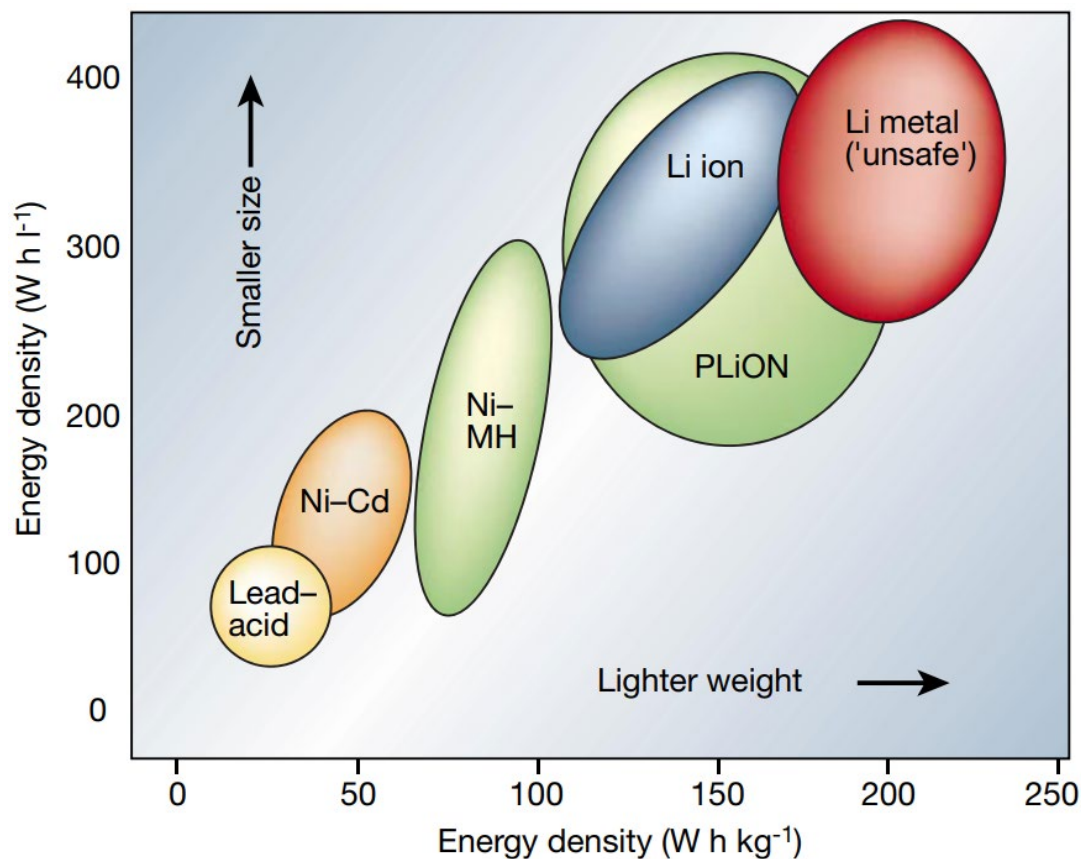


Figure 1.1. A comparison between the practical energy densities (both volumetric and gravimetric) achievable from various battery chemistries including lead acid, nickel cadmium, nickel metal hydride, lithium ion, solid state polymer lithium ion, and lithium metal batteries.¹⁰

Significant work in understanding the fundamental chemistries and degradation mechanisms of these next generation materials is necessary to realize their potential as mature technologies. To contribute to such an understanding, this thesis aims to explore various aspects affecting the stability, or lack thereof, of several next-generation battery chemistries, and provide

methodologies to circumvent deleterious processes which would otherwise preclude their real-world application. In the interest of providing the reader with a complete discussion, a brief overview of existing battery chemistries and common alternatives is provided, along with an overview of the benefits and shortfalls of each. Following this introductory discussion an exploration of state-of-the-art approaches to circumvent common decay pathways will be explored. Finally, a brief overview of the scope and goals of this thesis will be outlined to motivate the subsequent investigations into more specific battery chemistries and analyses.

1.2 Chemical Batteries: Principles of Operation

Before considering the operating characteristics of lithium-ion batteries specifically it is useful to address the mechanism by which batteries in general store and release energy. Regardless of their construction, chemical batteries store energy as a difference in electrochemical potential between two (typically solid) electrodes. This potential difference serves as the driving force by which electrons can move from a high energy (low potential) state to a low energy (high potential) state, with the magnitude of the potential difference and the total number of stored electrons dictating the work that can be done by the battery.²⁰ The electrochemical potential of each electrode is dictated by the nature of the redox active material contained within it and, while the general construction of most batteries is very similar, it is the choice of redox active materials which differentiates the various battery chemistries available for a given application.

The operating principals by which electrochemical energy can be stored and released using a battery are outlined here and are presented, in the simplest form, in Figure 1.2 for a cell operating under discharging conditions. During discharge, energy is released from the battery by allowing electrons to travel from a low voltage electrode (also referred to as the anode or

negative electrode) to a high voltage electrode (also referred to as the cathode or positive electrode). To accommodate this flow of electrons, electrochemical reactions occur at each electrode: a formal oxidation at the anode and a formal reduction at the cathode. These reactions dictate both the number of electrons that can be reversibly transported from one electrode to the other and the potentials developed at each electrode during charge/discharge. Finally, to maintain overall charge neutrality, ions must move to oppose the electronic current (thus anions move towards the anode to compensate the buildup of positive charge and cations move towards the cathode to compensate the buildup of negative charge). This ionic current, which must occur separately from the electronic current, is factored into the redox reactions that govern the cell. On charging, this process is reversed (reduction occurs at the anode and oxidation occurs at the cathode) as external energy is used to drive electrons and ions from low energy to high energy states.

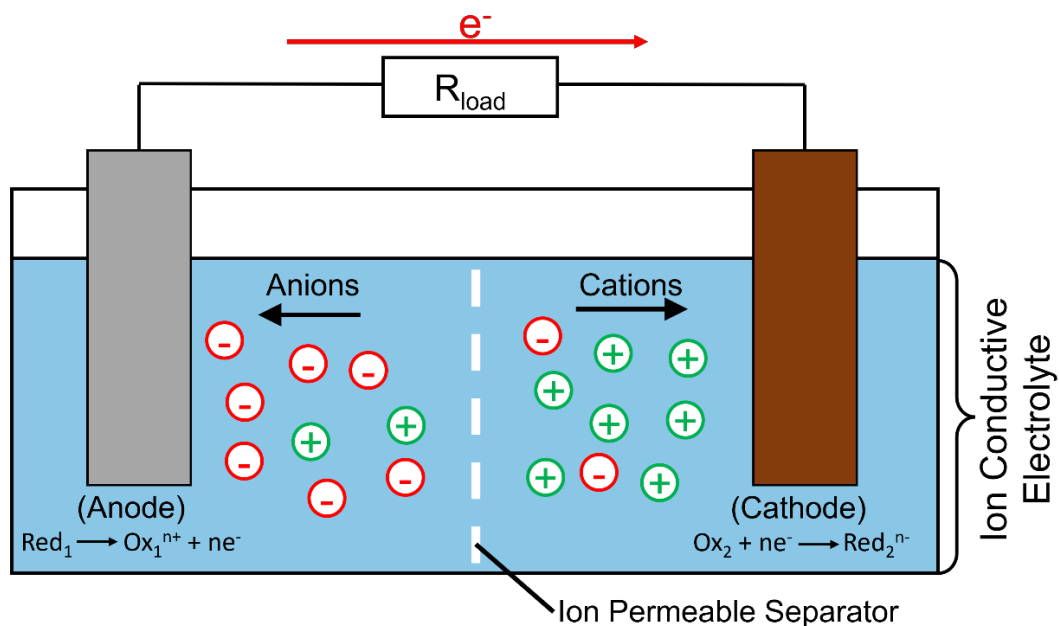


Figure 1.2. Simplified schematic of a chemical battery operating under discharging conditions allowing electrons to flow from anode to cathode through a resistive load (R_{load}). Here two different redox reactions are present, a formal oxidation on the anode converting some reduced species (Red_1) to some oxidized species (Ox_1) and a formal reduction on the cathode (converting Ox_2 to Red_2). Electronic charge is compensated by ionic motion in the electrolyte.

As an electrochemical energy storage medium, lithium-ion batteries operate under these fundamental principles and are only differentiated from other chemistries by the fact that the redox reactions governing the anode and cathode rely on lithium ions (Li^+) to maintain charge neutrality. As discussed above, the total energy that can be stored in a battery is dictated by its capacity (how many electrons it can reversibly transfer on charge and discharge) and the potential difference between the cathode and the anode. While LIBs exhibit similar capacities to other secondary batteries, they operate at significantly high voltages, giving them much higher gravimetric energy densities. The origin for this high operating potential is large difference in electrochemical potential associated with lithium intercalation and deintercalation into various host materials, a topic that will be discussed in the following section.

1.3 Common Materials and Architectures for LIBs

In almost all cases the electrodes used for commercially relevant rechargeable LIBs rely on Li^+ intercalation/deintercalation into a crystalline solid host material. Electrodes of this type undergo simultaneous reduction/oxidation and lithiation/delithiation wherein each single electron reduction is accompanied by the insertion of one Li^+ into the host structure. The chemical composition of the host determines both the electrochemical potential for intercalation as well as the overall gravimetric capacity of the material (the total number of lithium ions that can be inserted into the host per unit mass). In most LIB architectures the anode is composed of some form of graphitic hard carbon while the cathode consists of a lithiated transition metal oxide, each of which will be explored below.

1.3.1 Conventional Anode Materials

Graphitic hard carbons, and mixtures thereof, are by far the most common anode materials for modern LIBs.^{21, 22} It has long been known that the layered structure of graphitic

carbons can play host to a variety of foreign ion intercalants, both anionic and cationic, including lithium ions.²³⁻²⁶ The capability of graphite to undergo electrochemically mediated intercalation using Li^+ as the intercalant makes it ideal as an electrode material for LIBs. The so-called “lithiation,” of graphite occurs via the electrochemical reduction of the carbon host wherein incoming electrons occupy the vacant π^* antibonding states within the carbon lattice. Concomitant with this reduction, Li^+ intercalates between adjacent carbon sheets (left hand side of Figure 1.5), driven into the structure electrostatically by the buildup of negative charge associated with the ongoing reduction of the host lattice. Lithiation proceeds until the fully occupied LiC_6 stoichiometry is reached.²⁷⁻³¹ This process is encapsulated by the following electrochemical half reaction (Eq. 1).



The popularity of graphitic carbon as an anode material is largely due to the reasonably high gravimetric capacities (ca. 372 mA h g^{-1}) and low lithiation potentials (ca. 0.2 V vs Li/Li^+ , 2.8 V vs the standard hydrogen electrode, SHE) associated with lithium insertion. The extremely low electrochemical potential of the anode allows LIBs to easily exceed the operating voltages of other battery chemistries when paired with an appropriate cathode material.^{29, 32} Unfortunately, while this low lithiation voltage is beneficial in developing high energy density batteries, it poses a problem for the practical deployment of graphite as an electrode material. In particular, the reductive decomposition of liquid electrolytes on contact with the highly reactive lithiated graphite make the choice of an appropriate electrolyte non-trivial and has brought to light the importance of interfacial chemistry and surface reactivity towards the development of stable battery chemistries, a topic which will be covered in greater detail in sections 1.3.3 and 1.3.4.

1.3.2 Conventional Cathode Materials

Of equal importance to the choice of anode material is the choice of the cathode material. In virtually all cases, commercial LIBs employ a lithiated transition metal oxide or phosphate as the cathode, which can accommodate reversible lithium intercalation and deintercalation at potentials much greater than those associated with the intercalation of lithium into graphitic carbon.² In contrast to graphite which stores electrons in the antibonding states of π -conjugated carbon, the redox reactions associated with transition metal oxides are based on metal centered redox. The potentials at which these reactions occur depend on the specific metal (often a first-row transition metal such as Mn, Fe, Co, and/or Ni) and the anion network surrounding it (oxide, phosphate, or sulfide). While the cathode composition may vary, most cathode materials display standard reduction potentials between 3 and 5 V vs Li/Li⁺ (0 to 2 V vs SHE) yielding practical full cell operating potentials of roughly 4 V.^{2, 33, 34}

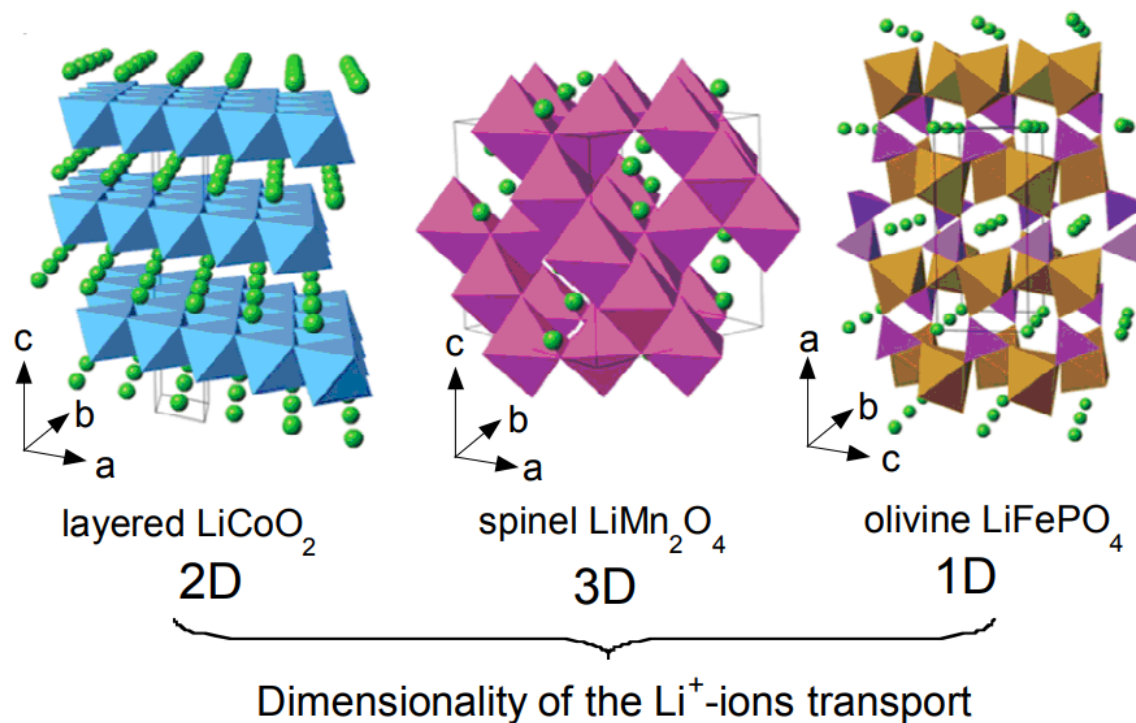
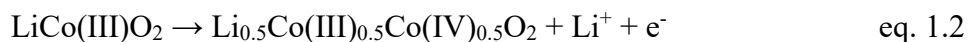


Figure 1.3. Most common crystal structures of transition metal oxide cathode materials along with their associated ion transport dimensionalities.³⁵

Common cathode materials generally possess one of three different archetypal crystal structures, each of which displays unique pathways for lithium transport within the solid. These structures include olivines (based on the mineral structure $(\text{Mg}^{2+}, \text{Fe}^{2+})_2\text{SiO}_4$), layered oxides, and spinels (based on the mineral structure MgAl_2O_4), corresponding to one-dimensional, two-dimensional, and three-dimensional networks for lithium transport respectively.³⁵ Figure 1.3 displays these structures along with the most popular exemplar cathode from each class. Among the variety of cathode materials that have been developed, layered LiCoO_2 (LCO) is by far the most popular material deployed for commercial applications. The popularity of this material is largely due to its good long-term cycling stability and high redox potential (ca. 4.2 V vs Li/Li^+). The operating potential of LCO is associated with the reversible oxidation of Co(III) to Co(IV) during charging which should, in principle, deliver 272 mA h g^{-1} of capacity. Unfortunately, due to crystallographic transitions during deep delithiation³⁶⁻³⁸, $\text{Li}_{1-x}\text{CoO}_2$ can only accommodate fully reversible redox for $0.5 \leq x \leq 1$ yielding a functional capacity of ca. 140 mA h g^{-1} with the following associated half reaction:



Combining equations 1 and 2 we arrive at the full discharge reaction governing the transfer of a single electron from anode to cathode in the most common LIB architecture (Eq. 1.3).



1.3.3. Conventional Electrolytes and General Stability Considerations

The preceding sections have explored the solid components that comprise the anode and cathode of a lithium-ion battery; however, they have neglected what is, arguably, the most important part of the whole assembly: the electrolyte. While the solid electrodes store the electrochemical energy which is eventually used to do work, the electrolyte facilitates the

reactions which allow this energy to be released on command. Specifically, the electrolyte allows lithium-ions to transport to and from the electrodes, thereby supplying the ions necessary to complete the redox reactions mentioned above. This section discusses the most common electrolyte compositions seen in commercial batteries.

Two considerations largely dictate the choice of an appropriate electrolyte for a given application. Firstly, the electrolyte must be chemically and electrochemically stable when in contact with the electrodes both at rest and while the electrodes are charging or discharging. Secondly, the electrolyte must be sufficiently ionically conductive to allow the quick transfer of ions between the two electrodes. Failure in the first area results in direct electrolyte decomposition when in contact with the electrodes, or during charging or discharging. In the best case this simply shuts down the electrochemical reactions driving the cell, and in the worst case can lead to more violent reactions including gas release, cell pressurization, and cell rupture. Failure in the second area results in significant energy losses during charge and discharge as the electrolyte acts as a resistor, effectively reducing the operating potential of the cell as dictated by Ohms law as described below (Eq. 1.4),

$$V_{cell} = V_{cathode} - V_{anode} - R_{electrolyte} \times I_{cell} \quad \text{eq. 1.4}$$

where V_{cell} is the total cell voltage during discharge, $V_{cathode}$ and V_{anode} are the standard potentials of the cathode and anode half reactions respectively, $R_{electrolyte}$ is the total ionic resistance associated with the electrolyte, and I_{cell} is the current driven between the two electrodes.

As discussed in previous sections, much of the appeal of LIBs as a technology is due to the high operating potentials achievable relative to other battery chemistries. Thus, the realization of electrolytes that can, not only survive these extreme potentials, but also reduce the

ionic resistance of the cell is of prime importance. It is easy to see that aqueous electrolytes cannot meet the demands of LIB architectures as the low potentials developed at the anode result in rapid hydrogen evolution which result in immediate cell failure.³⁹ To circumvent this stability issue non-aqueous solvents and combinations thereof have been intensively investigated as the liquid component of LIB electrolytes. Figure 1.4 demonstrates the benefits of these non-aqueous electrolytes (in this case a mixture of linear and cyclic carbonates) by comparing their electrochemical stability windows to that of aqueous electrolytes. The operating potentials of common electrode materials are also included as a reference for the electrochemical stability required for the stable operation of various LIB materials.

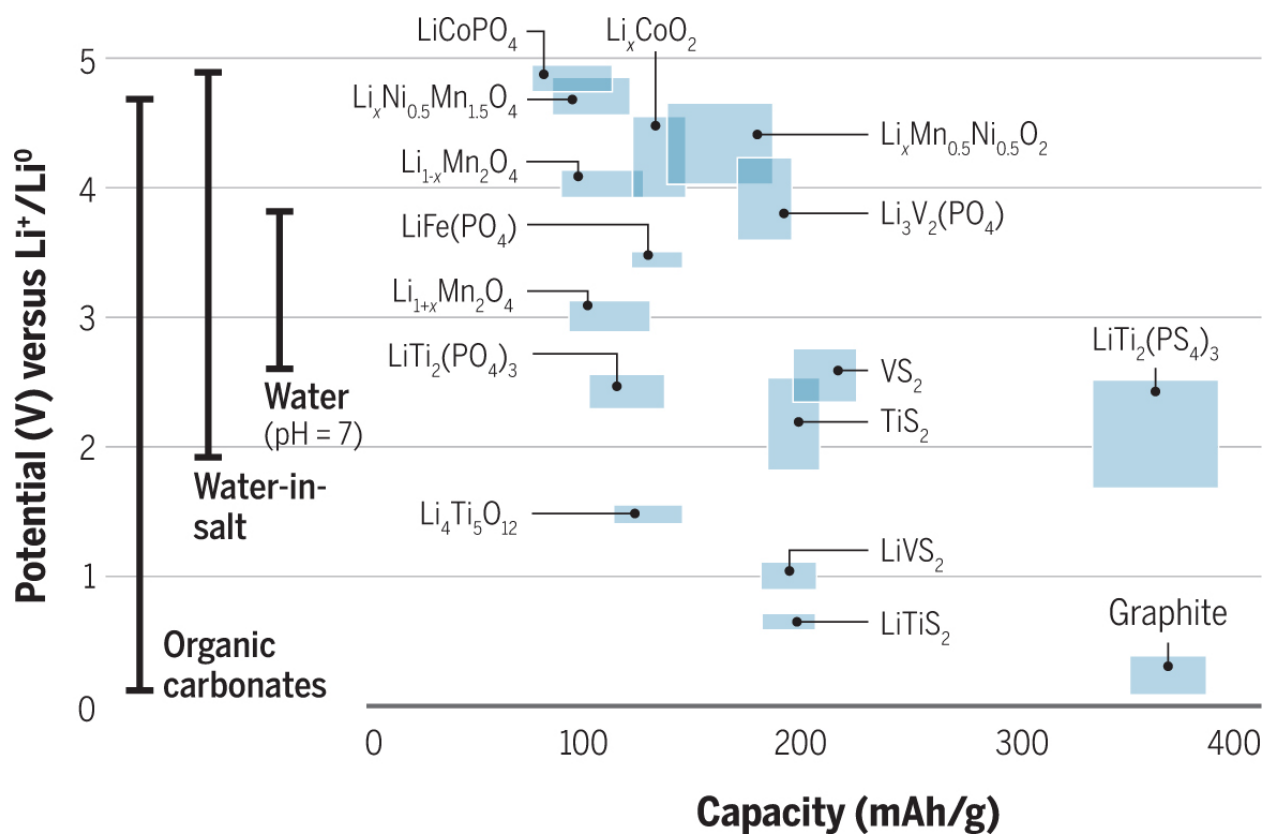


Figure 1.4. Electrochemical stability windows for various electrolyte systems (left) compared against the standard reduction potentials of common cathode and anode materials (right). The functional gravimetric capacities of the cathode materials are also displayed for reference.⁴⁰

While numerous non-aqueous solvents have been investigated, including nitriles, ethers, carbonates, sulfoxides, and glymes, a combination of linear and cyclic carbonates have been widely adopted due to their good electrochemical stability and their high solvating power which can accommodate adequate concentrations of lithium salts.^{41, 42} The specific composition that has found the most favor among commercial applications is a rough 1:1 volume mixture of ethylene carbonate (EC) and dimethyl carbonate (DMC). In this system the cyclic carbonate, EC, possesses a sufficiently high dielectric to solvate lithium salts while the linear DMC serves to reduce the viscosity of the electrolyte, thereby improving the mobility of ions within.

While the transition to non-aqueous electrolytes benefits electrolyte stability, it poses a problem when considering the salt species that will populate the electrolyte with Li^+ . While many lithium salts are very soluble in aqueous media their solubilities are generally low in non-aqueous solutions (as is the case for many common lithium salts including LiCl and Li_2SO_4).⁴³ Similarly salts which form strong ion pairs in solution are also non-ideal due to the energy necessary to overcome this pairing during ion intercalation.⁴⁴ These concerns led to the adoption of the lithium salts with strongly non-coordinating anions to address both low solubility and unfavorable ion pairing. The most popular among these weakly coordinating salts are lithium perchlorate, lithium bis(trifluoromethanesulfonyl)imide (LiTFSI), and lithium hexafluorophosphate. Of these LiPF_6 has emerged as the most widely used electrolyte salt largely due to its ease of production and adequate electrochemical properties.

The solvent and salt are typically combined to form a ca. 1M LiPF_6 solution in EC/DMC, along with various electrolyte additives and stabilizers, which has proven more than satisfactory in supporting the ionic currents demanded by conventional LIBs. With the electrolyte composition sorted we arrive at the prototypical architecture for the lithium-ion battery as

presented in Figure 1.5. In this scheme the anode and cathode face each other on opposite sides of the electrochemical cell with the electrolyte filling the space between. As mentioned in section 1.2, the electrodes must remain physically separated within the cell (thereby separating the ionic and electronic currents) as any physical contact will result in an internal short circuit which will prevent the battery from charging. Sufficient physical separation is accomplished by placing a thin (ca. 25 μm) electrolyte permeable polymer separator between the two electrodes. This separator, often fabricated from a mix of polyethylene and polypropylene, effectively prevents physical contact between the anode and the cathode, allowing the stable operation of the battery.

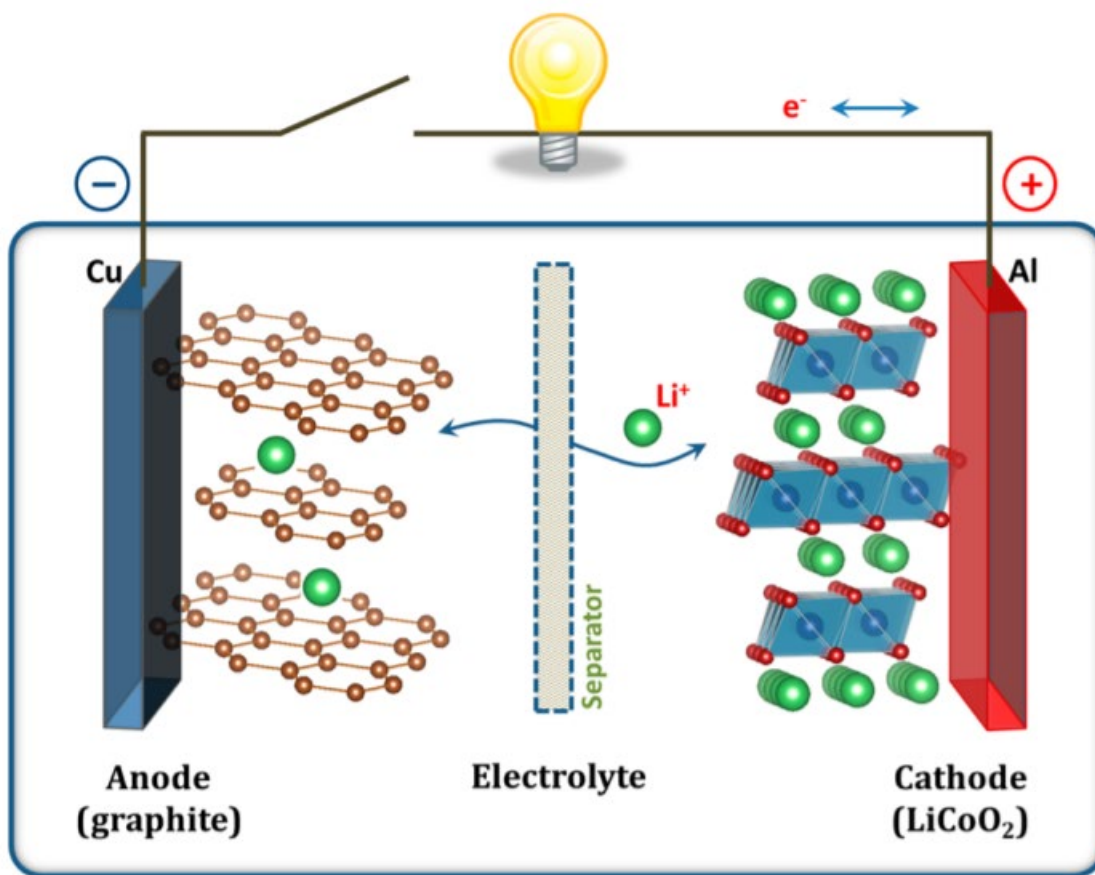


Figure 1.5. Schematic of a common lithium-ion battery, wherein Li^+ is transported between to different intercalation electrodes resulting in the transport of electrical charge through an external circuit. Ion transport to accommodate this electronic current is facilitated by a non-aqueous electrolyte in contact with both the anode and cathode.

1.3.4 The Vital Importance of the Solid Electrolyte Interphase

The previous section alluded to stability issues inherent to LIBs by virtue of the extreme potentials accessed by both the anode and the cathode during cell cycling. While non-aqueous electrolytes certainly have lower reactivity than their aqueous counterparts, they are, nonetheless, thermodynamically unstable when in contact with common anode materials and some cathode materials. This section will cover the passivation of electrodes by the beneficial (and some would say serendipitous) decomposition of the liquid electrolyte at the surface of solid electrodes. Many have demonstrated that on contact with lithiated graphite (as well as metallic lithium and lithiated silicon) common electrolyte formulations spontaneously decompose to form a thin, electrically insulating layer at the surface of the redox active component of the electrode. Two interesting observations were made about this decomposition layer. Firstly, the layer that formed remained, at least somewhat, ionically conductive, and maintained pathways for Li^+ intercalation and deintercalation. Secondly, the layer and its growth were self-passivating such that, upon reaching a certain thickness, electrolyte decomposition ceased, and the layer remained roughly stable. This latter behavior is due to the insulating character of the decomposed layer which became known as the solid electrolyte interphase (SEI). As the layer grew thicker it prevented the reducing potentials at the anode surface from electronically contacting the electrolyte thereby preventing continued reductive decomposition.

These two properties of the SEI, ionic conductivity and self-passivation, make LIBs tractable as a technology. Without the protection of the SEI the thermodynamic instabilities within the cell would completely decompose the electrolyte making the “battery,” useless. The importance of the SEI was quickly noticed, and the intervening years saw the publication of countless research articles attempting to characterize the structure and chemistry of the SEI. One

of the longest enduring, and most widely accepted descriptions of SEI composition posits a mosaic structure where inorganic domains of Li_2O and LiF (among others) form near the electrode surface while domains of polymeric species (polyolefins and semicarbonates) form on top of this patchwork inorganic layer.⁴⁵⁻⁴⁷ A schematic of this structure is presented in Figure 1.6. Recent research has also shown that through interface engineering, electrolyte additives, electrode pretreatments and other approaches which modify the chemistry of the SEI, significant improvements in the overall stability and performance of the battery system can be realized.⁴⁸⁻⁵² Finally, it is worth mentioning that while most of the work on the SEI interrogates the anode/electrolyte interface, there exists an analogous interface on the cathode side, comprised mainly of oxidation products which also serves to prevent the ongoing decomposition of the electrolyte at high potentials.⁵³⁻⁵⁵

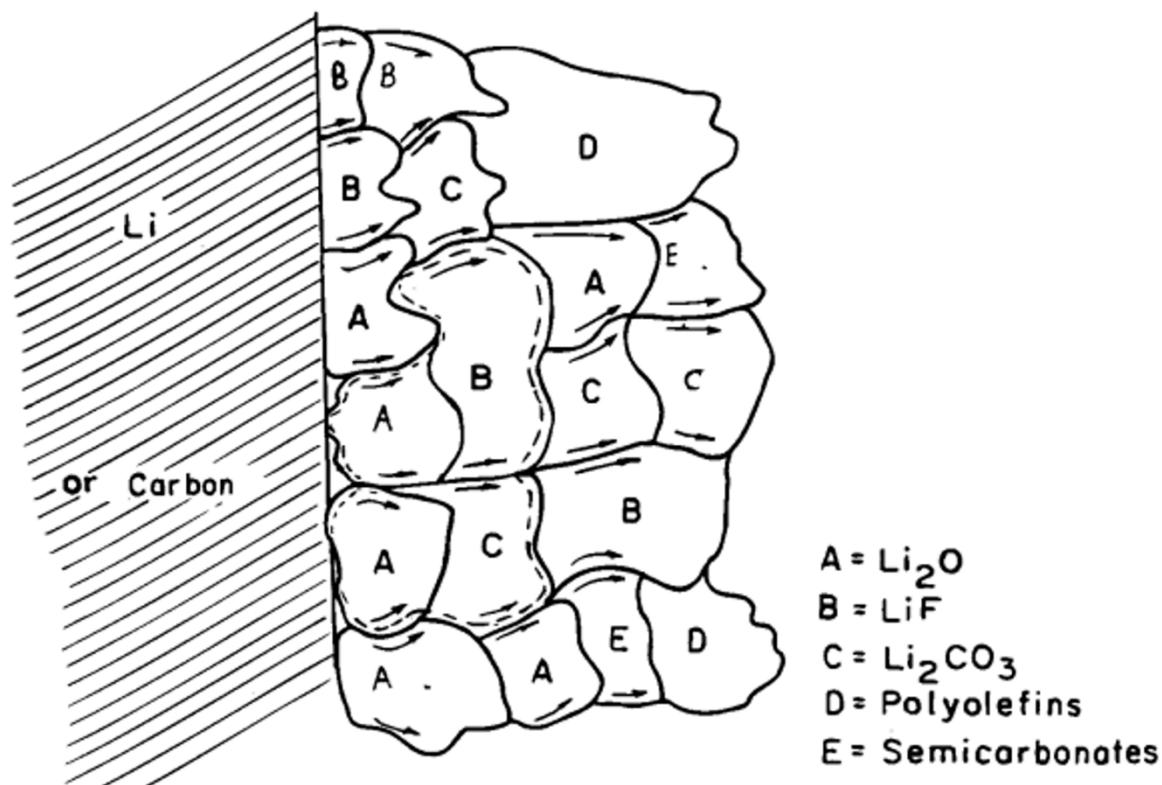


Figure 1.6. Commonly accepted structure of the solid electrolyte interphase formed at the anode surface when submerged in non-aqueous electrolyte.⁴⁶

1.4 High energy density alternatives to existing battery chemistries

Having described the operating principles of the most basic, and most commercially relevant, lithium-ion battery architecture we will proceed to explore the limitations of this chemistry and discuss approaches that are currently under investigation to improve the practical energy density accessible to LIBs. To a crude approximation, the energy density of a given LIB system is the product of its overall gravimetric capacity and the potential at which this capacity is obtained during discharge. The total capacity of a given LIB is dictated by the individual gravimetric capacities of both the anode and cathode in the following form:

$$\frac{1}{Q_{cell}} = \frac{1}{Q_{anode}} + \frac{1}{Q_{cathode}} \quad \text{eq. 1.5}$$

Where Q_{cell} is the total capacity of the cell, Q_{anode} is the gravimetric capacity of the anode, and $Q_{cathode}$ is the gravimetric capacity of the cathode. Similarly, the potential of the cell (E_{cell}) is simply the difference in electrochemical potential between the cathode ($E_{cathode}$) and the anode (E_{anode}) at a given state of charge (Q)*.

$$E_{cell}(Q) = E_{cathode}(Q) - E_{anode}(Q) \quad \text{eq. 1.6}$$

Importantly, the electrode potentials (and the overall cell potential) change with the degree of lithiation, also called the state of charge (SOC), of the electrodes. Thus, the potential of the cell is not independent of the state of charge but is rather a function of it. Thus, the total energy density of a cell can be described as an integration of the cell potential over its functional capacity as described in equation 1.7.

*Note that this formalism ignores many important contributions to the operating potential of the cell. Most importantly it does not consider ohmic contributions to the cell potential as described in section 1.3.3; however, this simplified approach is sufficient to motivate the subsequent discussion of high energy density battery chemistry.

$$U_{cell} = \int_0^{Q_{cell}} E_{cell}(Q) dQ \quad \text{eq. 1.7}$$

Here Q represents the capacity of the full cell, U describes the total energy density of the cell, and $E(Q)$ gives the potential at the specified SOC.

Equation 1.7 can be visualized graphically by plotting the voltage of a cell as a function of its capacity undergoing discharge at a constant current. Figure 1.7 shows such a discharge trace, wherein the total energy stored within the cell is given by the area beneath the red voltage curve. This graphical depiction also suggests that, to a crude approximation, the total energy density can be regarded as the product of the average cell voltage (E_{av}) by its capacity (Q_{cell}) as described in equation 1.8.

$$U_{cell} \approx E_{av} \times Q_{cell} \quad \text{eq. 1.8}$$

Thus, there are effectively two pathways to improve the energy density of an existing cell architecture: increase its gravimetric capacity or increase its operating voltage (or both). Both approaches have been extensively explored, with countless high voltage and high-capacity materials catalogued, spanning organic, inorganic, and organometallic chemistries. In the following sections we will explore these chemistries paying close attention to the fundamental operating principals of each and to the challenges that must be overcome before they can be realized as practical technologies.

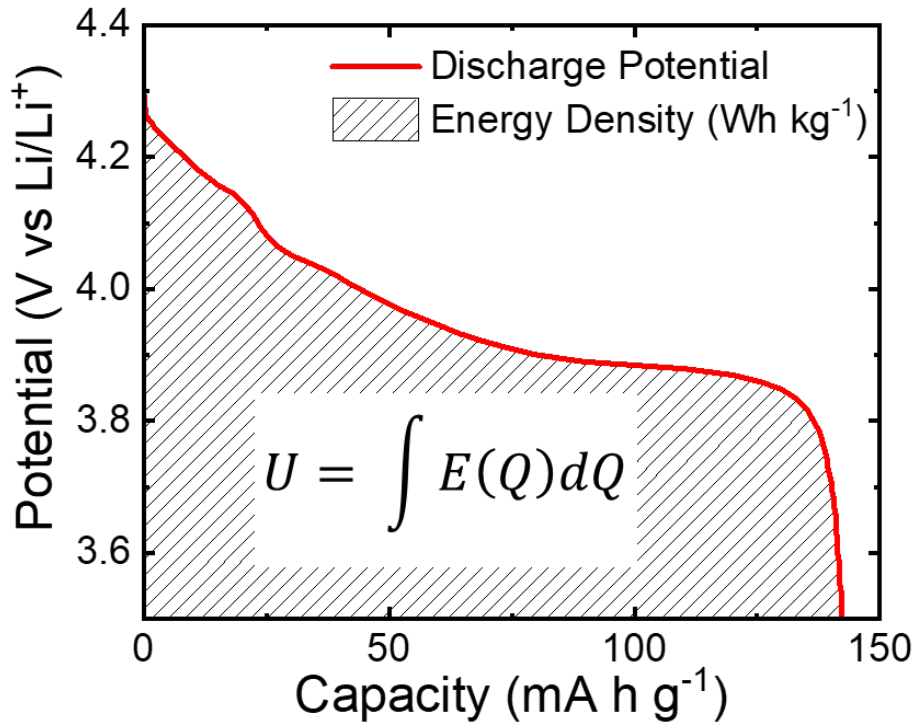


Figure 1.7. Characteristic discharge plot obtained from a LiCoO_2 cathode discharged at a rate of 15 mA g^{-1} . The shaded portion of the plot displays the overall energy density of the cathode material. Likewise, the equation describes, mathematically the energy density (U) of the system in terms of the potential ($E(Q)$) and the capacity (Q).

1.4.1 Increasing Cell Operating Voltages

As discussed in section 1.2, and detailed in equation 1.6, the total cell potential of a given LIB architecture is defined as the difference between the potentials of its cathode and its anode. The task of increasing this overall potential is then a simple matter of either increasing the operating potential of the cathode or decreasing the operating potential of the anode. Unfortunately, the lithiation potential of graphite is only 200 mV above the electrodeposition potential of metallic lithium, which represents the lowest formal potential of any element.⁵⁶ This simplifies the situation somewhat, as the anode potential cannot be any lower than it already is.

Thus, any increase in the cell potential must be realized by increasing the operating voltage of the cathode.

Fortunately, there exist many avenues to modify the operating potential of conventional metal oxide cathodes. The simplest among these is to transition from the conventional $\text{Co}^{3+}/\text{Co}^{4+}$ redox (standard potential ca 3.8 V vs Li/Li^+) toward a more electrophilic metal center with a higher standard potential. A very common approach is to supplant Co with Ni, making use of the $\text{Ni}^{2+}/\text{Ni}^{3+}/\text{Ni}^{4+}$ redox couple. Indeed, in the spinel electrode $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ (LMNO) the available nickel redox couple displays a standard potential near 4.7 V vs Li/Li^+ offering, roughly, a 26% increase in cell energy density when paired with a common graphite anode.^{57, 58} Other approaches, beyond simple metal replacement, include a transition to polyanionic solids such as LiCoPO_4 or to more exotic chemistries such as anion intercalation into graphitic carbon, although these approaches are far less popular.^{59, 60}

Regardless of the cathode chemistry deployed, a battery system relying on high voltage redox (at or near 5 V vs Li/Li^+) will, invariably, encounter stability issues due to the limited oxidative stability of common electrolytes.⁶¹ Analogous to the reductive instabilities encountered at the anode, these oxidative instabilities manifest at the cathode/electrolyte interface during cathode delithiation resulting in oxidative decomposition of the electrolyte. These oxidative decomposition reactions are often associated with gas evolution at the electrode surface (likely due to the oxidation of the electrolyte to CO_2 and fluorinated alkanes) and pose significant problems for stable battery operation.⁶²

The aforementioned, oxidative instability can be visualized by considering the energies associated with the electrochemical potential of the cathode surface relative to the highest occupied molecular orbital (HOMO) of the electrolyte as displayed in Figure 1.8. When the

electron energy associated with the valence band of the cathode drops below that of the HOMO of the electrolyte (as is the case at high potentials) electrons transfer from the electrolyte to the cathode surface, resulting in electrochemical decomposition. While numerous works have demonstrated that this electrolyte decomposition does result in the formation of a cathode electrolyte interphase (CEI), it does not passivate to the same extent that the SEI does at the anode.

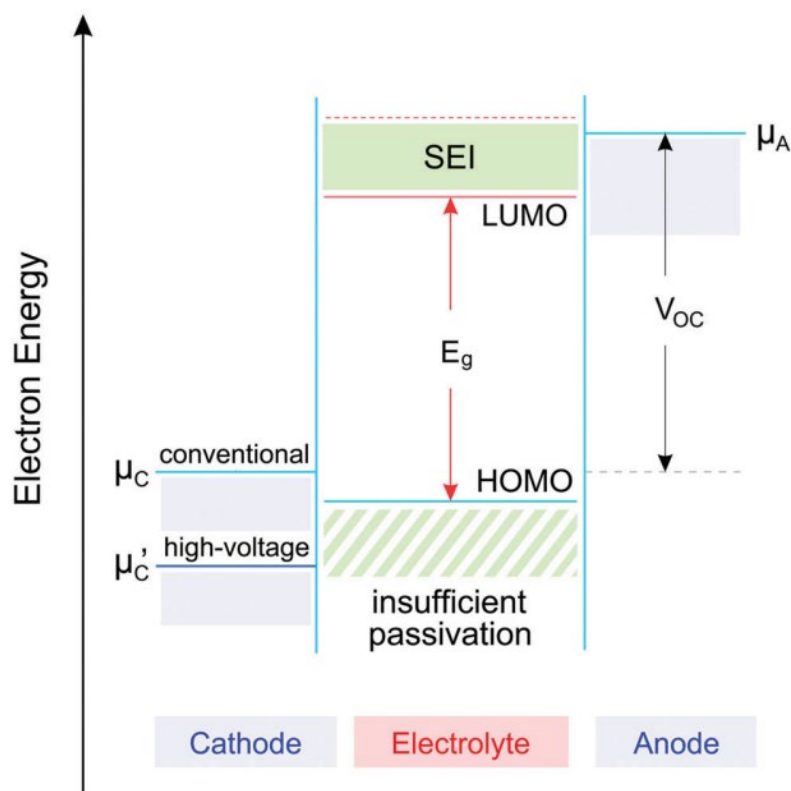


Figure 1.8. Energy diagram displaying the molecular orbital energies of an organic electrolyte relative to the energies associated with the operating potentials of the anode and the cathode.⁶¹

The realization that the instability associated with high voltage cathodes is largely due to interfacial processes has motivated research into interface modifications and electrolyte optimization (very similar to the early work on the anode SEI) to develop a cathode/electrolyte system with adequate cycling performance. To date none of these modifications have manifested

a high voltage cathode with adequate cycling stability for practical application, a fact which hints at a lack of understanding of the specific interfacial processes dictating the formation and stability of the CEI. The development of such an understanding forms the basis for the first project discussed in chapter 2 of this thesis, wherein organosilane self-assembled monolayers are used to probe and modify the surface chemistry of LMNO electrodes in search of key factors affecting the electrochemical stability of this interface.

1.4.2 Increasing Cell Capacity

While raising the operating voltage of a given cell can be accomplished by solely through modifications to cathode chemistry, increasing the capacity of a LIB requires increases to both the anode and cathode capacities. This behavior is reflected in equation 1.5, which suggests that the overall cell capacity will be dictated by whichever electrode displays the lowest gravimetric capacity. In this section we will explore some of the anode and cathode chemistries available to improve total cell capacity relative to that of the conventional metal oxide/graphite architecture.

(I) Anodes

Numerous materials have been proposed and characterized as high-capacity negative electrodes. While the specific chemistries of these materials are diverse, they typically rely on alloying or conversion reactions with lithium.⁶³⁻⁶⁵ Examples of this form include the alloying reaction of elemental silicon, tin, or germanium each of which can accommodate large amounts of Li at reducing potentials. In addition to these alloying reactions, and significantly more enticing, is the direct electrodeposition of metallic lithium at the anode. As mentioned in section 1.4.1 metallic lithium possesses the lowest standard reduction potential of any element (-3.04 V vs Li/Li^+). This, coupled with its enormous gravimetric capacity of 3860 mA h g^{-1} has made Li_m the holy grail of battery anodes. Unfortunately, the deployment of Li_m in practical batteries has

been plagued with safety concerns motivating ongoing research into the origins of and factors dictating the poor electrochemical performance of Li_m anodes.⁶⁶

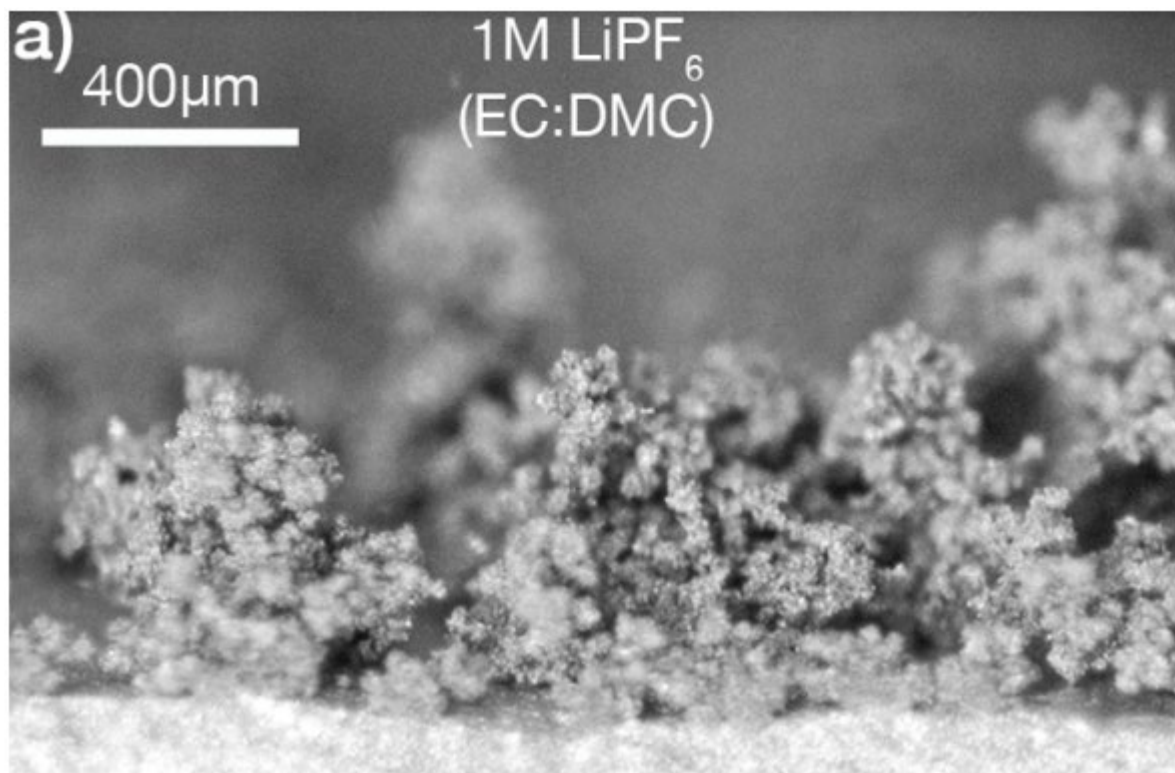


Figure 1.9. Mossy lithium grown during repeated deposition and stripping of Li^+ at a Li_m electrode in a liquid electrolyte.⁶⁷

The issues associated with Li_m electrodes are largely due to inhomogeneous deposition and stripping of Li^+ at the anode/electrolyte interface during charging and discharging. It is well known that lithium electrodeposition from non-aqueous electrolytes does not form a smooth homogenous film. Instead, lithium deposits as mossy or dendritic structures (Figure 1.9) due to the kinetic sluggishness of metal surface diffusion.⁶⁸ Most commonly, these dendritic structures simply accelerate the overall capacity fade of the cell due to continuous electrolyte consumption at the high surface area deposit. Similarly, due to the high aspect ratios associated with these dendritic structures, the deposit is often mechanically unstable and can separate from the anode resulting in so-called “dead lithium,” which is no longer electrochemically accessible. In some

situations, however, dendrites can bridge the anode and cathode, growing through the separator from one side of the cell to the other. The resulting internal short circuit can dramatically increase the temperature inside the cell leading to unregulated and catastrophic thermal runaway, a fact which has garnered much publicity in the form of battery fires.⁶⁹⁻⁷¹

Developing approaches to circumvent dendritic lithium growth during cycling represents a major area of research in advanced LIBs, with many researchers suggesting that a transition away from conventional liquid electrolytes to solid-state electrolytes may address many of the limitations of Li_m as an electrode.⁷² This solid-state approach relies primarily on mechanical dendrite suppression, with early work suggesting that any Li^+ -conducting solid electrolyte with a sufficiently high shear modulus should eliminate the formation of dendritic lithium structures due to the comparatively low bulk and shear moduli of metallic lithium. This idea was captured computationally in work by Monroe and Newman who demonstrated a transition to stable surface energies (and as a consequence stable electrodeposition) when electrolytes of sufficient mechanical strength were used against Li_m (Figure 1.10).⁷³

Despite numerous computational results affirming this initial conclusion, solid electrolytes experimentally exhibit mechanical and chemical decomposition pathways, which continue to plague the ongoing implementation of these electrolytes to real battery systems. Specifically, it has been shown by many that dendritic lithium can proliferate through the solid electrolyte along grain boundaries where the mechanical adhesion of the SE is insufficient to suppress dendrite growth.⁷⁴ Likewise deleterious chemical decomposition at the interface also plays a key role in the development of an effective solid-state battery system.^{75, 76} The work described in chapter 3 of this thesis analyzes, using real space imaging, the bulk and interfacial decay pathways extant during repeated cycling of a $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$ (LGPS) solid electrolyte. The

results demonstrate that chemical transformations at the Li/SE interface largely dictate the overall mechanical stability of the electrolyte.

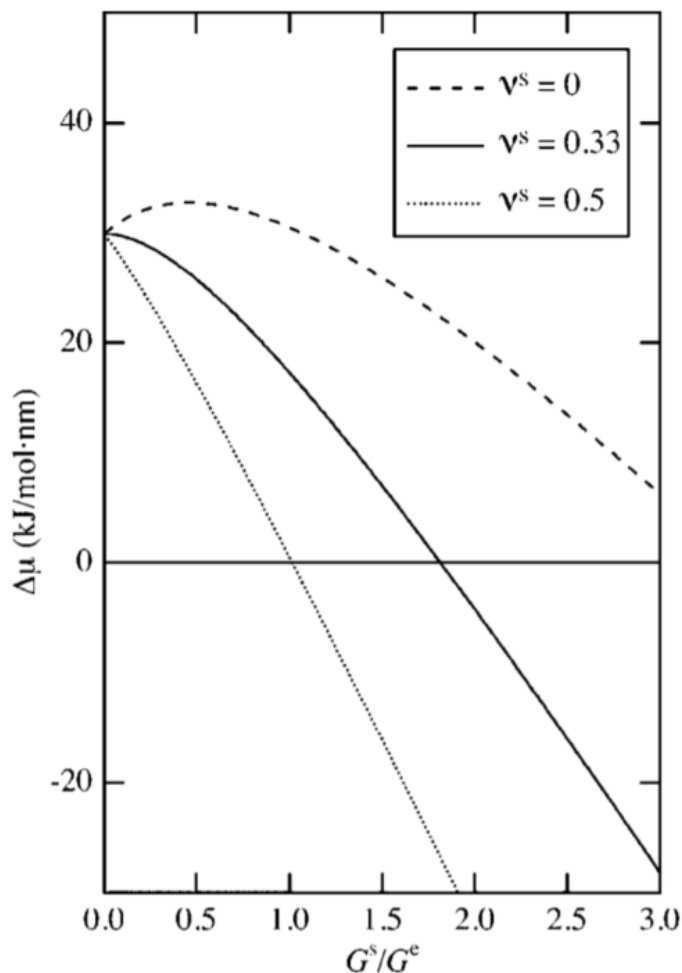


Figure 1.10. Computational results describing the surface energy associated with deposition of lithium from solid electrolytes with different shear moduli, normalized to the shear modulus of Li_m . At sufficiently high shear moduli the system achieves an overall negative energy of deposition suggesting stable Li_m plating. Here v represents the visco-elasticity of the solid electrolyte.⁷³

(II) Cathodes

Even if the anode problem is solved, and metallic lithium anodes become tractable as a reliable technology, the capacity limitations associated with the cathode still serve to limit the overall capacity of a given LIB architecture. While section 1.4.1 explored methods to increase the cathode operating potential, this section will focus on increasing the functional capacity of

cathode materials. As has been a theme throughout most of this introduction, there exist a huge number of materials proposed in the literature which exhibit gravimetric capacities greater than those of conventional transition metal oxides (typically around 150 mA h g⁻¹). The list of high-capacity cathodes contains a wide range of electrochemical storage mechanisms from intercalation type electrodes to conversion cathode to gas phase electrochemistry.⁷⁷⁻⁸⁰

One subset of materials, with which this thesis is directly concerned, that have been the subject of a significant research effort are organic solids as redox active materials.⁸¹⁻⁸³ In these materials lithium is stored reversibly by the reduction and oxidation of some organic species including amines, carboxylates, imines, and organosulfur compounds.⁸⁴⁻⁸⁹ The benefits of organic materials are manifold and include high synthetic tunability, very high achievable capacities, and low cost relative to the high price of Co based transition metal oxides. Unfortunately, while organic materials exhibit low molecular weights and concomitantly high gravimetric capacities, they often display significant issues with stability and longevity. In particular, materials based on carbonyl redox (quinones forming the most common representative of this group)⁹⁰ exhibit extremely high capacities (ca. 980 mA h g⁻¹ per redox active C=O)⁹¹ but display rapid capacity fade during cell cycling (often on the order of 50% capacity fade after 20 cycles).

The poor cycling stability of these materials is oft attributed to their high affinity for organic electrolytes which results in the dissolution of redox active species during prolonged cell operation.⁹² While active material dissolution is true in many cases, it is not the only reason for the observed capacity fade. In particular, the poor cycling stability of electrodes based on dilithium rhodizonate (Li₂C₆O₆, theoretical capacity: 580 mA h g⁻¹)⁹³ cannot be attributed to dissolution due to the very low solubility of the crystalline organic material in conventional

organic solvents.⁹⁴ Chapter 4 of this thesis details work interrogating the cycling stability of lithium rhodizonate from a physical standpoint. This work investigated the chemo-mechanical response of the electrode system, a property frequently overlooked in this sphere of battery analysis. Interestingly this work reveals a crystallographic origin for capacity fade which has not been previously described. Further, chapter 5 continues work in this area leveraging the information obtained from the preceding investigation. With the understanding that the origin of capacity fade is crystallographic in nature, chapter 5 describes efforts to mitigate performance decay by modifying the crystal structure of rhodizonate complexes.

1.5. Concluding remarks for this introduction

Overall, this work aims to understand the processes that lead to performance decays in LIB electrodes and electrolytes. This work addresses processes which occur both at interfaces, and in bulk materials, although much of the discussion throughout is motivated by the former. A common theme across much, but not all, of this work is the intimate coupling of chemical and mechanical properties that yield complex, and sometimes unexpected behaviors in these battery materials. Likewise, this work combines synthetic, analytical, and computational methods to understand the behaviors exhibited by these systems to the extent possible. It is hoped that such a multifaceted approach will provide new insights into important characteristics of these next generation battery materials which will be of interest to researchers developing these technologies for application.

1.6 References

- (1) Goodenough, J. B.; Park, K. S., The Li-Ion Rechargeable Battery: A Perspective. *J Am Chem Soc* **2013**, *135* (4), 1167-1176.
- (2) Manthiram, A., A reflection on lithium-ion battery cathode chemistry. *Nature Communications* **2020**, *11* (1), 1550.
- (3) Schmuch, R.; Wagner, R.; Hörpel, G.; Placke, T.; Winter, M., Performance and cost of materials for lithium-based rechargeable automotive batteries. *Nature Energy* **2018**, *3* (4), 267-278.
- (4) Nitta, N.; Wu, F.; Lee, J. T.; Yushin, G., Li-ion battery materials: present and future. *Materials Today* **2015**, *18* (5), 252-264.
- (5) Takeuchi, E. S.; Leising, R. A., Lithium Batteries for Biomedical Applications. *MRS Bulletin* **2002**, *27* (8), 624-627.
- (6) Passerini, S.; Owens, B. B., Medical batteries for external medical devices. *J Power Sources* **2001**, *97-8*, 750-754.
- (7) Bock, D. C.; Marschilok, A. C.; Takeuchi, K. J.; Takeuchi, E. S., Batteries used to power implantable biomedical devices. *Electrochim Acta* **2012**, *84*, 155-164.
- (8) Spurrett, R.; Thwaite, C.; Slimm, M.; Lizius, D. In *Lithium-ion batteries for space*, 6th European Space Power Conference (ESPC), Oporto, Portugal, May 06-10; Oporto, Portugal, 2002; pp 477-482.
- (9) Scholey, N., RECHARGEABLE BATTERIES FOR MOBILE COMMUNICATIONS. *Electronics & Communication Engineering Journal* **1995**, *7* (3), 93-96.
- (10) Tarascon, J. M.; Armand, M., Issues and challenges facing rechargeable lithium batteries. *Nature* **2001**, *414* (6861), 359-367.

- (11) Arshad, F.; Li, L.; Amin, K.; Fan, E.; Manurkar, N.; Ahmad, A.; Yang, J.; Wu, F.; Chen, R., A Comprehensive Review of the Advancement in Recycling the Anode and Electrolyte from Spent Lithium Ion Batteries. *ACS Sustainable Chemistry & Engineering* **2020**, *8* (36), 13527-13554.
- (12) Fernholm, A. They developed the world's most powerful battery.
<https://www.nobelprize.org/prizes/chemistry/2019/popular-information/> (accessed 17 Feb.).
- (13) Chen, T.; Jin, Y.; Lv, H.; Yang, A.; Liu, M.; Chen, B.; Xie, Y.; Chen, Q., Applications of Lithium-Ion Batteries in Grid-Scale Energy Storage Systems. *Transactions of Tianjin University* **2020**, *26* (3), 208-217.
- (14) Mongird, K.; Viswanathan, V.; Balducci, P.; Alam, J.; Fotedar, V.; Koritarov, V.; Hadjerioua, B., An Evaluation of Energy Storage Cost and Performance Characteristics. *Energies* **2020**, *13* (13).
- (15) Cano, Z. P.; Banham, D.; Ye, S. Y.; Hintennach, A.; Lu, J.; Fowler, M.; Chen, Z. W., Batteries and fuel cells for emerging electric vehicle markets. *Nature Energy* **2018**, *3* (4), 279-289.
- (16) Lin, D. C.; Liu, Y. Y.; Cui, Y., Reviving the lithium metal anode for high-energy batteries. *Nature Nanotechnology* **2017**, *12* (3), 194-206.
- (17) Tian, Y. S.; Zeng, G. B.; Rutt, A.; Shi, T.; Kim, H.; Wang, J. Y.; Koettgen, J.; Sun, Y. Z.; Ouyang, B.; Chen, T. N.; Lun, Z. Y.; Rong, Z. Q.; Persson, K.; Ceder, G., Promises and Challenges of Next-Generation "Beyond Li-ion" Batteries for Electric Vehicles and Grid Decarbonization. *Chem Rev* **2021**, *121* (3), 1623-1669.

- (18) Manthiram, A., Materials Challenges and Opportunities of Lithium Ion Batteries. *Journal of Physical Chemistry Letters* **2011**, 2 (3), 176-184.
- (19) Liu, C.; Li, F.; Ma, L. P.; Cheng, H. M., Advanced Materials for Energy Storage. *Advanced Materials* **2010**, 22 (8), E28-+.
- (20) Reddy, T. B.; Linden, D., *Linden's handbook of batteries*. 4th ed.; McGraw-Hill: New York, 2011.
- (21) Roselin, L. S.; Juang, R. S.; Hsieh, C. T.; Sagadevan, S.; Umar, A.; Selvin, R.; Hegazy, H. H., Recent Advances and Perspectives of Carbon-Based Nanostructures as Anode Materials for Li-ion Batteries. *Materials* **2019**, 12 (8).
- (22) Wu, Y. P.; Rahm, E.; Holze, R., Carbon anode materials for lithium ion batteries. *J Power Sources* **2003**, 114 (2), 228-236.
- (23) Guerard, D.; Herold, A., Intercalation of lithium into graphite and other carbons. *Carbon* **1975**, 13 (4), 337-345.
- (24) Zanini, M.; Basu, S.; Fischer, J. E., Alternate synthesis and reflectivity spectrum of stage 1 lithium—graphite intercalation compound. *Carbon* **1978**, 16 (3), 211-212.
- (25) Basu, S.; Zeller, C.; Flanders, P. J.; Fuerst, C. D.; Johnson, W. D.; Fischer, J. E., Synthesis and properties of lithium-graphite intercalation compounds. *Materials Science and Engineering* **1979**, 38 (3), 275-283.
- (26) Seel, J. A.; Dahn, J. R., Electrochemical Intercalation of PF₆ into Graphite. *J Electrochem Soc* **2000**, 147 (3), 892.
- (27) Levi, M. D.; Aurbach, D., The mechanism of lithium intercalation in graphite film electrodes in aprotic media. Part 1. High resolution slow scan rate cyclic voltammetric studies and modeling. *J Electroanal Chem* **1997**, 421 (1), 79-88.

- (28) Levi, M. D.; Levi, E. A.; Aurbach, D., The mechanism of lithium intercalation in graphite film electrodes in aprotic media. Part 2. Potentiostatic intermittent titration and in situ XRD studies of the solid-state ionic diffusion. *J Electroanal Chem* **1997**, *421* (1), 89-97.
- (29) Stevens, D. A.; Dahn, J. R., The Mechanisms of Lithium and Sodium Insertion in Carbon Materials. *J Electrochem Soc* **2001**, *148* (8), A803.
- (30) Asenbauer, J.; Eisenmann, T.; Kuenzel, M.; Kazzazi, A.; Chen, Z.; Bresser, D., The success story of graphite as a lithium-ion anode material – fundamentals, remaining challenges, and recent developments including silicon (oxide) composites. *Sustainable Energy & Fuels* **2020**, *4* (11), 5387-5416.
- (31) Zhang, L.; Li, X.; Augustsson, A.; Lee, C. M.; Rubensson, J. E.; Nordgren, J.; Ross, P. N.; Guo, J. H., Revealing the electronic structure of LiC₆ by soft X-ray spectroscopy. *Applied Physics Letters* **2017**, *110* (10), 104106.
- (32) Shellikeri, A.; Watson, V.; Adams, D.; Kalu, E. E.; Read, J. A.; Jow, T. R.; Zheng, J. S.; Zheng, J. P., Investigation of Pre-lithiation in Graphite and Hard-Carbon Anodes Using Different Lithium Source Structures. *J Electrochem Soc* **2017**, *164* (14), A3914-A3924.
- (33) Mizushima, K.; Jones, P. C.; Wiseman, P. J.; Goodenough, J. B., Li_xCoO₂ (0 < x ≤ 1): a new cathode material for batteries of high energy density. *Mater. Res. Bull.* **1980**, *15*, 783-789.
- (34) Padhi, A. K.; Nanjundaswamy, K. S.; Masquelier, C.; Okada, S.; Goodenough, J. B., Effect of structure on the Fe³⁺/Fe²⁺ redox couple in iron phosphates. *J. Electrochem. Soc.* **1997**, *144*, 1609-1613.

- (35) Julien, C. M.; Mauger, A.; Zaghib, K.; Groult, H., Comparative Issues of Cathode Materials for Li-Ion Batteries. *Inorganics* **2014**, 2 (1).
- (36) Manthiram, A.; Choi, J., Chemical and structural instabilities of lithium ion battery cathodes. *J Power Sources* **2006**, 159 (1), 249-253.
- (37) Liu, Q.; Su, X.; Lei, D.; Qin, Y.; Wen, J.; Guo, F.; Wu, Y. A.; Rong, Y.; Kou, R.; Xiao, X.; Aguesse, F.; Bareño, J.; Ren, Y.; Lu, W.; Li, Y., Approaching the capacity limit of lithium cobalt oxide in lithium ion batteries via lanthanum and aluminium doping. *Nature Energy* **2018**, 3 (11), 936-943.
- (38) Xia, H.; Meng, Y.; Lu, L.; Ceder, G., Electrochemical Behavior and Li Diffusion Study of LiCoO₂ Thin Film Electrodes Prepared by PLD. **2007**.
- (39) Suo, L.; Borodin, O.; Gao, T.; Olguin, M.; Ho, J.; Fan, X.; Luo, C.; Wang, C.; Xu, K., “Water-in-salt” electrolyte enables high-voltage aqueous lithium-ion chemistries. *Science* **2015**, 350 (6263), 938-943.
- (40) Smith, L.; Dunn, B., Opening the window for aqueous electrolytes. *Science* **2015**, 350 (6263), 918-918.
- (41) Xu, K., Nonaqueous Liquid Electrolytes for Lithium-Based Rechargeable Batteries. *Chem Rev* **2004**, 104 (10), 4303-4418.
- (42) Xu, K., Electrolytes and Interphases in Li-Ion Batteries and Beyond. *Chem Rev* **2014**, 114 (23), 11503-11618.
- (43) Dudley, J. T.; Wilkinson, D. P.; Thomas, G.; LeVae, R.; Woo, S.; Blom, H.; Horvath, C.; Juzkow, M. W.; Denis, B.; Juric, P.; Aghakian, P.; Dahn, J. R., Conductivity of electrolytes for rechargeable lithium batteries. *J Power Sources* **1991**, 35 (1), 59-82.

- (44) Morales, D.; Ruther, R. E.; Nanda, J.; Greenbaum, S., Ion transport and association study of glyme-based electrolytes with lithium and sodium salts. *Electrochim Acta* **2019**, *304*, 239-245.
- (45) Peled, E.; Menkin, S., Review-SEI: Past, Present and Future. *J Electrochem Soc* **2017**, *164* (7), A1703-A1719.
- (46) Peled, E.; Golodnitsky, D.; Ardel, G., Advanced Model for Solid Electrolyte Interphase Electrodes in Liquid and Polymer Electrolytes. *J Electrochem Soc* **1997**, *144* (8), L208-L210.
- (47) Peled, E., The Electrochemical Behavior of Alkali and Alkaline Earth Metals in Nonaqueous Battery Systems—The Solid Electrolyte Interphase Model. *J Electrochem Soc* **1979**, *126* (12), 2047-2051.
- (48) Zhang, S. S., A review on electrolyte additives for lithium-ion batteries. *J Power Sources* **2006**, *162* (2), 1379-1394.
- (49) Haregewoin, A. M.; Wotango, A. S.; Hwang, B.-J., Electrolyte additives for lithium ion battery electrodes: progress and perspectives. *Energ Environ Sci* **2016**, *9* (6), 1955-1988.
- (50) Zhou, W.; Chen, J.; Xu, X.; Han, X.; Chen, M.; Yang, L.; Hirano, S.-i., Interface Engineering of Silicon and Carbon by Forming a Graded Protective Sheath for High-Capacity and Long-Durable Lithium-Ion Batteries. *Acs Appl Mater Inter* **2021**, *13* (13), 15216-15225.
- (51) Guo, W.; Meng, Y.; Hu, Y.; Wu, X.; Ju, Z.; Zhuang, Q., Surface and Interface Modification of Electrode Materials for Lithium-Ion Batteries With Organic Liquid Electrolyte. *Frontiers in Energy Research* **2020**, *8*.

- (52) Delaporte, N.; Wang, Y.; Zaghib, K., Pre-treatments of Lithium Foil Surface for Improving the Cycling Life of Li Metal Batteries. *Frontiers in Materials* **2019**, *6*.
- (53) Wang, H.; Li, X.; Li, F.; Liu, X.; Yang, S.; Ma, J., Formation and modification of cathode electrolyte interphase: A mini review. *Electrochem Commun* **2021**, *122*, 106870.
- (54) Kühn, S. P.; Edström, K.; Winter, M.; Cekic-Laskovic, I., Face to Face at the Cathode Electrolyte Interphase: From Interface Features to Interphase Formation and Dynamics. *Adv Mater Interfaces* **2022**, *n/a* (n/a), 2102078.
- (55) Cabana, J.; Kwon, B. J.; Hu, L., Mechanisms of Degradation and Strategies for the Stabilization of Cathode–Electrolyte Interfaces in Li-Ion Batteries. *Accounts of Chemical Research* **2018**, *51* (2), 299-308.
- (56) Bard, A. J.; Parsons, R.; Jordan, J., *Standard Potential in Aqueous Solution*. 1st ed.; Routledge: 1985.
- (57) Liang, G.; Peterson, V. K.; See, K. W.; Guo, Z.; Pang, W. K., Developing high-voltage spinel $\text{LiNi}_{0.5}\text{Mn}_{1.5}\text{O}_4$ cathodes for high-energy-density lithium-ion batteries: current achievements and future prospects. *J Mater Chem A* **2020**, *8* (31), 15373-15398.
- (58) Manthiram, A.; Chemelewski, K.; Lee, E. S., A perspective on the high-voltage $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ spinel cathode for lithium-ion batteries. *Energ Environ Sci* **2014**, *7* (4), 1339-1350.
- (59) Sui, Y.; Liu, C.; Masse, R. C.; Neale, Z. G.; Atif, M.; AlSalhi, M.; Cao, G., Dual-ion batteries: The emerging alternative rechargeable batteries. *Energy Storage Mater* **2020**, *25*, 1-32.

- (60) Ludwig, J.; Nilges, T., Recent progress and developments in lithium cobalt phosphate chemistry- Syntheses, polymorphism and properties. *J Power Sources* **2018**, 382, 101-115.
- (61) Li, W. D.; Song, B. H.; Manthiram, A., High-voltage positive electrode materials for lithium-ion batteries. *Chem Soc Rev* **2017**, 46 (10), 3006-3059.
- (62) Mao, C.; Ruther, R. E.; Geng, L.; Li, Z.; Leonard, D. N.; Meyer, H. M.; Sacci, R. L.; Wood, D. L., Evaluation of Gas Formation and Consumption Driven by Crossover Effect in High-Voltage Lithium-Ion Batteries with Ni-Rich NMC Cathodes. *Acs Appl Mater Inter* **2019**, 11 (46), 43235-43243.
- (63) Bhatt, M. D.; Lee, J. Y., High capacity conversion anodes in Li-ion batteries: A review. *International Journal of Hydrogen Energy* **2019**, 44 (21), 10852-10905.
- (64) Zhang, W.-J., A review of the electrochemical performance of alloy anodes for lithium-ion batteries. *J Power Sources* **2011**, 196 (1), 13-24.
- (65) Demirocak, D. E.; Srinivasan, S. S.; Stefanakos, E. K., A Review on Nanocomposite Materials for Rechargeable Li-ion Batteries. *Applied Sciences* **2017**, 7 (7).
- (66) Cheng, X.-B.; Zhang, R.; Zhao, C.-Z.; Zhang, Q., Toward Safe Lithium Metal Anode in Rechargeable Batteries: A Review. *Chem Rev* **2017**, 117 (15), 10403-10473.
- (67) Wood, K. N.; Kazyak, E.; Chadwick, A. F.; Chen, K.-H.; Zhang, J.-G.; Thornton, K.; Dasgupta, N. P., Dendrites and Pits: Untangling the Complex Behavior of Lithium Metal Anodes through Operando Video Microscopy. *Acs Central Sci* **2016**, 2 (11), 790-801.
- (68) Frenck, L.; Sethi, G. K.; Maslyn, J. A.; Balsara, N. P., Factors That Control the Formation of Dendrites and Other Morphologies on Lithium Metal Anodes. *Frontiers in Energy Research* **2019**, 7.

- (69) Hou, J.; Lu, L.; Wang, L.; Ohma, A.; Ren, D.; Feng, X.; Li, Y.; Li, Y.; Ootani, I.; Han, X.; Ren, W.; He, X.; Nitta, Y.; Ouyang, M., Thermal runaway of Lithium-ion batteries employing LiN(SO₂F)₂-based concentrated electrolytes. *Nature Communications* **2020**, *11* (1), 5100.
- (70) Feng, X.; Ouyang, M.; Liu, X.; Lu, L.; Xia, Y.; He, X., Thermal runaway mechanism of lithium ion battery for electric vehicles: A review. *Energy Storage Mater* **2018**, *10*, 246-267.
- (71) Galushkin, N. E.; Yazvinskaya, N. N.; Galushkin, D. N., Mechanism of Thermal Runaway in Lithium-Ion Cells. *J Electrochem Soc* **2018**, *165* (7), A1303-A1308.
- (72) Zheng, F.; Kotobuki, M.; Song, S.; Lai, M. O.; Lu, L., Review on solid electrolytes for all-solid-state lithium-ion batteries. *J Power Sources* **2018**, *389*, 198-213.
- (73) Monroe, C.; Newman, J., The Impact of Elastic Deformation on Deposition Kinetics at Lithium/Polymer Interfaces. *J Electrochem Soc* **2005**, *152* (2), A396.
- (74) Shen, F.; Dixit, M. B.; Xiao, X.; Hatzell, K. B., Effect of Pore Connectivity on Li Dendrite Propagation within LLZO Electrolytes Observed with Synchrotron X-ray Tomography. *ACS Energy Letters* **2018**, *3* (4), 1056-1061.
- (75) Gurung, A.; Pokharel, J.; Baniya, A.; Pathak, R.; Chen, K.; Lamsal, B. S.; Ghimire, N.; Zhang, W.-H.; Zhou, Y.; Qiao, Q., A review on strategies addressing interface incompatibilities in inorganic all-solid-state lithium batteries. *Sustainable Energy & Fuels* **2019**, *3* (12), 3279-3309.
- (76) Xu, L.; Tang, S.; Cheng, Y.; Wang, K.; Liang, J.; Liu, C.; Cao, Y.-C.; Wei, F.; Mai, L., Interfaces in Solid-State Lithium Batteries. *Joule* **2018**, *2* (10), 1991-2015.

- (77) Zhou, H.; Xin, F.; Pei, B.; Whittingham, M. S., What Limits the Capacity of Layered Oxide Cathodes in Lithium Batteries? *ACS Energy Letters* **2019**, *4* (8), 1902-1906.
- (78) Zhu, K.; Wang, C.; Chi, Z.; Ke, F.; Yang, Y.; Wang, A.; Wang, W.; Miao, L., How Far Away Are Lithium-Sulfur Batteries From Commercialization? *Frontiers in Energy Research* **2019**, *7*.
- (79) Wu, F.; Yushin, G., Conversion cathodes for rechargeable lithium and lithium-ion batteries. *Energ Environ Sci* **2017**, *10* (2), 435-459.
- (80) Liu, T.; Vivek, J. P.; Zhao, E. W.; Lei, J.; Garcia-Araez, N.; Grey, C. P., Current Challenges and Routes Forward for Nonaqueous Lithium–Air Batteries. *Chem Rev* **2020**, *120* (14), 6558-6625.
- (81) Schon, T. B.; McAllister, B. T.; Li, P.-F.; Seferos, D. S., The rise of organic electrode materials for energy storage. *Chem Soc Rev* **2016**, *45* (22), 6345-6404.
- (82) Lu, Y.; Chen, J., Prospects of organic electrode materials for practical lithium batteries. *Nature Reviews Chemistry* **2020**, *4* (3), 127-142.
- (83) Liang, Y.; Yao, Y., Positioning Organic Electrode Materials in the Battery Landscape. *Joule* **2018**, *2* (9), 1690-1706.
- (84) Liang, Y.; Tao, Z.; Chen, J., Organic Electrode Materials for Rechargeable Lithium Batteries. *Adv Energy Mater* **2012**, *2* (7), 742-769.
- (85) Wu, X.; Ma, J.; Hu, Y.-S.; Li, H.; Chen, L., Nano-sized carboxylates as anode materials for rechargeable lithium-ion batteries. *Journal of Energy Chemistry* **2014**, *23* (3), 269-273.

- (86) Sun, T.; Li, Z.-J.; Yang, X.; Wang, S.; Zhu, Y.-H.; Zhang, X.-B., Imine-rich poly (o-phenylenediamine) as high-capacity trifunctional organic electrode for alkali-ion batteries. *CCS Chemistry* **2019**, *1* (4), 365-372.
- (87) Luo, C.; Borodin, O.; Ji, X.; Hou, S.; Gaskell, K. J.; Fan, X.; Chen, J.; Deng, T.; Wang, R.; Jiang, J.; Wang, C., Azo compounds as a family of organic electrode materials for alkali-ion batteries. *Proceedings of the National Academy of Sciences* **2018**, *115* (9), 2004.
- (88) Shadike, Z.; Tan, S.; Wang, Q.-C.; Lin, R.; Hu, E.; Qu, D.; Yang, X.-Q., Review on organosulfur materials for rechargeable lithium batteries. *Materials Horizons* **2021**, *8* (2), 471-500.
- (89) Wang, D.-Y.; Guo, W.; Fu, Y., Organosulfides: An Emerging Class of Cathode Materials for Rechargeable Lithium Batteries. *Accounts of Chemical Research* **2019**, *52* (8), 2290-2300.
- (90) Bitenc, J.; Pavčnik, T.; Košir, U.; Pirnat, K., Quinone Based Materials as Renewable High Energy Density Cathode Materials for Rechargeable Magnesium Batteries. *Materials* **2020**, *13* (3).
- (91) Lu, Y.; Hou, X.; Miao, L.; Li, L.; Shi, R.; Liu, L.; Chen, J., Cyclohexanone with Ultrahigh Capacity as Cathode Materials for Lithium-Ion Batteries. *Angewandte Chemie International Edition* **2019**, *58* (21), 7020-7024.
- (92) Belanger, R. L.; Commarieu, B.; Paolella, A.; Daigle, J.-C.; Bessette, S.; Vijh, A.; Claverie, J. P.; Zaghib, K., Diffusion Control of Organic Cathode Materials in Lithium Metal Battery. *Sci Rep-Uk* **2019**, *9* (1), 1213.

- (93) Chen, H.; Armand, M.; Demailly, G.; Dolhem, F.; Poizot, P.; Tarascon, J.-M., From Biomass to a Renewable LiXC₆O₆ Organic Electrode for Sustainable Li-Ion Batteries. *ChemSusChem* **2008**, *1* (4), 348-355.
- (94) Kim, H.; Seo, D.-H.; Yoon, G.; Goddard, W. A.; Lee, Y. S.; Yoon, W.-S.; Kang, K., The Reaction Mechanism and Capacity Degradation Model in Lithium Insertion Organic Cathodes, Li₂C₆O₆, Using Combined Experimental and First Principle Studies. *The Journal of Physical Chemistry Letters* **2014**, *5* (17), 3086-3092.

Chapter 2: Origin of Enhanced Cyclability in Covalently Modified $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$

Cathodes

*Reprinted with the permission of Madsen, K. E.; Wade, K. A.; Haasch, R. T.; Buchholz, D. B.; Bassett, K. L.; Nicolau, B. G.; Gewirth, A. A. Origin of Enhanced Cyclability in Covalently Modified $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ Cathodes. ACS Appl. Mater. Interfaces **2019**, 11, 43, 39890–39901. Copyright 2019 American Chemical Society.*

2.1 Introduction

Lithium-ion batteries (LIBs) represent one of the most promising energy storage architectures due to their high energy densities, low self-discharge, and low maintenance requirements relative to other electrochemical energy storage platforms.¹ Despite these favorable characteristics, current and future technologies demand systems with ever greater energy densities, surpassing those achievable with commercially-utilized metal oxide cathode materials.² At present, cobalt-based oxide cathodes impose limitations on both the operating potential (~ 4.0 V vs Li/Li^+) and gravimetric capacity (~ 150 mA h g^{-1}) of lithium-ion cells, capping their effective energy densities.¹ Transitioning to high voltage (~ 5 V vs Li/Li^+) cathode materials, which increase the operating voltage without sacrificing gravimetric capacity, would improve the overall energy density of LIB cells.^{1,2} Unfortunately, while a variety of high-voltage metal oxides and phosphates have been explored, they universally demonstrate poor long-term stability.^{3,4}

The spinel cathode $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ (LMNO) offers a promising operating potential of ~ 4.7 V vs Li/Li^+ but exhibits rapid capacity fade during cell cycling.⁵ This decay is due in part to the high potentials accessed by the LMNO surface, exceeding the electrochemical stability window of

carbonate-based electrolytes.⁶ Direct oxidation of the electrolyte at the cathode surface results in the formation of an interfacial layer deemed the cathode electrolyte interphase (CEI) analogous to the well-studied solid electrolyte interphase (SEI) formed on anode materials.⁷ In conventional (lower voltage) metal oxide cathodes the CEI electrically insulates the oxide surface thereby kinetically stabilizing it against further electrolyte decomposition. Unfortunately, the CEI formed on LMNO is known to be unstable, precluding such passivation and resulting in rapid capacity fade.⁸

Transition metal dissolution from the cathode lattice exacerbates capacity fade in transition metal oxides by promoting electrolyte decomposition at the anode as well as the cathode. Mn based spinel cathodes, including LiMn_2O_4 (LMO), and LMNO, are particularly susceptible to this due to the dissolution of ionic Mn species, primarily Mn^{2+} , from the electrode surface.⁹⁻¹¹ The dissolution of Mn^{2+} , mediated by the electrolyte, results in transition metal migration to the anode and deposition of metallic Mn into the native SEI, destabilizing the interface and promoting reductive decomposition of the electrolyte.¹¹⁻¹³

These deleterious processes are primarily mediated by the surface chemistry of the cathode material, making modification of the electrode surface a promising route to achieve stable electrochemical performance.¹⁴ While numerous approaches have been explored to control the interfacial properties of oxide cathodes, the most popular has been the application of thin surface coatings to stabilize the chemistry of the CEI.¹⁵ Many different surface coatings elicit improved stability including metal oxides (SiO_2 , ZrO_2 , Al_2O_3 , LiNbO_3) metal phosphates (ZrP_2O_7 , LiFePO_4), ionic salts (LiF , CaF_2 , MgF_2 , AlF_3), and organic species (polypyrrole, poly-imide).¹⁴ Unfortunately many of these coatings demonstrate poor uniformity and poor adhesion to the oxide surface.¹⁶ Recently the application of organic self-assembled monolayers (SAMs) to oxide electrodes has

emerged as a facile approach to achieve highly conformal coatings which, due to their covalent attachment to the surface, should exhibit fewer problems with adhesion during cell cycling.¹⁷⁻²⁰

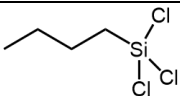
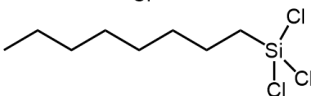
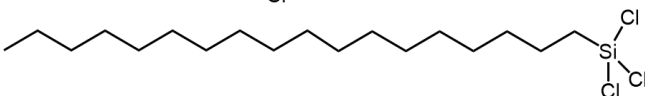
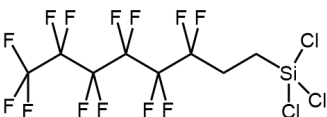
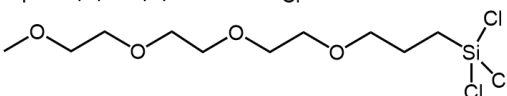
Amphiphilic molecules including alkanolic acids, organo-phosphonic acids, and organo-silanes spontaneously assemble on metal oxide surfaces. This assembly results in the formation of a well-defined monolayer coating bound to the surface by a polar head group leaving a pendant chain (often a long chain alkane) extending away from the surface.²¹ The thermodynamic favorability of self-assembly accommodates pendants of variable length and chemical functionality allowing the surface properties of the native oxide to be modified by molecular design of the coating moiety.

Previous work on LMO electrodes modified with alkylphosphonate monolayers demonstrates that both the overpotential towards Li^+ intercalation/deintercalation and the magnitude of Mn dissolution from the oxide surface can be controlled by varying the structure of the phosphonic acid.¹⁷ Similarly, results obtained from silane-based monolayers on LMNO have shown that long chain alkyl, and fluoroalkyl silanes, as well as fluorinated aromatic silanes, dramatically improve long term cycling performance.¹⁸⁻²⁰ In these cases the improved galvanostatic stability is largely attributed to reductions in both electrolyte oxidation and transition metal dissolution due to enhanced chemical stability at the interface.¹⁸⁻²⁰ Despite the exciting improvements to electrode cyclability attendant silane modification, few silane structures have been investigated. Similarly, the interplay between silane structure and the chemistry of the CEI remains unclear.

Herein we investigate a range of organo-silanes, presented in Table 2.1, to determine salient features of monolayer structure affecting the stability of SAM modified LMNO. We tune the structure of the monolayer by applying coatings of variable chain length (ranging from butane

to octane to octadecane chains) or by applying silanes with variable chemical functionality (including a perfluorinated-alkyl chain and a short polyethylene glycol (PEG) chain). Of particular interest is the impact of these surface structures on the long-term cyclability of modified LMNO electrodes. Additionally, we explore the interplay between monolayer structure and the composition of the CEI formed during battery operation. It is expected that such an understanding will help facilitate rational design of surface coatings to improve the outlook for high voltage electrode materials.

Table 2.1. Structures and abbreviations of studied alkyl-silanes.

Name	Abbreviation	Structure
Butyltrichlorosilane	BTS	
Octyltrichlorosilane	OTS	
Octadecyltrichlorosilane	ODTS	
(Tridecafluoro-1,1,2,2-tetrahydrooctyl)trichlorosilane	Fluoro	
Methoxy(triethyleneoxy)propyltrichlorosilane	PEG	

2.2 Experimental

2.2.1 Surface Modification of $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ Powder

$\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ (LMNO) powder (NEI Corporation, SP-10 Nanomyte) was used as received. Surface modification of LMNO powder was accomplished via a solution-phase deposition process.²² LMNO powder (0.5g) was immersed in 1 mM solutions of trichlorosilane (20 mL) in toluene (anhydrous, Aldrich) for 24 hours with vigorous stirring. Solutions were

centrifuged, and solids rinsed with toluene (30 mL) to remove residual silanes from the surface. Solids were then annealed at 120 °C for 1 h in an oven to covalently attach any physisorbed silanes. Samples were then sonicated in 10 mL of toluene for 5 minutes, separated by centrifugation, and dried at 80 °C under vacuum overnight. Silanes used were n-butyltrichlorosilane (BTS, Gelest, >95%), n-octyltrichlorosilane (OTS, Gelest, >95%), n-octadecyltrichlorosilane (ODTS, Gelest, 97%), (tridecafluoro-1,1,2,2-tetrahydrooctyl)trichloro-silane (Fluoro, Gelest, >95%), and methoxytriethyleneoxypropyltrichlorosilane (PEG, Gelest, >92%).

2.2.2 Surface Functionalization of $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ Thin Films

LMNO films were grown on $\hat{c}$ -sapphire substrates. Before film deposition, the substrates were cleaned in acetone and isopropanol. The films were grown on the substrates by pulsed-laser deposition (PLD) using a PVD Products nanoPLD 1000. The laser was a 248 nm KrF excimer laser with 25 ns duration. The system had a background pressure of 4×10^{-8} Torr. The beam was focused onto a 1.5×3.5 mm spot on the target. The target was rotated at 15 rpm to prevent localized heating, and the laser pulses were swept across the target radius to additionally prevent localized heating. The target-substrate separation was fixed at 8 cm. The LMNO layer was deposited using a dense hot-pressed LMNO target ~25 mm in diameter (Kurt J. Lesker). The target contained 30% excess lithium, $\text{Li}_{1.3}\text{Mn}_{1.5}\text{Ni}_{0.5}\text{O}_{4.15}$, to compensate for lithium loss during high temperature deposition. A laser pulse energy of 200 mJ/pulse* and a repetition rate of 10 Hz was used (*This is the energy as measured at the laser; there is an approximate 30% energy loss along the optical train between the laser and the target). The LMNO films were deposited in a 30 mTorr oxygen ambient at a temperature of 800 °C. After deposition the substrates were annealed in 300 Torr of oxygen at 750 °C for 1-hour then cooled in 300 Torr of oxygen at 10 °C/min. Films were deposited to an approximate thickness of 63 nm. X-ray diffraction (XRD) obtained from a film showed the

presence of a crystalline thin film epitaxial with the substrate (LMNO (111) coincident with surface, Figure 2.1). Film surface roughness was measured using a Veeco Dektak 150 surface profiler and reveals a mean roughness of ca. 30 nm, making the deposited films sufficiently smooth for subsequent analysis.

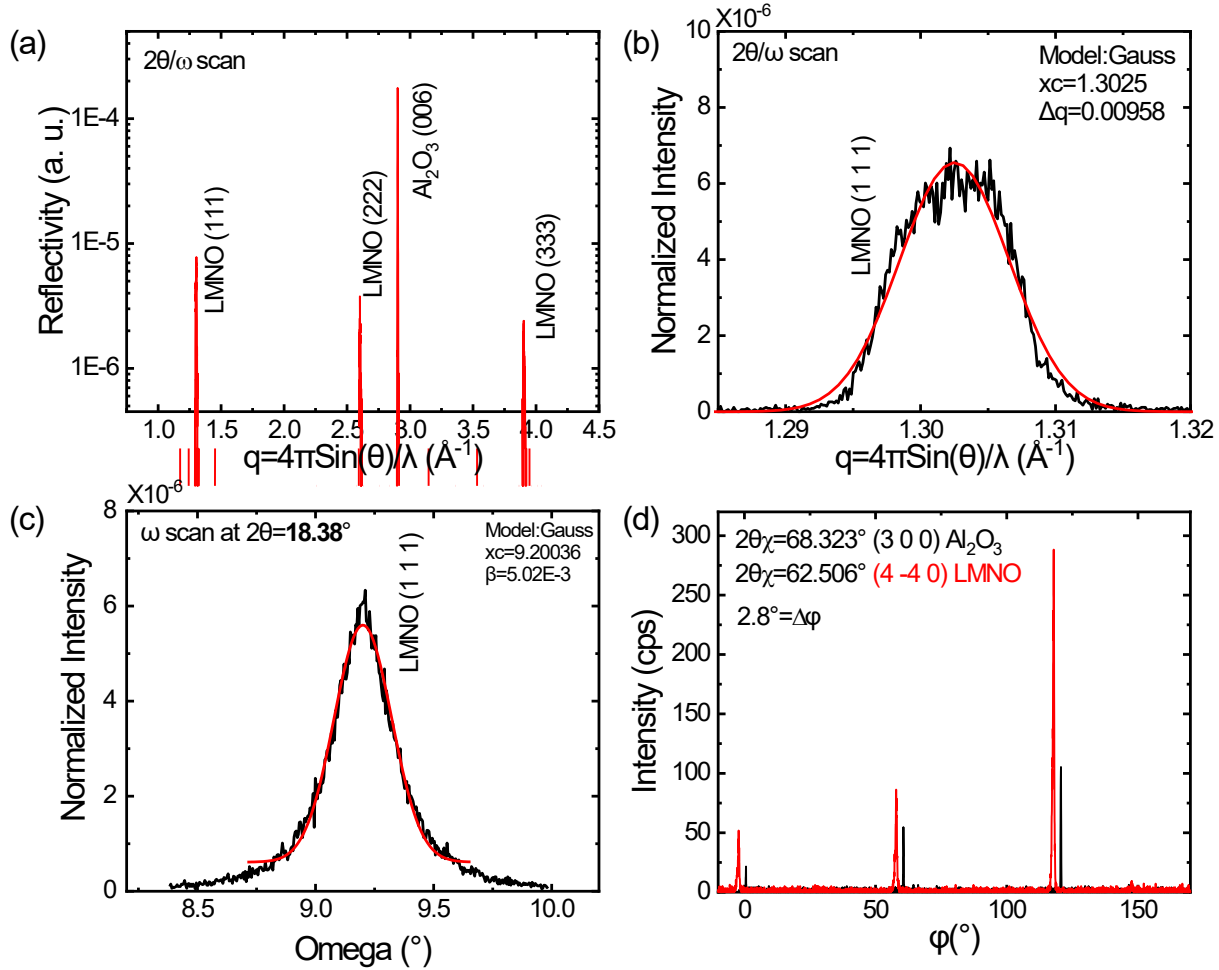


Figure 2.1. XRD spectra of PLD deposited LMNO thin films on $\hat{c}$ -sapphire. a) Out of plane diffraction pattern for the LMNO/ Al_2O_3 (0 0 6) heterostructure. b) $2\theta/\omega$ and c) ω scans of LMNO (1 1 1) peak (black) and Gaussian data fits (red). d) In plane ϕ -scan of LMNO (4 -4 0) (red) and Al_2O_3 (3 0 0) (black) to verify that LMNO is epitaxially deposited on the Al_2O_3 substrate.

PLD-grown films were coated using a similar process to LMNO powder. Films were scored and cut into 1 cm^2 pieces and immersed in a $10 \text{ }\mu\text{M}$ solution of silane in toluene (20 mL).

The concentration of the coating bath was reduced 100-fold to preclude polymer deposition at the substrate surface, which occurs readily at higher silane concentrations.²³ LMNO films were then rinsed with toluene (30 mL) and annealed as described earlier. Annealed films were sonicated in toluene to remove any residuum and dried at 80 °C under vacuum overnight.

2.2.3 Preparation of Composite Cathodes

Composite electrodes were fabricated from a slurry consisting of 8:1:1 wt% modified LMNO powder, conductive carbon (Super P Li, Timcal), and polyvinylidene difluoride (PVDF, Kynar Flex) in n-methyl pyrrolidone (NMP, Aldrich). Slurries were prepared by first grinding the active cathode powder with conductive carbon in a mortar and pestle until visually homogenous. This powder was then added to a 7 wt% PVDF solution in NMP and mixed in a planetary mixer (Thinky) three times to ensure homogeneity. The resulting slurry was cast on an aluminum foil current collector (MiniBin, Sigma-Aldrich) with a doctor blade to an approximate thickness of 100 μm and placed immediately into an oven at 80 °C for at least 2 hours before being punched into 1.27 cm diameter electrodes. Typical active mass loadings were ca. 2.5 mg cm^{-2} . Electrodes were then dried at 80 °C under vacuum for 12 hours before being transferred into an Ar-filled glovebox for cell assembly.

2.2.4 Coin Cell Fabrication and Electrochemical Analysis

Electrochemical measurements were conducted in CR2032 coin cells (MTI corporation, Richmond, CA). Battery stacks were assembled in an argon filled glovebox with O_2 and H_2O concentrations < 2 ppm. Stacks were assembled as follows: the positive electrode was placed and two drops of electrolyte (1 M LiPF_6 in 50:50 ethylene carbonate (EC): ethyl methyl carbonate (EMC), Battery grade, Aldrich) were added to wet the surface. A polyethylene/polypropylene

separator (Celgard) was placed onto the positive electrode, and two more drops of electrolyte were added to the surface of the separator (total electrolyte volume ca. 40 μL). To complete the stack a 14 mm diameter lithium foil (0.75 mm thickness, 99.9%, Alfa Aesar) counter/reference electrode was added, followed by a stainless-steel disc and wave spring. Cells were sealed using a hydraulic coin cell crimper (MTI corporation, Richmond, CA).

Cyclic voltammetry was performed with a CH Instruments electrochemical workstation (Model 1020E, Austin, TX) between 3.5 V and 5.1 V at a scan rate of 50 $\mu\text{V s}^{-1}$ for a minimum of 3 cycles to ensure nominal cell performance before additional characterization was carried out. Electrochemical impedance spectroscopy was carried out with a BioLogic Science Instruments impedance analyzer (SP-150) between 1 MHz and 10 mHz with an amplitude of 30 mV centered at 3.5 V. Galvanostatic cycling was conducted on an Arbin electrochemical workstation (BT2043, Arbin Instruments, College Station, TX) with switching potentials of 3.5 V and 5.0 V.

2.2.5 XPS Characterization of Modified LMNO Particles and Cycled Electrodes

X-ray photoelectron spectroscopy (XPS) measurements were carried out on a Kratos Axis Ultra X-ray photoelectron spectrometer (Kratos Analytical, Inc., Manchester, UK) using monochromatic Al K α radiation (1486.6 eV). Due to the spectral overlap of the Ni 2p and F KLL Auger lines the X-ray source was switched to Mg K α (1253.6 eV) for Ni spectra. All samples were mounted to the analysis stage using insulating double-sided tape. Modified LMNO particles and uncycled electrodes were mounted in air, however due to their air-sensitivity cycled electrodes were prepared and mounted in an Ar-filled glovebox as follows. Cycled coin cells were disassembled and positive electrodes recovered from the electrode stack. The recovered cathodes were then rinsed with ethyl methyl carbonate (5 mL, Aldrich) to remove residual electrolyte.²⁴ Rinsed electrodes were dried under active vacuum at room temperature in the glove box

antechamber for 24 hours. Dried electrodes were mounted on the sample holder in the glovebox and quickly transferred into the instrument to minimize atmospheric exposure. Typical exposure times were less than 1 min. Spectra obtained from LMNO particles and PLD thin films were energy calibrated to the aliphatic carbon peak at 285.0 eV associated with adventitious carbon, whereas spectra obtained from LMNO electrodes were calibrated to the sp^2 carbon peak at 284.5 eV associated with the Super P carbon additive.²⁵

2.2.6 Contact Angle Measurements

Surface interactions between modified LMNO and electrolyte were investigated using contact angle measurements. LMNO thin films were analyzed using a Ramé-Hart contact angle goniometer (Model 250). 1 μ L of either DI water or electrolyte (1 M $LiPF_6$ in EC/EMC) was dropped on the surface using a micro-syringe and contact angles computed from an average of 20 tangent line measurements between the substrate and droplet.

2.2.7 Mn Dissolution Quantification

Mn dissolution from pristine and modified electrodes was carried out using inductively coupled plasma mass spectrometry (ICP-MS, Nexion 350D, Perkin Elmer). Modified LMNO particles (0.2 g) were submerged in electrolyte (5 mL) with stirring for 96 h in an Ar-filled glove box. The electrolyte was then collected after syringe filtration through a 0.22 μ m PTFE filter (VMW) and submitted under inert atmosphere for analysis.

2.3 Results and Discussion

2.3.1 Electrochemical Performance of Modified LMNO

Cyclic voltammetry (CV) was used to correlate changes in the electrochemical characteristics of LMNO to the structure of the coating moiety. Figure 2.2a presents voltammograms of composite electrodes cast with modified and unmodified LMNO, collected at a scan rate of $50 \mu\text{V s}^{-1}$ between 3.5 V and 5.1 V vs Li/Li⁺. Two distinct electrochemical events are observed for bare LMNO with half wave potentials at approximately 3.90 V and 4.75 V vs Li/Li⁺ associated with the Mn³⁺/Mn⁴⁺ and Ni²⁺/Ni³⁺/Ni⁴⁺ redox couples respectively*, consistent with a cation disordered spinel LMNO^{18, 26} (*Due to the proximity of the Ni²⁺/Ni³⁺ and Ni³⁺/Ni⁴⁺ redox waves (ca. 70 mV) and peak broadening due to internal cell resistance a single peak is observed rather than two distinct peaks).²⁷ Surface modification yields voltammograms which preserve these redox couples but exhibit lower peak current densities as well as shifts in anodic and cathodic peak potentials to higher and lower voltages, respectively. This effect is quantified in Figure 2.2b, which displays the potential difference (ΔE_p) between both the Mn³⁺/Mn⁴⁺ and the Ni²⁺/Ni³⁺/Ni⁴⁺ anodic and cathodic peaks plotted against coating length. Due to the high currents elicited during Ni redox the magnitude of ΔE_p is much greater than for the Mn redox couple, however both couples demonstrate the same trend. For alkyl and fluoroalkyl coatings the magnitude of ΔE_p correlates with the length of the coating molecule, with longer coatings exhibiting greater peak splitting. Interestingly, PEG modified LMNO exhibits the largest ΔE_p , deviating from the length dependent trend observed with alkyl and fluoroalkyl coatings. The observed reduction in peak current and the shifting peak positions are indicative of an increased overpotential for Li⁺ intercalation/deintercalation. As the magnitude of this effect appears dependent on monolayer structure, the origin of the overpotential is thought to be the result of

higher interfacial resistance towards Li^+ transport across the cathode electrolyte interphase (CEI) due to the presence of the SAM.

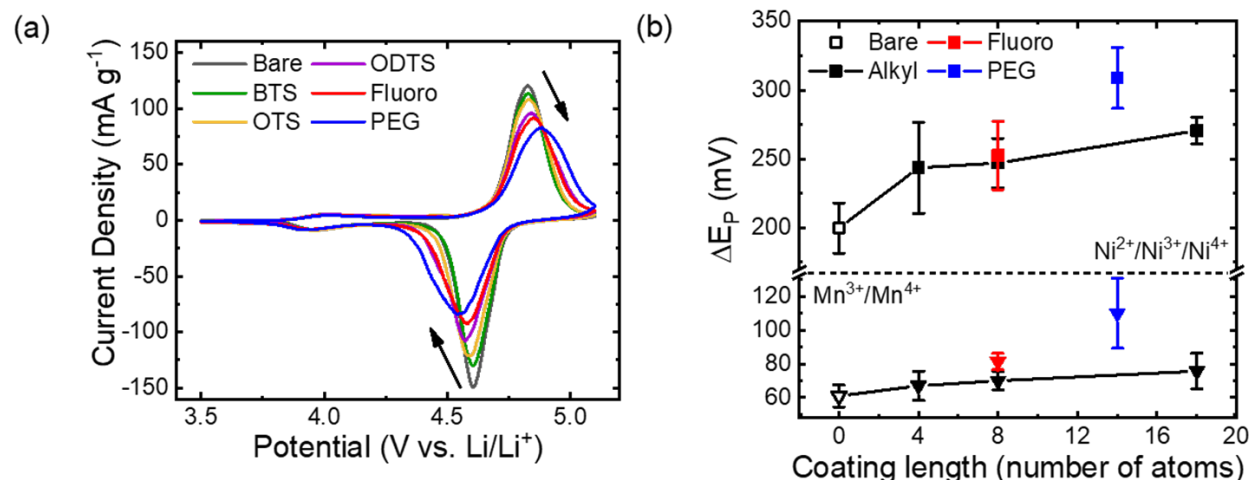


Figure 2.2. a) Cyclic voltammetry of coated LMNO particles with arrows highlighting the increasing overpotential towards lithium intercalation and concomitant drop in current density with increasing coating length. b) quantified peak splittings between both the $\text{Mn}^{3+}/\text{Mn}^{4+}$ (triangles) and the $\text{Ni}^{2+}/\text{Ni}^{3+}/\text{Ni}^{4+}$ (squares) cathodic and anodic waves.

To verify that the origin of these behaviors is the result of interfacial charge transport, electrochemical impedance spectroscopy (EIS) was carried out on modified LMNO electrodes. Figure 2.3a presents impedance spectra for bare and coated electrodes collected at 3.5 V vs. Li/Li^+ , along with a proposed equivalent circuit used to fit experimental data. Impedance spectra demonstrate a single semicircle in the high/mid-frequency region and a steep tail in the low frequency region. These characteristics are attributed to the charge transport of Li^+ at the CEI and ionic diffusion in the solid oxide cathode respectively, as reported in previous literature.²⁸

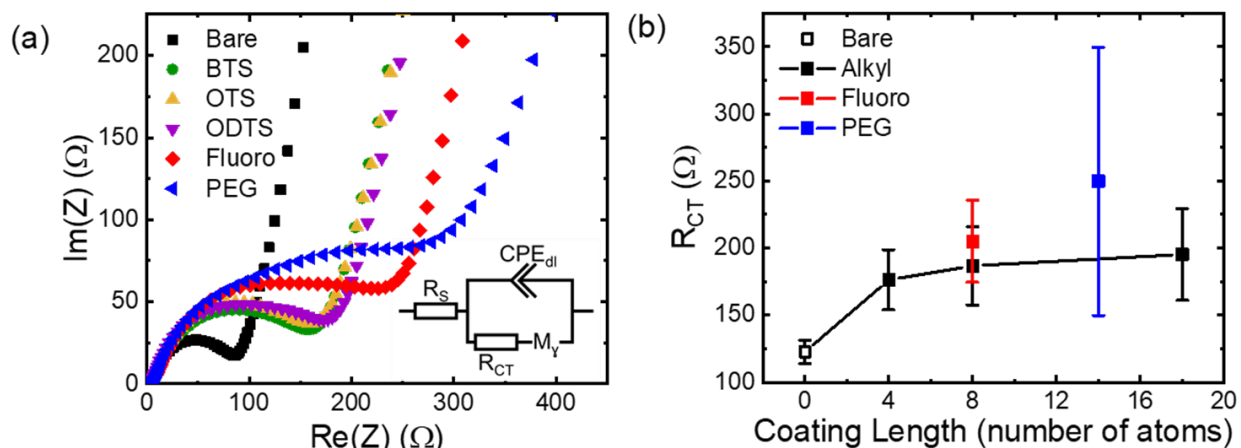


Figure 2.3. a) Electrochemical impedance spectra and proposed equivalent circuit for pristine and surface functionalized LMNO particles. Equivalent circuit elements are as follows: solution resistance (R_s), constant phase element describing the capacitive double layer (CPE_{dl}), charge transfer resistance (R_{CT}), Anomalous diffusion element (M_y). b) Interfacial charge transfer resistance plotted against coating species. Error bars represent standard deviations computed from a set of four independent trials.

A modified Randles circuit (inset of Figure 2.3a) was used to extract physical parameters from the impedance spectra. Two modifications were implemented to achieve acceptable data fits. In place of a pure capacitor, a constant phase element was used as previously described in the analysis of composite electrodes.^{6, 28} Additionally, an anomalous diffusion element was used in place of the traditional Warburg element to better model solid-state ion diffusion within the spinel lattice, as implemented in other intercalation type electrode systems.²⁹⁻³¹ For clarity, data fits have been omitted from the plot, however example fits are presented in Figure 2.4a along with the computed capacitance values (Figure 2.4b). Figure 2.3b presents the interfacial charge transfer resistance (R_{CT}), extracted from fitted experimental data, plotted against coating length. The trends in R_{CT} mirror those seen in Fig 1b. with longer coatings imparting greater interfacial charge transfer resistances. Similarly, PEG modified LMNO exhibits the highest interfacial impedance. This supports the conclusion that the increased overpotentials observed in voltammetry are due to interfacial charge transfer dynamics at the electrode surface. The cause of the increased interfacial impedance is thought to be diminished ionic conductivity across the electrode/electrolyte

interphase. This is expected for alkyl and fluoroalkyl species due to their low ionic conductivities,³² however the high impedance exhibited by the PEG coated surface is unexpected. The ionic conductivity of ether-based molecules has been studied in great detail in the context of solid polymer electrolytes,³³ and it is known that ether functionality imparts higher Li^+ conductivity than aliphatic species. Thus, it is expected that additional PEG functionality at the LMNO surface should provide less resistance to Li^+ transport than its alkyl or fluoroalkyl counterparts. The subversion of this expectation suggests that other interfacial processes are the origin of the low ionic conductivity at the PEG surface, the nature of which will be explored in future sections.

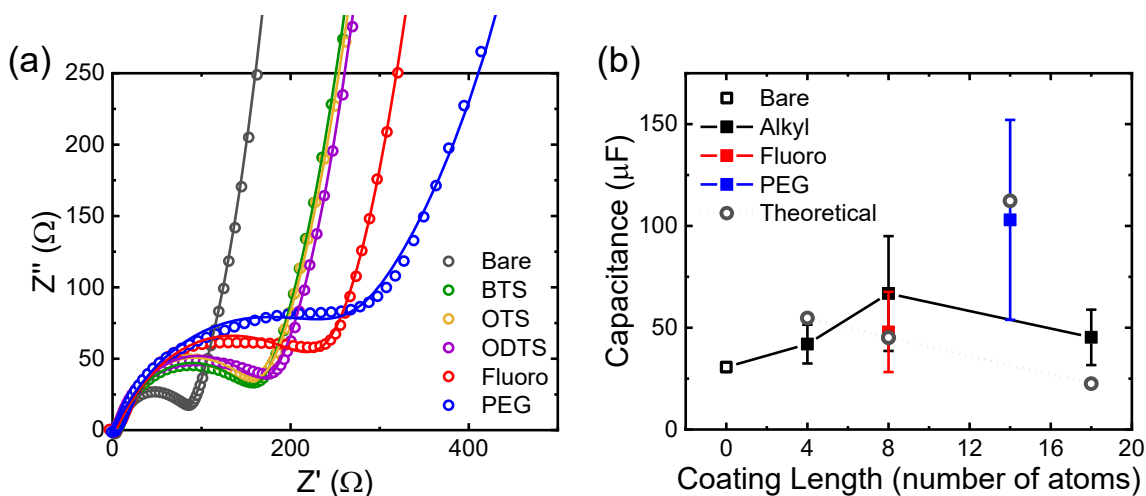


Figure 2.4. a) Fitted impedance spectra collected from modified LMNO electrodes b) Capacitance values obtained from fitted spectra as well as predicted capacitance based on a parallel plate capacitor model (See Appendix A for additional discussion).

To probe the long-term stability of modified LMNO cathodes, cells were subjected to galvanostatic cycling between 3.5 and 5.0 V vs Li/Li^+ . Figure 2.5 presents the galvanostatic cycling performance obtained from coated and uncoated electrodes. At low cycling rates (C/10, Figure 2.5a) the charge/discharge traces of modified and unmodified electrode materials present nearly identical voltage plateaus at 3.8, 4.7, and 4.8 V vs Li/Li^+ . These plateaus can be indexed to the

$\text{Mn}^{3+}/\text{Mn}^{4+}$, $\text{Ni}^{2+}/\text{Ni}^{3+}$, and $\text{Ni}^{3+}/\text{Ni}^{4+}$ redox events respectively. Similarly, all electrode materials demonstrate capacities over 120 mA h g^{-1} , near the theoretical capacity for LMNO of 148 mA h g^{-1} ,^{1, 5} suggesting acceptable utilization of electrode active material regardless of surface modification. At higher cycling rates (1C, Figure 2.5b) the positions of redox plateaus shift to higher potentials during charge, and lower potentials during discharge. These shifts are attributed to ohmic drop and are a consequence of the higher currents applied during 1C cycling. Similarly, as ohmic drop is directly correlated to internal cell resistance,³⁴ the potential shift is more pronounced for LMNO modified with long chain silanes. Further, the increased cell resistance of coated electrodes results in diminished gravimetric capacity at elevated C-rates. This effect is particularly severe for the more resistive coatings (ODTS, PEG, Fluoro) which present significantly lower capacities (87, 93, and 101 mA h g^{-1} respectively) relative to unmodified LMNO (119 mA h g^{-1}) at a 1C cycling rate.

Figure 2.5c presents the capacity retentions and Coulombic efficiencies of LMNO electrodes over 100 charge/discharge cycles. The high Coulombic efficiencies (greater than 99%) exhibited by all electrodes are consistent with previous work on LMNO cathodes³⁵ and indicate that, regardless of surface modification, highly reversible electrochemistry is maintained. Further, all coated electrodes exhibit improved capacity retention relative to unmodified LMNO, suggesting that surface modification imparts additional stability to the oxide cathode. Figure 2.5d presents the overall capacity retention after 100 cycles plotted against coating length. It is apparent that, irrespective of coating length, simple alkyl-silanes demonstrate roughly equivalent capacity retentions (~94%). Changing the functionality of the chain from alkyl to fluoro-alkyl imparts additional improvement up to 96% retention, with some cells demonstrating retentions near 99%, consistent with previous investigations of vapor phase deposited perfluoroalkyl-silanes on

LMNO.¹⁸ Surprisingly, PEG functionalized LMNO demonstrates similar capacity retention (~95%) to alkyl-silanes, contrasting previous investigations of alkylphosphonates on LMO.¹⁷

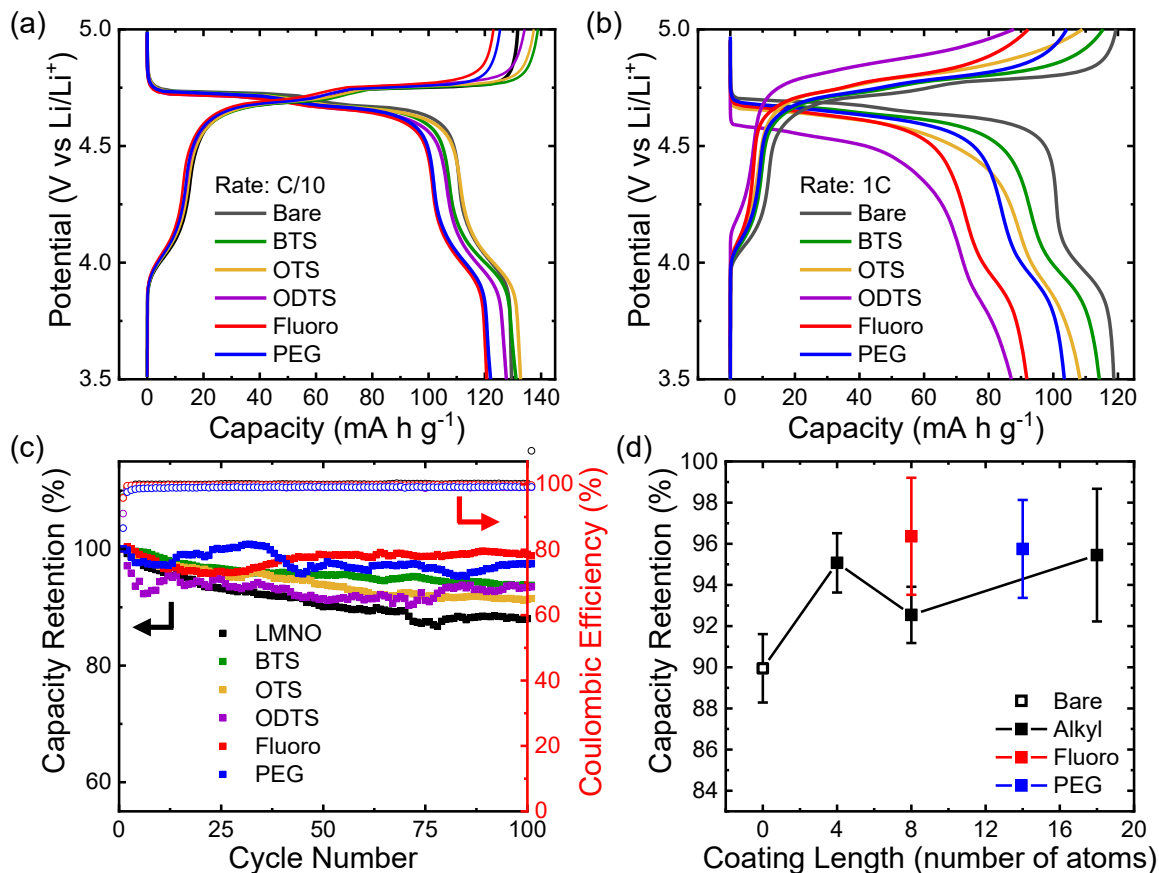


Figure 2.5. Galvanostatic charge/discharge traces collected at a) C/10 and b) 1C cycling rates. c) Capacity retention and Coulombic efficiencies for as-synthesized materials. d) Capacity retention after 100 charge/discharge cycles plotted against coating chain length.

2.3.2 Surface Chemistry of Modified LMNO Particles

While the changes observed in the electrochemical performance of modified LMNO seem to correlate with the structure of the silane, it is possible that variations in packing density, or chemical changes to the substrate are partly responsible for the observed behavior. To address these possibilities, the surface chemistry of modified LMNO was interrogated with XPS.

Figure 2.6 shows XPS spectra obtained from silane-modified LMNO particles along with molecular packing densities computed from XPS analysis of PLD thin films. Figure 2.6a shows Si

2p XPS obtained from unmodified LMNO particles (black) as well as those modified with BTS (green), OTS (orange), ODTS (purple), Fluoro (red), and PEG (blue) silanes. All silane-modified particles demonstrate a single peak attributed to the siloxane head-group anchoring the monolayer to the surface. Alkyl-silanes (BTS, OTS, and ODTS) exhibit a Si 2p binding energy of 102.5 eV, consistent with previous results obtained from organo-silanes on oxide substrates,³⁶ whereas Fluoro and PEG silanes are shifted towards higher binding energies by 0.8 and 0.3 eV respectively. These shifts have been observed in silane monolayers of similar composition^{18, 37, 38} and are attributed to the higher electron withdrawing character of PEG and fluoroalkyl chains relative to that of simple alkanes.

Figure 2.6b shows the C 1s spectra of coated and bare LMNO particles, with peaks indexed as follows: CH₂ (285.0 eV), C-O (286.7 eV), CF₂ (290.9 eV), and CF₃ (293.3 eV).³⁹ With the exception of the Fluoro-modified surface, all modified particles demonstrate higher CH₂ signal intensities than that of unmodified LMNO as expected for an alkane-coated surface. Due to the small amount of aliphatic carbon (2 carbon atoms) present in the Fluoro-silane, as well as the strongly attenuating fluorinated overlayer, the Fluoro-modified particles exhibit reduced CH₂ signal intensity relative to particles with other coatings. Additional peaks are observed for both the Fluoro and PEG silanes associated with fluorinated (290.9 eV, 293.3 eV) and ether bound (286.7 eV) carbon respectively. Both the Si and C spectra present signals consistent with the presence and chemical identity of the coating, suggesting that surface modification of the LMNO substrate was successful.

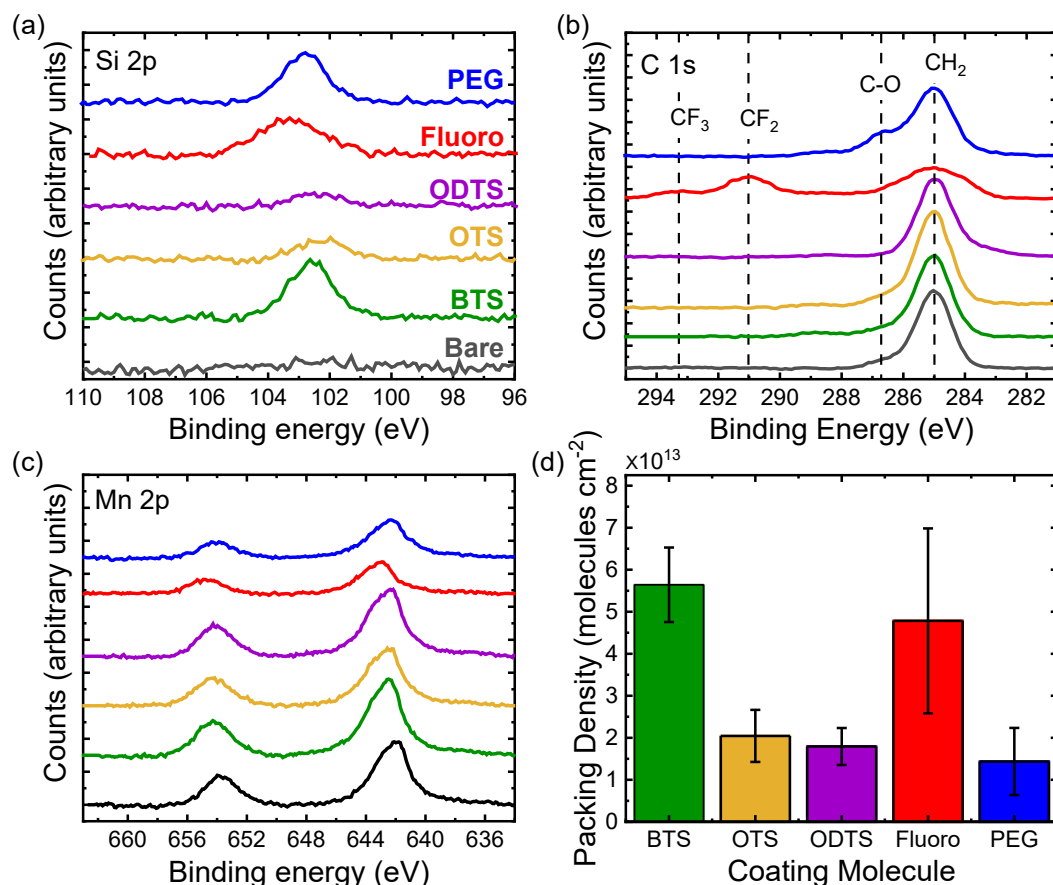


Figure 2.6. XPS spectra of a) Si 2p b) C 1s and c) Mn 2p for pristine and functionalized LMNO particles. Spectra are stacked vertically in ascending order: LMNO-black, BTS-green, OTS-orange, ODTS-purple, Fluoro-red, PEG-blue. d) Molecular packing density obtained using a two-layer model to fit experimental data from decorated LMNO thin films.

Figure 2.6c shows Mn 2p XPS spectra obtained from bare and coated electrode particles. Two distinct features are observed at 641.9 eV and 654.0 eV due to Mn 2p_{3/2} and Mn 2p_{1/2} respectively. Deconvolution of the Mn 2p_{3/2} feature yields two peaks at 641.6 eV and 642.4 eV indexed to Mn³⁺ and Mn⁴⁺ respectively, consistent with previous work on LMNO electrodes.^{39, 40} The Mn spectra change little following addition of the coating which is unsurprising as the binding of trichlorosilane at the surface does not change the oxidation state of the substrate²¹. The fluoro-modified sample exhibits a 0.3 eV shift to higher binding energies in the Mn 2p XPS, similar to the effect observed in the Si 2p spectra. This effect has been observed previously and is attributed to the extremely high electron withdrawing capability of fluoro and perfluoroalkyl moieties.⁴¹

Despite this shift the lack of change to the substrate oxidation state suggests that the active material does not degrade appreciably during the coating process and makes it unlikely that substrate effects are the cause of the observed changes to electrochemical behavior. Notably, the presence of characteristic Mn and Ni peaks suggests that the surface layer exhibits an average thickness less than ca. 3 nm as greater film thicknesses would fully attenuate Ni photoelectrons (see Appendix A for a complete discussion)

Further investigation of the monolayer structures was carried out on LMNO thin films as the surface roughness and inhomogeneous size distribution of LMNO particles make quantitative analysis with XPS unreliable. Figure 2.6d displays packing densities of silane molecules computed from XPS collected on PLD deposited LMNO films (XPS spectra in Figure 2.7) which exhibit sufficiently low surface roughness (ca. 30 nm) for quantitative modelling. Densities are calculated from the relative intensities of the Si and Mn signals using a double layer model to approximate the composition of the surface as described elsewhere.^{42, 43} Briefly, the double layer model treats the surface as a smooth, semi-infinite substrate (LMNO) with an attenuating overlayer (silane coating) which inelastically scatters emitted photoelectrons. From this model theoretical signal intensities for Mn and Si were computed for a perfectly packed monolayer. These values were then compared to values obtained from experiment to obtain an approximation of the actual packing density. A more detailed description of the model can be found in Appendix A.

From the double layer model, packing densities on the order of 10^{13} molecules cm^{-2} are obtained for each of the coatings studied. For context, the packing density of a complete, crystalline, silane monolayer on silica is on the order of 10^{14} molecules cm^{-2} .²¹ The decrease in packing density of the film compared to those expected from monolayers on silica is likely due to the polycrystallinity of the PLD substrate, and the lower degree of hydroxylation at the spinel

oxide surface.⁴⁴ Nonetheless, the different coatings display packing densities within a factor of four of one another. This result suggests that while packing density may play a role, it is likely not the primary cause of the electrochemical behavior of modified electrodes. Additionally, the sub-monolayer packing densities obtained from XPS analysis suggest that deposition of polysiloxane at the LMNO surface, which would interfere with the quality of the surface coating, is absent during surface modification. These results coupled with electrochemical capacitance values obtained from impedance spectra (figure 2.4b) support the conclusion that the silane coating forms a sub-monolayer film conformally coating the LMNO surface.

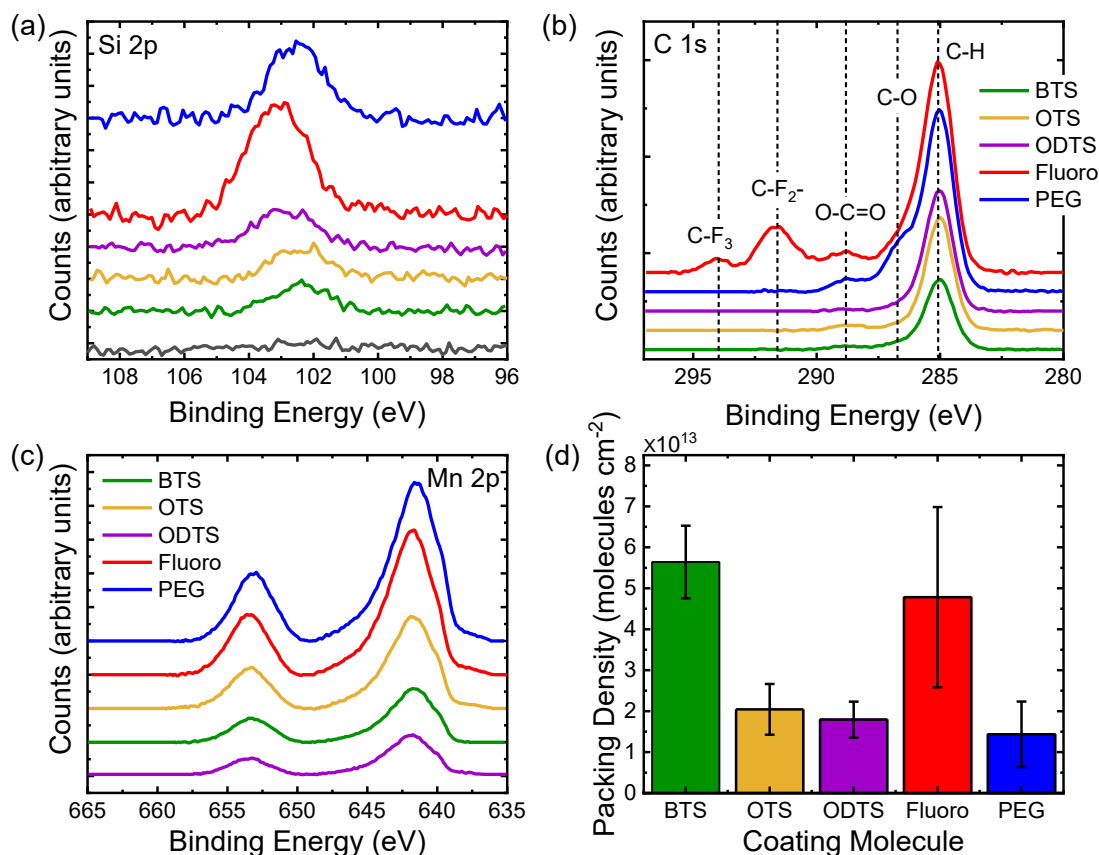


Figure 2.7. XPS characterization of functionalized LMNO thin films. Spectra are stacked in ascending order as follows: LMNO-black, BTS-green, OTS-orange, ODTS-purple, Fluoro-red, PEG-blue. XPS spectra are collected in the a) Si 2p b) C 1s and c) Mn 2p spectral regions. d) Results of double layer analysis used to approximate the molecular packing density of organosilanes at the LMNO surface.

2.3.3 Quantification of Mn Dissolution

To elucidate the origin of the improved galvanostatic performance seen in Figure 2.5d, we explored the impact of surface modification on known decomposition pathways in high-voltage electrodes. Previous literature has identified transition metal dissolution as one of the principal causes of capacity fade in oxide electrodes.^{19, 45} Specifically, spinel cathode materials containing Mn are plagued by transition metal dissolution from the electrode surface, mediated by both the electrolyte and its decomposition products.^{11, 46} To explore the impact silane coatings have on Mn dissolution, active materials were soaked in electrolyte for an extended period of time and analyzed with ICP-MS. Figure 2.8 presents the total Mn concentration detected in recovered electrolytes after soaking. All coatings reduce the quantity of Mn leached into the electrolyte relative to the unmodified electrode. Alkane and fluoroalkane coatings result in the most substantial changes, while PEG functionalization only slightly changes the behavior of the substrate. With the exception of the PEG modified surface, these results align with the overall improvements observed in galvanostatic capacity retention. Notably, previous investigations of fluoroalkyl modified LMNO reported similar effects and attributed improvements in galvanostatic cycling largely to the reduction in Mn dissolution at the electrode surface.^{18, 19} Moreover, results obtained from alkyl and PEG silanes are consistent with those obtained from alkylphosphonate modified LMO.¹⁷

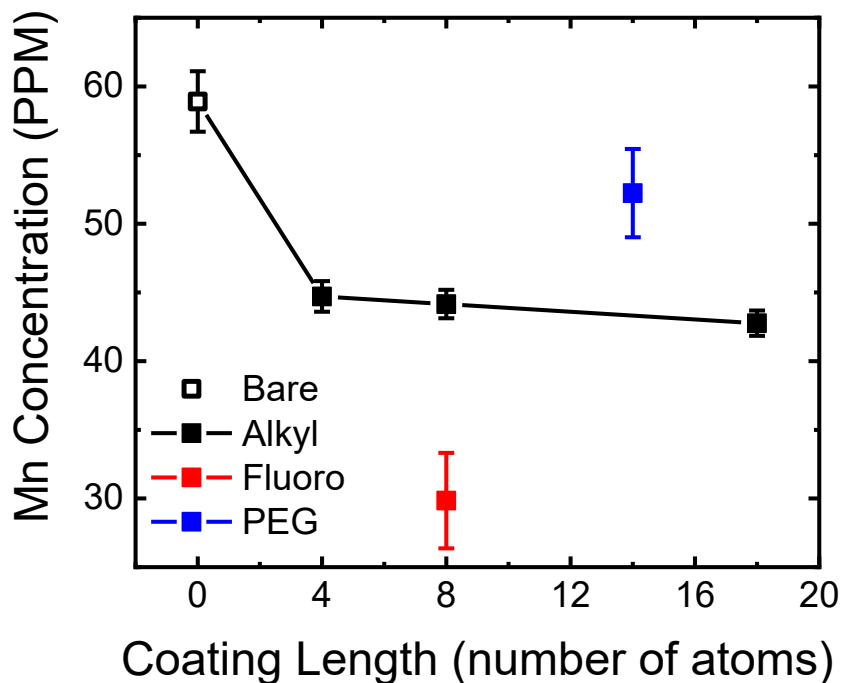


Figure 2.8. Mn concentrations obtained from electrolyte after soaking modified LMNO powders for 96 h.

2.3.4 Interfacial Interactions of Modified LMNO

Modification of the LMNO surface likely changes the electrode/electrolyte interactions and thereby transition-metal dissolution from the cathode. To verify that this is the origin of the behavior observed in Figure 5 and to further understand how electrolyte interacts with the modified LMNO surface, contact angle measurements were conducted on coated and uncoated LMNO thin films. Figure 2.9a displays images obtained from measurements using both 1M LiPF₆ in EC/EMC electrolyte (top) and water (bottom) as the droplet. Contact angles were computed from an average of 20 tangent line measurements between the droplet and the substrate. These values were then plotted against the length of the silane coating for both electrolyte and water as shown in Figures 2.9b and 2.9c, respectively. The LMNO surface exhibits a substantial change in wetting behavior after silane modification. All alkyl chains increase the contact angle for both electrolyte and water,

with increasing chain length resulting in increased contact angles. This result is consistent with prior work utilizing silane monolayers on glass surfaces.⁴⁷ Changing the composition of the chain from an alkane to a fluorinated alkane or PEG chain results in even more dramatic changes to the contact angle. Surfaces coated with the fluorinated silane demonstrates the highest contact angle, while PEG demonstrates the lowest (even lower than that of the unmodified LMNO surface when in contact with electrolyte). These behaviors are similar to those reported previously using SAMs with chains of similar composition.^{48, 49}

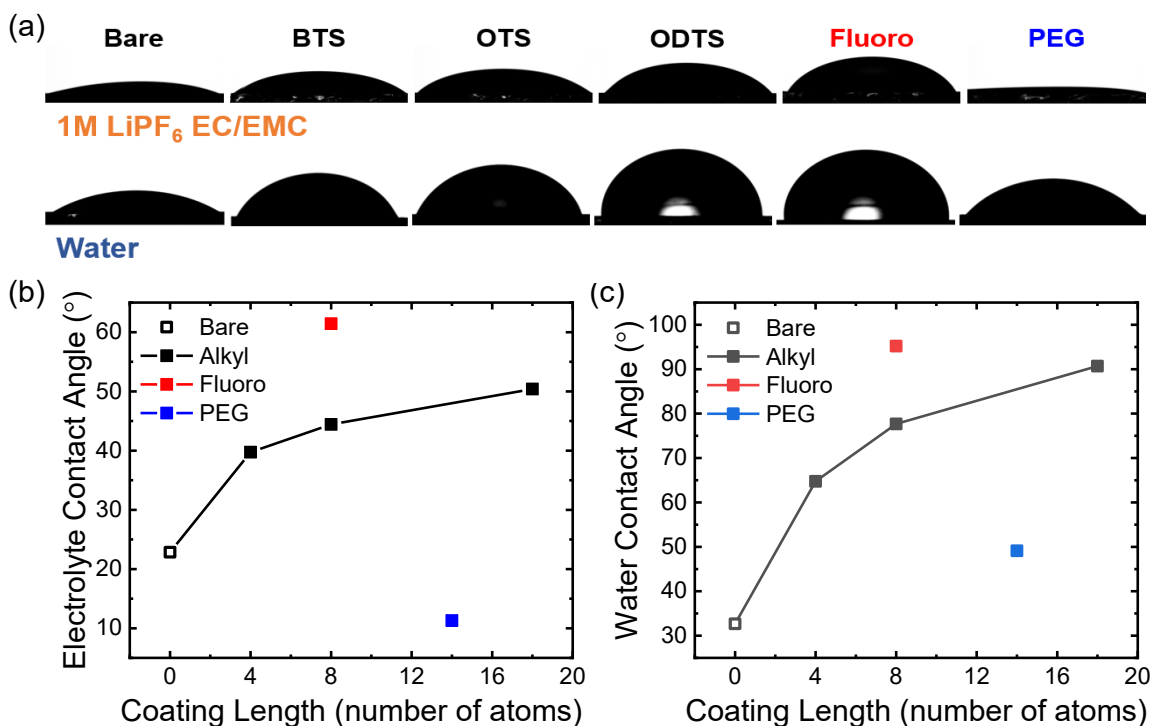


Figure 2.9. a) Contact angle measurements conducted on coated LMNO thin films using either water or an electrolyte solution (1M LiPF₆, EC/EMC). Plots depict measured contact angles plotted against coating length using b) electrolyte or c) water

Comparing the two solvent systems (water and electrolyte) electrolyte droplets demonstrate lower contact angles than water droplets at equivalent surfaces. This effect is attributed to the change in dielectric constant of the solvent. Water has a high relative dielectric constant of 80.4 at room temperature, while the mixed EC/EMC solvent system has a substantially

lower dielectric of 33.6.⁵⁰ This decrease results in lower surface tension at the interface between the droplet and the substrate and, consequently, more acute contact angles.⁵¹ The trends observed in contact angle measurements demonstrate that changing the identity of the monolayer affects the affinity of the surface towards electrolyte, with alkyl and fluoroalkyl coatings exhibiting progressively lower electrolyte affinity, and PEG coatings enhancing surface/electrolyte interactions. This mirrors the trends observed in Figure 5 and suggests that reducing the electrolyte affinity of the surface results in a diminution in Mn dissolution. This aligns well with previous results obtained from alkylphosphonate modified LMO electrodes.¹⁷

2.3.5 Quantifying Electrolyte Oxidation

While Mn dissolution offers insights into the improved stability afforded by surface modification with hydrophobic silanes, it cannot adequately explain the improved capacity retention exhibited by PEG functionalized LMNO. To elucidate the origin of this stability another well-known decomposition pathway was explored, namely electrolyte oxidation. Due to the high potentials accessed by LMNO electrodes oxidative electrolyte decomposition is inevitable during charge. The continuous decomposition of the electrolyte progressively worsens battery performance during prolonged operation. To investigate the impact surface modification has on this process, modified electrodes were cycled voltammetrically to elevated potentials (5.5 V vs Li/Li⁺). At such high potentials, lithium deintercalation has occurred, and the observed current can be attributed to deleterious oxidative processes including electrolyte decomposition.¹⁸ Figure 2.10 displays voltammograms collected from coated and uncoated electrodes at elevated potentials. Coated LMNO electrodes exhibit lower current densities at 5.5 V relative to unmodified LMNO, with the PEG-coated electrode demonstrating the lowest oxidative current density overall. This suggests that the PEG coating is the most effective at electronically isolating the electrode surface

from the electrolyte. This result is surprising as the PEG coating is neither the longest, or the most densely packed of the systems studied.

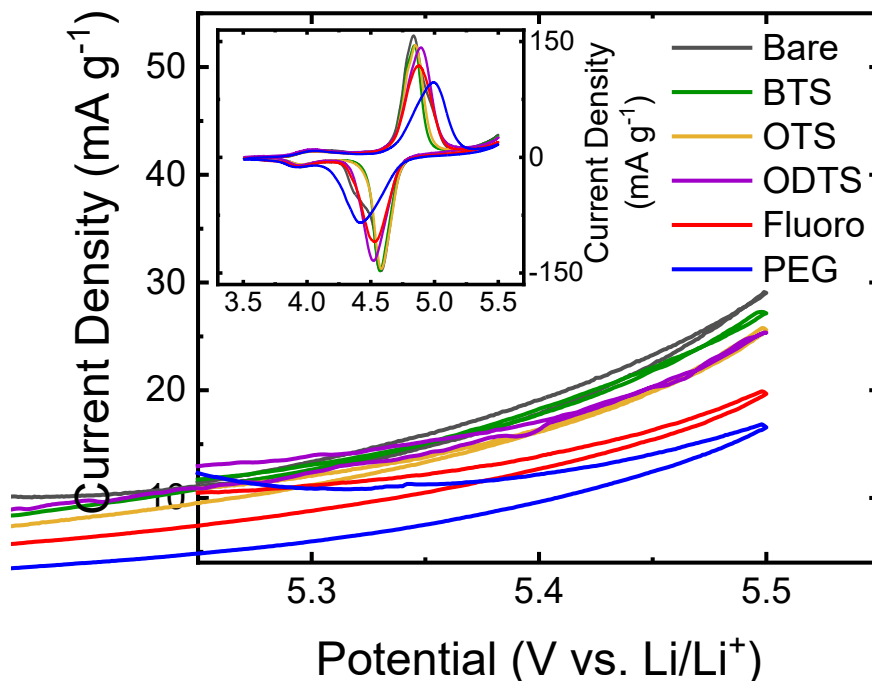


Figure 2.10. Cyclic voltammograms collected between 3.5 and 5.5 V vs Li/Li⁺ focusing on the high-potential region to highlight the current attributed to electrolyte oxidation. The full voltammograms for each electrode are presented in the inset.

2.3.6 Post-Mortem XPS Analysis

To determine the origin of the diminished electrolyte oxidation observed in PEG coated electrodes, as well as the role silane surface modification plays on the chemical composition of the CEI, post-mortem XPS characterization was conducted on all electrodes after 100 charge/discharge cycles. Figure 2.11a presents Si 2p XPS of composite electrodes (A collection of all XPS spectral regions investigated is presented in Appendix A). Each of the coated electrodes demonstrates a peak in the same binding energy region as those observed for the uncycled particles, suggesting that the silane coating remains adhered to the surface throughout cell cycling. The Si

signal intensities of OTS and ODTs, already low for uncycled electrodes, are difficult to discern above background for cycled electrodes due to the deposition of electrolyte decomposition products comprising the cathode electrolyte interphase. Despite the poor signal intensity, the similarity in structure and binding mechanism among silane molecules suggest that OTS and ODTs likely remain at the electrode surface as well. While the retention of the Si signal reflects the strength of the siloxane-surface bond anchoring the chain to the cathode, it does not guarantee that the pendant chain remains chemically unchanged during battery operation.

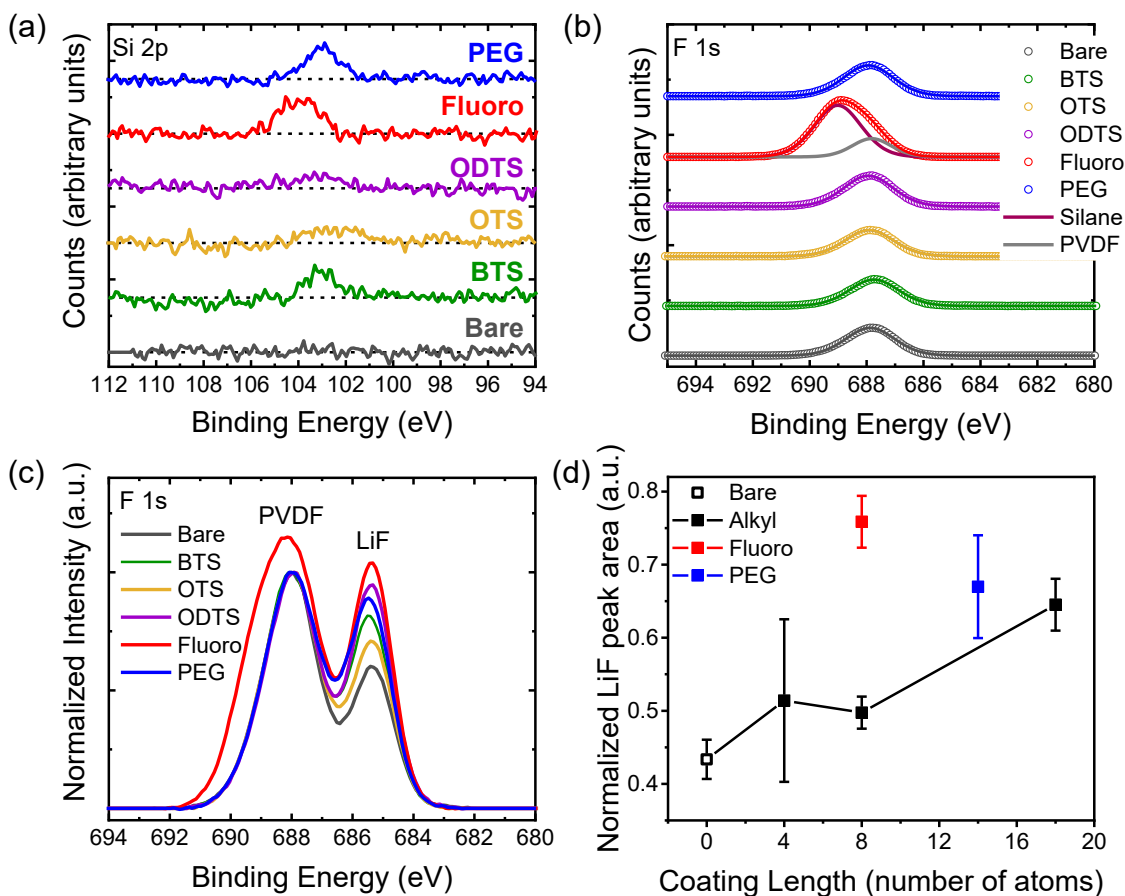


Figure 2.11. a) Si 2p XPS of cycled electrodes, dashed lines signify signal baseline. b) F 1s XPS spectra and peak fittings of pre-cycled composite cathodes. c) Normalized F 1s XPS spectra of cycled LMNO electrodes. d) normalized LiF peak area plotted against coating length for cycled electrodes.

Figure 2.11b displays F 1s XPS of uncycled electrodes. With the exception of fluorinated LMNO, a single peak is observed in the F 1s region associated with PVDF binder (688.1 eV). The presence of an additional feature at 689.2 eV for the Fluoro-modified surface matches the binding energy expected for a perfluorinated alkyl chain.³⁹ Figure 2.11c presents F 1s XPS from the same electrodes after cycling normalized to the PVDF signal. A second peak at lower binding energy (685.4 eV) is observed after cycling, matching literature reports of LiF salt deposited at the surface.⁵² To compare the surface concentrations of LiF between different electrodes, the PVDF peak is treated as a reference signal as its concentration within composite electrodes remains roughly constant. Additionally, previous work has shown that decomposition of electrolyte on PVDF can be regarded as minimal, making it a more stable reference signal than other, electroactive components of the electrode.⁵² Figure 2.11d presents the peak area of LiF calculated from the spectra in Figure 8c after normalization. All coated surfaces demonstrate an enhancement in LiF deposition relative to unmodified LMNO. Hydrophobic silane coatings (BTS, OTS, ODTs, Fluoro) demonstrate a positive correlation between the hydrophobicity of the surface (as determined by contact angle measurements) and the amount of LiF deposited during battery cycling (Figure 2.12). Further, and more surprisingly, the PEG coated surface also demonstrates elevated LiF deposition, greater than that observed for any of the alkyl coatings.

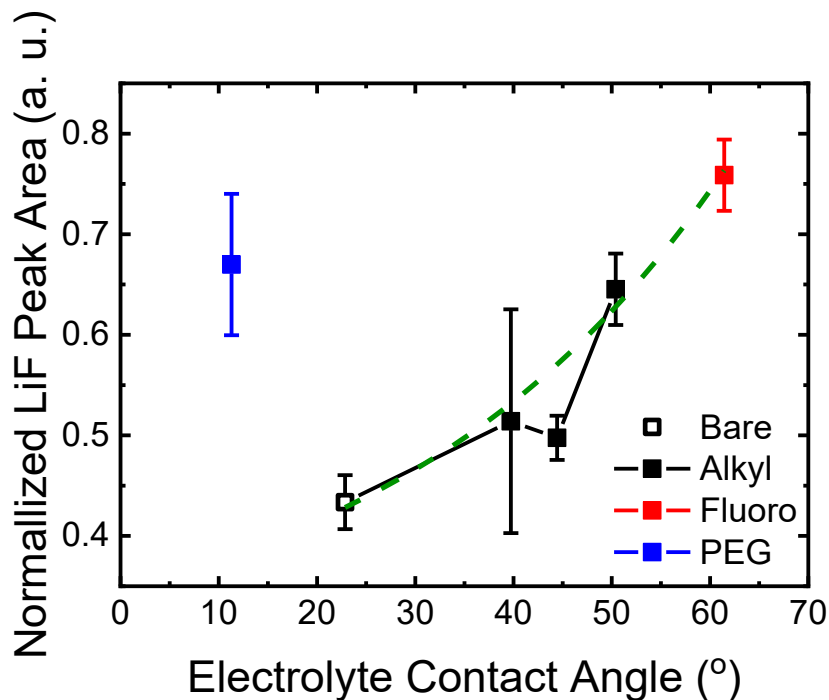


Figure 2.12. Normalized LiF peak area obtained from XPS of modified and unmodified LMNO electrodes after 100 charge/discharge cycles plotted against electrolyte contact angle. The dashed line is meant to highlight the general upward trend between the hydrophobicity of the surface, and the concentration of LiF present in the CEI.

The high concentrations of LiF on PEG and Fluoro modified LMNO electrodes reflect the high galvanostatic capacity retentions observed for each of these coatings (Figure 2.13), suggesting that the presence of LiF imparts greater electrode stability during extended galvanostatic cycling. Similarly, this result explains the low electrolyte oxidation current observed with PEG-modified LMNO. Due to the extremely low electronic conductivity of LiF,⁵³ it is likely that the increased deposition of LiF at the electrode surface effectively insulates the electrolyte from the highly oxidizing character of the LMNO surface during charge. Thus, the elevated concentration of LiF for both the Fluoro and PEG modified LMNO results in less severe electrolyte oxidation during cell operation and consequently greater long-term cycling stability. This conclusion is supported by previous work on both LiF-coated anodes and cathodes which demonstrate improved reductive and oxidative stabilities respectively relative to their uncoated counterparts.^{54, 55} Additionally, the

high concentration of LiF at the PEG-modified surface explains the unexpectedly high impedance observed for PEG functionalized LMNO (Figure 2.3) due to the low ionic conductivity of LiF.⁵⁶

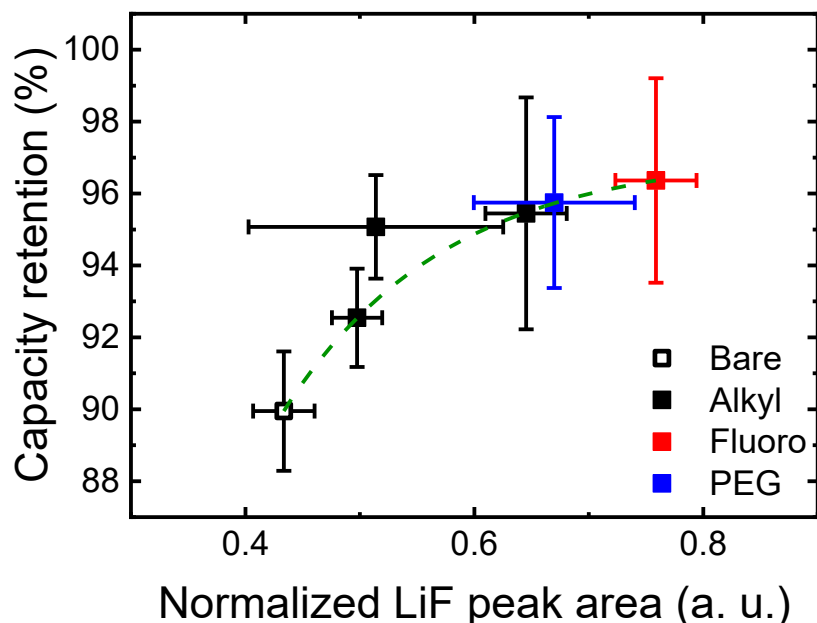


Figure 2.13. Capacity retention of modified and unmodified LMNO cells after 100 charge/discharge cycles plotted against the normalized LiF peak area collected from post-mortem XPS of cycled electrodes. The dashed line highlights the positive correlation between capacity retention and LiF surface concentration.

2.3.7. Proposed Mechanism for LiF Deposition at Modified Electrode Surfaces

XPS results demonstrate that silane coatings facilitate the deposition of LiF at modified electrode surfaces, and that, with the exception of the PEG silane, this effect is more pronounced as the hydrophobicity of the surface increases. From previous computational and experimental investigations of hydrophobic interfaces it is known that at hydrophobic boundaries, polar solvents preferentially partition away from the interface to maximize interactions with other solvent molecules.⁵⁷ Similarly, the dielectric constant and solvating power of polar solvent molecules diminishes when in close proximity to hydrophobic surfaces.⁵⁸ These behaviors directly impact

the ability of solvent molecules to solvate ionic species at these interfaces. Both calculations and experiments show that at hydrophobic boundaries ion pairs and salts are not as readily solvated by the solvent,^{59, 60} and that the ability of solvent to dissolve salt species at the surface diminishes as the surface becomes more hydrophobic.⁶¹ These observations suggest that salt precipitation is enhanced at hydrophobic interfaces due to the unfavorable interactions between the interface and polar solvent molecules.

The presence of a hydrophobic modifier at the LMNO surface affords an explanation for the observed phenomena. On unmodified oxide electrodes the high dielectric solvent EC preferentially adsorbs over linear carbonates which exhibit lower polarity.⁶² Upon modification of the LMNO surface with hydrophobic silanes, EC is no longer preferentially adsorbed at the LMNO surface, but rather favors the bulk solvent where its dipole moment can be maximally shielded. As cyclic carbonates are primarily responsible for dissolving the highly polar LiPF₆ salt,⁶³ the loss of EC (and the reduction of the dipole moment of any residual EC that remains) results in LiPF₆ precipitation at the surface. Un-solvated LiPF₆ can spontaneously disproportionate to form LiF and PF₅.⁶⁴ Thus, increasing the hydrophobicity of the LMNO surface increases the local concentration of LiPF₆ favoring the disproportionation and deposition of LiF at the electrode surface.

While desolvation of LiPF₆ explains the increased LiF deposition at LMNO surfaces modified with hydrophobic silanes, this concept does not explain the high concentration of LiF found at PEG functionalized electrodes. Another common decomposition reaction for the electrolyte involves LiPF₆ hydrolysis in the presence of adventitious water or reactive oxo and hydroxo species to form HF, which can react with a lithium source (solvated Li ions or the oxide surface) to form LiF.⁶⁵ Contact angle measurements reveal that the hydrophilic PEG silane has an

affinity for water comparable to that exhibited by the unmodified LMNO surface. This affinity, coupled with the ether functionality of the chain allows PEG to bind water much more strongly than the other LMNO electrodes, potentially resulting in an elevated surface concentration of water capable of hydrolyzing LiPF_6 . Further, it is well-known that ether-based molecules (including short chain glycols) exhibit oxidative instability at the surface of high-voltage electrodes.⁶³ It is likely that the PEG chain is, at least partially, oxidized producing reactive intermediaries which are capable of decomposing LiPF_6 or PF_6^- to LiF at the surface as observed in previous investigations of high-energy Li-ion cathodes.⁶⁵ These formation pathways suggest that both hydrophobic and hydrophilic silanes enhance the deposition of LiF at the LMNO surface, imparting additional stability to the CEI, and resulting in the observed improvements in galvanostatic capacity retention.

2.4 Conclusions

In this investigation we demonstrated successful functionalization of the high-voltage spinel cathode $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ with organo-silane monolayers through a facile solution phase deposition reaction. We show that by controlling the structure of the coating moiety the surface properties, and consequently the electrochemical behavior, of LMNO electrodes can be tuned. All modified electrodes demonstrate improved capacity retention relative to unmodified LMNO and the extent of this improvement is tied to the structure of the monolayer. While surface modification with alkyl chains elicits modest improvements to capacity retention, Fluoroalkyl and PEG modification demonstrate the highest overall capacity retentions suggesting a substantial increase in the stability of the electrode/electrolyte interphase.

Electrolyte soaking experiments demonstrate that while the suppression of Mn dissolution from the cathode surface afforded by hydrophobic coatings (BTS, OTS, ODTS, and Fluoro)

reflects their improved capacity retentions, it cannot adequately explain the high capacity retention exhibited by PEG modified LMNO. Post-mortem XPS characterization of modified electrodes reveals that the operando deposition of LiF at the electrode surface is enhanced by the presence of the silane coating. Further, the surface concentration of LiF correlates with the observed improvements to galvanostatic capacity retention. High-voltage cyclic voltammetry of modified LMNO suggests that the elevated surface concentration of LiF diminishes electrolyte oxidation due to the insulating character of LiF. From these results we conclude that while the suppression of Mn dissolution plays a role in improving galvanostatic performance, improving the oxidative stability of the cathode surface by LiF deposition is of equal importance.

Overall, we demonstrate that surface modification of LMNO by organo-silanes offers a way to control the interfacial dynamics of the cathode during battery operation. The composition and stability of the CEI are intimately tied to the structure of the coating moiety with the most evident consequence being the deposition of inorganic species like LiF at the electrode surface. Furthermore, while suppression of Mn dissolution is partly responsible for improved electrode stability, a key component to the good capacity retention of modified electrodes is the presence of LiF at the interphase, improving the anodic stability of the CEI. This enhanced anodic stability diminishes the severity of electrolyte oxidation allowing for more stable long-term cell performance. Thus, the application of simple, insulating inorganic salts at the electrode interphase is expected to benefit overall electrode stability, and electrode modifications which facilitate the deposition of these species will likely elicit improved long-term performance from high-voltage lithium-ion cathodes.

2.5 References

- (1) Goodenough, J. B.; Park, K. S., The Li-Ion Rechargeable Battery: A Perspective. *J Am Chem Soc* **2013**, *135* (4), 1167-1176.
- (2) Manthiram, A., An Outlook on Lithium Ion Battery Technology. *Acs Central Sci* **2017**, *3* (10), 1063-1069.
- (3) Li, W. D.; Song, B. H.; Manthiram, A., High-voltage positive electrode materials for lithium-ion batteries. *Chem Soc Rev* **2017**, *46* (10), 3006-3059.
- (4) Schoiber, J.; Berger, R. J. F.; Yada, C.; Miki, H.; Husing, N., A Two-Step Synthesis for Li₂CoPO₄F as High-Voltage Cathode Material. *J Electrochem Soc* **2015**, *162* (14), A2679-A2683.
- (5) Manthiram, A.; Chemelewski, K.; Lee, E. S., A perspective on the high-voltage LiMn_{1.5}Ni_{0.5}O₄ spinel cathode for lithium-ion batteries. *Energ Environ Sci* **2014**, *7* (4), 1339-1350.
- (6) Aurbach, D., Review of selected electrode-solution interactions which determine the performance of Li and Li ion batteries. *J Power Sources* **2000**, *89* (2), 206-218.
- (7) Peled, E.; Menkin, S., Review-SEI: Past, Present and Future. *J Electrochem Soc* **2017**, *164* (7), A1703-A1719.
- (8) Duncan, H.; Abu-Lebdeh, Y.; Davidson, I. J., Study of the Cathode-Electrolyte Interface of LiMn_{1.5}Ni_{0.5}O₄ Synthesized by a Sol-Gel Method for Li-Ion Batteries. *J Electrochem Soc* **2010**, *157* (4), A528-A535.
- (9) Jang, D. H.; Shin, Y. J.; Oh, S. M., Dissolution of spinel oxides and capacity losses in 4V Li/LixMn₂O₄ coils. *J Electrochem Soc* **1996**, *143* (7), 2204-2211.

- (10) Gummow, R. J.; Dekock, A.; Thackeray, M. M., Improved Capacity Retention in Rechargeable 4v Lithium Lithium Manganese Oxide (Spinel) Cells. *Solid State Ionics* **1994**, *69* (1), 59-67.
- (11) Bhandari, A.; Bhattacharya, J., Review-Manganese Dissolution from Spinel Cathode: Few Unanswered Questions. *J Electrochem Soc* **2017**, *164* (2), A106-A127.
- (12) Harris, O. C.; Tang, M. H., Molecular Probes Reveal Chemical Selectivity of the Solid-Electrolyte Interphase. *J Phys Chem C* **2018**, *122* (36), 20632-20641.
- (13) Esbenschade, J. L.; Gewirth, A. A., Effect of Mn and Cu Addition on Lithiation and SEI Formation on Model Anode Electrodes. *J Electrochem Soc* **2014**, *161* (4), A513-A518.
- (14) Xu, X. L.; Deng, S. X.; Wang, H.; Liu, J. B.; Yan, H., Research Progress in Improving the Cycling Stability of High Voltage LiNi_{0.5}Mn_{1.5}O₄ Cathode in Lithium-Ion Battery. *Nano-Micro Lett* **2017**, *9* (2).
- (15) Chen, Z. H.; Qin, Y.; Amine, K.; Sun, Y. K., Role of surface coating on cathode materials for lithium-ion batteries. *J Mater Chem* **2010**, *20* (36), 7606-7612.
- (16) Zuo, D. X.; Tian, G. L.; Li, X.; Chen, D.; Shu, K. Y., Recent progress in surface coating of cathode materials for lithium ion secondary batteries. *J Alloy Compd* **2017**, *706*, 24-40.
- (17) Nicolau, B. G.; Petronico, A.; Letchworth-Weaver, K.; Ghadar, Y.; Haasch, R. T.; Soares, J. A. N. T.; Rooney, R. T.; Chan, M. K. Y.; Gewirth, A. A.; Nuzzo, R. G., Controlling Interfacial Properties of Lithium-Ion Battery Cathodes with Alkylphosphonate Self-Assembled Monolayers. *Adv Mater Interfaces* **2018**, *5* (10).

- (18) Zettsu, N.; Kida, S.; Uchida, S.; Teshima, K., Sub-2 nm Thick Fluoroalkylsilane Self-Assembled Monolayer-Coated High Voltage Spinel Crystals as Promising Cathode Materials for Lithium Ion Batteries. *Sci Rep-Uk* **2016**, *6*.
- (19) Kim, D. W.; Uchida, S.; Shiiba, H.; Zettsu, N.; Teshima, K., New Insight for Surface Chemistries in Ultra-thin Self-assembled Monolayers Modified High-voltage Spinel Cathodes. *Sci Rep-Uk* **2018**, *8*.
- (20) Chae, S.; Soon, J.; Jeong, H.; Lee, T. J.; Ryu, J. H.; Oh, S. M., Passivating film artificially built on $\text{LiNi}_{0.5}\text{Mn}_{1.5}\text{O}_4$ by molecular layer deposition of (pentafluorophenylpropyl)trimethoxysilane. *J Power Sources* **2018**, *392*, 159-167.
- (21) Ulman, A., Formation and structure of self-assembled monolayers. *Chemical Reviews* **1996**, *96* (4), 1533-1554.
- (22) Bennes, J.; Ballandras, S.; Cherioux, F., Easy and versatile functionalization of lithium niobate wafers by hydrophobic trichlorosilanes. *Appl Surf Sci* **2008**, *255* (5), 1796-1800.
- (23) McGovern, M. E.; Kallury, K. M. R.; Thompson, M., Role of Solvent on the Silanization of Glass with Octadecyltrichlorosilane. *Langmuir* **1994**, *10* (10), 3607-3614.
- (24) Schulz, N.; Hausbrand, R.; Dimesso, L.; Jaegermann, W., XPS-Surface Analysis of SEI Layers on Li-Ion Cathodes: Part I. Investigation of Initial Surface Chemistry. *J Electrochem Soc* **2018**, *165* (5), A819-A832.
- (25) G. Johansson, J. H., A. Berndtsson, M. Klasson, R. Nilsson, Calibration of electron spectra. *Journal of Electron Spectroscopy and Related Phenomena* **1973**, *2* (3), 295-317.
- (26) Hai, B.; Shukla, A. K.; Duncan, H.; Chen, G. Y., The effect of particle surface facets on the kinetic properties of $\text{LiMn}_{1.5}\text{Ni}_{0.5}\text{O}_4$ cathode materials. *J Mater Chem A* **2013**, *1* (3), 759-769.

- (27) Kebede, M. A.; Yannopoulos, S. N.; Sygellou, L.; Ozoemena, K. I., High-Voltage LiNi_{0.5}Mn_{1.5}O₄-delta Spinel Material Synthesized by Microwave-Assisted Thermo-Polymerization: Some Insights into the Microwave-Enhancing Physico-Chemistry. *J Electrochem Soc* **2017**, *164* (13), A3259-A3265.
- (28) Aurbach, D.; Levi, M. D.; Levi, E.; Teller, H.; Markovsky, B.; Salitra, G.; Heider, U.; Heider, L., Common electroanalytical behavior of Li intercalation processes into graphite and transition metal oxides. *J Electrochem Soc* **1998**, *145* (9), 3024-3034.
- (29) Bisquert, J.; Compte, A., Theory of the electrochemical impedance of anomalous diffusion. *J Electroanal Chem* **2001**, *499* (1), 112-120.
- (30) Zhuang, Q. C.; Wei, T.; Du, L. L.; Cui, Y. L.; Fang, L.; Sun, S. G., An Electrochemical Impedance Spectroscopic Study of the Electronic and Ionic Transport Properties of Spinel LiMn₂O₄. *J Phys Chem C* **2010**, *114* (18), 8614-8621.
- (31) Hjelm, A. K.; Lindbergh, G., Experimental and theoretical analysis of LiMn₂O₄ cathodes for use in rechargeable lithium batteries by electrochemical impedance spectroscopy (EIS). *Electrochim Acta* **2002**, *47* (11), 1747-1759.
- (32) Maoz, R.; Sagiv, J., Penetration-Controlled Reactions in Organized Monolayer Assemblies .2. Aqueous Permanganate Interaction with Self-Assembling Monolayers of Long-Chain Surfactants. *Langmuir* **1987**, *3* (6), 1045-1051.
- (33) Long, L. Z.; Wang, S. J.; Xiao, M.; Meng, Y. Z., Polymer electrolytes for lithium polymer batteries. *J Mater Chem A* **2016**, *4* (26), 10038-10069.
- (34) Schweiger, H. G.; Obeidi, O.; Komesker, O.; Raschke, A.; Schiemann, M.; Zehner, C.; Gehnen, M.; Keller, M.; Birke, P., Comparison of Several Methods for Determining the Internal Resistance of Lithium Ion Cells. *Sensors-Basel* **2010**, *10* (6), 5604-5625.

- (35) Hur, J.; Kim, I. T., Synthesis and Electrochemical Performance of Reduced Graphene Oxide/AlPO₄-coated LiMn_{1.5}Ni_{0.5}O₄ for Lithium-ion Batteries. *B Korean Chem Soc* **2014**, *35* (12), 3553-3558.
- (36) Shircliff, R. A.; Stradins, P.; Moutinho, H.; Fennell, J.; Ghirardi, M. L.; Cowley, S. W.; Branz, H. M.; Martin, I. T., Angle-Resolved XPS Analysis and Characterization of Mono layer and Multi layer Si lane Films for DNA Coupling to Silica. *Langmuir* **2013**, *29* (12), 4057-4067.
- (37) Wilson, K.; Stancescu, M.; Das, M.; Rumsey, J.; Hickman, J., Direct patterning of coplanar polyethylene glycol alkylsilane monolayers by deep-ultraviolet photolithography as a general method for high fidelity, long-term cell patterning and culture. *J Vac Sci Technol B* **2011**, *29* (2).
- (38) Wang, H. X.; Fang, J.; Cheng, T.; Ding, J.; Qu, L. T.; Dai, L. M.; Wang, X. G.; Lin, T., One-step coating of fluoro-containing silica nanoparticles for universal generation of surface superhydrophobicity. *Chem Commun* **2008**, (7), 877-879.
- (39) John F. Moulder, W. F. S., Peter E. Sobol, Kenneth D. Bomben, *Handbook of X-ray Photoelectron Spectroscopy*. Perkin-Elmer Corporation: 1992.
- (40) Wang, H.; Ben, L. B.; Yu, H. L.; Chen, Y. Y.; Yang, X. N.; Huang, X. J., Understanding the effects of surface reconstruction on the electrochemical cycling performance of the spinel LiNi_{0.5}Mn_{1.5}O₄ cathode material at elevated temperatures. *J Mater Chem A* **2017**, *5* (2), 822-834.
- (41) Taucher, T. C.; Hehn, I.; Hofmann, O. T.; Zharnikov, M.; Zojer, E., Understanding Chemical versus Electrostatic Shifts in X-ray Photoelectron Spectra of Organic Self-Assembled Monolayers. *J Phys Chem C* **2016**, *120* (6), 3428-3437.

- (42) Petrovykh, D. Y.; Kimura-Suda, H.; Whitman, L. J.; Tarlov, M. J., Quantitative analysis and characterization of DNA immobilized on gold. *J Am Chem Soc* **2003**, *125* (17), 5219-5226.
- (43) Haasch, R. T., X-Ray Photoelectron Spectroscopy (XPS) and Auger Electron Spectroscopy (AES). In *Practical Materials Characterization*, M., S., Ed. Springer, New York, NY: 2014; pp 93-132.
- (44) Chen, L.; Warburton, R. E.; Chen, K. S.; Libera, J. A.; Johnson, C.; Yang, Z. Z.; Hersam, M. C.; Greeley, J. P.; Elam, J. W., Mechanism for Al₂O₃ Atomic Layer Deposition on LiMn₂O₄ from In Situ Measurements and Ab Initio Calculations. *Chem-US* **2018**, *4* (10), 2418-2435.
- (45) Ziv, B.; Borgel, V.; Aurbach, D.; Kim, J. H.; Xiao, X. C.; Powell, B. R., Investigation of the Reasons for Capacity Fading in Li-Ion Battery Cells. *J Electrochem Soc* **2014**, *161* (10), A1672-A1680.
- (46) Komaba, S.; Kumagai, N.; Sasaki, T.; Miki, Y., Manganese dissolution from lithium doped Li-Mn-O spinel cathode materials into electrolyte solution. *Electrochemistry* **2001**, *69* (10), 784-787.
- (47) Booth, B. D.; Vilt, S. G.; Ben Lewis, J.; Rivera, J. L.; Buehler, E. A.; McCabe, C.; Jennings, G. K., Tribological Durability of Silane Monolayers on Silicon. *Langmuir* **2011**, *27* (10), 5909-5917.
- (48) Cerruti, M.; Fissolo, S.; Carraro, C.; Ricciardi, C.; Majumdar, A.; Maboudian, R., Poly(ethylene glycol) monolayer formation and stability on gold and silicon nitride substrates. *Langmuir* **2008**, *24* (19), 10646-10653.

- (49) Colorado, R.; Lee, T. R., Wettabilities of self-assembled monolayers on gold generated from progressively fluorinated alkanethiols. *Langmuir* **2003**, *19* (8), 3288-3296.
- (50) Hall, D. S.; Self, J.; Dahn, J. R., Dielectric Constants for Quantum Chemistry and Li-Ion Batteries: Solvent Blends of Ethylene Carbonate and Ethyl Methyl Carbonate. *J Phys Chem C* **2015**, *119* (39), 22322-22330.
- (51) Berim, G. O.; Ruckenstein, E., On the shape and stability of a drop on a solid surface. *J Phys Chem B* **2004**, *108* (50), 19330-19338.
- (52) Schulz, N.; Hausbrand, R.; Wittich, C.; Dimesso, L.; Jaegermann, W., XPS-Surface Analysis of SEI Layers on Li-Ion Cathodes: Part II. SEI-Composition and Formation inside Composite Electrodes. *J Electrochem Soc* **2018**, *165* (5), A833-A846.
- (53) Sirdeshmukh, D. B.; Sirdeshmukh, L.; Subhadra, K. G., Dielectric and Electrical Properties. In *Alkali Halides: A Handbook of Physical Properties*, Springer Berlin Heidelberg: Berlin, Heidelberg %@ 978-3-662-04341-7, 2001; pp 137-153.
- (54) Shi, S. J.; Tu, J. P.; Tang, Y. Y.; Zhang, Y. Q.; Liu, X. Y.; Wang, X. L.; Gu, C. D., Enhanced electrochemical performance of LiF-modified LiNi_{1/3}Co_{1/3}Mn_{1/3}O₂ cathode materials for Li-ion batteries. *Journal of Power Sources* **2013**, *225*, 338-346.
- (55) Lang, J. L.; Long, Y. Z.; Qu, J. L.; Luo, X. Y.; Wei, H. H.; Huang, K.; Zhang, H. T.; Qi, L. H.; Zhang, Q. F.; Li, Z. C.; Wu, H., One-pot solution coating of high quality LiF layer to stabilize Li metal anode. *Energy Storage Mater* **2019**, *16*, 85-90.
- (56) Pan, J.; Cheng, Y. T.; Qi, Y., General method to predict voltage-dependent ionic conduction in a solid electrolyte coating on electrodes. *Phys Rev B* **2015**, *91* (13).

- (57) Jaynes, W. F.; Boyd, S. A., Hydrophobicity of Siloxane Surfaces in Smectites as Revealed by Aromatic Hydrocarbon Adsorption from Water. *Clay Clay Miner* **1991**, *39* (4), 428-436.
- (58) Fumagalli, L.; Esfandiar, A.; Fabregas, R.; Hu, S.; Ares, P.; Janardanan, A.; Yang, Q.; Radha, B.; Taniguchi, T.; Watanabe, K.; Gomila, G.; Novoselov, K. S.; Geim, A. K., Anomalously low dielectric constant of confined water. *Science* **2018**, *360* (6395), 1339-+.
- (59) Sato, T.; Sasaki, T.; Ohnuki, J.; Umezawa, K.; Takano, M., Hydrophobic Surface Enhances Electrostatic Interaction in Water. *Phys Rev Lett* **2018**, *121* (20).
- (60) Venkateshwaran, V.; Vembanur, S.; Garde, S., Water-mediated ion-ion interactions are enhanced at the water vapor-liquid interface. *P Natl Acad Sci USA* **2014**, *111* (24), 8729-8734.
- (61) Chen, S.; Itoh, Y.; Masuda, T.; Shimizu, S.; Zhao, J.; Ma, J.; Nakamura, S.; Okuro, K.; Noguchi, H.; Uosaki, K.; Aida, T., Subnanoscale hydrophobic modulation of salt bridges in aqueous media. *Science* **2015**, *348* (6234), 555-559.
- (62) Yu, L.; Liu, H. J.; Wang, Y.; Kuwata, N.; Osawa, M.; Kawamura, J.; Ye, S., Preferential Adsorption of Solvents on the Cathode Surface of Lithium Ion Batteries. *Angew Chem Int Edit* **2013**, *52* (22), 5753-5756.
- (63) Xu, K., Nonaqueous liquid electrolytes for lithium-based rechargeable batteries. *Chem Rev* **2004**, *104* (10), 4303-4417.
- (64) Stich, M.; Gottlinger, M.; Kurniawan, M.; Schmidt, U.; Bund, A., Hydrolysis of LiPF₆ in Carbonate-Based Electrolytes for Lithium-Ion Batteries and in Aqueous Media. *Journal of Physical Chemistry C* **2018**, *122* (16), 8836-8842.

- (65) Gueguen, A.; Streich, D.; He, M. L.; Mendez, M.; Chesneau, F. F.; Novak, P.; Berg, E. J., Decomposition of LiPF₆ in High Energy Lithium-Ion Batteries Studied with Online Electrochemical Mass Spectrometry. *J Electrochem Soc* **2016**, *163* (6), A1095-A1100.

Chapter 3: Direct Observation of Interfacial Mechanical Failure in Thiophosphate Solid Electrolytes with *Operando* X-Ray Tomography

Reprinted with the permission of Madsen, K. E.; Bassett, K. L.; Ta, K.; Sforzo B. A.; Matusik, K. E.; Kastengren, A. L.; Gewirth, A. A. Direct Observation of Interfacial Mechanical Failure in Thiophosphate Solid Electrolytes with Operando X-Ray Tomography. Adv. Mater. Interfaces 2020, 7, 2000751. Copyright 2020 Wiley-VCH GmbH.

3.1 Introduction

Since their commercialization in 1991 by the SONY corporation, lithium-ion batteries (LIBs) have become the energy storage medium of choice for small-scale applications due to their high energy and power densities relative to other battery chemistries.¹ Despite the successful implementation of LIBs in portable electronics, proliferation of Li⁺-based energy storage at larger scales (electric vehicle or grid-scale energy storage) has been hampered by insufficient energy densities, prohibitive material costs, and safety concerns.² While significant progress has been made in the development of high energy density battery materials, the implementation of these materials in practical systems without sacrificing cell longevity or safety remains a pressing scientific challenge.

One of the best studied approaches to improve the energy density of LIBs is to replace existing graphitic anodes (372 mA h g⁻¹ capacity) with metallic lithium (3860 mA h g⁻¹ capacity).³ Unfortunately, uneven lithium plating and stripping in conventional organic liquid electrolytes promotes the growth of lithium dendrites, accelerating the formation of internal short circuits. Additionally, the low thermal stability of electrolyte solvents (often mixtures of cyclic

and linear carbonates) incurs exothermic decomposition of the electrolyte in the event of cell failure, often resulting in catastrophic thermal runaway.⁴⁻⁶ While many approaches have been explored to address these issues, one of the most promising options is the elimination of the liquid electrolyte in favor of a solid electrolyte.⁷

Transitioning from conventional liquid electrolytes to Li^+ -conducting solid electrolytes (SEs) presents two primary advantages. The high mechanical rigidity of inorganic SEs may act to suppress the formation of dendrites at lithium anodes, reducing the possibility of internal short circuits.⁸ Additionally, the negligible flammability of most SEs dramatically lowers the risk of uncontrolled thermal runaway in the event of cell failure.^{9, 10} These potential benefits motivate the search for materials with sufficiently high ionic conductivities to serve as replacements for their liquid counterparts. Among the known SE formulations, the thiophosphate class of superionic conductors exhibit exceptionally fast lithium transport at room temperature.¹¹ Specifically, $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$ (LGPS) demonstrates an ionic conductivity of ca. 10 mS cm^{-1} , comparable to conductivities achievable in conventional liquid electrolytes.¹²

Despite the favorable ionic conductivity of LGPS, its poor chemical and electrochemical stabilities remain key challenges for practical application. The narrow window of electrochemical stability of LGPS and many other thiophosphate SEs results in decomposition of the SE at both the anode and cathode surfaces.¹³⁻¹⁸ While progress has been made in the implementation of low voltage cathodes, the reductive instability of LGPS at lithium anodes remains a considerable issue. Previous investigations of the LGPS/ Li^0 interface have identified Li_2S , Li_3P , Li-Ge alloys, and other P-S and S-S containing species as some of the most common reductive decomposition products formed during lithium plating.^{19, 20} In general these species are poor ionic conductors, and their buildup at the interface results in a progressive decrease in

overall ionic conductivity. Likewise, the formation of electronically conductive alloys of Ge allows the reductive decomposition of the SE to propagate into bulk LGPS, precluding the stable passivation of the interface.

In addition to their (electro)chemical instabilities, mechanical instabilities in SEs have also been reported. Recent analyses of garnet and thiophosphate SEs with *operando* x-ray tomography have demonstrated crack propagation through bulk SE pellets during electrochemical cycling. While the mechanisms of crack formation are material-dependent, they have generally been attributed to either lithium dendrite propagation through voids and defects in the SE or to volumetric changes at the electrode/electrolyte interphase due to the electrochemical decomposition of the electrolyte.²¹⁻²⁴ Regardless of the mechanism, the progressive mechanical failure of SE pellets with cycling results in a sharp decrease in ionic conductivity and allows the facile growth of lithium dendrites through the crack network resulting in subsequent cell shorting.⁸

One approach to address the (electro)chemical instabilities of SEs is the application of a coating or interlayer to the SE surface, thereby protecting the electrolyte from the extreme potentials of the electrodes.^{16, 25} While both organic and inorganic coatings have been investigated, the application of inorganic coatings often leads to the formation of insufficiently conductive interfacial domains, limiting the practicality of these coatings in a real system.²⁶ In contrast, the application of polymeric or liquid interlayers has proven quite effective at mitigating electrochemical decomposition while maintaining fast ion transport.²⁷⁻²⁹ Indeed, previous work from our group has demonstrated the improved electrochemical stability of thiophosphate solid electrolytes upon modification of the $\text{Li}^0/\text{electrolyte}$ interface with a highly concentrated (solvate) liquid electrolyte.³⁰

The enhanced electrochemical stability resulting from the application of a solvate interlayer has been largely attributed to thermodynamic factors, specifically the suppression of reductive decomposition at the SE/Li⁰ interface. While this offers insights into the electrochemical stability of the modified interface, the impact of interlayer modification on the mechanical stability of the SE remains unclear. Much progress has been made in the development of *operando* real-space imaging techniques to study such processes in battery materials under electrochemical load.³¹ In the present work we employ *operando* x-ray tomography to interrogate the mechanical responses of both interlayer-modified and unmodified LGPS solid electrolyte pellets during electrochemical cycling. Leveraging the spatial resolution of the technique we directly image the growth of interfacial decomposition domains responsible for the eventual fracture of the bulk LGPS pellet. We then demonstrate that by the application of a liquid interlayer the formation of these domains can be suppressed, thereby improving both the electrochemical performance and mechanical stability of the solid electrolyte.

3.2 Experimental Section

Electrolyte Formulation and Cell Fabrication: Unless otherwise stated, all chemicals were used as received without further purification and all sample manipulations were carried out in an Ar-filled glovebox with O₂ and H₂O concentrations <2 ppm. Electrochemical and x-ray measurements were carried out on cylindrical solid-electrolyte pellets fabricated from commercial Li₁₀GeP₂S₁₂ powder (LGPS, 99.99%, MSE Supplies LLC). Pellets were assembled by first grinding the as-received LGPS powder in a mortar and pestle to break up large aggregates. This powder was then cold pressed under static load (1500 PSI, 3 min) in a stainless-steel pellet dye (International Crystal Laboratories) to a diameter of ca. 2 mm and a thickness of ca. 150 μ m. Pressed pellets were stored under an Ar atmosphere until use. The solvated liquid

electrolyte used for interlayer cells was prepared as described previously.³⁰ Briefly, lithium bis(trifluoromethanesulfonyl)imide (LiTFSI, 99.95% Sigma-Aldrich) was dissolved in 50:50 wt.% 1, 3-dioxolane (DOL, anhydrous, Sigma-Aldrich): 1, 2-dimethoxyethane (DME, anhydrous, 99.5%, Sigma-Aldrich) in a ratio of 7 mols of LiTFSI per liter of solvent. This neat electrolyte was then diluted with one part 1,1,2,2-tetrafluoroethyl 2,2,3,3-tetrafluoropropyl ether (TTE, 99%, Synquest Laboratories) to five parts neat solvate by volume to reduce viscosity and improve ionic conductivity.^{30, 32}

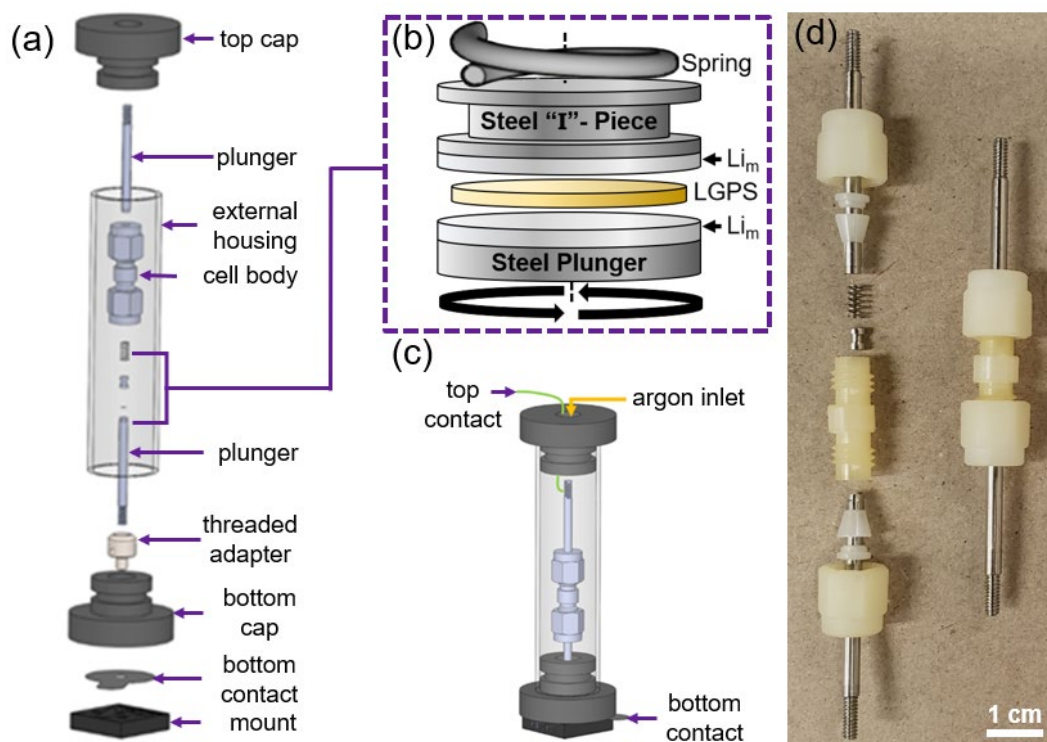


Figure 3.1. (a) Schematic depiction of the experimental setup used for operando tomography measurements. (b) Focused view on the electrode stack. (c) Completed cell as assembled for x-ray measurements. (d) Photograph depicting the disassembled (left) and assembled (right) electrochemical cell used for tomography experiments.

Lithium symmetric cells were used to evaluate mechanical changes to the solid-electrolyte pellets during electrochemical cycling both in the presence and absence of a solvate interlayer. To maintain adequate x-ray transmission for tomographic reconstructions, electrode

stacks were assembled in custom nylon Swagelok cells. LGPS cells assembled without an interlayer were constructed from bottom to top as follow: lithium metal was mechanically polished and affixed to a 3 mm-diameter stainless-steel post to serve as the counter/reference electrode. This electrode was then inserted into the nylon cell body followed by an LGPS pellet and the working electrode, which consisted of polished lithium pressed onto a stainless-steel "I"-piece. To complete the assembly a spring was added to maintain stack pressure followed by another steel post to maintain electrical contact (Figure 3.1). Interlayer cells were fabricated in the same order with one modification in that a small amount ($<1\ \mu\text{L}$) of solvate was added to the surface of each lithium electrode prior to contacting the LGPS pellet.

X-Ray Measurements: X-ray tomography was conducted at the 7-BM beamline at the Advanced Photon Source at Argonne National Laboratory. Fully assembled electrochemical cells were transported to the beamline under an Ar atmosphere and mounted on a rotating stage in the path of the x-ray beam. To mitigate potential atmospheric exposure, cells were housed in a cylindrical acrylic enclosure and purged continuously with Ar gas for the duration of the experiment. Transmission x-ray images were collected using monochromatic x-rays ($30.25 \pm 1\ \text{keV}$), which illuminated a $100\ \mu\text{m}$ LuAG:Ce scintillator, followed by a visible light turning mirror. A visible light camera (FLIR Grasshopper3) equipped with a 5x long-distance microscope collected the images. The system afforded an effective pixel resolution of $1.172\ \mu\text{m}$ horizontally and vertically. A fly scanning procedure was used for the tomography rotation to minimize the radiation dose on the cell.

Typical *operando* measurements consisted of a short (ca. 1000 s) open circuit potential hold followed by voltammetric cycling at a scan rate of $200\ \mu\text{V s}^{-1}$ between $\pm 100\ \text{mV}$ vs Li/Li^+ to verify nominal cell performance. After ensuring normal cell function, the potential window

was expanded to ± 500 mV vs Li/Li⁺ (still scanning at $200 \mu\text{V s}^{-1}$) and the cell was cycled until failure (defined as a total cell short). In tandem with electrochemical measurements, tomographic data was collected every 500 seconds during which time the cell was rotated 360 degrees around its z-axis to provide a complete view of the interior cell volume. Prior to each scan, images were collected both with the cell removed from the camera field of view and with the x-ray beam off to produce a series of brightfield and darkfield images respectively.

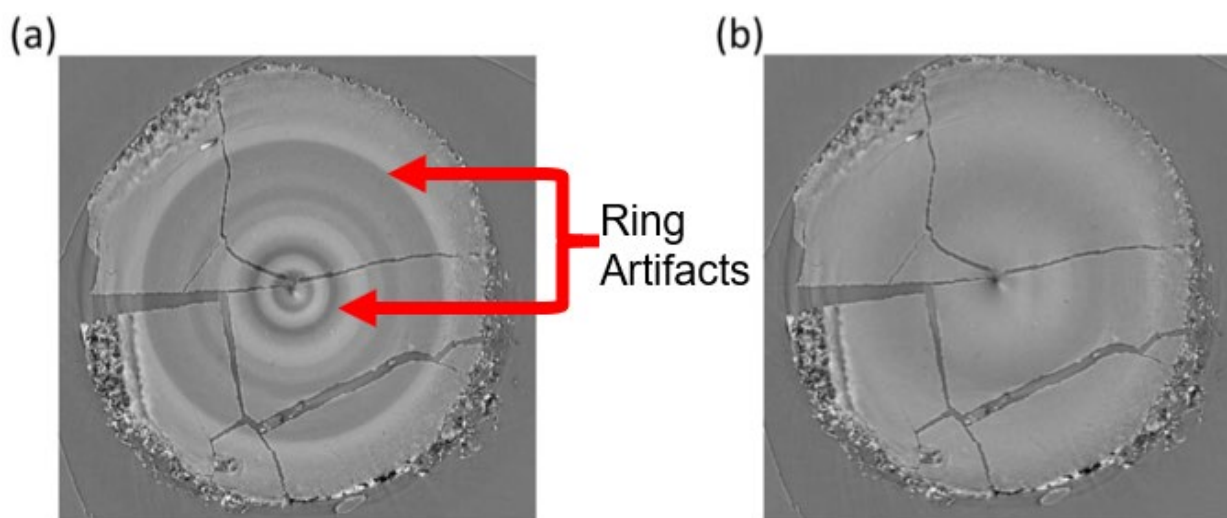


Figure 3.2. Reconstructed tomographic slices of an LGPS solid electrolyte pellet generated using either (a) a simple smoothing filter or (b) a Fourier-based wavelet algorithm. Obvious ring aberrations, coaxial with the center of rotation are largely removed by the Fourier algorithm leaving images clean enough for subsequent data analysis.

Tomographic Reconstruction and Data Processing: Raw tomographic data was processed using the open-source Python utility TomoPy to generate x-ray attenuation data across the cell volume, as described elsewhere.³³ Briefly, the reconstructions used a fast Fourier transform from the projection domain into the volume domain to determine the pixel intensity at each point within the sample volume. Brightfield and darkfield images were used to normalize the projection data. Additional image corrections were undertaken in the form of a post-processing algorithm to remove obvious artifacts left by the reconstruction, primarily ring artifacts (Figure 3.2). Ring

removal relied again on Fourier and wavelet image transforms to remove aberrations exhibiting periodicity about the center of rotation.

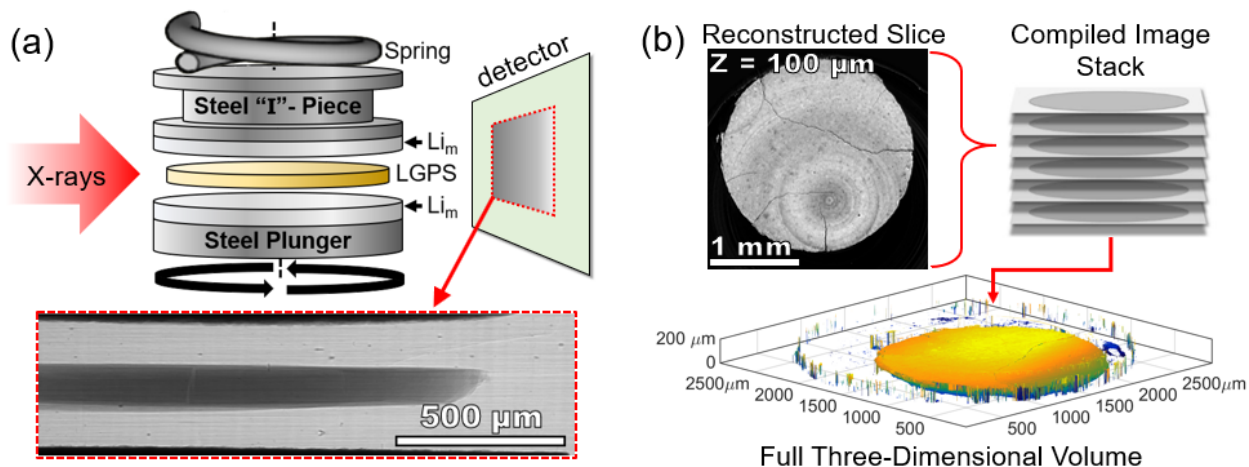


Figure 3.3. (a) Schematic depiction of the experimental setup used to collect tomographic data (top) and an example of the obtained transmission mode images used in subsequent data analysis (bottom). (b) An example of a cross-sectional image at a fixed height within and LGPS pellet (top left) reconstructed from a series of the transmission mode images depicted in (a). Cross-sectional images were further compiled into rough image stacks (top right) and subsequently processed into full three-dimensional renderings of the solid electrolyte pellet (bottom)

Analysis and rendering of the volume data, including determination of pellet void volume, was accomplished in MATLAB (Figure 3.3). Wherever appropriate, computation relied on population analyses of pixel intensity to maintain statistically valid interpretations of the data and to avoid erroneous pixel counting. A detailed description of the computational methods employed is provided in Appendix F.

3.3 Results and Discussion

3.3.1 Electrochemical Performance of Interlayer-Modified and All-Solid-State LGPS Cells

Operando electrochemical measurements were conducted on cold-pressed LGPS pellets (ca. 150 μm thick, 2mm in diameter), cycled voltammetrically in lithium-lithium symmetric cells. Cells without interlayer modification, herein referred to as all-solid-state cells, were

assembled by placing the LGPS pellet between two pristine lithium electrodes, creating direct contact between the SE and the Li^0 surface. For cells assembled with an interlayer, herein referred to as interlayer cells, interface modification was accomplished by painting the surfaces of both lithium electrodes with solvate electrolyte before contacting the LGPS pellet. The solvate, identified in previous investigations to exhibit favorable interactions with LGPS,³⁰ consisted of a 6.6 molal solution of lithium bis(trifluoromethanesulfonyl) imide (LiTFSI) in 1,3-dioxolane (DOL)/1,2-dimethoxyethane (DME)/ 1,1,2,2-tetrafluoroethyl 2,2,3,3-tetrafluoropropyl ether (TTE).³⁰

Figure 3.4a presents voltammetry obtained from *operando* LGPS cells in both the all-solid-state (black) and interlayer (red) configurations, cycled between ± 500 mV at a scan rate of $200 \mu\text{V s}^{-1}$. After several cycles, both cells demonstrate a roughly linear correlation between applied potential and current, reflecting the ohmic behavior expected from a high ionic-conductivity solid electrolyte.³⁴ Interestingly, at early cycles (cycles 1-3) the all-solid-state cell deviates from this behavior, exhibiting hysteresis characteristic of diffusion limitations at the electrode/electrolyte interface.³⁵ Previous investigations of similar systems attribute the observed diffusion limitations to local Li^+ depletion at the interface, and are indicative of insufficient ion transport to maintain the Li^+ flux demanded by the potential.^{19, 34} Significantly, the lack of a diffusion limitation in the interlayer cell suggests improved Li^+ mobility across the interface relative to the purely solid boundary.

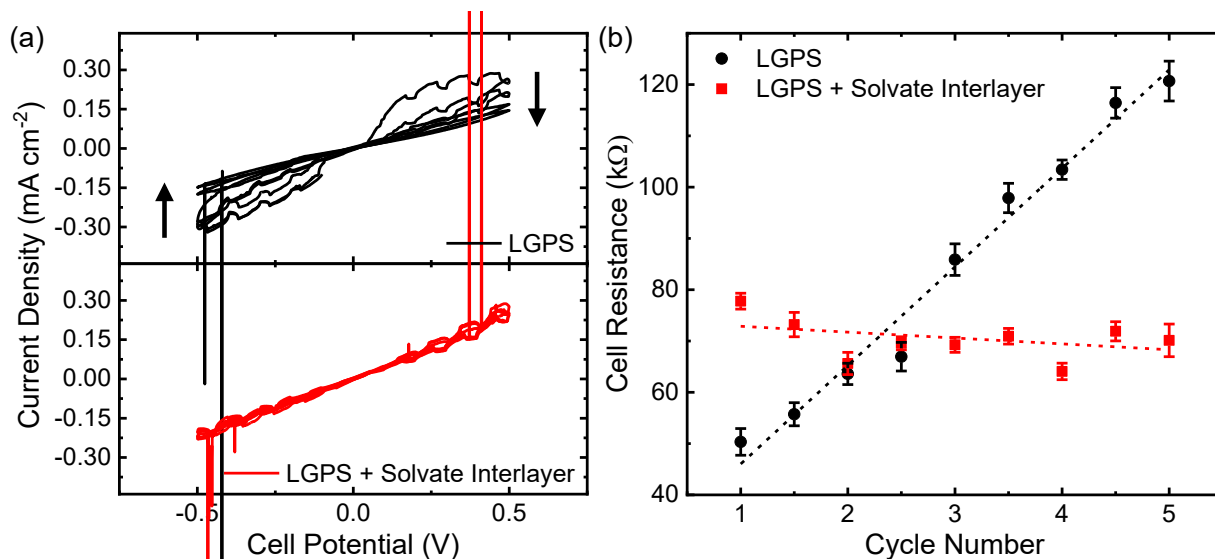


Figure 3.4. (a) Cyclic voltammograms of an all-solid-state LGPS cell (black) and an LGPS cell with solvate interlayers applied to the surfaces of each lithium electrode (red). (b) Cell resistance for each potential sweep for both the all-solid-state (black) and interlayer cell (red) plotted as a function of cycle number.

In addition to the differences in interfacial ion depletion, the two cell configurations exhibit different long-term stabilities with cycling. The all-solid-state cell demonstrates a progressive drop in peak current density and concomitant increase in total cell resistance (calculated as the slope of a linear regression applied to each potential sweep) with cycle number, as depicted in Figure 3.4b. This increased resistance is attributed to reductive decomposition of LGPS at the electrode/electrolyte interface due to the generally poor thermodynamic stability of thiophosphates against metallic lithium.^{14, 15, 36, 37} Notably, the interlayer cell does not display this fade, exhibiting a slowly decreasing cell resistance around 70 kΩ, nearly half the 120 kΩ reached in the all-solid-state cell after 5 cycles. This observation demonstrates the enhanced reductive stability of the modified interface, aligning well with previous reports on both interlayer-modified, and quasi-solid-state electrolytes.^{30, 38, 39}

The origins of the beneficial electrochemical properties of LGPS attendant interlayer modification have been investigated in a previous report from our group where it was found that

the presence of the solvate electrolyte impacts both the interfacial and bulk properties of the LGPS pellet.³⁰ Specifically, the presence of a highly concentrated liquid electrolyte at the interface serves to both effectively drop the highly reducing potentials induced during lithium plating and to homogenize Li^+ transport between the electrode and the LGPS pellet, thereby enhancing peak current stability. Additionally, the observed decrease in overall cell resistance for the interlayer modified cell has been attributed to the slow permeation of the liquid electrolyte into the LGPS pellet, gradually decreasing grain boundary impedance within the SE and facilitating faster ion conduction across the bulk electrolyte pellet.

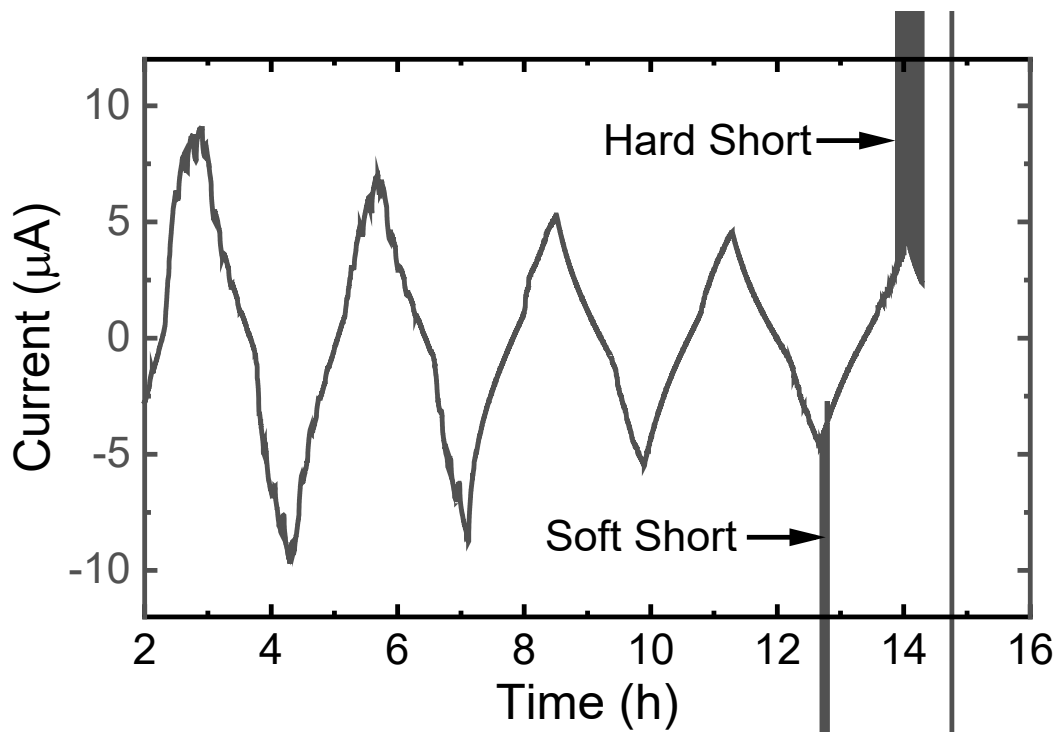


Figure 3.5. Exemplary data from an all-solid-state LGPS cell depicting both a soft short, wherein the cell remains nominally functional, as well as a hard short, which results in total, and irreparable cell failure.

In both *operando* cells, sharp current spikes are observed in the later cycles indicative of lithium dendrimers bridging the solid electrolyte pellet resulting in “soft” internal short circuits. In these cases, the sharp jump in current is enough to decompose the dendrimer and return the

cell to approximately normal function.^{40, 41} Cells were cycled until failure (Figure 3.5) wherein an irreparable cell short occurred, typified by a permanent jump in cell current. Generally, the interlayer modified cells exhibit enhanced lifetimes relative to the all solid-state cells, suggesting moderate suppression of lithium dendrite propagation.³⁰ One other electrochemical response of note is the presence of periodic current oscillations in all *operando* cells. These oscillations have the same periodicity as tomographic imaging (500 s) and are attributed to photoelectron emission resulting from beam interactions with the electrochemical cell. Notably, the peak current densities, and the rates of peak current fade elicited by *operando* cells show good agreement with ex-situ cells (Figure 3.6) suggesting that any putative radiation damage as a result of the x-ray beam is likely negligible.

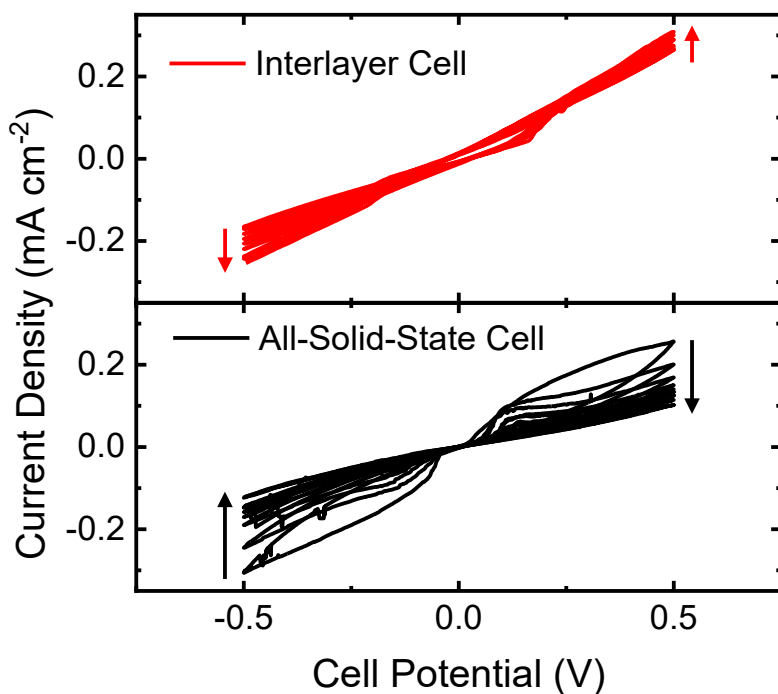


Figure 3.6. Ex-situ voltammetry of both an all-solid-state cell (black) depicting the progressive decay of peak current density in the absence of an interlayer, largely mirroring the behavior of *operando* electrochemical cells. In the presence of an interlayer (red) current densities increase slightly as cycling progresses.

3.3.2 Bulk Mechanical Decomposition of All-Solid-State and Interlayer-Modified Cells

Structural changes to bulk LGPS pellets were visualized using cross-sectional reconstructions of tomographic data as a way to monitor the mechanical stability of the SE. Figure 3.7a and b presents cross-sectional slices from the centers of an all-solid-state and interlayer cell, respectively, along with corresponding electrochemical data. Pixel intensities in the grey-scale images reflect total x-ray attenuation with bright pixels signifying regions of higher total attenuation (LGPS) and dim pixels signifying regions of lower total attenuation (empty space or metallic lithium), scale bars are 1 mm. In most greyscale images, concentric rings of slightly different pixel intensity are found about the center of rotation of the electrochemical cell; these represent reconstruction ring artifacts, as depicted in Figure 3.2. As the change in pixel intensity due to ring artifacts is generally small, and the subsequent analyses rely primarily on global averaging, the impact of these artifacts is expected to be minimal.

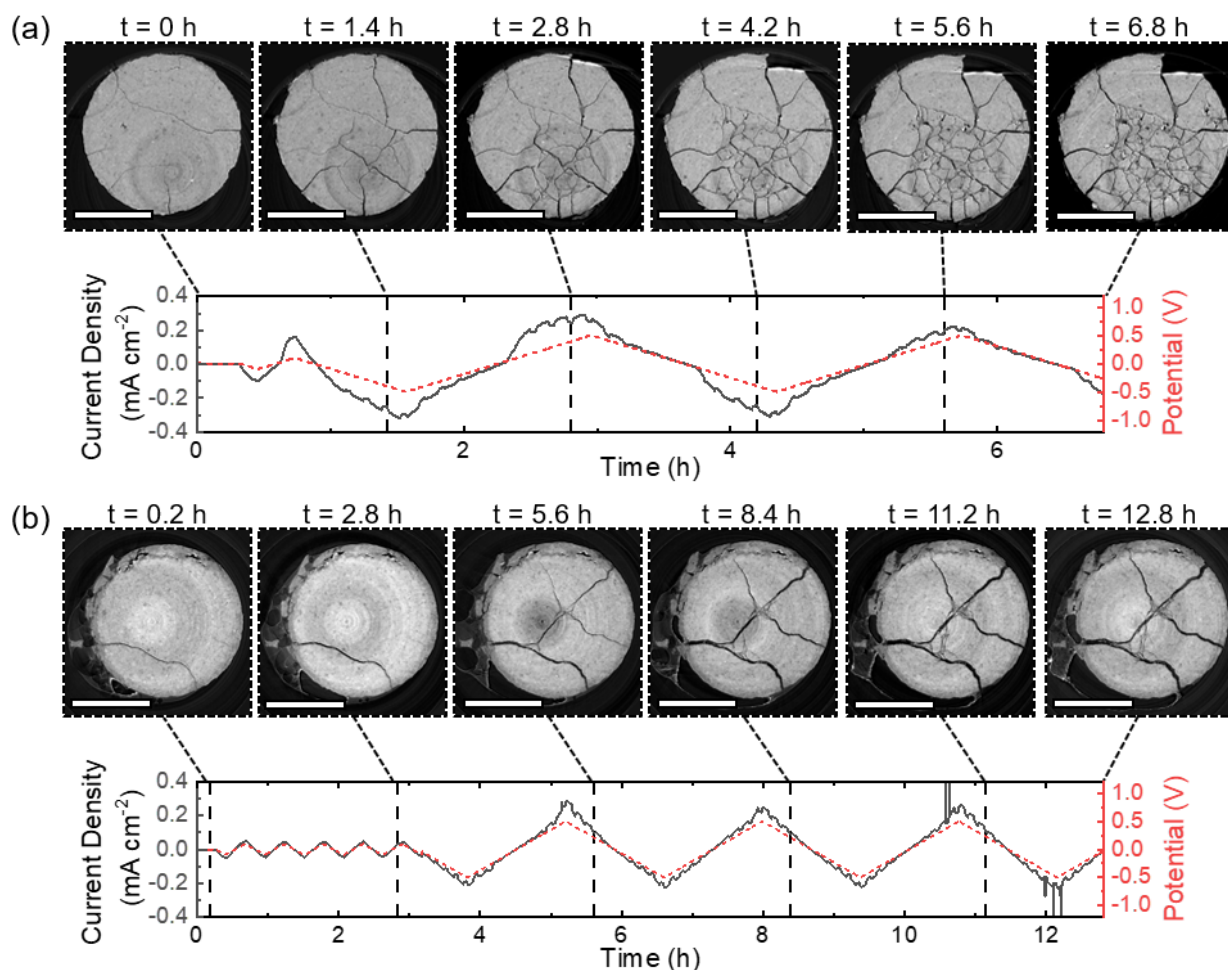


Figure 3.7. Cross-sectional images of an (a) all-solid-state and (b) interlayer cell during voltammetric cycling plotted as a function of time along with the cell voltage (dashed red traces) and cell current (black traces) for each cell. Pixel brightness reflects x-ray attenuation with bright pixels representing the LGPS pellet and darkest pixels representing empty space. Images were taken at the center of each LGPS pellet with scale bars denoting 1 mm.

Mechanical decomposition is obvious in both cells, as evidenced by crack formation through the LGPS pellets during cycling. The extent of this decomposition is greatest when cycling within a wide potential window (± 0.5 V), with fewer significant mechanical changes observed during narrow window cycling (± 0.1 V) or during open circuit potential holds. In addition to crack propagation, large scale movement of material is seen in the all-solid-state cell, as demonstrated by a piece of material tilting out of the image frame in the upper right-hand corner at later cycling times. Despite the observed mechanical decomposition in both cells,

several differences are evident when comparing the mechanical responses of the two configurations.

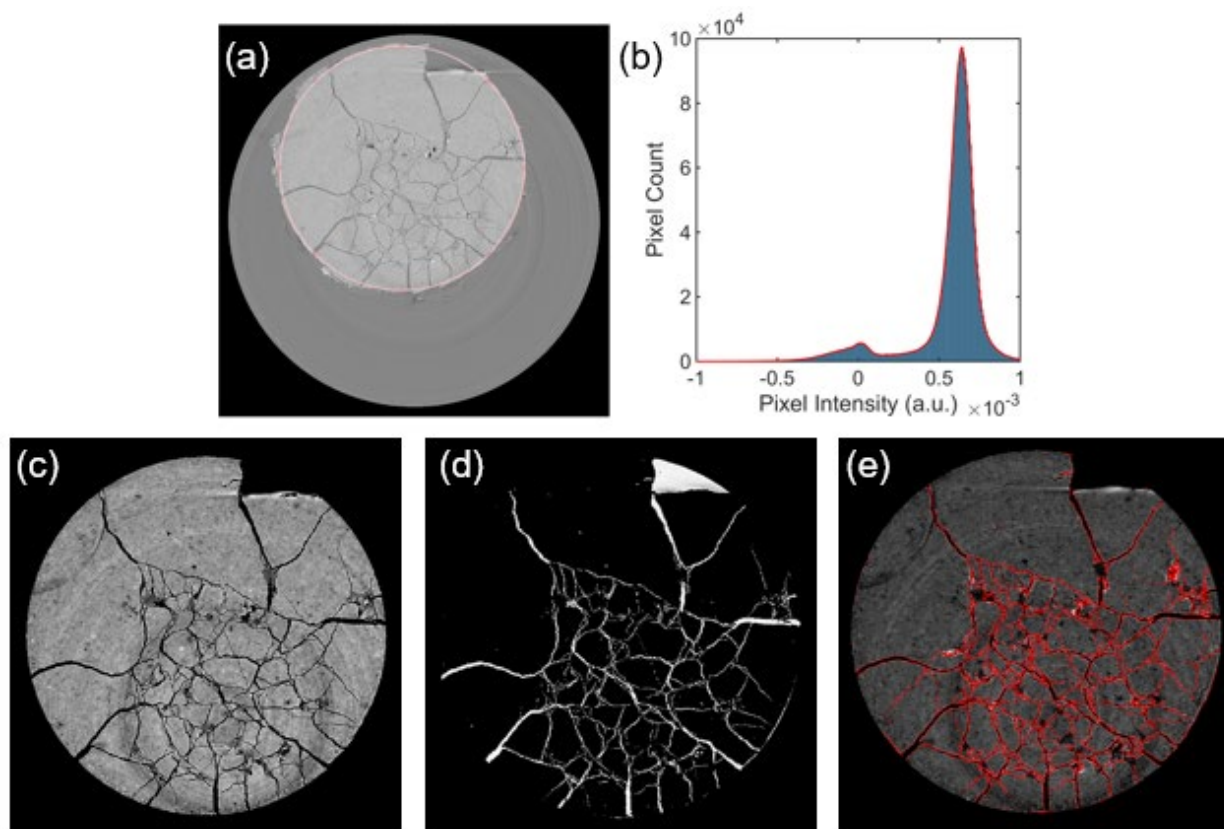


Figure 3.8. Image segmentation process used to compute internal void volume and void surface area. Raw reconstructed tomographic images (a) are used as the input and are processed within an elliptical region of interest projected through the entire thickness of the pellet in the z -direction. (b) Histograms of pixel intensities are computed and used for statistics-based differentiation of material domains. (c) Isolated LGPS and (d) isolated void domains are identified and used for subsequent volume integration. Finally, images are differentiated to isolate the internal surfaces of the crack network as depicted in (e) overlaid on the original pellet image.

Visually, the most obvious of these differences is the continuity of the LGPS pellet during cycling. The all-solid-state cell demonstrates substantial pulverization of the SE, similar to other investigations of ceramic solid electrolytes,^{21, 23} whereas the interlayer cell maintains larger continuous domains of LGPS. Furthermore, the rate of crack formation in the all-solid-state cell appears much greater than that of the interlayer cell. Indeed, while the all-solid-state

cell appears profoundly fractured after 6.8 h, the interlayer cell never reaches the same extent of structural decomposition, even after 12.8 h of continuous cycling.

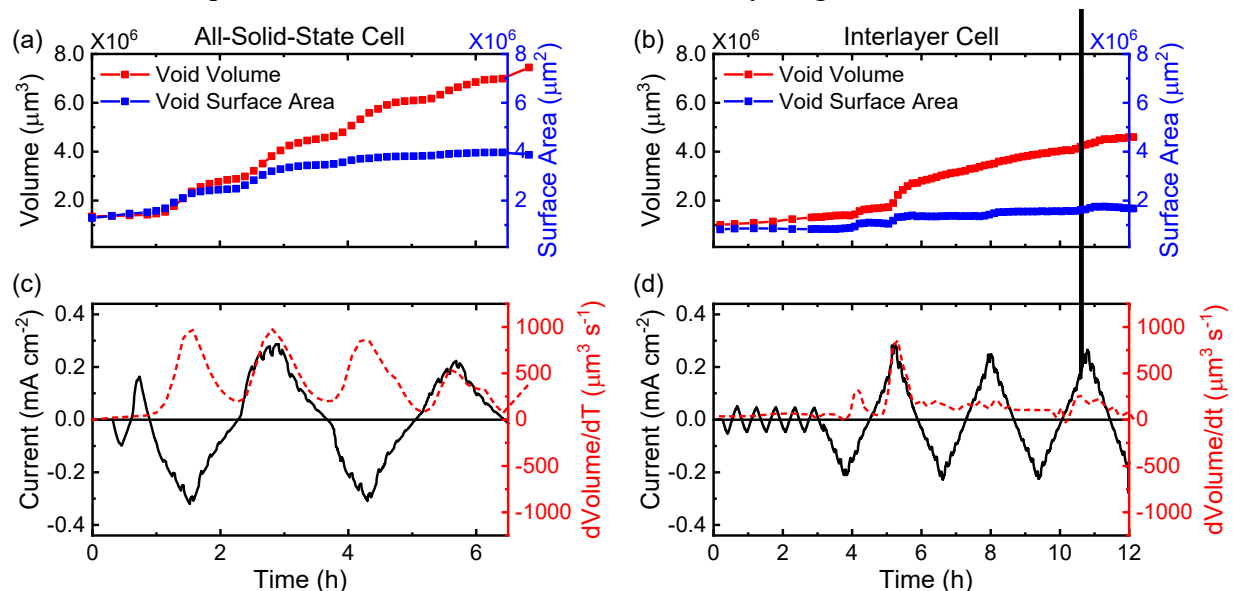


Figure 3.9. Computed void volumes (red) and void surface areas (blue) of solid electrolyte pellets during voltammetric cycling in the (a) absence and (b) presence of a solvate interlayer. (c) and (d) depict the derivative void volumes (dashed red traces) plotted as a function of time for the all-solid-state and interlayer cells, respectively, along with the currents (solid black traces) elicited by these cells during cycling.

To better quantify these observations, the total void volume was computed along with the internal surface area of the crack network for both the all-solid-state and interlayer cells, using the difference in pixel intensity associated with void space and LGPS to facilitate pixel counting (figure 3.8). The results of this analysis for the all-solid-state and interlayer cells are presented in Figure 3.9a and b, respectively. In this interpretation, the void volume reflects bulk structural decomposition, while the void surface area quantifies the continuity of the pellet, as reported in previous analyses of fatigue driven mechanical failure.⁴²⁻⁴⁴ For both cells void volume and void surface area increase monotonically with cycling, reflecting the irreversible mechanical decomposition of the pellet. The enhanced mechanical stability afforded by the solvate is evident, as the interlayer cell demonstrates substantially lower void volume and surface area even after nearly twice the cycling time.

To better understand the mode of mechanical failure, derivative traces of void volume with respect to time for the all-solid-state and interlayer cells were computed and are presented in Figure 3.9c and d, respectively. In the all-solid-state cell, a clear correlation exists between the rate of void formation and the cell potential, suggesting that, either the passage of current, or the polarization of the electrode to extreme potentials results in mechanical decomposition of the LGPS pellet. In contrast, the overall rate of void formation is lower in the interlayer cell, and the correlation between void formation and cell potential is much less apparent. This suggests that the presence of the solvate interlayer both reduces the rate at which void space forms and works to decouple the mechanical decomposition of the SE from electrochemical processes.

3.3.3 Interfacial Properties of All-Solid-State LGPS Cells

The results displayed in Figure 3 demonstrate that modifying the LGPS/Li⁰ interface by the addition of a solvate interlayer strongly impacts the mechanical evolution of the solid-electrolyte pellet. To further understand the origins of this effect, three-dimensional renderings were computed for both the all-solid-state cell and the interlayer cell, using grayscale pixel intensities to segment volume data. These renderings directly capture the interfacial structure of the LGPS pellet as cycling progresses. Figure 3.10 shows a time-dependent sequence of these

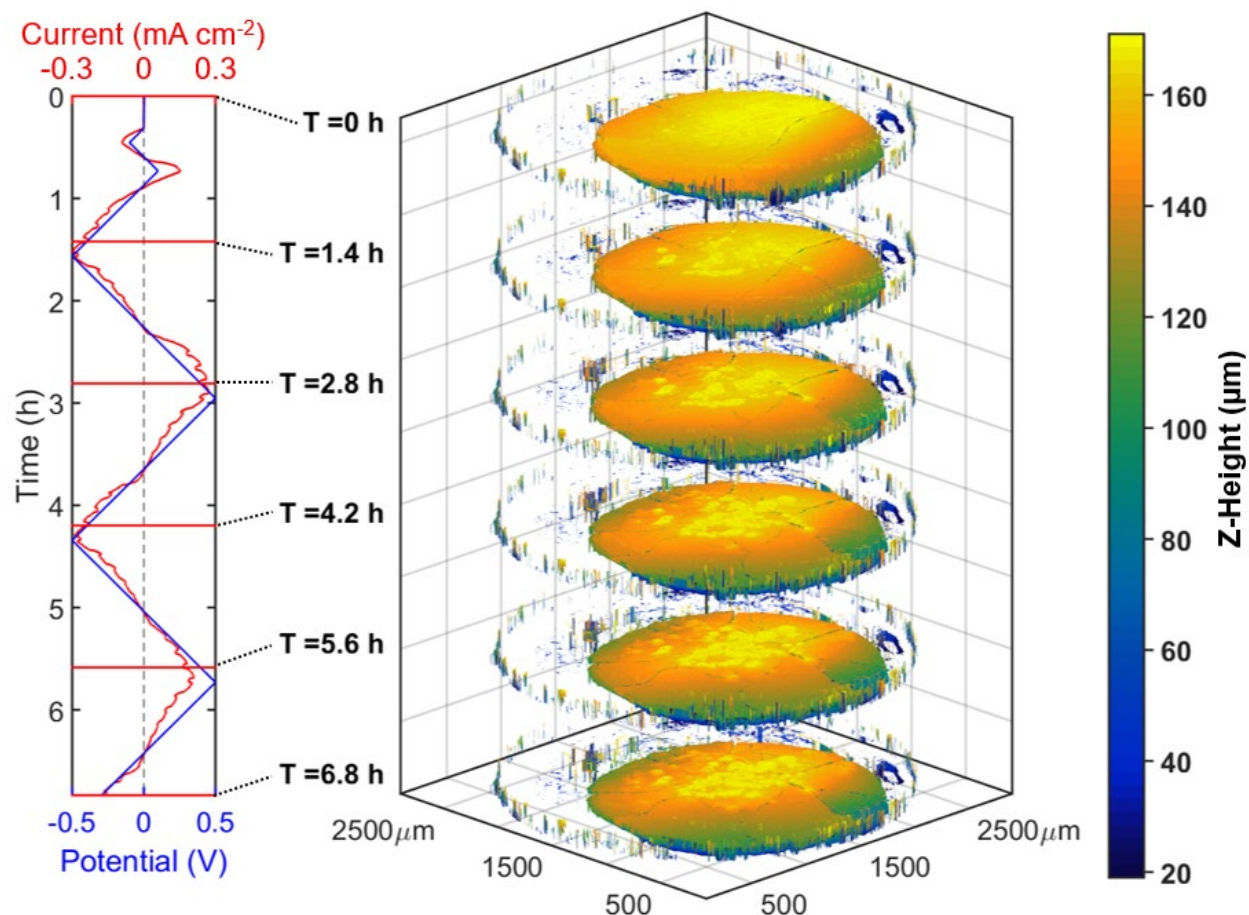


Figure 3.10. Electrochemical cycling and accompanying three dimensional renderings of an LGPS pellet in the all-solid-state configuration. As cycling progresses a substantial morphological change is observed at the pellet surface as evidenced by the increased roughness and accumulation of high elevation (bright yellow) domains.

renderings for the all-solid-state cell along with the corresponding electrochemical data. For each time point a full three-dimensional reconstruction of the SE pellet is depicted in real space with the height of each point along the pellet reflected by the color map (the top of the pellet is depicted in yellow and the bottom of the pellet is depicted in blue). The figure shows a prominent morphological change at the LGPS/Li⁰ interface through the formation of topologically heterogeneous regions (depicted here as bright yellow domains) protruding vertically from the SE surface. These regions likely result from inhomogeneous contact between the SE and the Li surface. Indeed, heterogeneous contact between Li and other SE materials has long been an issue.^{45, 46} The formation of these protrusions appears to be irreversible, supporting

the conclusion that the species formed at the interface are the result of electrochemical decomposition of the solid electrolyte. Further, the large size of the domains formed suggests that the decomposition reaction grows from the interface into the bulk, likely due to the formation of electronically conductive $\text{Li}_{3.75}\text{Ge}$ as one of the primary decomposition products. Electronic conductivity through the decomposed domains allows the reductive decomposition to continue even when the domain size is large.

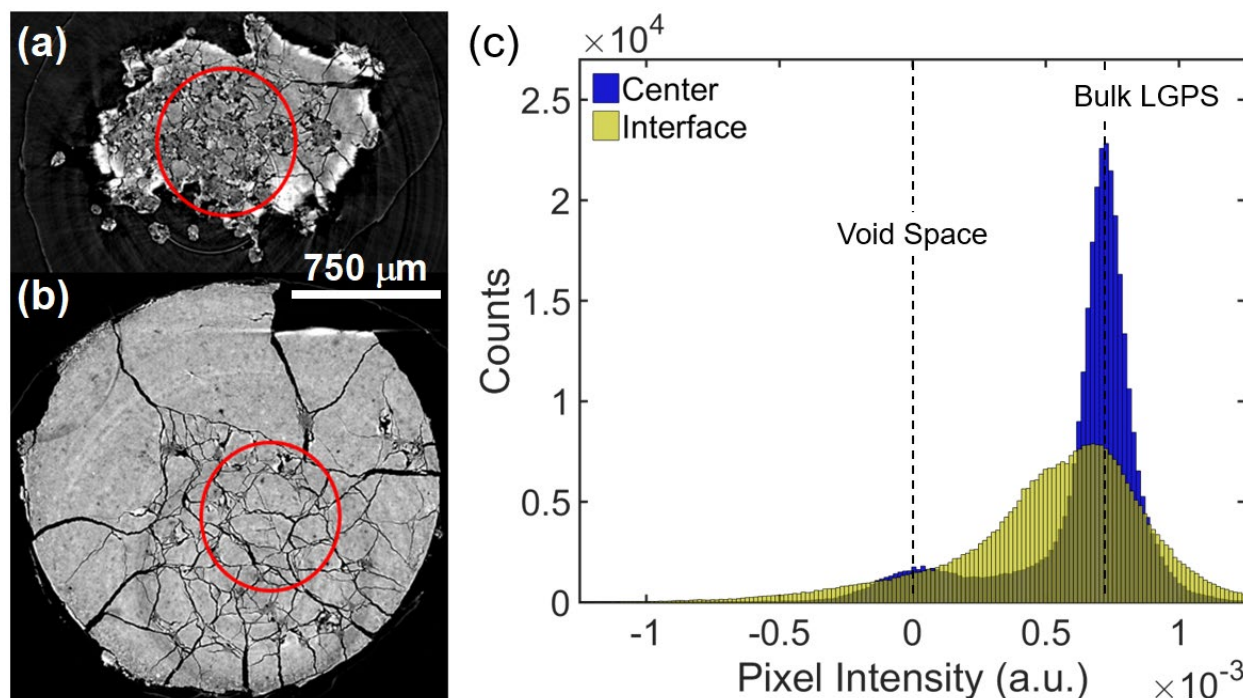


Figure 3.11. Reconstructed tomographic images collected from an all-solid-state cell (a) near the LGPS/ Li^0 interface and (b) at the center of the LGPS pellet at the end of cycling ($t \approx 6.8$ h). (c) Presents a histogram analysis of pixel intensities collected from images a and b demonstrating a large population of pixels with intermediate intensity near the LGPS/Li interface that cannot be attributed to either bulk LGPS or void space/Li.

Additional characterization of the decomposition species for the all-solid-state cell is presented in Figure 3.11, which shows tomographic slices at the end of cell cycling ($t \approx 6.8$ h) taken from positions near the LGPS/ Li^0 interface (Figure 3.11a) and at the center of the SE pellet (Figure 3.11b). There is a substantial difference in the pixel intensity distributions near the interface compared with those of the bulk pellet. Large domains of material with intermediate

pixel intensity (between that of void space and bulk LGPS) are found near the interface which are absent in the center of the pellet. To quantify this phenomena, histograms of pixel intensities were computed in the areas demarcated by red circles in Figure 3.11a and b, the results of which are presented in Figure 3.11c. At the center of the pellet, pixel intensities largely fall within a bimodal distribution, with pixels binned either as void space/Li⁰ or bulk LGPS. A third population of intermediate pixel intensity is observed at the interface signifying the presence of a new material phase that cannot be attributed to either bulk LGPS or void space/metallic lithium.

The approximate composition of the domains formed near the LGPS/Li⁰ interface was estimated by comparing the pixel intensities of these domains to the intensities expected of the common decomposition products of LGPS. Previous work has demonstrated that the primary species formed during the reduction of LGPS are Li₂S, Li₃P, and Li_{3.75}Ge in molar ratios dictated by the stoichiometry of LGPS.^{11, 19, 20} As these species cannot be isolated directly from grayscale images, their expected pixel intensities were estimated based on their total x-ray attenuation coefficients (μ , cm⁻¹). Attenuation coefficients were calculated for each component as the product of its density (ρ , g cm⁻³) and its x-ray mass attenuation coefficient (μ/ρ , cm² g⁻¹) and normalized by the total attenuation coefficient of the LGPS pellet. The theoretical pixel intensity for each component was then calculated as the product of the pixel intensity of the LGPS pellet and the normalized attenuation coefficient. Mass attenuation coefficients for each species were obtained from the NIST photon cross-section database,⁴⁷ and material densities were obtained from previous literature with the exception of the LGPS pellet whose density was determined experimentally.

Table 3.1 presents the results of the above analysis, as well as all parameters used to estimate the pixel intensities of the species under investigation. Of the decomposition products

considered here, both Li_2S and Li_3P are expected to exhibit lower pixel intensities than LGPS as a result of their low mass attenuation coefficients. In contrast, the mass attenuation coefficient, and therefore the expected pixel intensity, of $\text{Li}_{3.75}\text{Ge}$ is substantially greater than that of LGPS due the large electron density of Ge relative to P and S. While Li_2S and Li_3P agree qualitatively with the low pixel intensities observed experimentally, their calculated intensities are too low for a confident assignment. Similarly, the high calculated pixel intensity of $\text{Li}_{3.75}\text{Ge}$ disagrees qualitatively with experiment, suggesting that the decomposition domains at the interface cannot be attributed to a single decomposition product.

As no single species can be assigned to the observed decomposition domains, a mixed phase of the three species is considered. The composition of this phase was treated as $12 \text{ Li}_2\text{S} + 2 \text{ Li}_3\text{P} + \text{Li}_{3.75}\text{Ge}$ as reported elsewhere for the reduction of LGPS against lithium.²⁰ The absolute pixel intensity of this phase was treated as an average of the pixel intensities of each species weighted by their molar ratios within the phase. The calculated intensity from this approximation ($4.4 \pm 0.5 \times 10^{-4}$) agrees closely with the experimental pixel intensity of the decomposition domains ($4.5 \pm 1.0 \times 10^{-4}$) supporting the conclusion that the decomposition products observed near the interface are comprised, primarily, of Li_2S as well as small amounts of Li_3P and $\text{Li}_{3.75}\text{Ge}$. While this approximation cannot, unambiguously, assign the composition of the interfacial species, the results are in good agreement with the previously reported composition expected from the stoichiometric reduction of LGPS into its common decomposition products.

Table 3.1. Parameters used and results obtained from a pixel intensity analysis near the LGPS/Li⁰ interface for an all-solid-state LGPS cell. The table includes the density ρ , x-ray mass attenuation coefficient μ/ρ , total attenuation coefficient μ , experimental pixel intensity if known, and estimated pixel intensity for each species under consideration.

Species	ρ (g cm ⁻³)	μ/ρ (cm ² g ⁻¹)	μ (cm ⁻¹)	Pixel Intensity (Experimental) ^{a)}	Pixel Intensity (Calculated)
Li ₁₀ GeP ₂ S ₁₂ pellet	1.5 ± 0.1	3.29	4.9 ± 0.3	6.4 ± 0.7 × 10 ⁻⁴	-
Li ⁰	0.53 ⁴⁸	0.16	8.8 × 10 ⁻²	0.0 ± 0.3 × 10 ⁻⁴	-
Ar (void space)	1.7 × 10 ⁻³ ⁴⁸	2.7	4.6 × 10 ⁻³	0.0 ± 0.4 × 10 ⁻⁴	-
Interfacial species	-	-	-	4.5 ± 1.0 × 10 ⁻⁴	-
Li ₂ S	1.66 ⁴⁹	1.5	2.53	-	3.3 ± 0.4 × 10 ⁻⁴
Li ₃ P	1.43 ⁵⁰	1.1	1.55	-	2.0 ± 0.2 × 10 ⁻⁴
Li _{3.75} Ge	1.76 ⁵¹	10.2	17.8	-	23.1 ± 2.5 × 10 ⁻⁴
12 Li ₂ S + 2 Li ₃ P + Li _{3.75} Ge		-	3.42	-	4.4 ± 0.5 × 10 ⁻⁴

^{a)} Uncertainties were determined from the standard deviation in the pixel distribution of the LGPS pellet and were propagated to the calculated pixel intensities.

To quantitatively monitor the formation of these decomposition products at the LGPS/Li⁰ interface as a function of electrochemical cycling, the total volume of decomposed species was computed by deconvolution of the pixel intensity distributions as a linear sum of three Gaussian populations as detailed in Figure 3.12. The results of this volume analysis align very well with the expected behavior of this system when looking at line scans of pixel distribution over time (Figure 3.13). Specifically, at the start of cycling the interface varies sharply from LGPS to Li_m reflecting the quality of the pristine interface. As cycling proceeds this boundary becomes less distinct as decomposition domains begin to cover the LGPS surface.

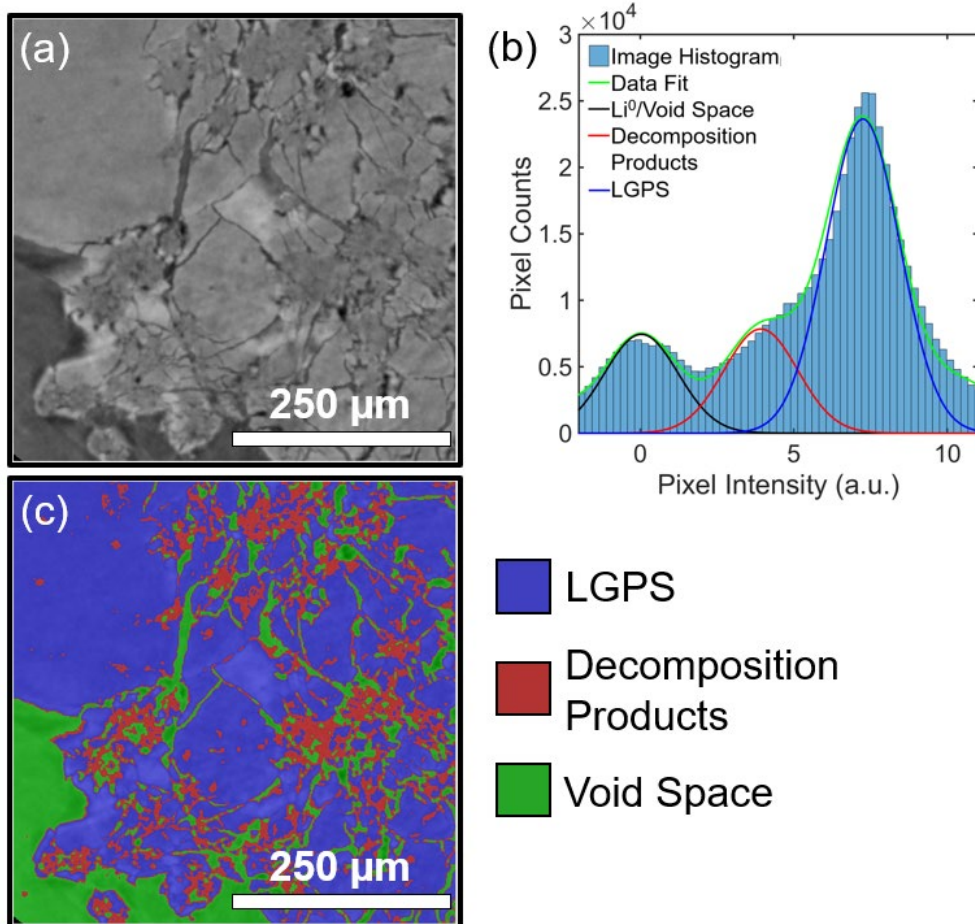


Figure 3.12. (a) Raw tomographic reconstruction used as the starting point for image segmentation. (b) Histogram of image (a) deconvoluted as the sum of three gaussian populations. (c) Final spatial map of differentiated domains using the statistically calculated ranges from the histogram analysis in (b).

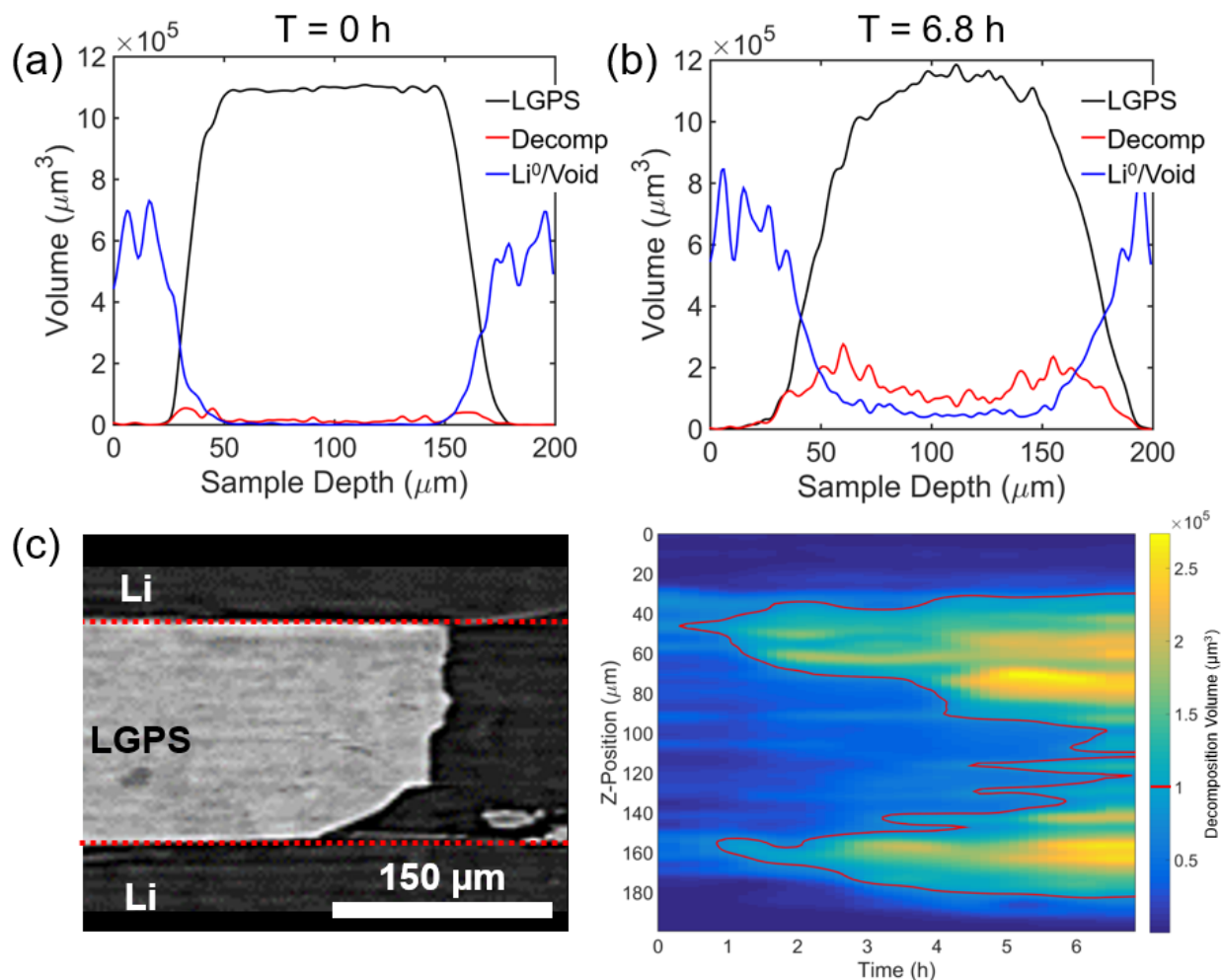


Figure 3.13. a) Pixel counting analysis on an all-solid-state LGPS cell monitoring the three primary domains, LGPS (black), decomposed material (red) and empty space/Li (blue) at the beginning of cycling plotted as a function of sample depth. b) Pixel counting analysis performed at the end of cycling. c) decomposition volume plotted as a function of time and sample position with the red line indicating populations significantly above the background signal. Image to the right is a reference to guide the eye to the original position of the electrode/electrolyte interfaces at the beginning of cycling.

Figure 3.14a, qualitatively shows the increase in decomposition volume on both sides of the LGPS pellet as cycling progresses. In the absence of a true working or counter electrode in the symmetric cell geometry, electrodes are referred to herein by their physical position relative to the LGPS pellet. In all measurements the “Top” electrode (red traces) was subjected to reductive polarization first, followed by the bottom electrode (blue traces). Derivative traces of the data presented in Figure 3.14a are provided in Figure 3.14b, where a clear correlation exists

between the potential of the electrode and the rate of surface decomposition at the LGPS/Li⁰ interface. Specifically, upon reductive polarization of the top electrode LGPS decomposes readily, in good agreement with previous work in our group suggesting that LGPS decomposes at potentials below 0.1 V vs. Li^{+/0}.¹⁹ Further, upon oxidative polarization of the top electrode, decomposition on the bottom electrode is evident as this electrode becomes reducing towards LGPS.

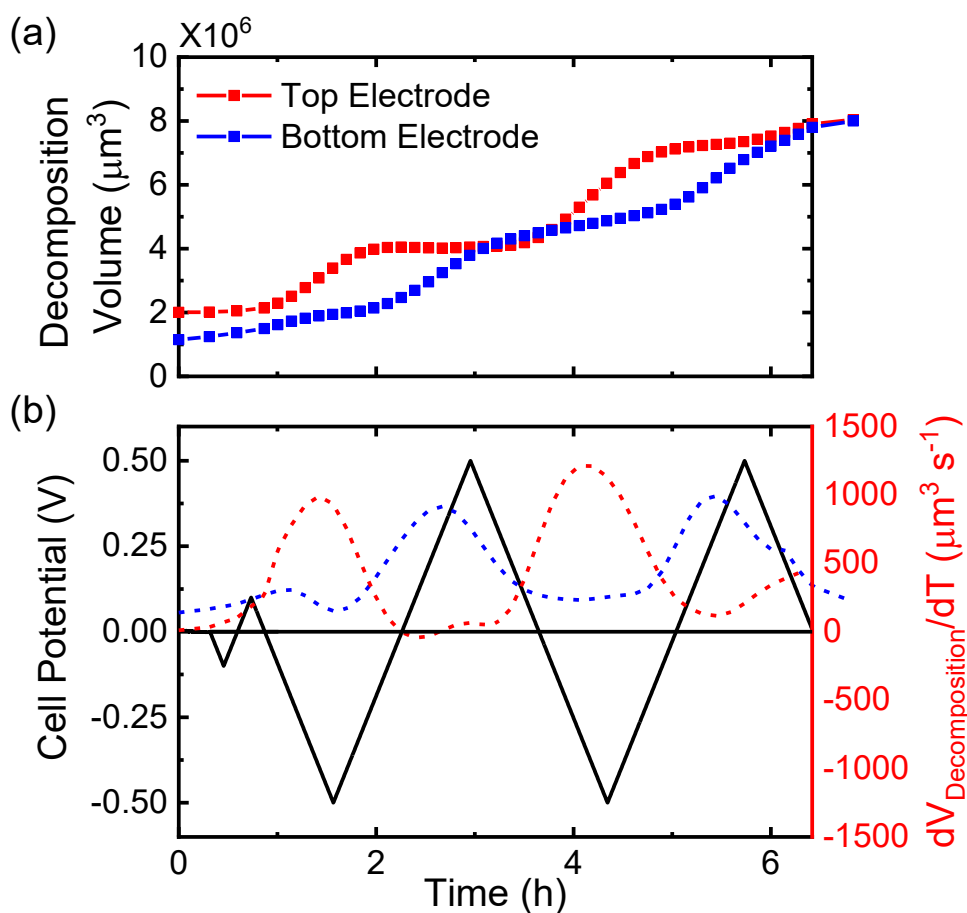


Figure 3.14. (a) Computed volumes of decomposed material formed at the top (red) and bottom (blue) electrode/LGPS interfaces. (b) Derivative traces of the decomposition volume at the top (red dashed trace) and bottom (blue dashed trace) electrodes plotted against time along with the accompanying cell voltage profile (solid black trace).

To understand the consequences of reductive decomposition, individual decomposition sites were investigated as a function of electrochemical cycling. Figure 3.15 presents a series of

3D renderings of an individual decomposition site at different time points during a single reductive sweep. These renderings were obtained from the top surface of the pellet shown in Figure 4 focusing on the formation of a single protrusion. Again, the color depicts the height of each point along the pellet surface with blue denoting the native surface and yellow representing vertical protrusions from it. As the cell is polarized negatively the decomposed inclusion grows outward across the surface, forming a roughly hemispherical protrusion from the LGPS pellet. Eventually (after ca. 1.4 h), the inclusion reaches a critical size (ca. 150 μm in diameter), resulting in crack formation at the locus of the nucleation site. After formation, the resultant crack propagates through the entire thickness of the pellet at rates faster than the time resolution of the experiment. Direct comparison between cross-sectional images and surface maps reveals that the origin of nearly all cracks in the material overlap with such a nucleation site at one of the electrodes (Figure 3.16). Notably we do not observe dendritic lithium growth through these decomposition domains which would appear as void space inside the domains themselves. Similarly, as lithium dendrites found in other solid and liquid electrolytes typically exhibit diameters between 1 and 100 μm they should be visible within the spatial resolution of this technique.^{8, 24, 52}

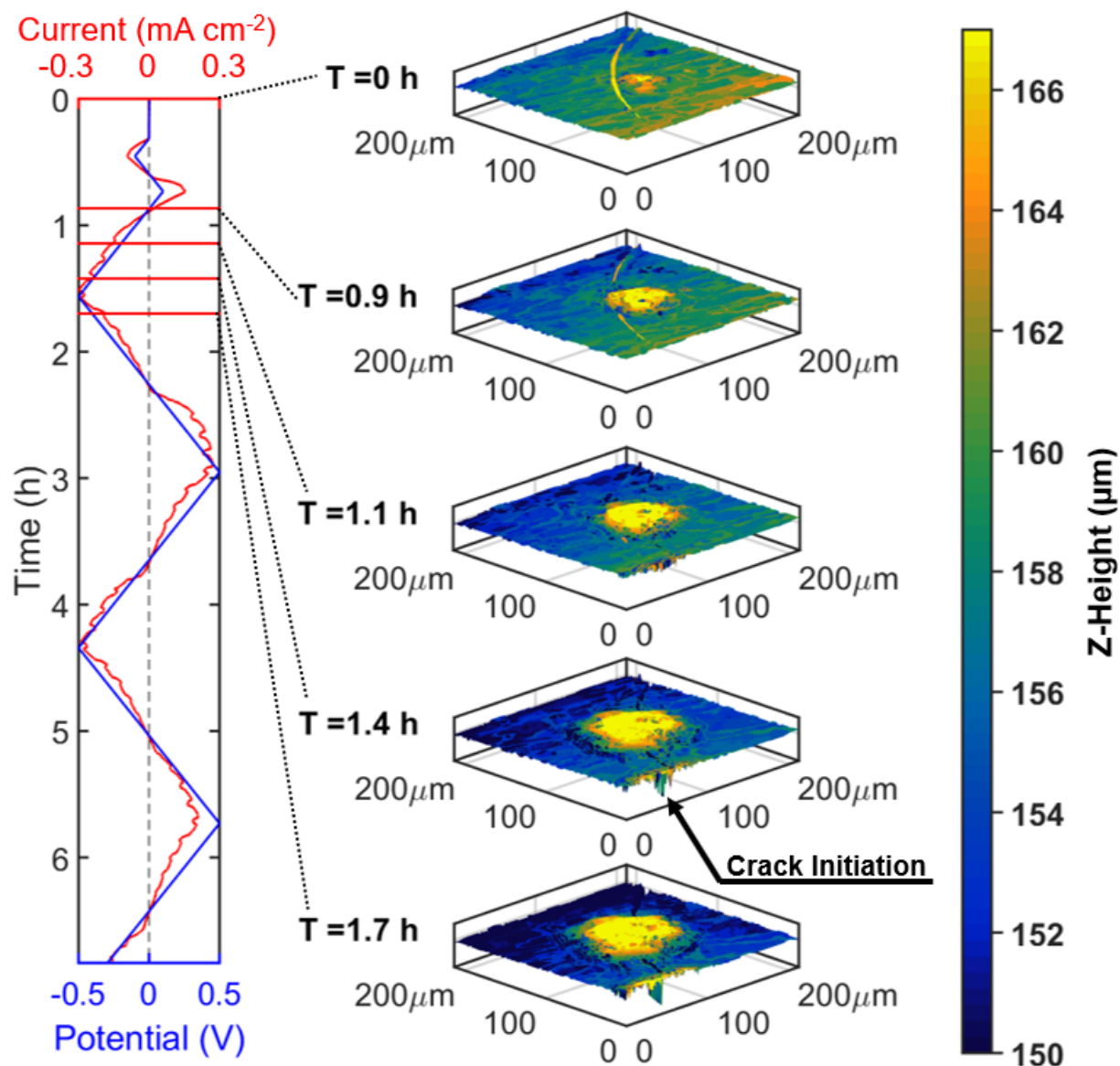


Figure 3.15. Three-dimensional renderings of a single decomposition domain while polarizing the top electrode reductively. As the potential is reduced the size of the inclusion increases radially and vertically. After a certain point ($t = 1.4$ h) the inclusion exerts sufficient pressure on the LGPS sub-structure and a crack forms at the locus of the inclusion.

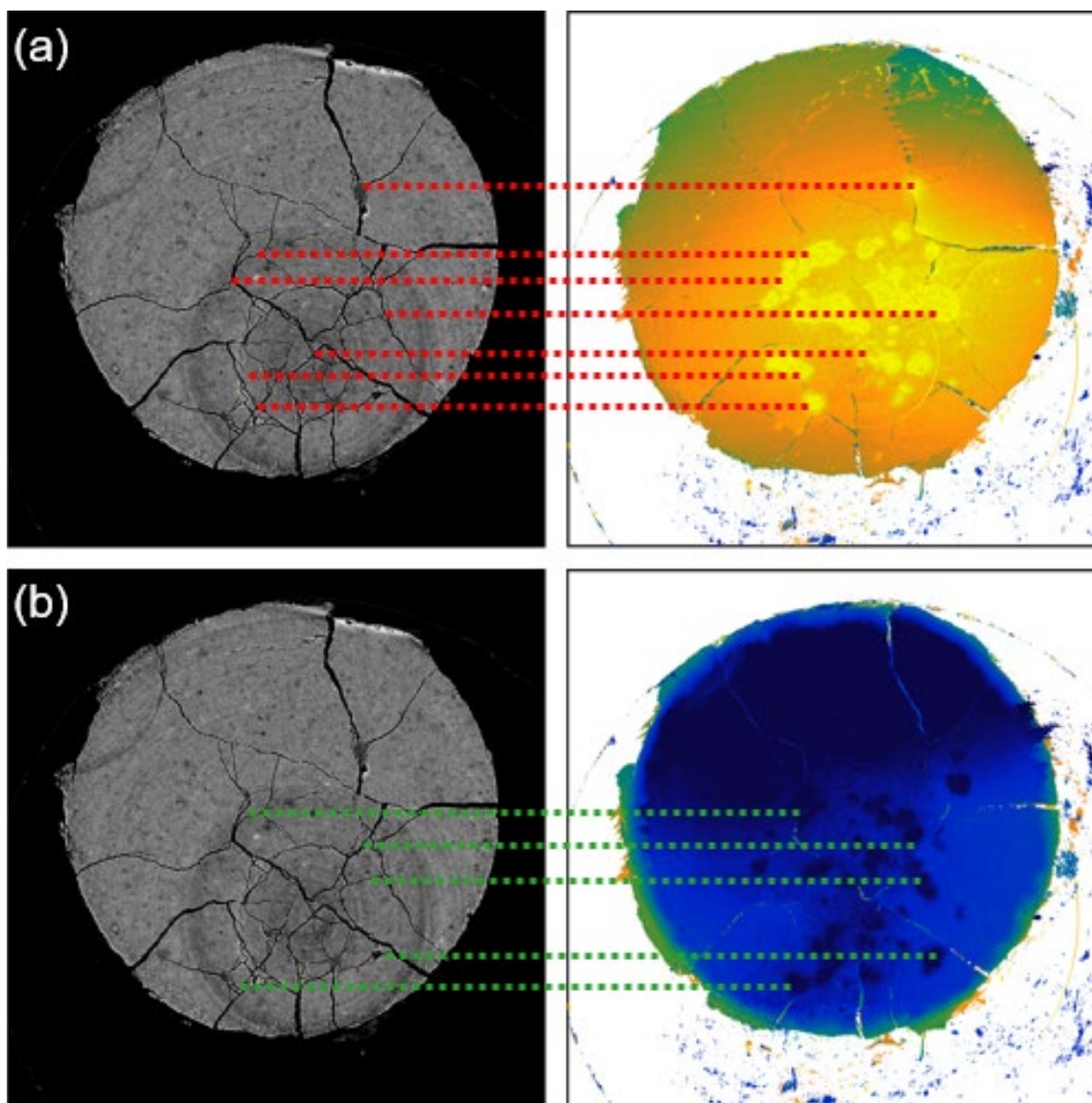


Figure 3.16. Correlation between the origin points of cracks in the bulk LGPS pellet and nucleation sites of material decomposition both at the (a) top and (b) bottom of the LGPS pellet.

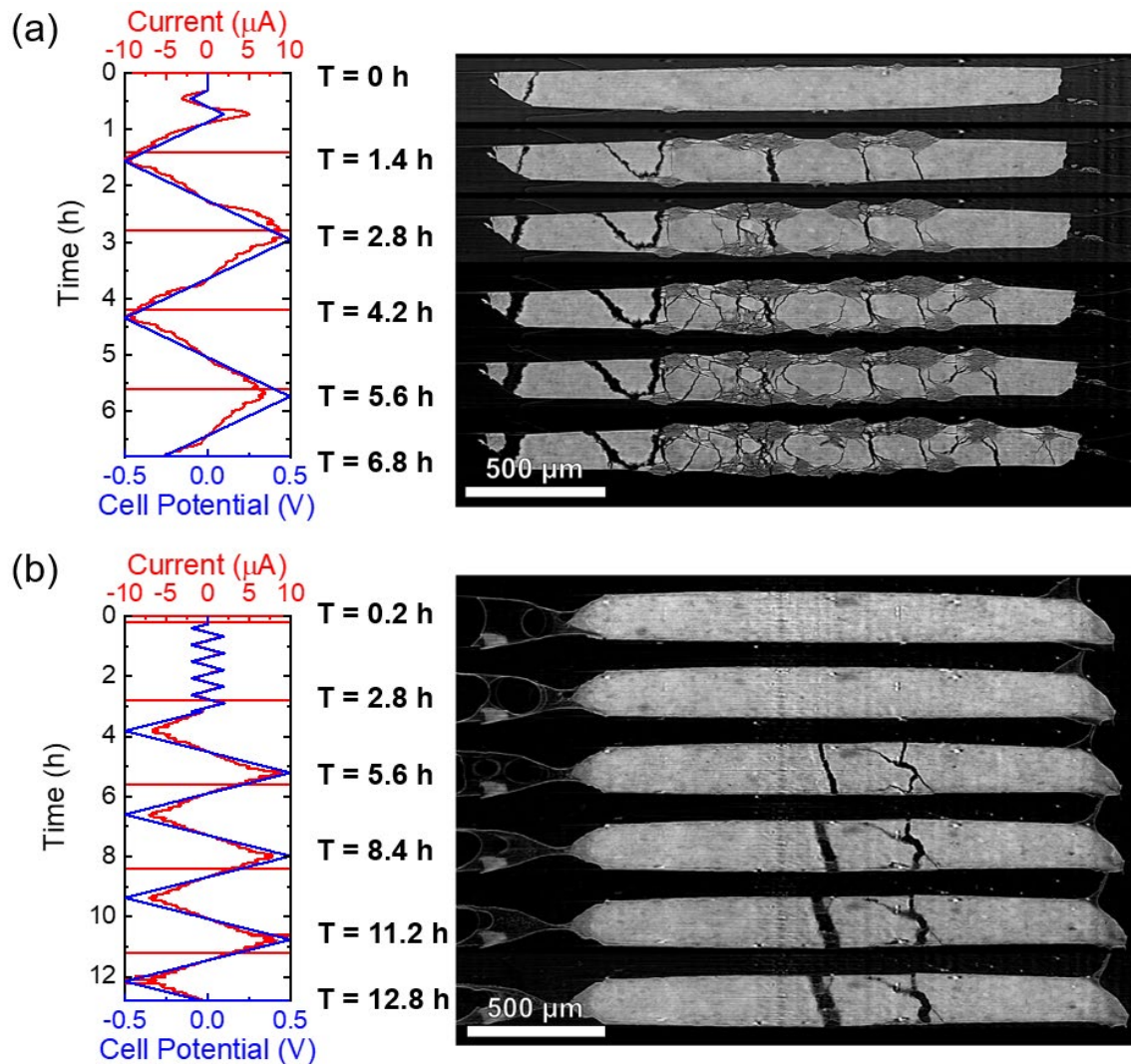


Figure 3.17. a) Orthographic projections of an all-solid-state cell in the x, z-plane as a function of cycle time. b) Orthographic projections of an interlayer cell in the x, z-plane as a function of cycling time.

We next quantify the stresses incurred by the SE at the LGPS/Li⁰ interface. An approximate critical stress was calculated using the average size of the inclusions formed during cycling. The shape of these inclusions was determined from both contrast imaging and orthographic projections of the SE pellet (Figures 3.15 and 3.17, respectively) and can be approximated by an oblate spheroid with lateral radii of ca. 72 μm and heights of ca. 19 μm . From this approximate shape a critical stress value was calculated using the bulk elastic modulus of LGPS pellets as reported elsewhere.⁵³ The results of this analysis (a detailed description of

which is given Appendix B) are presented in Table 3.2 and reveal a critical stress of ca. 0.8 GPa. This calculated critical stress is substantially lower than the shear modulus reported for cold-pressed LGPS pellets (4.5 GPa)⁵⁴ and an order of magnitude lower than the proposed shear modulus necessary to suppress lithium dendrite formation.^{55, 56} While the mechanical properties of SE pellets are strongly impacted by experimental factors such as synthesis temperature, grain size and pelletizing pressure,⁵⁷ such a large discrepancy between the predicted shear modulus and the apparent measured critical stress must reflect a substantial mechanical instability resulting from the electrochemical decomposition of LGPS in the all-solid-state configuration.

Table 3.2. Computed critical volumes of material inclusions and critical stress (σ_c) developed at the point of pellet fracture.

Inclusion Height (μm)	Inclusion Radius (μm)	Inclusion Volume (μm^3)	E_{LGPS} (GPa) ⁵³	E_{Li} (GPa) ⁴⁸	σ_c (GPa)
19 ± 3	72 ± 20	$1.7 \pm 0.3 \times 10^5$	10.5	4.9	0.8 ± 0.2

3.3.4 Interfacial Properties of Interlayer-Modified LGPS Cells

To understand the impact a solvate interlayer has on the LGPS/Li⁰ interface, 3D renderings of the interlayer cell were computed and are presented in Figure 3.18. Several differences are apparent when comparing the renderings obtained from the interlayer cell with those of the all-solid-state cell. One of the most obvious is that, in contrast to the all-solid-state cell, some of the empty volume within the interlayer cell contains liquid electrolyte, which can be directly imaged in the reconstructions due to x-ray attenuation by LiTFSI. 3D renderings show that excess electrolyte (electrolyte which is not pressed between the LGPS and the electrode) forms a corona around the LGPS pellet, pooling (likely due to gravity) at the bottom of the cell. Importantly, as this liquid electrolyte does not extend across the entirety of the cell volume it is not expected to contribute to ionic conductivity during cell cycling.

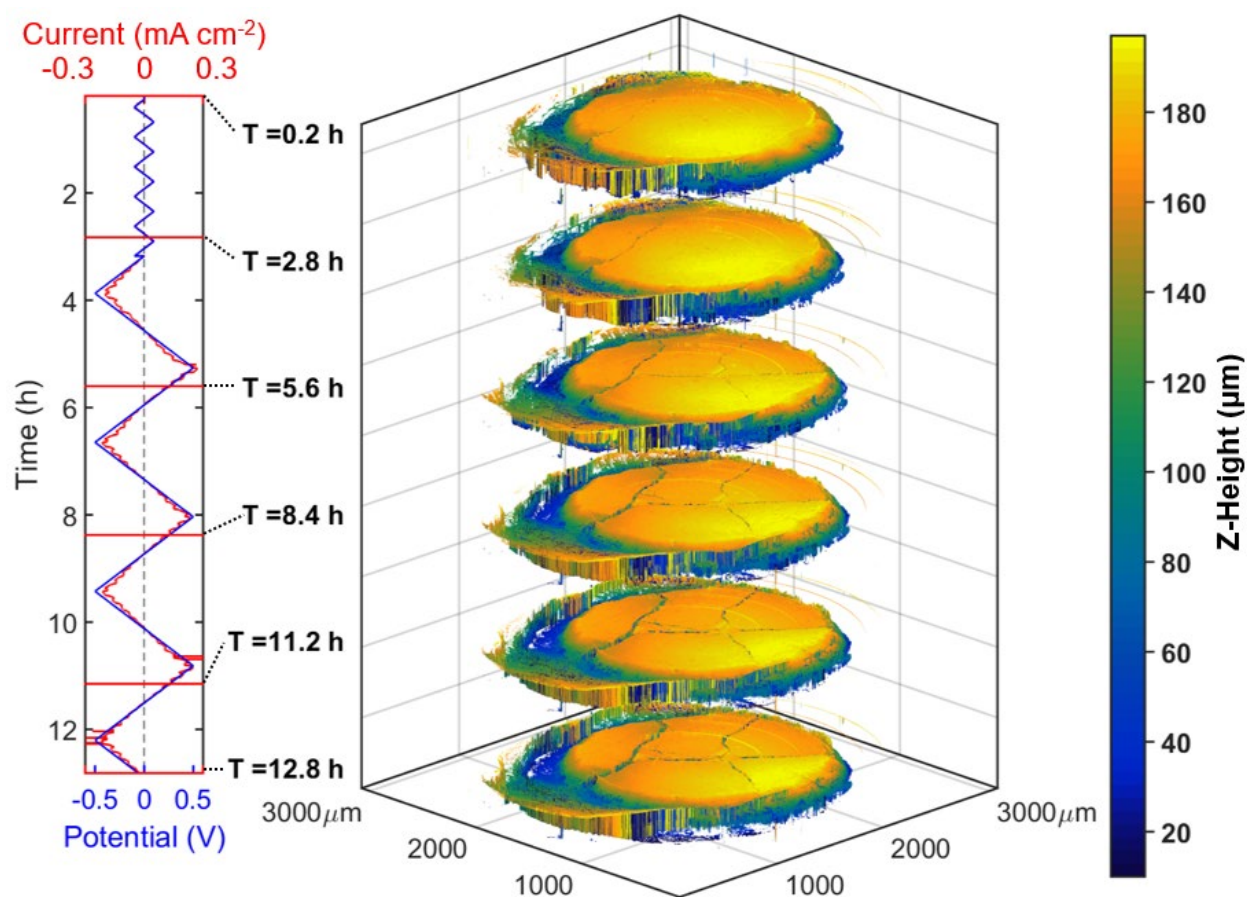


Figure 3.18. Three-dimensional renderings of an LGPS cell with solvate interlayers applied to both sides of the SE pellet. Crack formation is evident as cycling progresses, however in contrast to the all-solid-state-cell no obvious morphological changes are observed at the pellet surface during cell cycling

The mechanical behavior of the interlayer-modified interface strongly contrasts the behaviors exhibited by the all-solid-state interface. As the cell is cycled, the LGPS/Li⁰ interfaces in the interlayer cell remain smooth and nearly constant in appearance, showing no signs of the obvious morphological changes found in the all-solid-state cell. Notably, while cracks form in the pellet with cycling, they cannot be directly tied to interfacial evolution, as they were in the all-solid-state cell. To further confirm this observation, a histogram analysis of pixel distributions near the interface was performed and is presented in Figure 3.19. Interestingly, pixel intensities attributable to decomposition products are completely absent from the interface, suggesting that the reduction of LGPS is suppressed by the solvate interlayer.

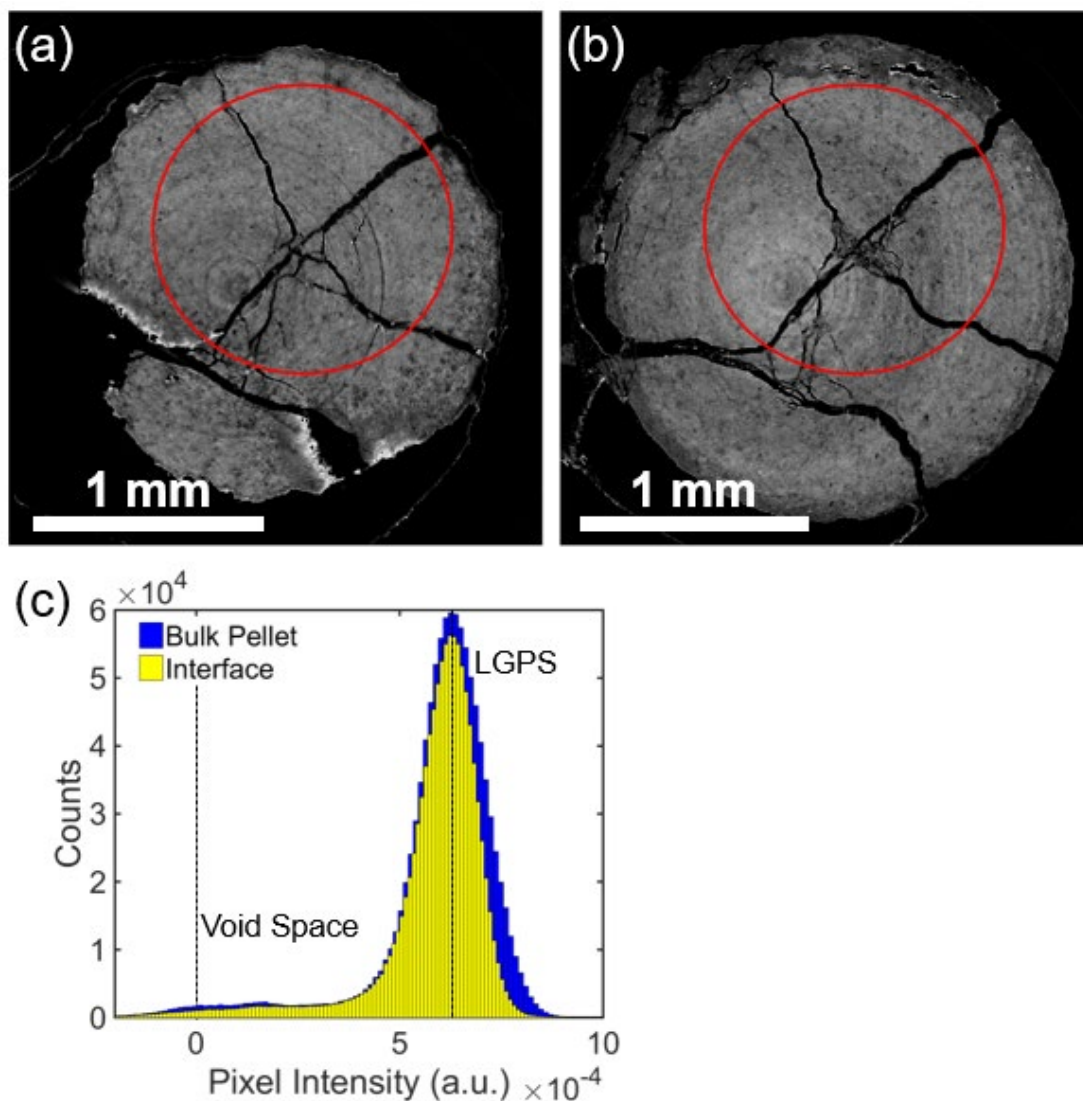


Figure 3.19. Images collected from an interlayer cell collected (a) near the LGPS/Li⁰ interface and (b) at the center of the LGPS pellet. (c) Histogram analysis from the region demarcated by the red circles in images (a) and (b).

The contrast between the two cell configurations reported here clearly demonstrates the impact of interfacial processes on the mechanical stability of the SE. In the absence of any interfacial modification, LGPS is subjected to untenably reducing potentials due to its direct contact with the Li⁰ surface. These highly reducing potentials result in the nucleation of decomposition domains at the LGPS/Li⁰ interface, the appearance of which coincides with the onset of mechanical pulverization. Interestingly, there is no evidence that lithium dendrite

growth plays a role in the local mechanical decomposition of the SE pellet. Upon the application of an interlayer to the electrode/electrolyte interface the behavior of the LGPS pellet is markedly different.

The application of a solvate interlayer addresses the thermodynamic instabilities of the SE, which would otherwise result in the interfacial decomposition of LGPS. The highly concentrated electrolyte serves to screen the reducing potentials at the electrode interface, thereby mitigating the reductive decomposition of the LGPS pellet. The absence of interfacial decomposition, and consequently the absence of large decomposition domains, results in lower interfacial stresses which would otherwise result in crack formation across the SE pellet, thereby engendering greatly enhanced mechanical stability to the interlayer-modified electrolyte. Additionally, the interlayer provides the simultaneous advantage of homogenizing lithium transport across the interface, suppressing the formation of highly localized nucleation sites for material decomposition. This combination of homogeneous lithium transport and enhanced thermodynamic stability afforded by the solvated interlayer is thus the origin of the enhanced mechanical stability and electrochemical performance of solvate modified LGPS pellets.

It is worth noting that despite the absence of discernable decomposition at the interface, crack formation is still present in the interlayer cell, albeit to a lesser extent than in the all-solid-state cell. As quantified in Figure 3.9, crack formation in the interlayer cell is largely decoupled from the cell potential with the exception of the first cycle within the widened voltage window. This suggests that while the application of the interlayer is effective at mitigating interface driven stress, other stresses within the cell volume still contribute to some mechanical instability in the electrolyte. While the origin of these stresses cannot be determined unambiguously from tomographic reconstructions, it is likely that they arise from nonuniformities in stack pressure

developed during cycling, potentially as a result of macroscopic lithium movement and redistribution at the electrode as has been suggested for both liquid, and solid-polymer electrolytes when employing metallic lithium anodes.⁵⁸⁻⁶⁰

3.4 Conclusion

Herein the mechanical responses of both unmodified and interlayer modified $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$ solid electrolytes were investigated during electrochemical cycling using *operando* x-ray tomography. We demonstrate that, in the case of the all-solid-state configuration, the electrochemical reduction of LGPS at the Li^0 electrode does not effectively passivate the interface, allowing the decomposition of LGPS to proceed unhindered during lithium plating resulting in the formation of the large, decomposed domains at the electrolyte surface. The low density of these domains relative to LGPS imparts compressive stress on the LGPS substrate, apparently independent of the growth of lithium dendrites from the electrode. As the decomposed domain size approaches a critical dimension (ca. 150 μm in diameter) the imposed stress on the substrate exceeds the strength of the pellet forcing the formation of cracks across the bulk SE. The close spatial correlation between decomposition sites and material fracture points suggests that the formation of cracks in the bulk electrolyte is a direct result of stresses imposed by electrochemical decomposition at the LGPS/ Li^0 interface. Further, using the dimensions of these domains, we show that the critical shear stress necessary to fracture the LGPS pellet is 0.8 ± 0.2 GPa, roughly an order of magnitude lower than the proposed shear modulus necessary to prevent lithium dendrite formation. This suggests that electrochemical instabilities at the Li^0 /LGPS interface dramatically reduce the mechanical stability of the SE and, consequently, its ability to prevent lithium dendrite formation during cycling.

In stark contrast to the interfacial changes observed in the all-solid-state cell, the interfaces formed in the interlayer cell exhibit enhanced uniformity during lithium plating and stripping. Specifically, there is no evidence that the decomposition domains found in the all-solid-state cell form at the LGPS/Li⁰ interface when an interlayer is present. This reflects a suppression of the reductive decomposition of LGPS at the interface, in good agreement with previous results from our group demonstrating the thermodynamic stability afforded by the application of the interlayer.³⁰ The absence of decomposition domains results in a substantial decrease in interfacial stress during cycling, thereby affording the observed improvements to the bulk mechanical stability of the LGPS pellet. Additionally, the mechanical pliability of the liquid interlayer homogenizes ion transport at the interface, suppressing the formation of inhomogeneous current densities, which are known to lead to additional mechanical instabilities. Thus, while crack formation in the material is evident, the bulk continuity of the pellet is greatly enhanced relative to the all-solid-state cell, likely improving the electrolyte's ability to suppress dendrite propagation. The consequences of this added stability are born out in the electrochemical performance of the interlayer cell, which exhibits more stable cell resistance and longer cycle life relative to the all-solid-state cell.

From these results, we conclude that the modification of the electrode/electrolyte interface is of paramount importance in the development of all-solid-state lithium batteries to suppress both the electrochemical decomposition and mechanical failure of the SE. Further we show that the application of a mechanically pliable liquid interlayer is sufficient to thermodynamically protect LGPS solid electrolytes from reductive decomposition, thereby mitigating interface driven mechanical failure which would otherwise fracture the SE pellet. From this we propose that the utilization of mechanically pliable interface modifications is

necessary to address many of the mechanical instabilities resulting from interfacial decomposition reactions at the electrode/electrolyte interface during cycling.

3.5 References

- (1) Goodenough, J. B.; Park, K. S., The Li-Ion Rechargeable Battery: A Perspective. *J Am Chem Soc* **2013**, *135* (4), 1167-1176.
- (2) Manthiram, A., An Outlook on Lithium Ion Battery Technology. *Acs Central Sci* **2017**, *3* (10), 1063-1069.
- (3) Cheng, X.-B.; Zhang, R.; Zhao, C.-Z.; Zhang, Q., Toward Safe Lithium Metal Anode in Rechargeable Batteries: A Review. *Chem Rev* **2017**, *117* (15), 10403-10473.
- (4) Xu, K., Nonaqueous liquid electrolytes for lithium-based rechargeable batteries. *Chem Rev* **2004**, *104* (10), 4303-4417.
- (5) Feng, X.; Ouyang, M.; Liu, X.; Lu, L.; Xia, Y.; He, X., Thermal runaway mechanism of lithium ion battery for electric vehicles: A review. *Energy Storage Mater* **2018**, *10*, 246-267.
- (6) Galushkin, N. E.; Yazvinskaya, N. N.; Galushkin, D. N., Mechanism of Thermal Runaway in Lithium-Ion Cells. *J Electrochem Soc* **2018**, *165* (7), A1303-A1308.
- (7) Zheng, F.; Kotobuki, M.; Song, S.; Lai, M. O.; Lu, L., Review on solid electrolytes for all-solid-state lithium-ion batteries. *J Power Sources* **2018**, *389*, 198-213.
- (8) Frenck, L.; Sethi, G. K.; Maslyn, J. A.; Balsara, N. P., Factors That Control the Formation of Dendrites and Other Morphologies on Lithium Metal Anodes. *Frontiers in Energy Research* **2019**, *7* (115).
- (9) Inoue, T.; Mukai, K., Are All-Solid-State Lithium-Ion Batteries Really Safe?– Verification by Differential Scanning Calorimetry with an All-Inclusive Microcell. *Acs Appl Mater Inter* **2017**, *9* (2), 1507-1515.

- (10) Xia, S.; Wu, X.; Zhang, Z.; Cui, Y.; Liu, W., Practical Challenges and Future Perspectives of All-Solid-State Lithium-Metal Batteries. *Chem-Us* **2019**, *5* (4), 753-785.
- (11) Chen, S.; Xie, D.; Liu, G.; Mwizerwa, J. P.; Zhang, Q.; Zhao, Y.; Xu, X.; Yao, X., Sulfide solid electrolytes for all-solid-state lithium batteries: Structure, conductivity, stability and application. *Energy Storage Mater* **2018**, *14*, 58-74.
- (12) Kamaya, N.; Homma, K.; Yamakawa, Y.; Hirayama, M.; Kanno, R.; Yonemura, M.; Kamiyama, T.; Kato, Y.; Hama, S.; Kawamoto, K.; Mitsui, A., A lithium superionic conductor. *Nature Materials* **2011**, *10* (9), 682-686.
- (13) Dewald, G. F.; Ohno, S.; Kraft, M. A.; Koerver, R.; Till, P.; Vargas-Barbosa, N. M.; Janek, J.; Zeier, W. G., Experimental Assessment of the Practical Oxidative Stability of Lithium Thiophosphate Solid Electrolytes. *Chem Mater* **2019**, *31* (20), 8328-8337.
- (14) Han, F.; Zhu, Y.; He, X.; Mo, Y.; Wang, C., Electrochemical Stability of $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$ and $\text{Li}_7\text{La}_3\text{Zr}_2\text{O}_{12}$ Solid Electrolytes. *Adv Energy Mater* **2016**, *6* (8), 1501590.
- (15) Richards, W. D.; Miara, L. J.; Wang, Y.; Kim, J. C.; Ceder, G., Interface Stability in Solid-State Batteries. *Chem Mater* **2016**, *28* (1), 266-273.
- (16) Pervez, S. A.; Cambaz, M. A.; Thangadurai, V.; Fichtner, M., Interface in Solid-State Lithium Battery: Challenges, Progress, and Outlook. *Acs Appl Mater Inter* **2019**, *11* (25), 22029-22050.
- (17) Nie, K.; Hong, Y.; Qiu, J.; Li, Q.; Yu, X.; Li, H.; Chen, L., Interfaces Between Cathode and Electrolyte in Solid State Lithium Batteries: Challenges and Perspectives. *Frontiers in Chemistry* **2018**, *6* (616).
- (18) Xiao, Y.; Wang, Y.; Bo, S.-H.; Kim, J. C.; Miara, L. J.; Ceder, G., Understanding interface stability in solid-state batteries. *Nat Rev Mater* **2020**, *5* (2), 105-126.

- (19) Sang, L.; Haasch, R. T.; Gewirth, A. A.; Nuzzo, R. G., Evolution at the Solid Electrolyte/Gold Electrode Interface during Lithium Deposition and Stripping. *Chem Mater* **2017**, 29 (7), 3029-3037.
- (20) Wenzel, S.; Randau, S.; Leichtweiß, T.; Weber, D. A.; Sann, J.; Zeier, W. G.; Janek, J., Direct Observation of the Interfacial Instability of the Fast Ionic Conductor Li₁₀GeP₂S₁₂ at the Lithium Metal Anode. *Chem Mater* **2016**, 28 (7), 2400-2407.
- (21) Shen, F.; Dixit, M. B.; Xiao, X.; Hatzell, K. B., Effect of Pore Connectivity on Li Dendrite Propagation within LLZO Electrolytes Observed with Synchrotron X-ray Tomography. *ACS Energy Letters* **2018**, 3 (4), 1056-1061.
- (22) Sun, F.; Dong, K.; Osenberg, M.; Hilger, A.; Risse, S.; Lu, Y.; Kamm, P. H.; Klaus, M.; Markötter, H.; García-Moreno, F.; Arlt, T.; Manke, I., Visualizing the morphological and compositional evolution of the interface of InLi-anode|thio-LISiON electrolyte in an all-solid-state Li–S cell by in operando synchrotron X-ray tomography and energy dispersive diffraction. *J Mater Chem A* **2018**, 6 (45), 22489-22496.
- (23) Tippens, J.; Miers, J. C.; Afshar, A.; Lewis, J. A.; Cortes, F. J. Q.; Qiao, H.; Marchese, T. S.; Di Leo, C. V.; Saldana, C.; McDowell, M. T., Visualizing Chemomechanical Degradation of a Solid-State Battery Electrolyte. *ACS Energy Letters* **2019**, 4 (6), 1475-1483.
- (24) Seitzman, N.; Guthrey, H.; Sulas, D. B.; Platt, H. A. S.; Al-Jassim, M.; Pylypenko, S., Toward All-Solid-State Lithium Batteries: Three-Dimensional Visualization of Lithium Migration in β -Li₃PS₄ Ceramic Electrolyte. *J Electrochem Soc* **2018**, 165 (16), A3732-A3737.

- (25) Du, M.; Liao, K.; Lu, Q.; Shao, Z., Recent advances in the interface engineering of solid-state Li-ion batteries with artificial buffer layers: challenges, materials, construction, and characterization. *Energy Environ Sci* **2019**, *12* (6), 1780-1804.
- (26) Gurung, A.; Pokharel, J.; Baniya, A.; Pathak, R.; Chen, K.; Lamsal, B. S.; Ghimire, N.; Zhang, W.-H.; Zhou, Y.; Qiao, Q., A review on strategies addressing interface incompatibilities in inorganic all-solid-state lithium batteries. *Sustainable Energy & Fuels* **2019**, *3* (12), 3279-3309.
- (27) Bosubabu, D.; Sivaraj, J.; Sampathkumar, R.; Ramesha, K., LAGP|Li Interface Modification through a Wetted Polypropylene Interlayer for Solid State Li-Ion and Li-S batteries. *ACS Applied Energy Materials* **2019**, *2* (6), 4118-4125.
- (28) Simon, F. J.; Hanauer, M.; Richter, F. H.; Janek, J., Interphase Formation of PEO20:LiTFSI-Li6PS5Cl Composite Electrolytes with Lithium Metal. *Acs Appl Mater Inter* **2020**, *12* (10), 11713-11723.
- (29) Liu, Q.; Yu, Q.; Li, S.; Wang, S.; Zhang, L.; Cai, B.; Zhou, D.; Li, B., Safe LAGP-based all solid-state Li metal batteries with plastic super-conductive interlayer enabled by in-situ solidification. *Energy Storage Mater* **2020**, *25*, 613-620.
- (30) Philip, M. A.; Sullivan, P. T.; Zhang, R.; Wooley, G. A.; Kohn, S. A.; Gewirth, A. A., Improving Cell Resistance and Cycle Life with Solvate-Coated Thiophosphate Solid Electrolytes in Lithium Batteries. *Acs Appl Mater Inter* **2019**, *11* (2), 2014-2021.
- (31) Wu, Y.; Liu, N., Visualizing Battery Reactions and Processes by Using In Situ and In Operando Microscopies. *Chem-US* **2018**, *4* (3), 438-465.
- (32) Shin, M.; Wu, H.-L.; Narayanan, B.; See, K. A.; Assary, R. S.; Zhu, L.; Haasch, R. T.; Zhang, S.; Zhang, Z.; Curtiss, L. A.; Gewirth, A. A., Effect of the Hydrofluoroether

- Cosolvent Structure in Acetonitrile-Based Solvate Electrolytes on the Li⁺ Solvation Structure and Li–S Battery Performance. *Acs Appl Mater Inter* **2017**, 9 (45), 39357-39370.
- (33) Gürsoy, D. a.; De Carlo, F.; Xiao, X.; Jacobsen, C., TomoPy: a framework for the analysis of synchrotron tomographic data. *J Synchrotron Radiat* **2014**, 21 (Pt 5), 1188-1193.
- (34) Sang, L.; Bassett, K. L.; Castro, F. C.; Young, M. J.; Chen, L.; Haasch, R. T.; Elam, J. W.; Dravid, V. P.; Nuzzo, R. G.; Gewirth, A. A., Understanding the Effect of Interlayers at the Thiophosphate Solid Electrolyte/Lithium Interface for All-Solid-State Li Batteries. *Chem Mater* **2018**, 30 (24), 8747-8756.
- (35) Bard, A. J.; Faulkner, L. R., Fundamentals and applications. *Electrochemical methods* **2001**, 2 (482), 580-632.
- (36) Zhu, Y.; He, X.; Mo, Y., First principles study on electrochemical and chemical stability of solid electrolyte–electrode interfaces in all-solid-state Li-ion batteries. *J Mater Chem A* **2016**, 4 (9), 3253-3266.
- (37) Wenzel, S.; Weber, D. A.; Leichtweiss, T.; Busche, M. R.; Sann, J.; Janek, J., Interphase formation and degradation of charge transfer kinetics between a lithium metal anode and highly crystalline Li₇P₃S₁₁ solid electrolyte. *Solid State Ionics* **2016**, 286, 24-33.
- (38) Shin, M.; Gewirth, A. A., Incorporating Solvate and Solid Electrolytes for All-Solid-State Li₂S Batteries with High Capacity and Long Cycle Life. *Adv Energy Mater* **2019**, 9 (26), 1900938.

- (39) Judez, X.; Martinez-Ibañez, M.; Santiago, A.; Armand, M.; Zhang, H.; Li, C., Quasi-solid-state electrolytes for lithium sulfur batteries: Advances and perspectives. *J Power Sources* **2019**, *438*, 226985.
- (40) Bonnick, P.; Niitani, K.; Nose, M.; Suto, K.; Arthur, T. S.; Muldoon, J., A high performance all solid state lithium sulfur battery with lithium thiophosphate solid electrolyte. *J Mater Chem A* **2019**, *7* (42), 24173-24179.
- (41) Yang, S.; Wang, W.; Lin, C.; Shen, W.; Li, Y., Investigation of Internal Short Circuits of Lithium-Ion Batteries under Mechanical Abusive Conditions. *Energies* **2019**, *12* (10).
- (42) Suzuki, Y.; Cousins, D. S.; Dorgan, J. R.; Stebner, A. P.; Kappes, B. B., Dual-energy X-ray computed tomography for void detection in fiber-reinforced composites. *Journal of Composite Materials* **2019**, *53* (17), 2349-2359.
- (43) Pietsch, P.; Wood, V., X-Ray Tomography for Lithium Ion Battery Research: A Practical Guide. *Annual Review of Materials Research* **2017**, *47* (1), 451-479.
- (44) Lai, J.; Wang, G.; Fan, Z.; Chen, J.; Qin, Z.; Xiao, C.; Wang, S.; Fan, X., Three-dimensional quantitative fracture analysis of tight gas sandstones using industrial computed tomography. *Sci Rep-Uk* **2017**, *7* (1), 1825.
- (45) Xu, L.; Tang, S.; Cheng, Y.; Wang, K.; Liang, J.; Liu, C.; Cao, Y.-C.; Wei, F.; Mai, L., Interfaces in Solid-State Lithium Batteries. *Joule* **2018**, *2* (10), 1991-2015.
- (46) Zhang, X.; Wang, Q. J.; Harrison, K. L.; Roberts, S. A.; Harris, S. J., Pressure-Driven Interface Evolution in Solid-State Lithium Metal Batteries. *Cell Reports Physical Science* **2020**, *1* (2), 100012.
- (47) Berger, M. J.; Hubbell, J. *XCOM: Photon cross sections on a personal computer*; National Bureau of Standards, Washington, DC (USA). Center for Radiation ...: 1987.

- (48) Samsonov, G. V., *Handbook of the Physicochemical Properties of the Elements*. Springer Science & Business Media: 2012.
- (49) Park, J.; Yu, S.-H.; Sung, Y.-E., Design of structural and functional nanomaterials for lithium-sulfur batteries. *Nano Today* **2018**, *18*, 35-64.
- (50) Herrmann, W. A.; Auner, N.; Klingebiel, U., *Synthetic Methods of Organometallic and Inorganic Chemistry, Volume 2, 1996: Volume 2: Groups 1, 2, 13 and 14*. Georg Thieme Verlag: 2014.
- (51) Weker, J. N.; Liu, N.; Misra, S.; Andrews, J. C.; Cui, Y.; Toney, M. F., In situ nanotomography and operando transmission X-ray microscopy of micron-sized Ge particles. *Energy Environ Sci* **2014**, *7* (8), 2771-2777.
- (52) Liu, H.; Cheng, X.-B.; Huang, J.-Q.; Yuan, H.; Lu, Y.; Yan, C.; Zhu, G.-L.; Xu, R.; Zhao, C.-Z.; Hou, L.-P.; He, C.; Kaskel, S.; Zhang, Q., Controlling Dendrite Growth in Solid-State Electrolytes. *ACS Energy Letters* **2020**, *5* (3), 833-843.
- (53) Weber, D. A.; Senyshyn, A.; Weldert, K. S.; Wenzel, S.; Zhang, W.; Kaiser, R.; Berendts, S.; Janek, J.; Zeier, W. G., Structural Insights and 3D Diffusion Pathways within the Lithium Superionic Conductor Li₁₀GeP₂S₁₂. *Chem Mater* **2016**, *28* (16), 5905-5915.
- (54) Wang, Z. Q.; Wu, M. S.; Liu, G.; Lei, X. L.; Xu, B.; Ouyang, C. Y., Elastic Properties of New Solid State Electrolyte Material Li₁₀GeP₂S₁₂: A Study from First-Principles Calculations. *Int J Electrochem Sc* **2014**, *9* (2), 562-568.
- (55) Monroe, C.; Newman, J., The Impact of Elastic Deformation on Deposition Kinetics at Lithium/Polymer Interfaces. *J Electrochem Soc* **2005**, *152* (2), A396.

- (56) Yu, S.; Schmidt, R. D.; Garcia-Mendez, R.; Herbert, E.; Dudney, N. J.; Wolfenstine, J. B.; Sakamoto, J.; Siegel, D. J., Elastic Properties of the Solid Electrolyte $\text{Li}_7\text{La}_3\text{Zr}_2\text{O}_{12}$ (LLZO). *Chem Mater* **2016**, 28 (1), 197-206.
- (57) Sakuda, A.; Hayashi, A.; Tatsumisago, M., Sulfide Solid Electrolyte with Favorable Mechanical Property for All-Solid-State Lithium Battery. *Sci Rep-Uk* **2013**, 3 (1), 2261.
- (58) Ferrese, A.; Newman, J., Modeling Lithium Movement over Multiple Cycles in a Lithium-Metal Battery. *J Electrochem Soc* **2014**, 161 (6), A948-A954.
- (59) Ferrese, A.; Newman, J., Mechanical Deformation of a Lithium-Metal Anode Due to a Very Stiff Separator. *J Electrochem Soc* **2014**, 161 (9), A1350-A1359.
- (60) Ferrese, A.; Albertus, P.; Christensen, J.; Newman, J., Lithium Redistribution in Lithium-Metal Batteries. *J Electrochem Soc* **2012**, 159 (10), A1615-A1623.

Chapter 4: Pressure Dependent Electrochemical Behavior in Di-Lithium Rhodizonate Electrodes

Reprinted with the permission of Madsen, K. E.; Shin, M.; Gewirth, A. A. Pressure-Dependent Electrochemical Behavior of Di-Lithium Rhodizonate Cathodes. Chemistry of Materials, Chem. Mater. 2021, 33, 14, 5738–5747. Copyright 2019 American Chemical Society.

4.1 Introduction

Lithium-ion batteries (LIBs) represent one of the most prolific forms of energy storage for small scale electronic devices.¹ Despite their ubiquity in portable electronics, the adaptation of existing LIB chemistries towards high-capacity applications, including electric vehicles and grid-scale energy storage, has been met with numerous challenges.^{2, 3} In particular, the limited gravimetric capacity (ca. 150 mA h g⁻¹) and high cost of existing metal oxide cathode materials remain pressing issues.^{4, 5} This makes the development of alternative chemistries which support stable, high-capacity Li⁺ storage from abundant chemical feedstocks an active area of research. One promising approach to address these limitations is to transition from conventional metal oxide-based cathodes to redox active organic electrodes.⁶⁻⁸ The low cost of carbonaceous precursors, enormous synthetic versatility afforded by organic chemistry, and the high gravimetric capacities achievable in many organic materials make these materials very attractive as replacements for existing inorganic cathodes.⁹

To date, a vast number of redox-active organic materials have been identified, including amines, carboxylates, imines, and organosulfur compounds.¹⁰⁻¹⁵ Possibly the most well studied of these are quinone-based materials due to their high operating voltages (ca. 3 V vs Li/Li⁺), and

high capacities ($2e^-$ per quinone unit).^{16, 17} Both small molecule quinones and quinone-containing polymers have been considered as high-performance Li^+ storage materials.¹⁸ While polymers generally exhibit higher capacity retention than small molecules (largely due to the low solubility of polymers in conventional organic electrolytes) their gravimetric capacities are limited by the presence of the often electrochemically inactive polymer backbone.¹⁹⁻²² To circumvent this limitation, efforts have been made towards the development of quinone-based small molecules which maximize the number of redox active sites (and therefore the gravimetric capacity) of each molecule per unit mass. The maximum possible active site density for quinone materials is realized in molecular crystals of cyclic oxo-carbons ($C_xO_x^{n-}$), including squarate, croconate and rhodizonate ($x = 4, 5$, and 6 respectively).^{23, 24} While these materials exhibit extremely high theoretical capacities ($> 950 \text{ mA h g}^{-1}$ if all $C=O$ units are electrochemically active) they display rapid capacity fade during cycling, making them ill-fit for practical application.

The oxo-carbon salt di-lithium rhodizonate ($Li_2C_6O_6$) is of particular interest, as it demonstrates a high theoretical capacity (ca. 600 mA h g^{-1}) and can be produced by the oxidation of naturally abundant myo-inositol.²⁵⁻²⁷ Lithium storage in $Li_2C_6O_6$ is facilitated by sequential reduction of the two redox active ortho-quinones resulting in the formation of two lithiated enediols (Figures 4.1a and 4.1b). This reaction is accomplished in the solid-state, by Li^+ insertion between hexagonally close-packed layers of rhodizonate ions (Figures 4.1c and 4.1d) similar to intercalation and deintercalation of Li^+ into layered oxides.²⁵ Previous work has demonstrated that capacities very near theoretical can be achieved during the first discharge; however, the capacity fades rapidly with further cycling.²⁷ Early work attributed this fade to non-negligible solubility of the active material (or one of its discharge products) in organic electrolytes, similar to the behavior displayed by many other redox active small molecules.^{28, 29} More recent work,

however, implicates mechanical decomposition as the primary mode of capacity fade. In particular, large lattice changes during charge and discharge result in active material exfoliation and subsequent electronic isolation after multiple cycles.²⁵ Despite this assignment, much remains unclear about the chemo-mechanical properties of $\text{Li}_2\text{C}_6\text{O}_6$ and the impacts they might have on its electrochemical performance.

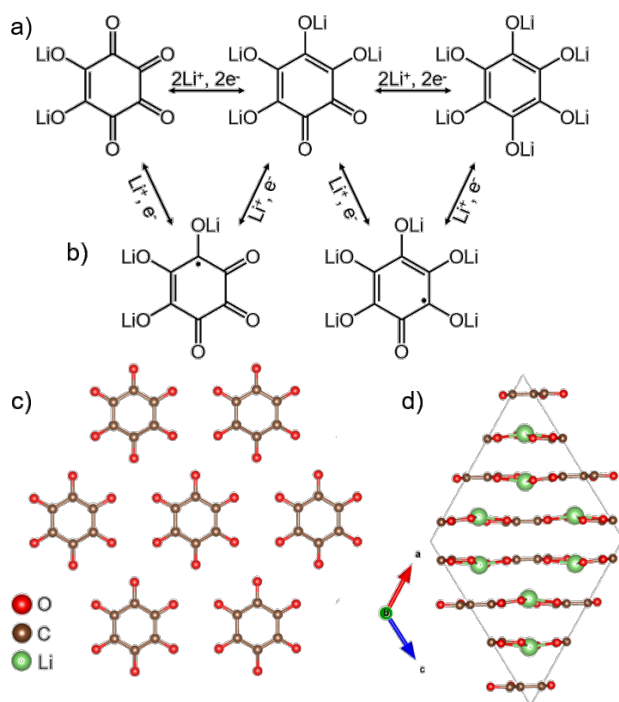


Figure 4.1. Redox processes governing the lithiation and delithiation of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes via a) a concerted two electron transfer and b) a series of one-electron radical intermediaries. c) In plane arrangement of hexagonally close-packed rhodizonate ions. d) Previously reported layered structure of $\text{Li}_2\text{C}_6\text{O}_6$.²⁵

One area of interest in this regard is the structural response of $\text{Li}_2\text{C}_6\text{O}_6$ to the application of external pressure. While this has not been interrogated specifically for $\text{Li}_2\text{C}_6\text{O}_6$, recent experimental work on the high-pressure behavior of $\text{Na}_2\text{C}_6\text{O}_6$ demonstrated changes to both the crystal structure and electrical conductivity as a function of applied pressure in a diamond anvil cell.³⁰ Additionally, computational investigations of other cyclic oxocarbons (including deltic,

squaric, and croconic acids) have suggested that these materials display interesting mechanical properties at elevated pressures including pressure dependent phase transitions and negative Poisson's ratios.³¹⁻³³ Likewise, numerous other works have demonstrated the beneficial impact of high pressures on the electrochemical reversibility and long-term stability of electrode stacks during battery cycling.³⁴⁻³⁸ Thus, the pressure dependent electrochemical characteristics of oxo-carbon materials may be very important in developing strategies to improve their stability and longevity.

Herein, we interrogate the pressure dependent electrochemical properties of di-lithium rhodizonate electrodes. We demonstrate that both the electrochemical availability and the long-term stability of $\text{Li}_2\text{C}_6\text{O}_6$ is strongly impacted by the application of an applied external load. Almost no electrochemical activity is observed from the active material at low pressures. This behavior is reversed at higher pressures, resulting in a substantial decrease in charge transfer impedance and an increase in voltammetric current density. Further, the application of high pressures during galvanostatic cycling results in increased capacity retention, reflecting improved active material stability. To explain these phenomena, we interrogate the crystallographic response of cycled electrodes and demonstrate that at low pressures $\text{Li}_2\text{C}_6\text{O}_6$ rapidly converts from the redox active structure to a redox inactive structure. This crystallographic transition is suppressed by the application of sufficiently high cell pressures. Thus, the origin of the high-pressure stability of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes is attributed to the interplay between the redox active and redox inactive forms of the crystal accessed during electrochemical cycling.

4.2 Experimental

4.2.1 Synthesis of $\text{Li}_2\text{C}_6\text{O}_6$

All chemicals were used as received without further purification. Di-lithium rhodizonate powder was synthesized by the neutralization of rhodizonic acid as reported elsewhere.²⁷ Briefly, rhodizonic acid di-hydrate (99%, Alpha Aesar) was ground together with a 1 molar equivalent of Li_2CO_3 (99%, Sigma-Aldrich) in a mortar and pestle until visually homogeneous and transferred to a scintillation vial. DI water was added dropwise until the powder was fully dissolved (ca. 1 mL H_2O to 100 mg rhodizonic acid) and the mixture was stirred in air for 12 hours followed by separation of the solids by centrifugation (3200 RPM, 20 min). The resulting black powder ($\text{Li}_2\text{C}_6\text{O}_6 \cdot 2\text{H}_2\text{O}$) was rinsed three times with acetone (99%, Sigma-Aldrich) and dried at 80 °C under vacuum (12 h). The obtained, hydrated, product was then dehydrated at 180 °C in air (12 h) to yield the desired $\text{Li}_2\text{C}_6\text{O}_6$ (bright green solid, Figure S1). The structure of the as-synthesized material was verified with vibrational spectroscopy and X-ray diffraction (Figures 4.2 and 4.3 respectively) both of which agree with previous literature.²⁷

4.2.2 Electrode Fabrication

(I) $\text{Li}_2\text{C}_6\text{O}_6$

To prevent rehydration of $\text{Li}_2\text{C}_6\text{O}_6$ during electrode fabrication all manipulations were performed in an argon filled glovebox (O_2 and $\text{H}_2\text{O} < 2$ ppm). Composite electrodes were prepared from a slurry consisting of 80:10:10 wt% $\text{Li}_2\text{C}_6\text{O}_6$, conductive carbon (Super P Li, Timcal) and polyvinylidene difluoride (PVDF, Kynar-Flex) in n-methyl pyrrolidone (NMP, anhydrous, Sigma-Aldrich). The active slurry was homogenized in a planetary mixer (Thinky) with a mixing time of 3 minutes at 2000 RPM without defoaming. This process was repeated a

minimum or three times, or until the slurry appeared smooth and glossy. The resulting slurry was then cast onto Al foil (MiniBin, Sigma-Aldrich) using a doctor blade to an approximate thickness of 300 μm . The cast was then dried at room temperature under active vacuum for 12 hours. Circular electrodes (9.5 mm in diameter) were punched from this cast and exhibited active material loadings of approximately 2 mg cm^{-2} .

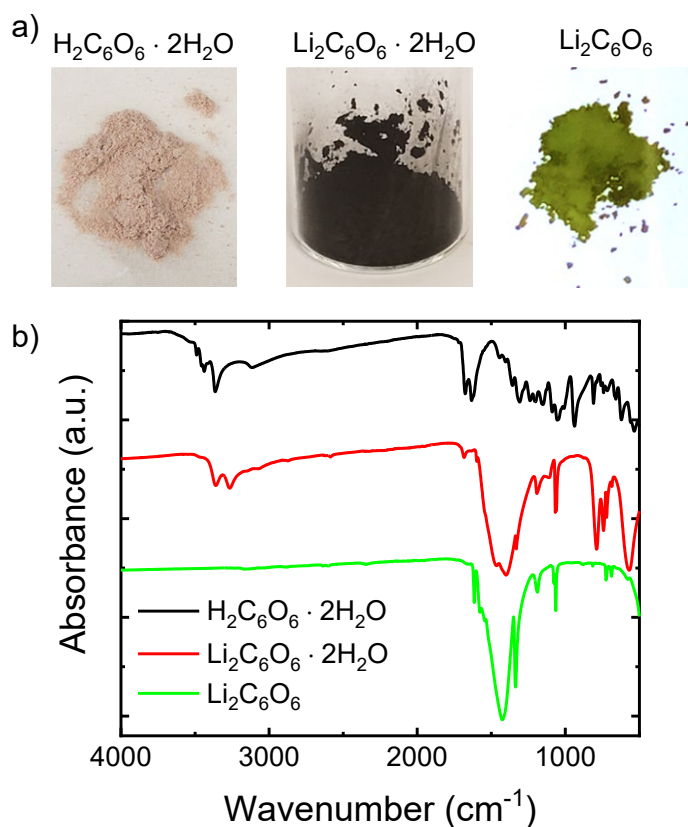


Figure 4.2. a) Photographs of rhodizonic acid, hydrated lithium rhodizonate, and dehydrated di-lithium rhodizonate powders. B) FTIR spectra of rhodizonic acid (black), lithium rhodizonate dihydrate (red), and anhydrous lithium rhodizonate (LRO) powders.

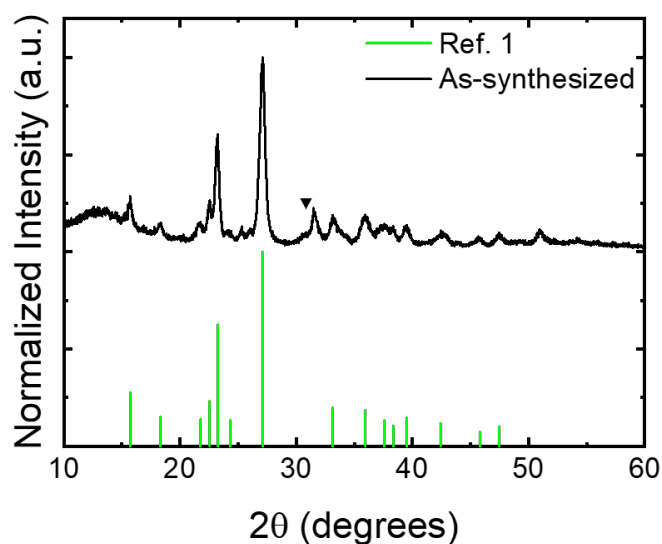


Figure 4.3. Powder X-ray diffraction pattern obtained from the as-synthesized (black trace) compared against the diffraction pattern reported in previous literature.¹ The marked peak at ca. 32 degrees is attributed to a small amount of the redox inactive phase.

(II) Li₇Ti₅O₁₂ anode preparation

Li₇Ti₅O₁₂ (LTO) electrodes were prepared electrochemically from pre-made Li₄Ti₅O₁₂ casts (90% active material by weight, MTI Corporation). Lithiation was accomplished galvanostatically in a large area (ca. 20 cm²) electrochemical cell using Li_m as the counter/reference electrode and 1M Lithium bis(trifluoromethanesulfonyl)-imide (LiTFSI, 99.95%, Sigma-Aldrich) in 1:1 by volume dioxolane (DOL, anhydrous, Sigma-Aldrich): dimethoxy-ethane (DME, anhydrous, Sigma-Aldrich) as the electrolyte. This electrolyte formulation was chosen in place of the more common LiPF₆ based electrolyte to avoid possible hydrolysis of the electrolyte (which is known to form HF along with other reactive intermediaries) which could affect the chemistry of the active materials during post-mortem measurements. The LTO electrode was lithiated under a constant reducing current of 100 μA cm⁻² until the cell registered a potential of 0.5 V vs Li/Li⁺ (Figure 4.4). Individual electrodes (12.7

mm diameter) were then punched from this large electrode, displaying active material loadings of approximately 8 mg cm^{-2} .

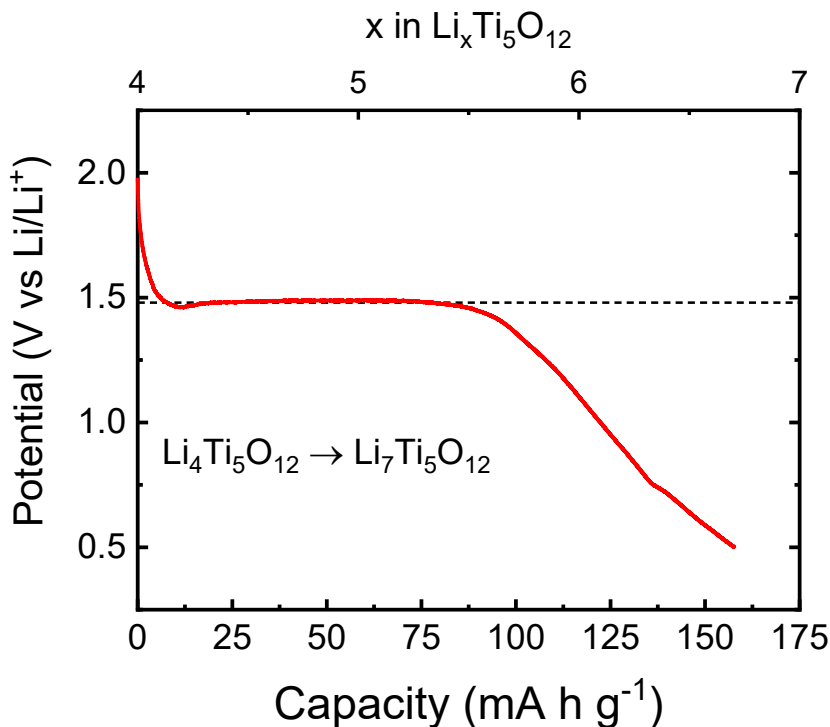


Figure 4.4. Electrochemical lithiation of a large area (ca. 20 cm^2) $\text{Li}_4\text{Ti}_5\text{O}_{12}$ electrode to produce the anode material used in pressure dependent electrochemical measurements.

4.2.3 Electrochemical Measurements

Initial electrochemical characterization of $\text{Li}_2\text{C}_6\text{O}_6$ was conducted in CR2032 coin cells (MTI Corporation, Richmond, CA). Battery stacks were assembled in an argon-filled glovebox (O_2 and $\text{H}_2\text{O} < 2 \text{ ppm}$). Cells were constructed as follows: the positive electrode was placed at the base of the cell followed by a polyethylene/polypropylene separator (Celgard) which was wet with $40 \mu\text{L}$ of electrolyte (1 M LiTFSI in 1:1 DOL/DME). A lithium anode was then prepared as follows: a 14 mm diameter circular electrode was punched from lithium metal foil (0.75 mm thickness, 99.9%, Alfa Aesar). The native oxide layer was removed from both sides of the electrode by scraping the surface with a razor blade. The electrode was then smoothed with a

stainless-steel roller to yield a flat, lustrous surface. The anode was placed onto the separator to serve as the counter/reference electrode. A stainless-steel disc and wave spring were then added onto the lithium to complete the stack, and cells were sealed using a hydraulic coin cell crimper (MTI Corporation, Richmond, CA). $\text{Li}_2\text{C}_6\text{O}_6$ electrodes were also cycled in conical three electrode cells (referred to herein as cone cells, a schematic of which is presented in Figure 4.5) to consider the electrochemical behavior of the active material in the absence of an applied load. Cells were assembled in an argon-filled glovebox with lithium foil serving as the counter and reference electrodes and 1M LiTFSI in 1:1 DOL/DME (ca. 3 mL) serving as electrolyte. Electrochemical cycling of cone cells was likewise conducted in the glovebox.

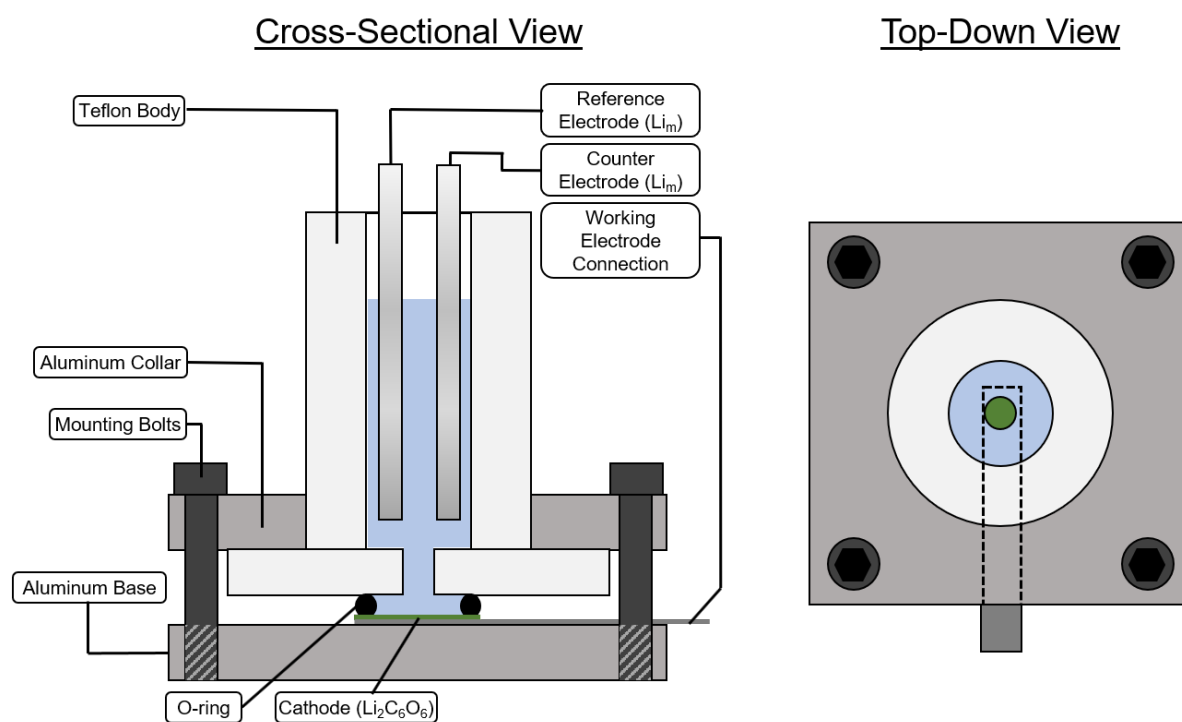


Figure 4.5. Schematic of the cone cell setup used to initially probe the pressure dependent behavior of di-lithium rhodizonate electrodes.

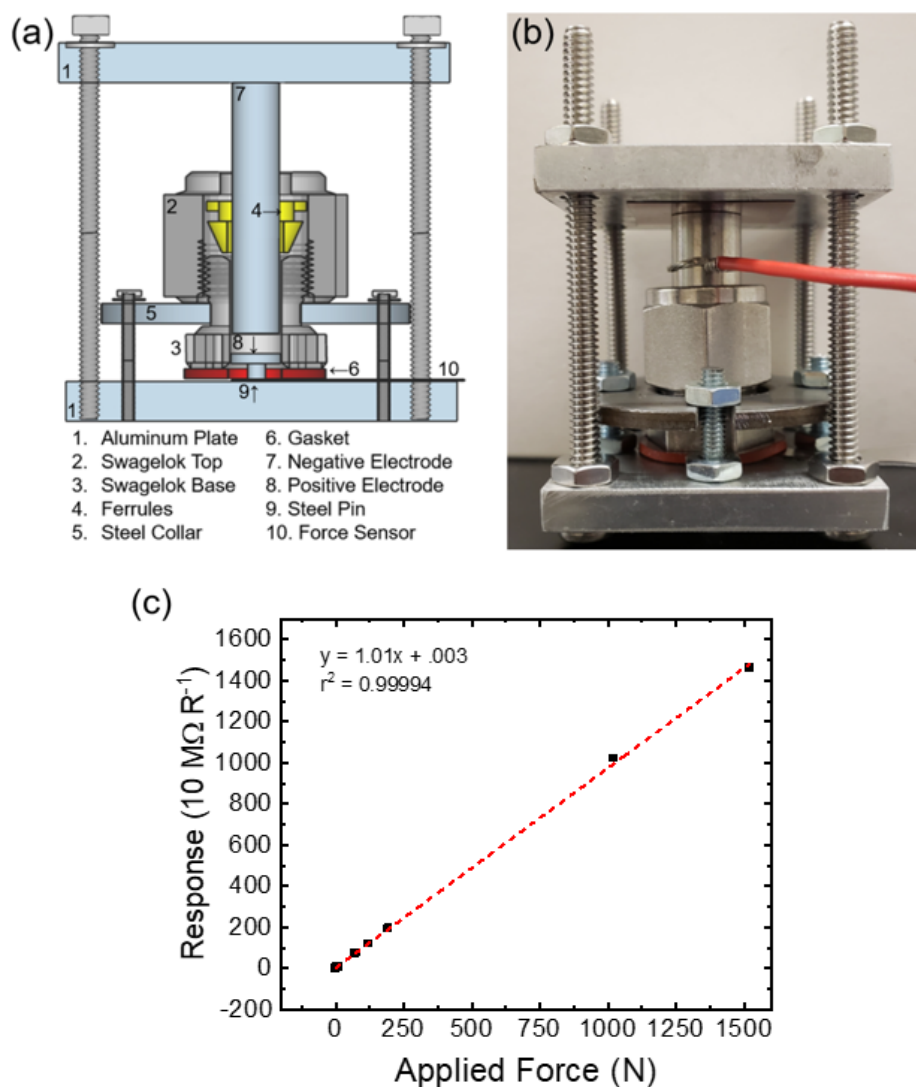


Figure 4.6. a) Cross-sectional schematic of the pressure sensing electrochemical cell used in subsequent measurements. b) Photograph of the fully assembled cell. c) Linearity of the force sensor response to various applied loads.

Pressure dependent electrochemical measurements were conducted in a custom-built pressure sensing electrochemical cell (Figure 4.6) utilizing 1.27 cm diameter A201 force sensors (Tekscan) to monitor the transverse load applied to the electrode stack.³⁷ Cells were assembled in an argon filled glovebox (O_2 and $\text{H}_2\text{O} < 2 \text{ ppm}$), with electrode stacks constructed as follows: first a $\text{Li}_2\text{C}_6\text{O}_6$ cathode was placed at the bottom of the cell, followed by two $25 \mu\text{m}$ polyethylene/polypropylene separators (Celgard). $100 \mu\text{L}$ of electrolyte (1 M LiTFSI in 1:1 DOL/DME) were then added dropwise to wet the separator and an LTO anode was placed onto

the separator to complete the stack. Cells were then sealed, removed from the glovebox, and pressurized until the force sensor registered the desired cell load. Cells were allowed to rest for a minimum of 2 hours to allow the pressure to equilibrate prior to any electrochemical measurements. Table 4.1 presents the mechanical load applied across the electrode stack during pressure dependent measurements along with the pressure developed within the cell as a result.

Table 4.1. Forces and associated pressures applied during electrochemical measurements.

Force (N)	Pressure (MPa)
20	0.28
200	2.82
500	7.05
1000	14.11
1500	21.16

Cells were cycled voltammetrically between -0.5 and 2.0 V vs LTO (1.0 to 3.5 V vs Li/Li⁺) at a scan rate of 200 $\mu\text{V s}^{-1}$ using a CH Instruments electrochemical workstation (CHI, Model 6002E, Austin, TX). Galvanostatic cycling was also conducted with a CHI between -0.5 V and 2.0 V vs LTO at a current rate of 100 mA g⁻¹ (ca. C/5). Electrochemical impedance spectroscopy (EIS) was performed on a BioLogic Science Instruments impedance analyzer (SP-150) between 1 MHz and 10 mHz with an amplitude of 10 mV.

4.2.4 X-Ray Diffraction

Powder X-Ray diffraction (PXRD) was carried out on a Bruker D8 Advance XRD system equipped with a Cu $k\alpha$ source and 2D Eiger2 R 500K detector. Bulk powder samples were analyzed between 10- and 60-degrees 2θ in the typical coupled Bragg-Brentano geometry. Due to the comparatively low active mass loading in electrode casts, both cycled and uncycled electrodes were analyzed using grazing incidence diffraction (again with an analysis window of 10-60 degrees 2θ) at an incident angle of 1 degree. Post-mortem samples were recovered from

the electrochemical cell after three full charge/discharge cycles and rinsed with ca. 5 mL of DME to remove residual LiTFSI. The recovered electrodes were then dried at 80 °C under active vacuum for 12 hours prior to measurement.

4.2.5 Computational Details

Structural optimizations were carried out using the Vienna ab initio simulation package (VASP 5.4.4) under the generalized-gradient-approximation (GGA) using Perdew-Burke-Ernzerhof (PBE) pseudo-potentials as described elsewhere.³⁹ All computations were conducted with a plane-wave basis and projector-augmented wave (PAW) method.⁴⁰ VDW contributions to the total energy were considered by applying a vdw-DF correction to the conventional Kohn-Sham DFT energy as discussed elsewhere.⁴¹⁻⁴³ Structures were optimized with a reduced unit cell consisting of two formula units of $\text{Li}_2\text{C}_6\text{O}_6$. Energy calculations were performed with $7 \times 7 \times 7$ Monkhorst-Pack k-point mesh with an energy cutoff of 520 eV and convergence criterion of 2 meV $\text{\AA}^{-1}$. Elevated pressure computations were considered by the application of an isotropic Pulay correction to the computed stress tensor to obtain optimized structures and energies under non-ambient conditions.

4.3 Results and Discussion

The pressure dependent behavior of as-synthesized $\text{Li}_2\text{C}_6\text{O}_6$ was initially probed with cyclic voltammetry in both coin cell and cone cell geometries, the results of which are presented in Figure 4.7. When cycled in a coin cell (black trace), $\text{Li}_2\text{C}_6\text{O}_6$ exhibits two primary redox events with half-wave potentials at ca. 2.10 and 2.67 V vs Li/Li^+ (peaks 2 and 3 respectively) along with two less intense redox events with half-wave potentials at ca. 1.86 and 3.06 V vs Li/Li^+ (peaks 1 and 4 respectively). Peaks 2 and 3 are attributed to the oxidation/reduction of the

two available ortho-quinones, each of which undergo a concerted two-electron transfer to form the corresponding lithiated enediols (Figure 1a).²⁷ Peaks 1 and 4 are associated with one-electron redox events which result in the formation of the intermediary radical semiquinone (Figure 1b).⁴⁴ The instability of the semiquinone, and subsequently fast radical quenching, results in the low intensity of these features when compared to the concerted pathway.^{44, 45} This behavior aligns well with previous reports of $\text{Li}_2\text{C}_6\text{O}_6$.²⁷

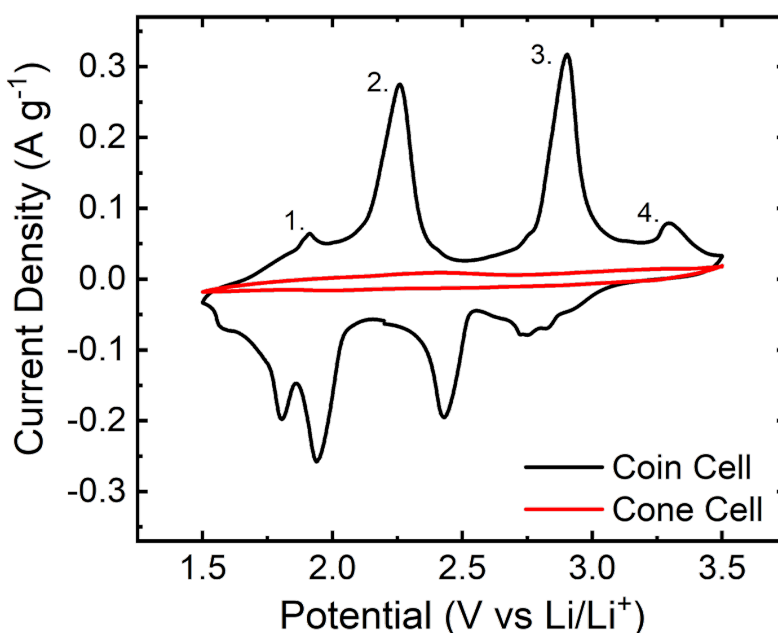


Figure 4.7. Cyclic voltammetry obtained from $\text{Li}_2\text{C}_6\text{O}_6$ electrodes in both a coin cell (black trace) and cone cell (red trace) geometry.

By contrast, the active material yields a much different response when cycled in a cone cell geometry (red trace) with an open solution volume. The change in cell geometry results in a considerable drop in the current density elicited by the active material and a disappearance of the redox events associated with lithiation/delithiation. The lack of coherent electrochemistry suggests that the active material is no longer available to uptake Li^+ either because it has dissolved into the electrolyte, or because it has been deactivated as a result of other processes. To

consider the first of these explanations, the electrolyte was recovered from the cell after voltammetry to determine the extent of active material dissolution. Despite prolonged cycling (up to 5 days), the electrolyte recovered from the cell remained clear and colorless, distinctly different from the vibrant orange color expected from the lithium rhodizonate in solution (Figure 4.8). This suggests that dissolution cannot adequately explain the observed lack of electrochemical activity. We likewise considered solvation of $\text{Li}_2\text{C}_6\text{O}_6$ by the electrolyte as a possible decomposition mechanism, however after prolonged contact with the electrolyte no change could be discerned in either the crystal structure or the vibrational spectrum of the active material (Figure 4.9). As the cone cell geometry does not exert any force on the redox active material, it is possible that the observed deactivation is related to a strong pressure dependence of the $\text{Li}_2\text{C}_6\text{O}_6$ active material.

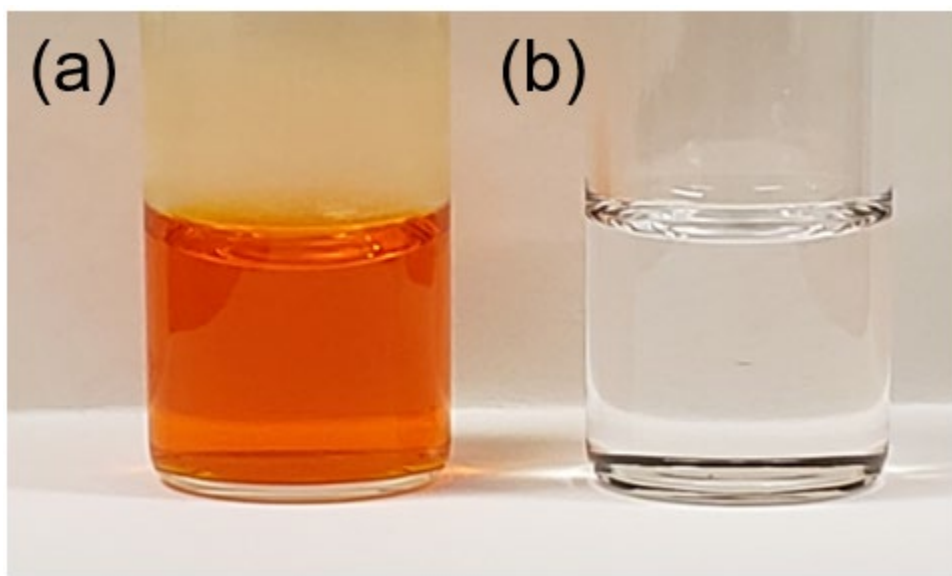


Figure 4.8. a) 1mM aqueous solution of $\text{Li}_2\text{C}_6\text{O}_6$. b) Electrolyte recovered after voltammetric cycling (20 cycles, ca. 5 days) in a cone cell.

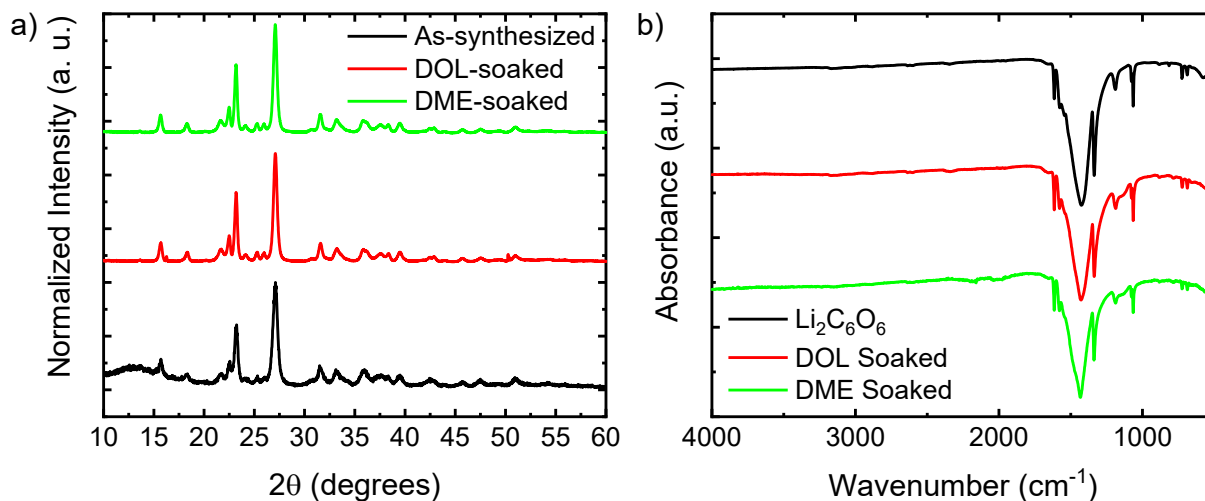


Figure 4.9. a) powder x-ray diffraction comparing the as-synthesized $\text{Li}_2\text{C}_6\text{O}_6$ powder (black trace) to diffraction obtained after soaking the active material in dioxolane (DOL, red trace) or dimethoxyethane (DME, green trace). b) Infrared spectra collected from the same samples analyzed in (a).

To probe the possible pressure dependence of $\text{Li}_2\text{C}_6\text{O}_6$, electrodes were cycled under different isostatic loads in a custom-built pressure sensing electrochemical cell. The cell consisted of a modified Swagelok housing fitted with an A201 force sensor (Tekscan) which allowed the determination of electrode stack pressure over a wide dynamic range (Figure 4.6c).³⁷ To mitigate the possibility of internal short circuits at high pressure, and to deconvolute the effect of anode volume change contributing to the overall cell pressure, $\text{Li}_7\text{Ti}_5\text{O}_{12}$ (LTO) was used as the anode material in place of metallic lithium (which deforms easily at high pressures) or graphite (which exhibits a large volume change of ca. 13% with cycling).⁴⁶

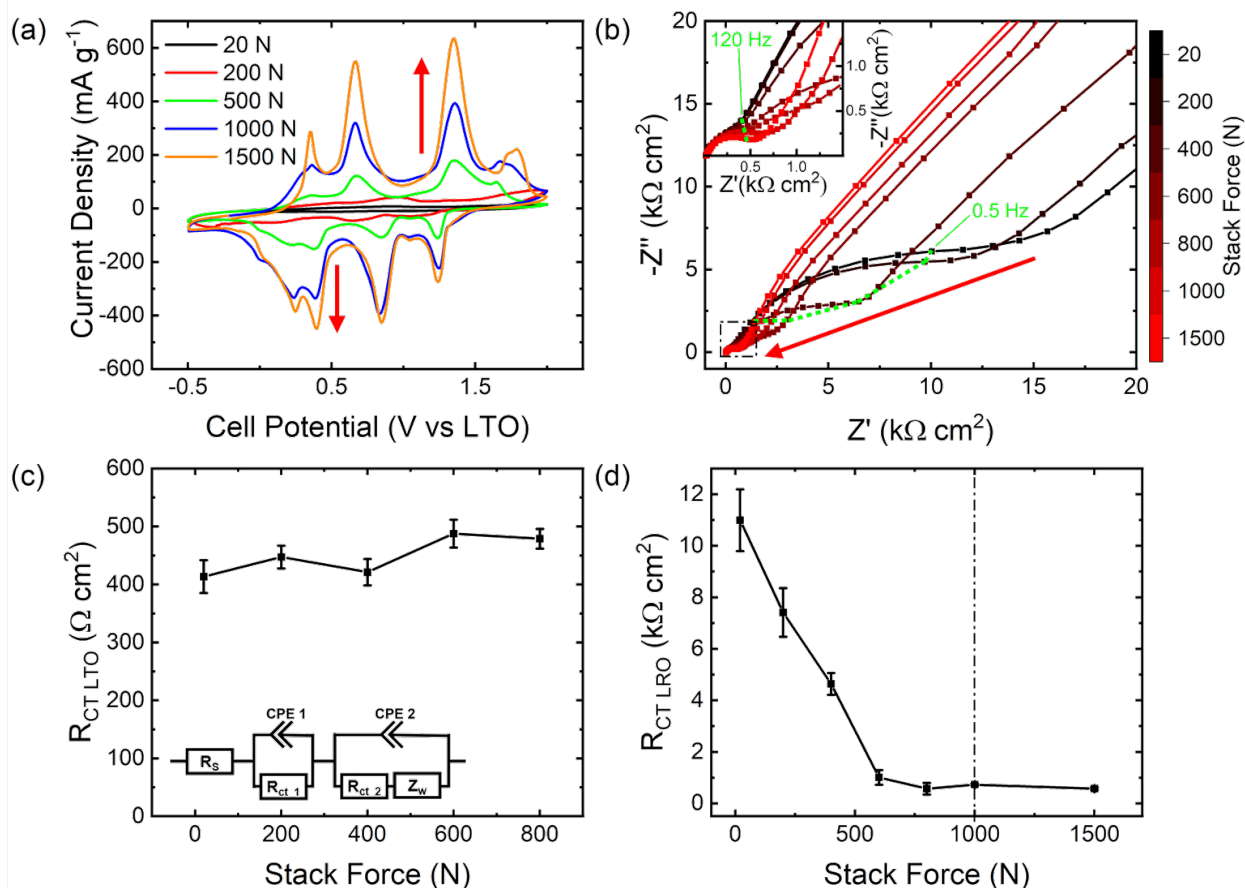


Figure 4.10. a) Force dependent cyclic voltammetry of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes. Arrows highlight the increase in peak current density as stack force is increased. b) Force dependent electrochemical impedance spectra collected from $\text{Li}_2\text{C}_6\text{O}_6$ electrodes. Inset depicts the high frequency region of the spectra wherein a second semicircle is observed associated with the LTO anode. c) Charge transfer impedance for the high frequency semicircle as a function of stack pressure, as well as the equivalent circuit used to fit experimental data. Equivalent circuit elements are as follows: solution resistance (R_s), constant phase elements describing the anode and cathode double layer capacitances (CPE 1 and CPE 2 respectively), charge transfer impedances at the anode and cathode interfaces ($R_{\text{CT 1}}$ and $R_{\text{CT 2}}$ respectively), and a Warburg diffusion element to consider bulk ion diffusion (Z_w). d) Charge transfer impedance associated with the low frequency semicircle as a function of applied pressure. The dotted line marks the point at which the low frequency semicircle can no longer be confidently differentiated from the high frequency semicircle

The pressure dependent electrochemical performance of $\text{Li}_2\text{C}_6\text{O}_6$ was initially probed using cyclic voltammetry, the results of which are presented in Figure 4.10a. Voltammetry was conducted between cell potentials of -0.5 and 2.0 V vs LTO (1 V and 3.5 V vs Li/Li^+) at a scan rate of $200 \mu\text{V s}^{-1}$, using fresh electrodes for each pressure under evaluation. As the applied load

across the electrode stack is increased, the peak current density exhibited by the cell likewise increases, reflecting improved active material utilization during cycling. The peak splitting between anodic and cathodic waves also responds to the magnitude of the applied load. At low force loadings ($F < 200$ N) the difference between anodic and cathodic peak potentials is nearly 1 V, reflecting a very high overpotential towards lithiation and delithiation. This peak splitting drops rapidly to a final value of ca. 140 mV as the stack pressure is increased signifying a dramatic decrease in the electrochemical overpotential.

To further characterize the pressure dependent electrochemical response, the charge transfer dynamics of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes were interrogated with electrochemical impedance spectroscopy at different stack pressures. Figure 4.10b presents impedance spectra obtained at progressively increasing cell pressures, collected at the open circuit potential (ca. 2.8 V vs Li/Li^+). Two semicircular features are evident in the EIS spectra, the first of which appears at lower frequencies and varies with the applied cell load. The second feature appears at higher frequencies (figure 3b, inset) and remains nearly constant with applied load. The low frequency feature is attributed to dynamics at the $\text{Li}_2\text{C}_6\text{O}_6$ /electrolyte interface while the high frequency semicircle is associated with the $\text{Li}_7\text{Ti}_5\text{O}_{12}$ /electrolyte interface. This assignment was verified by conducting the same series of impedance measurements in symmetric $\text{Li}_2\text{C}_6\text{O}_6/\text{Li}_2\text{C}_6\text{O}_6$ and $\text{Li}_7\text{Ti}_5\text{O}_{12}/\text{Li}_7\text{Ti}_5\text{O}_{12}$ cells (figure 4.11). Here, the impedance associated with the $\text{Li}_7\text{Ti}_5\text{O}_{12}$ symmetric cell was nearly pressure-independent while that associated with the $\text{Li}_2\text{C}_6\text{O}_6$ symmetric cell was not.

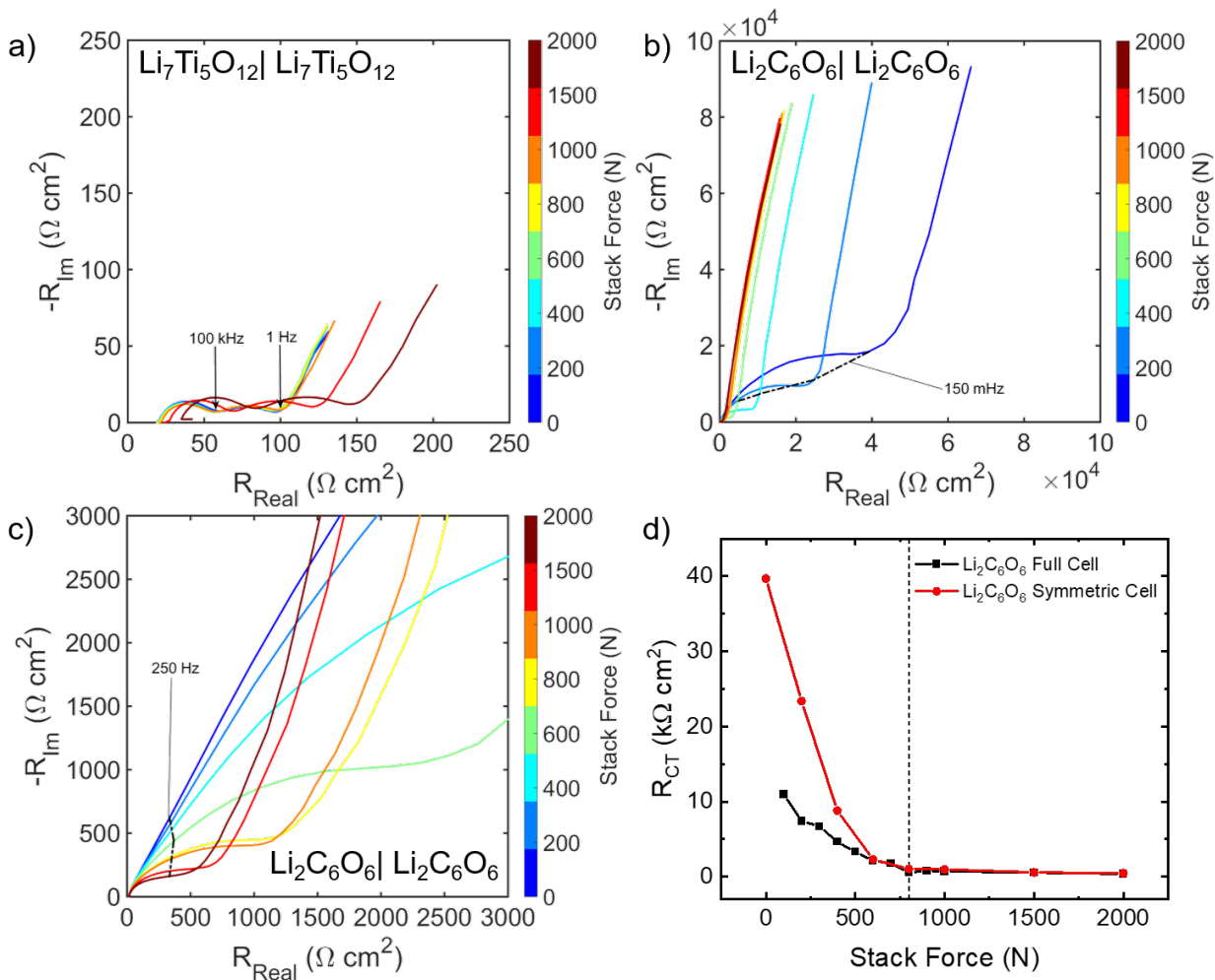


Figure 4.11. Pressure dependent electrochemical impedance spectra obtained from symmetric a) $\text{Li}_7\text{Ti}_5\text{O}_{12}$ and b-c) $\text{Li}_2\text{C}_6\text{O}_6$ cells as well as a comparison between the behavior of the symmetric and full $\text{Li}_2\text{C}_6\text{O}_6$ cells.

Spectra were fit to extract the charge transfer impedance associated with both the anode and cathode interfaces (full fitted spectra are presented Figure 4.12). Figure 3c presents the extracted charge transfer impedance values associated with the anode interface (R_{CT1}), as well as the equivalent circuit used to fit experimental spectra. Figure 4.10d presents the charge transfer impedance values associated with the cathode interface. Figures 4.10c and 4.10d reflect the qualitative pressure dependencies displayed in Figure 4.10b. Specifically, the charge transfer impedance at the $\text{Li}_2\text{C}_6\text{O}_6/\text{electrolyte}$ interface drops considerably as the applied load is increased, while charge transfer at the anode interface remains nearly constant over the pressure

range of interest. The electronic resistance of the electrode stack R_s is likewise independent of the applied external load (Figure 4.13). The impedance results align well with the results obtained from pressure dependent voltammetry and suggest that the origin of the poor electrochemical performance of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes at low pressure is untenably high charge transfer resistance, which is ameliorated by the application of adequate stack pressure.

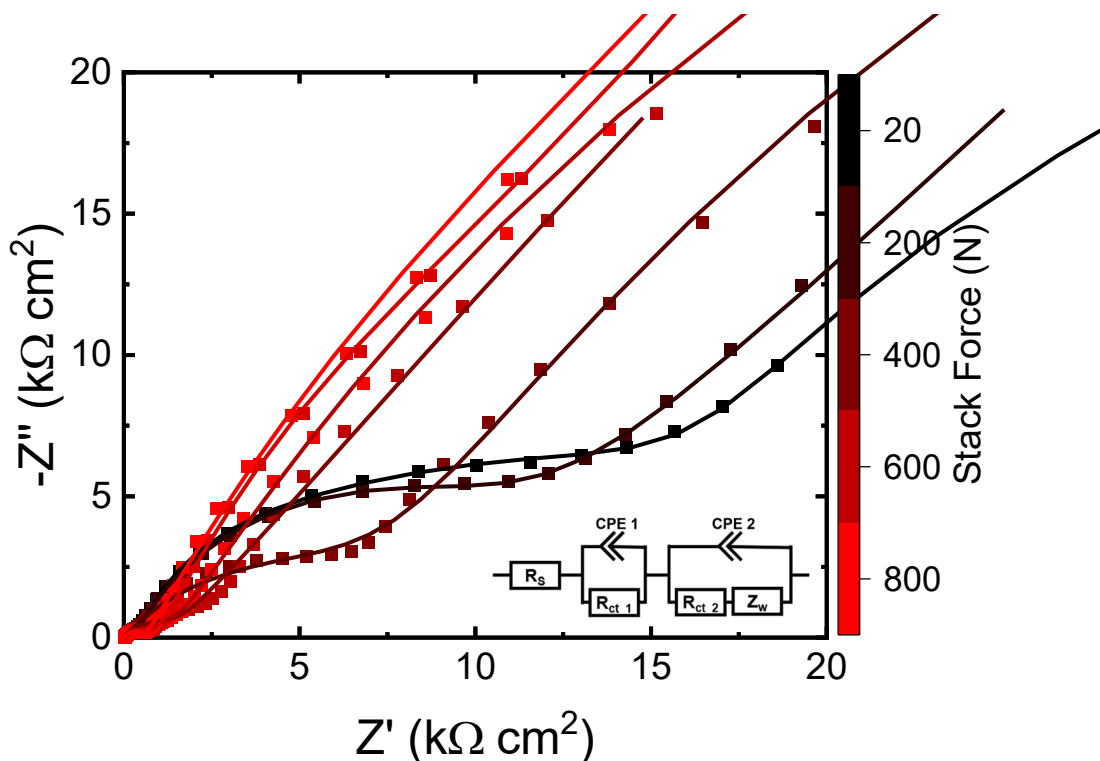


Figure 4.12. Exemplary fits to pressure dependent impedance data. Inset depicts the equivalent circuit used to fit the data.

The results obtained from voltammetry and impedance suggest that the application of adequate pressure to the $\text{Li}_2\text{C}_6\text{O}_6$ electrode effectively reduces the charge transfer impedance at the cathode/electrolyte interface making continued lithiation and delithiation possible. To understand the implications this has on the long-term performance of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes, LTO/ $\text{Li}_2\text{C}_6\text{O}_6$ full cells were cycled galvanostatically between -0.5 and 2.0 V vs LTO (1.0 and 3.5 V vs Li/Li^+) at a constant current of 100 mA g^{-1} (ca. $\text{C}/5$). Figure 4.14a presents the capacity

retention and Coulombic efficiency of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes cycled under different applied loads. At low force loadings ($F < 200$ N) very little electrochemical activity is obtained from the electrode stack, likely due to the high overpotential necessary to lithiate the active material. As the force loading is increased ($200 < F < 1000$ N) the active material displays close to theoretical capacity in the first cycle. The Coulombic efficiency of the active material is low, however, and the available capacity fades rapidly as a result. Finally, at even higher applied loads ($F \geq 1000$ N) the Coulombic efficiency of the cell approaches 100% and the overall capacity retention continues to increase, reflecting an improvement to the electrochemical reversibility of the $\text{Li}_2\text{C}_6\text{O}_6$ electrode.

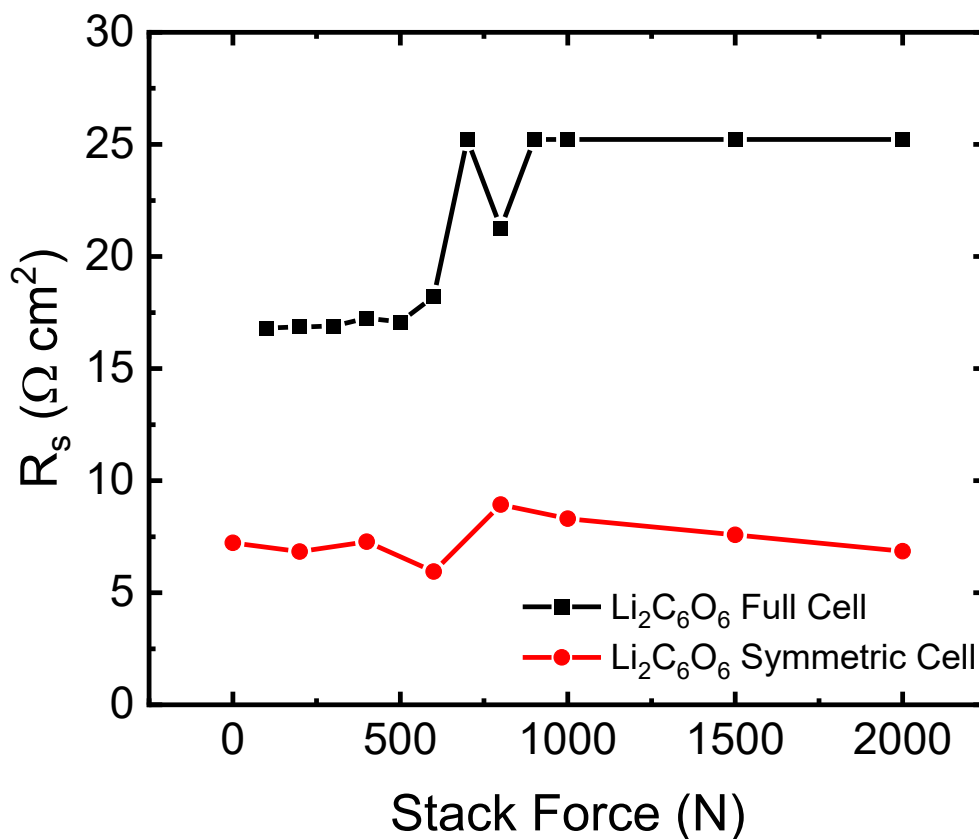


Figure 4.13. Pressure dependent R_s values obtained from fits presented in Figure 4.12.

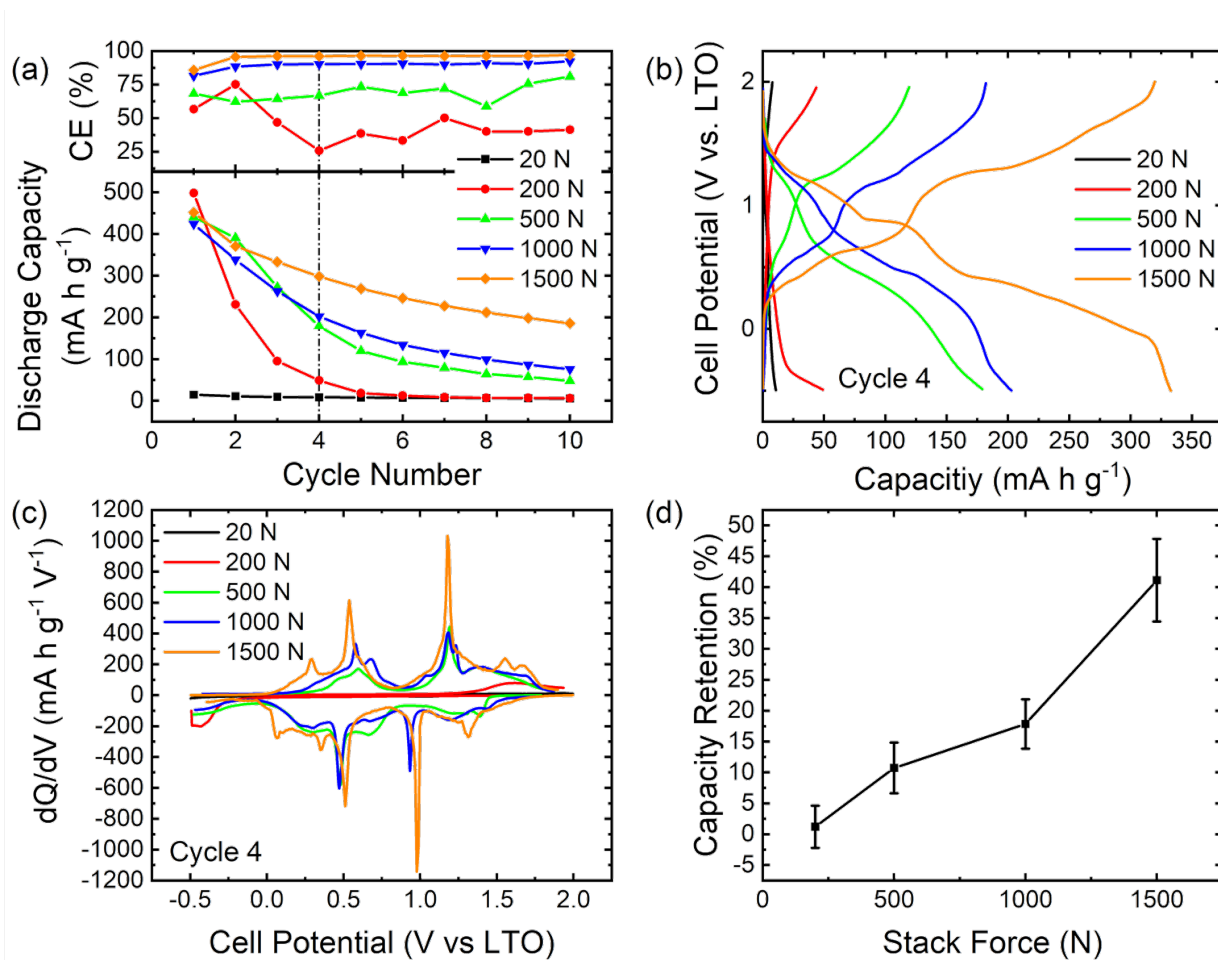


Figure 4.14. a) Discharge capacities and Coulombic efficiencies obtained by galvanostatically cycling $\text{Li}_2\text{C}_6\text{O}_6$ electrodes under different applied loads. b) Galvanostatic charge/discharge traces taken from the fourth galvanostatic cycle (demarcated by the dotted line in a). c) Differential capacity traces computed from the charging profiles in b, demonstrating a progressive increase in differential charge transfer as the stack pressure is increased. d) Overall capacity retention of electrodes cycled at different stack pressures (The 20 N sample is omitted as it does not exhibit any meaningful capacity).

Figure 4.14b presents individual charge/discharge traces obtained from the fourth galvanostatic cycle at each cell load (The first five charge/discharge cycles at each pressure are presented in Figure 4.15). At sufficiently high force loadings ($F > 200 \text{ N}$), redox plateaus are evident at potentials matching those obtained voltammetrically; however, at lower cell pressures, substantial polarization is found as evidenced by an increased separation between the corresponding anodic and cathodic plateaus. This effect is likewise demonstrated from the

differential capacity traces computed for each cycle, as displayed in Figure 4.14c. The large polarization observed at low pressures decreases upon the application of adequate stack pressure, consistent with the lowered interfacial charge transfer resistance obtained from EIS. Additionally, as the applied stack pressure increases, the peak intensity of the differential capacity traces likewise increases, which is often attributed to increased crystallinity in the active material during lithiation and delithiation.⁴⁷ Finally, Figure 4.14d quantifies the overall capacity retention as a function of cell pressure, reflecting the qualitative results displayed in Figure 4.14a. This result suggests that not only does an increase in cell pressure facilitate efficient charge transfer at the interface, it also improves the overall capacity retention and longevity of the active material.

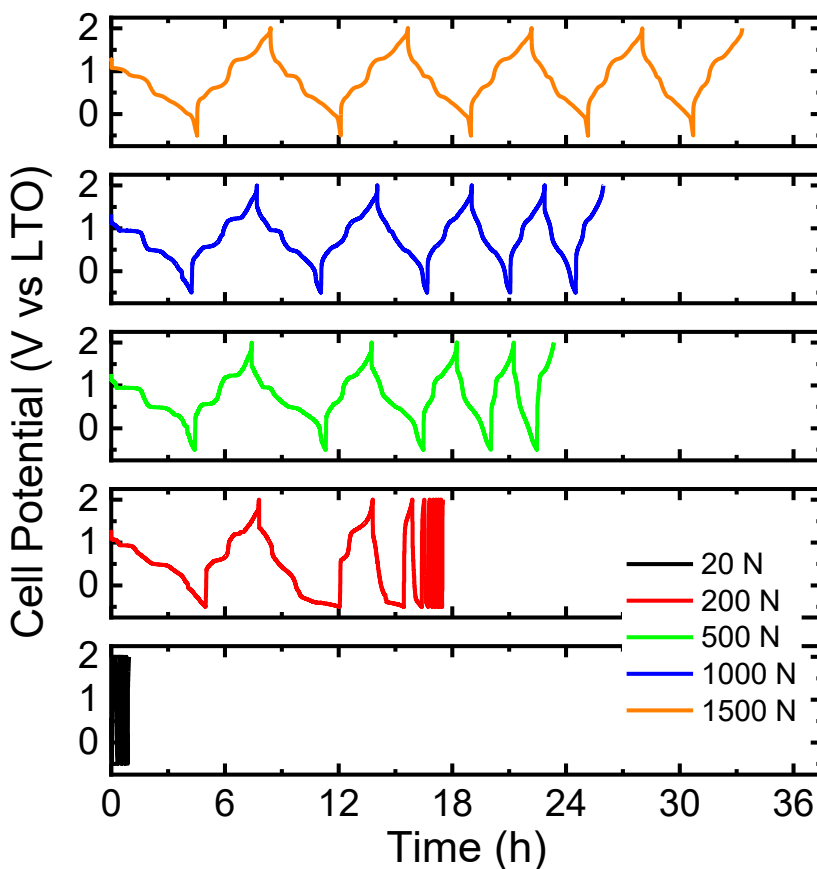


Figure 4.15. First five galvanostatic cycles for $\text{Li}_2\text{C}_6\text{O}_6$ electrodes at different pressures.

The origin of the improved cycling stability with applied pressure presented in Figure 4.14 was interrogated using post-mortem grazing incidence X-ray diffraction (GIXD) of cycled $\text{Li}_2\text{C}_6\text{O}_6$ electrodes. Figure 4.16a presents GIXD patterns obtained from a pristine (black trace) and cycled (red trace) electrode. The pristine electrode exhibits two main reflections at ca. 23.3° and 27.3° 2θ (peaks 1 and 2 respectively), while the cycled electrode displays an additional feature at ca. 32.0° (peak 3). Peaks beyond 35 degrees were attributed to the Al substrate, which displayed variable intensity depending on the quality of the recovered electrode (high pressure samples often suffered from active material delamination during cell disassembly). Previous reports on the structure of di-lithium rhodizonate suggest that the material adopts a layered structure comprised of pseudo-planar layers of hexagonally close-packed rhodizonate ions (Figure 4.1c).^{25, 27, 48} The highest intensity reflections in both the as-synthesized and cycled materials (peaks 2 and 3 respectively) correspond to the Van der Waals (VDW) spacing between these rhodizonate layers. The d-spacing calculated from the experimental diffraction patterns reflect a VDW gap of $3.36 \text{ \AA}$ for the as-synthesized material and $2.90 \text{ \AA}$ for the cycled material, suggesting an electrochemically induced contraction along the c-axis of the crystal. The disappearance of peaks 1 and 2, associated with the as-synthesized material, and the appearance of peak 3 resulting from galvanostatic cycling reflect the irreversible conversion of the redox active phase of $\text{Li}_2\text{C}_6\text{O}_6$ to the redox inactive phase. This phase transition is very similar to the previously reported electrochemical phase transitions observed in di-sodium rhodizonate electrodes during repeated sodiation-desodiation.⁴⁹

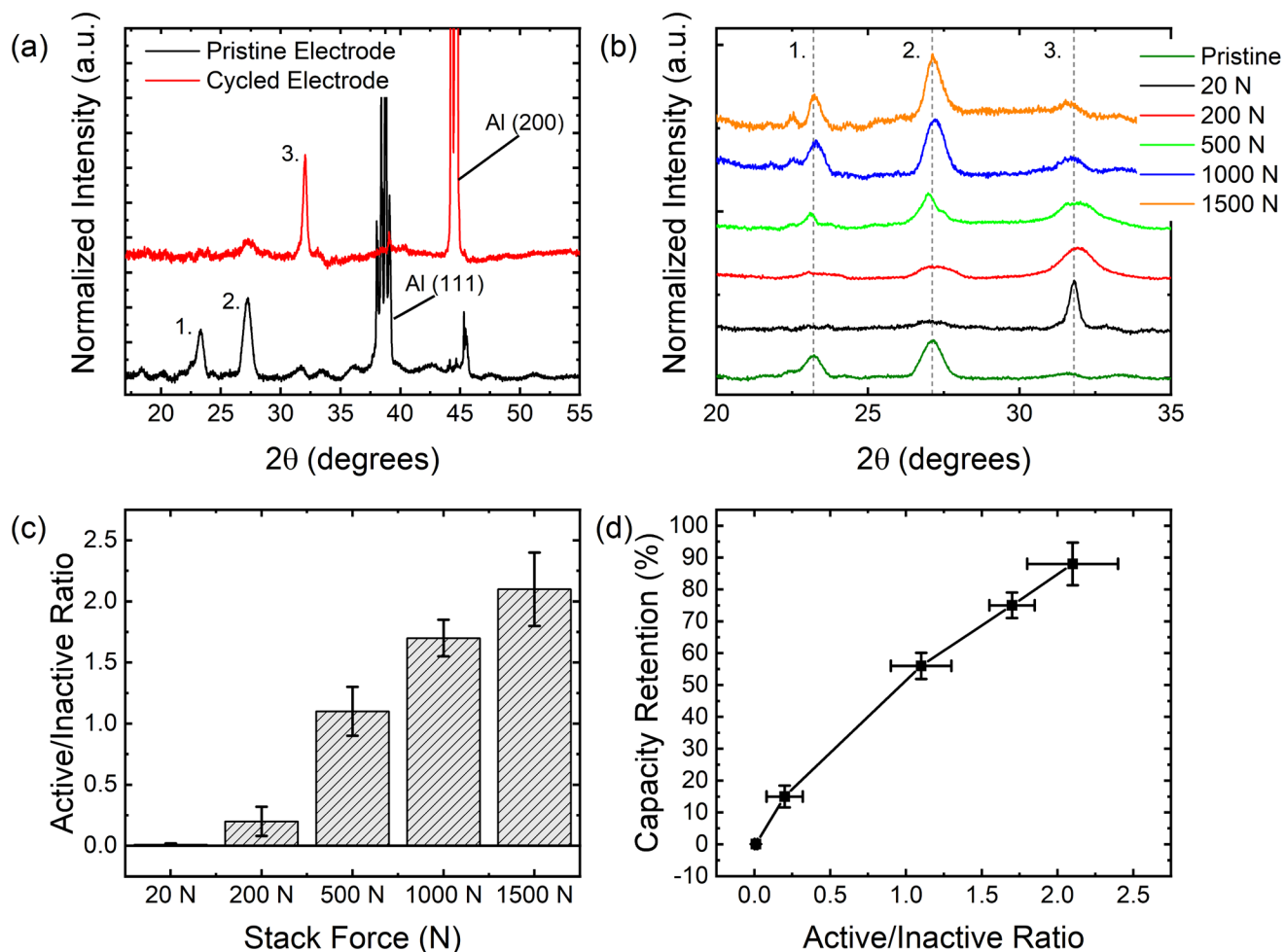


Figure 4.16. a) Grazing incidence X-ray diffractograms collected from a pristine (black) and a cycled (red) $\text{Li}_2\text{C}_6\text{O}_6$ electrode. b) Post-mortem grazing incidence XRD of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes cycled three times under different applied stack force. The uncycled electrode pattern is included in the dark green trace for reference. c) Ratio of redox active to redox inactive material computed from the integrated peak intensities of peaks 1-3 in b. d) Total capacity retention after three cycles plotted as a function of the ratio between the redox active and redox inactive material.

Figure 4.16b presents grazing incidence diffraction patterns obtained from electrodes after three galvanostatic charge/discharge cycles at different applied pressures. The ratio of redox-active to redox-inactive material was calculated by comparing the integrated peak intensities of peaks 1 and 2 to that of peak 3. The results of this analysis are presented in Figure 4.16c and demonstrate higher active material ratios at elevated pressures, aligning well with the results obtained from galvanostatic cycling. This correlation is quantified in Figure 4.16d where

the overall capacity retention is plotted against the active/inactive ratio, demonstrating the expected positive trend. This result suggests that cycling at elevated pressures suppresses the formation of the redox-inactive species, the presence of which is associated with rapid capacity fade. While this assignment aligns well with the pressure dependent behavior obtained from galvanostatic cycling, previous work has implicated active material exfoliation as a contributor to the overall capacity fade.²⁵ To consider this possibility, post-mortem scanning electron microscopy (SEM) was conducted on electrodes cycled three times at variable pressure (Figure 4.17). While the material displayed some pressure dependent morphological changes, obvious exfoliation was not observed, even for the low-pressure samples, which exhibit the most rapid capacity fade.

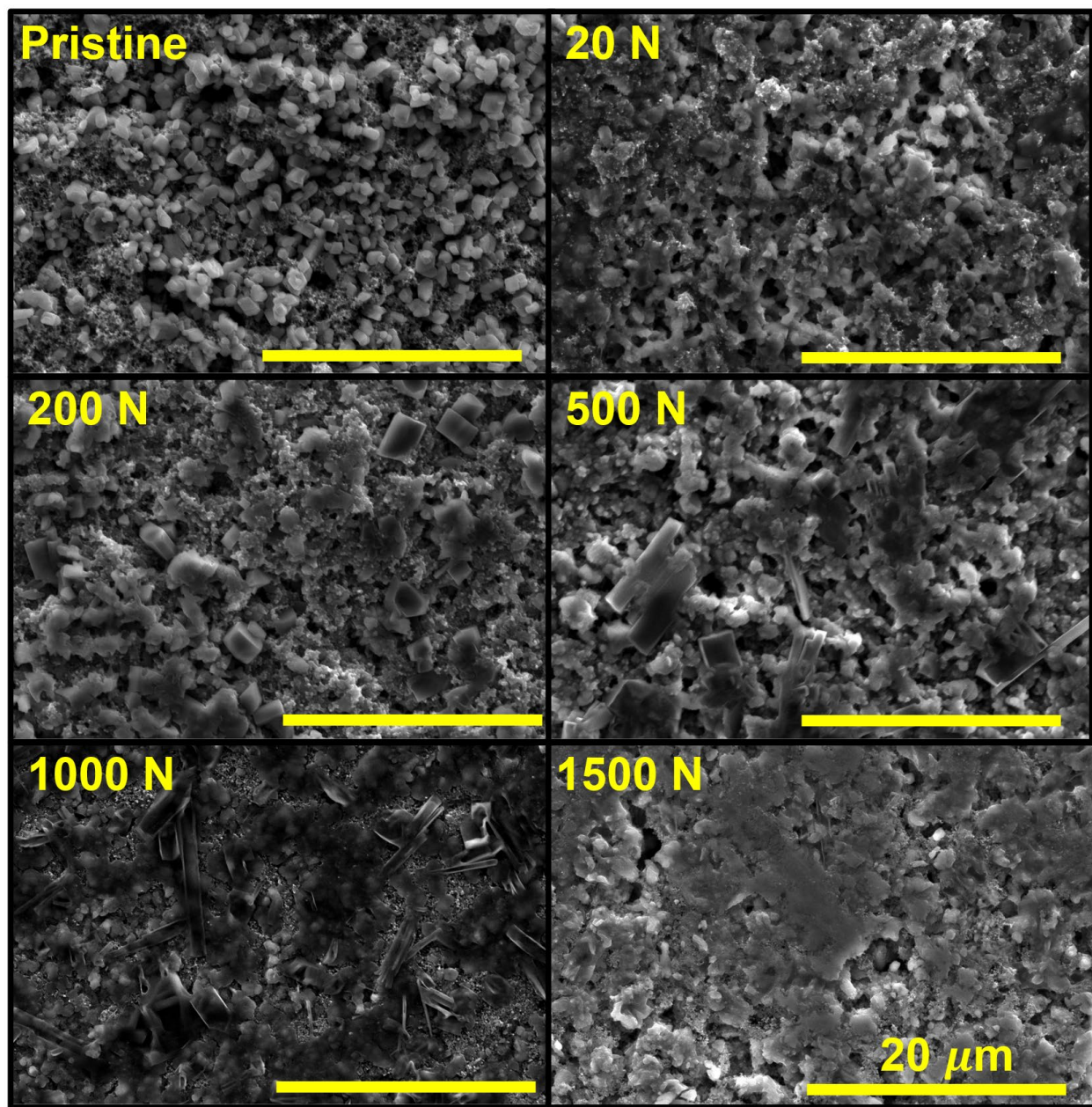


Figure 4.17. Post-mortem SEM micrographs of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes after three galvanostatic charge/discharge cycles. All scale bars are $20\ \mu\text{m}$.

To understand the origin of the improved active material stability at elevated pressures, DFT calculations were used to compare the energies of the redox-active and redox-inactive phases under different applied loads. The previously reported ground state structure of $\text{Li}_2\text{C}_6\text{O}_6$ was used as the redox-active phase and consisted of hexagonally close packed rhodizonate layers

stacked in an AABB pattern, with rhodizonate layers aligning directly above each other (Figure 4.18a). The structure of the redox-inactive phase likewise consisted of hexagonally close-packed rhodizonate layers; however, adjacent layers were offset in an ABAB stacking pattern in the stacking order exhibited by $\text{Na}_2\text{C}_6\text{O}_6$ (Figure 4.18b).⁴⁹ As the Li coordination could not be determined directly from experimental data,⁵⁰ structures were considered with Li^+ occupying sites either within the rhodizonate layer or within the VDW gap (figure 4.19). The correct Li^+ coordination was assigned to the configuration which most closely resembled the experimentally obtained diffraction patterns. The preferred Li^+ geometry for the redox-active phase was pseudo-square planar with Li-ions occupying sites within the rhodizonate layer. In contrast, the preferred coordination for the redox-inactive phase placed Li^+ within the VDW gap in a pseudo-tetrahedral geometry. This change in Li^+ coordination is accompanied by a decrease in the VDW gap from 3.3 Å to 2.9 Å reflecting the results observed experimentally.

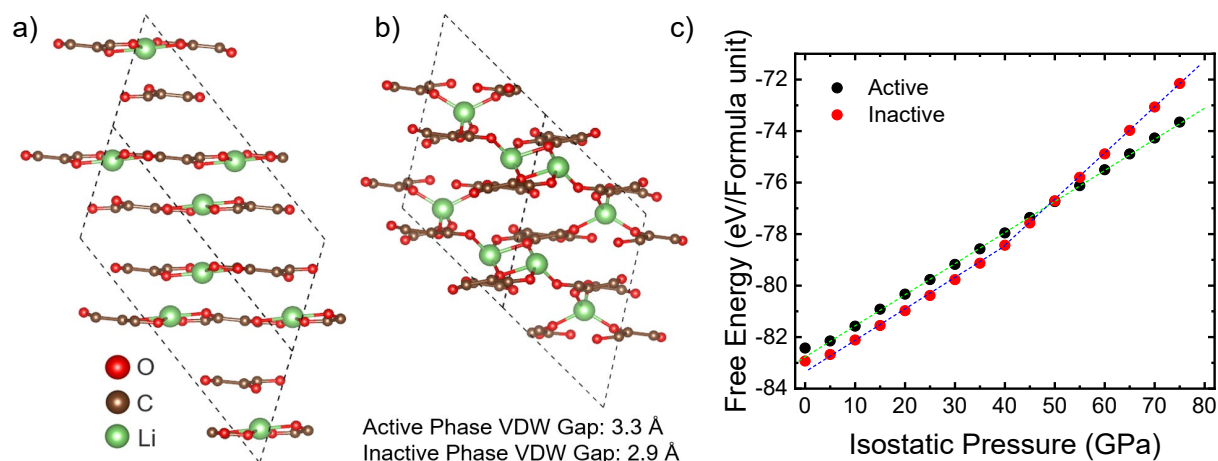


Figure 4.18. a) and b) depict the computed ground state structures for the active and inactive phases of lithium rhodizonate, respectively. c) Computed free energies for both the active and inactive phases as a function of an applied isostatic load.

The obtained structures for the redox-active and redox-inactive materials were then used to evaluate the pressure dependence of each conformation by computing the optimized geometry

and free energy at progressively increasing isostatic pressures. Figure 4.18c displays the free energies determined for both structures as a function of external pressure. At low pressures, the redox-inactive phase exhibits lower lattice free-energies, suggesting that the redox-active phase is only metastable at ambient conditions. This supports the facile interconversion from the active to inactive phase during electrochemical cycling. Similar electrochemically driven polymorphisms have been reported elsewhere for other molecular crystals as well as for disodium rhodizonate.^{49, 51} As the pressure is increased, the free energy of the inactive phase increases more rapidly than that of the active phase, likely due to greater electrostatic repulsion between the rhodizonate and Li^+ layers. At sufficiently high pressures an inversion occurs wherein the redox-active phase becomes the energetically preferred conformer. While this result aligns qualitatively with the results obtained from pressure dependent galvanostatic cycling, the pressure at which this inversion is expected to occur is ca. 50 GPa, far exceeding the pressures achievable during cycling experiments. This discrepancy between calculation and experiment suggests that pressure-dependent kinetic factors are of considerable importance in suppressing the crystallographic transition from the redox active to the redox inactive phase. As this phase transition involves a substantial change in the stacking order of the rhodizonate lattice, it likely occurs via the migration of planes of rhodizonate ions. The motion of these planes is expected to depend on the magnitude of the pressure applied. In particular, previous investigations of other VDW materials have demonstrated a considerable increase in the strength of VDW interactions at elevated pressures.⁵² This increase in inter-layer interactions likely restricts the free motion of rhodizonate layers, thereby hindering the conversion from the active to the inactive structure.

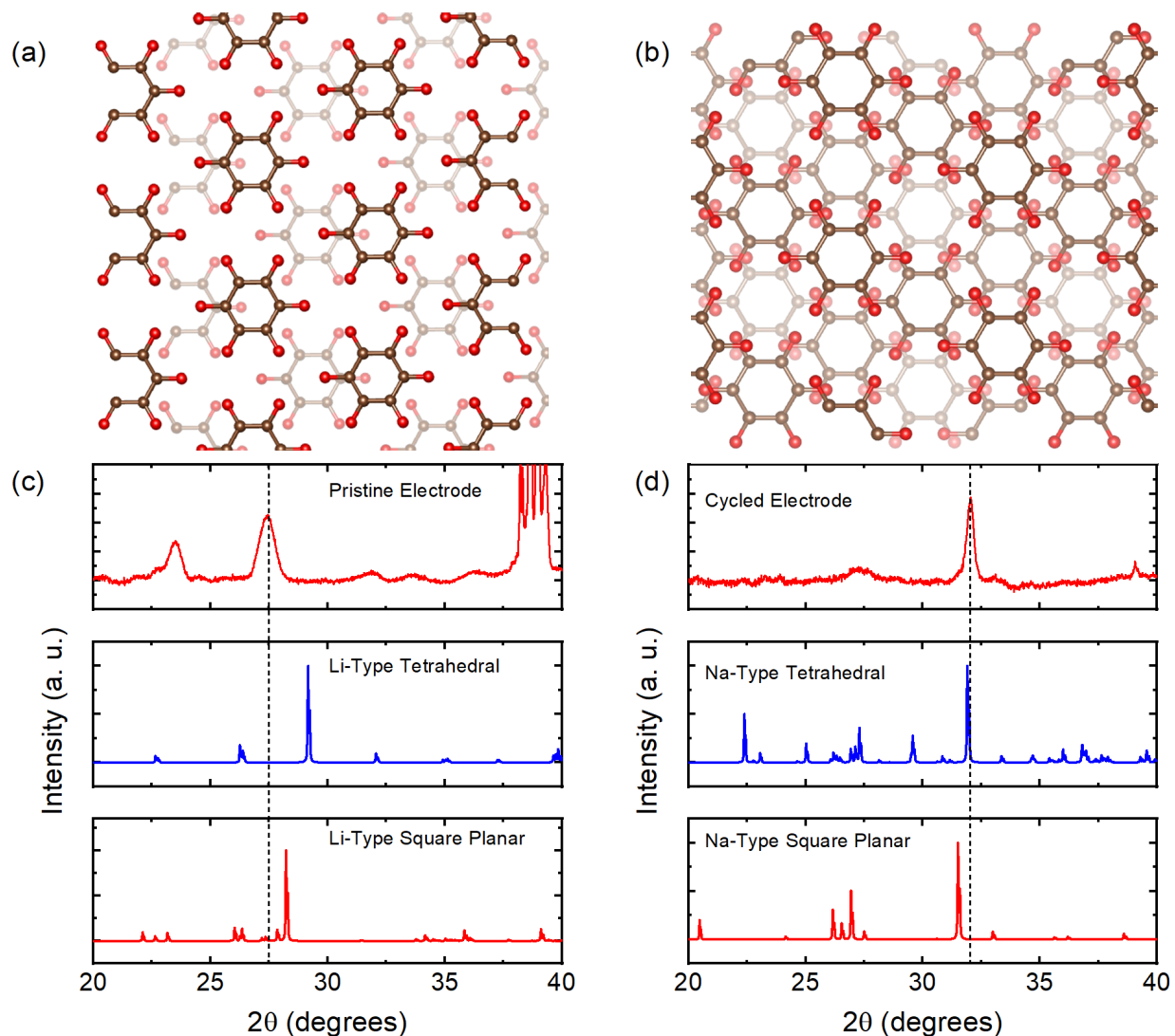


Figure 4.19. a) AABB stacking order of the rhodizonate lattice reported for as synthesized dilithium rhodizonate. b) ABAB stacking order of the rhodizonate lattice associated with disodium rhodizonate.⁴⁹ c) Experimental (top row) and simulated diffraction patterns for the Li-Type rhodizonate lattice with Li^+ occupying either a tetrahedral (within the VDW gap) or square planar (within the rhodizonate layer) site. d) Experimental (top row) and simulated diffraction patterns for the Na-Type rhodizonate lattice with Li^+ occupying either a tetrahedral (within the VDW gap) or square planar (within the rhodizonate layer) site.

In addition to the long-term cycling characteristics the high lithiation/delithiation overpotentials associated with the redox inactive structure can be understood by considering the structural characteristics of the material. First the contraction of the VDW gap induced by a change in the lithium coordination likely reduces the diffusion coefficient of Li^+ along the

interlayer space. Additionally, the high binding energy of Li^+ in the tetrahedral geometry (6.9 eV compared with 6.7 eV for Li^+ in the square planar geometry) reduces the mobility of these ions, and their presence within the VDW gap likely hampers the 2-dimensional diffusion of additional Li^+ through the structure,⁵³ resulting in the high overpotentials observed in voltammetry and impedance. These effects are similar to those observed in layered metal oxides, specifically nickel rich metal oxides upon transition metal migration to the Li^+ layer.⁵⁴

4.4 Conclusions

This work addresses the electrochemical responses of di-lithium rhodizonate electrodes under different applied pressures. Using pressure dependent cyclic voltammetry and electrochemical impedance spectroscopy we show that the charge transfer kinetics of Li^+ at the cathode/electrolyte interface are strongly impacted by the mechanical load applied to the electrode stack. At low applied stack pressures $\text{Li}_2\text{C}_6\text{O}_6$ electrodes exhibit untenably high charge transfer impedances and extremely low current densities when cycled voltammetrically. By contrast, electrodes held under large external loads show a considerable drop in charge transfer impedance, and a concomitant increase in voltammetric peak current densities. This suggests that the application of a mechanical load to the $\text{Li}_2\text{C}_6\text{O}_6$ electrode significantly improves the kinetics of lithium intercalation and deintercalation at the cathode/electrolyte interface.

In addition to the improved charge transfer kinetics, the cycling stability of $\text{Li}_2\text{C}_6\text{O}_6$ also improves as the stack pressure is increased. Cells cycled at low pressures exhibit rapid capacity fade and low Coulombic efficiencies, whereas cells cycled at high pressure show considerable improvements to both the Coulombic efficiency and overall capacity retention. Post-mortem diffraction demonstrates that the rapid capacity fade in low pressure cells is due to an irreversible crystallographic phase transition consisting of changes to both the rhodizonate stacking order and

the coordination of Li^+ within the rhodizonate lattice. Importantly, as the stack pressure is increased the conversion from the redox active to redox inactive species is suppressed, which agrees well with the observed increase in capacity retention. The suppression of this deleterious phase transition is thought to be the result of increasing kinetic stability of the redox active phase at elevated pressures.

Overall, this work demonstrates the importance of chemo-mechanical properties on the electrochemical performance of $\text{Li}_2\text{C}_6\text{O}_6$ electrodes. We show that crystallographic transitions induced by electrochemical stresses suppress the redox activity of $\text{Li}_2\text{C}_6\text{O}_6$. These deleterious phase transitions can be directly controlled by the application of external pressure, resulting in improvements to both charge transfer kinetics and capacity retention. This affirms the importance of considering chemo-mechanical effects when investigating organic electrode materials. Further, we suggest that the origin of the poor electrochemical activity of the redox-inactive phase is due to a change in the Li^+ coordination and subsequent drop in Li^+ mobility within the rhodizonate lattice. Thus, the development of oxo-carbon structures which suppress tetrahedral Li^+ coordination and maintain fast 2-dimensional solid-state diffusion is expected to result in improvements in both electrochemical kinetics and long-term cycling stability.

4.5 References

- (1) Goodenough, J. B.; Park, K. S., The Li-Ion Rechargeable Battery: A Perspective. *J Am Chem Soc* **2013**, *135* (4), 1167-1176.
- (2) Manthiram, A., An Outlook on Lithium Ion Battery Technology. *Acs Central Sci* **2017**, *3* (10), 1063-1069.
- (3) Chen, T.; Jin, Y.; Lv, H.; Yang, A.; Liu, M.; Chen, B.; Xie, Y.; Chen, Q., Applications of Lithium-Ion Batteries in Grid-Scale Energy Storage Systems. *Transactions of Tianjin University* **2020**, *26* (3), 208-217.
- (4) Nitta, N.; Wu, F.; Lee, J. T.; Yushin, G., Li-ion battery materials: present and future. *Materials Today* **2015**, *18* (5), 252-264.
- (5) Schmuch, R.; Wagner, R.; Hörpel, G.; Placke, T.; Winter, M., Performance and cost of materials for lithium-based rechargeable automotive batteries. *Nature Energy* **2018**, *3* (4), 267-278.
- (6) Schon, T. B.; McAllister, B. T.; Li, P.-F.; Seferos, D. S., The rise of organic electrode materials for energy storage. *Chem Soc Rev* **2016**, *45* (22), 6345-6404.
- (7) Lu, Y.; Chen, J., Prospects of organic electrode materials for practical lithium batteries. *Nature Reviews Chemistry* **2020**, *4* (3), 127-142.
- (8) Yin, X.; Sarkar, S.; Shi, S.; Huang, Q.-A.; Zhao, H.; Yan, L.; Zhao, Y.; Zhang, J., Recent Progress in Advanced Organic Electrode Materials for Sodium-Ion Batteries: Synthesis, Mechanisms, Challenges and Perspectives. *Advanced Functional Materials* **2020**, *30* (11), 1908445.
- (9) Liang, Y.; Yao, Y., Positioning Organic Electrode Materials in the Battery Landscape. *Joule* **2018**, *2* (9), 1690-1706.

- (10) Liang, Y.; Tao, Z.; Chen, J., Organic Electrode Materials for Rechargeable Lithium Batteries. *Adv Energy Mater* **2012**, 2 (7), 742-769.
- (11) Wu, X.; Ma, J.; Hu, Y.-S.; Li, H.; Chen, L., Nano-sized carboxylates as anode materials for rechargeable lithium-ion batteries. *Journal of Energy Chemistry* **2014**, 23 (3), 269-273.
- (12) Sun, T.; Li, Z.-J.; Yang, X.; Wang, S.; Zhu, Y.-H.; Zhang, X.-B., Imine-rich poly (o-phenylenediamine) as high-capacity trifunctional organic electrode for alkali-ion batteries. *CCS Chemistry* **2019**, 1 (4), 365-372.
- (13) Luo, C.; Borodin, O.; Ji, X.; Hou, S.; Gaskell, K. J.; Fan, X.; Chen, J.; Deng, T.; Wang, R.; Jiang, J.; Wang, C., Azo compounds as a family of organic electrode materials for alkali-ion batteries. *Proceedings of the National Academy of Sciences* **2018**, 115 (9), 2004.
- (14) Shadike, Z.; Tan, S.; Wang, Q.-C.; Lin, R.; Hu, E.; Qu, D.; Yang, X.-Q., Review on organosulfur materials for rechargeable lithium batteries. *Materials Horizons* **2021**, 8 (2), 471-500.
- (15) Wang, D.-Y.; Guo, W.; Fu, Y., Organosulfides: An Emerging Class of Cathode Materials for Rechargeable Lithium Batteries. *Accounts of Chemical Research* **2019**, 52 (8), 2290-2300.
- (16) Wu, Y.; Zeng, R.; Nan, J.; Shu, D.; Qiu, Y.; Chou, S.-L., Quinone Electrode Materials for Rechargeable Lithium/Sodium Ion Batteries. *Adv Energy Mater* **2017**, 7 (24), 1700278.

- (17) Bitenc, J.; Pavčnik, T.; Košir, U.; Pirnat, K., Quinone Based Materials as Renewable High Energy Density Cathode Materials for Rechargeable Magnesium Batteries. *Materials* **2020**, *13* (3).
- (18) Bhosale, M. E.; Chae, S.; Kim, J. M.; Choi, J.-Y., Organic small molecules and polymers as an electrode material for rechargeable lithium ion batteries. *J Mater Chem A* **2018**, *6* (41), 19885-19911.
- (19) Petronico, A.; Bassett, K. L.; Nicolau, B. G.; Gewirth, A. A.; Nuzzo, R. G., Toward a Four-Electron Redox Quinone Polymer for High Capacity Lithium Ion Storage. *Adv Energy Mater* **2018**, *8* (5), 1700960.
- (20) Song, Z.; Qian, Y.; Gordin, M. L.; Tang, D.; Xu, T.; Otani, M.; Zhan, H.; Zhou, H.; Wang, D., Polyanthraquinone as a Reliable Organic Electrode for Stable and Fast Lithium Storage. *Angewandte Chemie International Edition* **2015**, *54* (47), 13947-13951.
- (21) Sieuw, L.; Ernould, B.; Gohy, J.-F.; Vlad, A., On the improved electrochemistry of hybrid conducting-redox polymer electrodes. *Sci Rep-Uk* **2017**, *7* (1), 4847.
- (22) Kapaev, R. R.; Zhidkov, I. S.; Kurmaev, E. Z.; Stevenson, K. J.; Troshin, P. A., Hexaazatriphenylene-based polymer cathode for fast and stable lithium-, sodium- and potassium-ion batteries. *J Mater Chem A* **2019**, *7* (39), 22596-22603.
- (23) Zhao, Q.; Wang, J.; Lu, Y.; Li, Y.; Liang, G.; Chen, J., Oxocarbon Salts for Fast Rechargeable Batteries. *Angewandte Chemie International Edition* **2016**, *55* (40), 12528-12532.
- (24) Doane, L. M.; Fatiadi, A. J., Electrochemical oxidation of several oxocarbon salts in N,N-dimethylformamide. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry* **1982**, *135* (2), 193-209.

- (25) Kim, H.; Seo, D.-H.; Yoon, G.; Goddard, W. A.; Lee, Y. S.; Yoon, W.-S.; Kang, K., The Reaction Mechanism and Capacity Degradation Model in Lithium Insertion Organic Cathodes, Li₂C₆O₆, Using Combined Experimental and First Principle Studies. *The Journal of Physical Chemistry Letters* **2014**, 5 (17), 3086-3092.
- (26) Preisler, P. W.; Berger, L., Preparation of Tetrahydroxyquinone and Rhodizonic Acid Salts from the Product of the Oxidation of Inositol with Nitric Acid*. *J Am Chem Soc* **1942**, 64 (1), 67-69.
- (27) Chen, H.; Armand, M.; Demailly, G.; Dolhem, F.; Poizot, P.; Tarascon, J.-M., From Biomass to a Renewable LiXC₆O₆ Organic Electrode for Sustainable Li-Ion Batteries. *ChemSusChem* **2008**, 1 (4), 348-355.
- (28) Belanger, R. L.; Commarieu, B.; Paolella, A.; Daigle, J.-C.; Bessette, S.; Vijh, A.; Claverie, J. P.; Zaghbi, K., Diffusion Control of Organic Cathode Materials in Lithium Metal Battery. *Sci Rep-Uk* **2019**, 9 (1), 1213.
- (29) Wang, X.; Shang, Z.; Yang, A.; Zhang, Q.; Cheng, F.; Jia, D.; Chen, J., Combining Quinone Cathode and Ionic Liquid Electrolyte for Organic Sodium-Ion Batteries. *Chem-US* **2019**, 5 (2), 364-375.
- (30) Wang, X.; Zhang, P.; Tang, X.; Guan, J.; Lin, X.; Wang, Y.; Dong, X.; Yue, B.; Yan, J.; Li, K.; Zheng, H.; Mao, H.-k., Structure and Electrical Performance of Na₂C₆O₆ under High Pressure. *The Journal of Physical Chemistry C* **2019**, 123 (28), 17163-17169.
- (31) Colmenero, F., Anomalous mechanical behavior of the deltic, squaric and croconic cyclic oxocarbon acids. *Materials Research Express* **2019**, 6 (4), 045610.

- (32) Ciezak-Jenkins, J. A., High-pressure polymorphism of the electrochemically active organic molecule tetrahydroxy-p-benzoquinone. *Journal of Molecular Structure* **2016**, *1119*, 71-77.
- (33) Rychkov, D. A., A Short Review of Current Computational Concepts for High-Pressure Phase Transition Studies in Molecular Crystals. *Crystals* **2020**, *10* (2).
- (34) Müller, V.; Scurtu, R.-G.; Memm, M.; Danzer, M. A.; Wohlfahrt-Mehrens, M., Study of the influence of mechanical pressure on the performance and aging of Lithium-ion battery cells. *J Power Sources* **2019**, *440*, 227148.
- (35) Cannarella, J.; Arnold, C. B., Stress evolution and capacity fade in constrained lithium-ion pouch cells. *J Power Sources* **2014**, *245*, 745-751.
- (36) Louli, A. J.; Genovese, M.; Weber, R.; Hames, S. G.; Logan, E. R.; Dahn, J. R., Exploring the Impact of Mechanical Pressure on the Performance of Anode-Free Lithium Metal Cells. *J Electrochem Soc* **2019**, *166* (8), A1291-A1299.
- (37) Berckmans, G.; De Sutter, L.; Marinaro, M.; Smekens, J.; Jaguemont, J.; Wohlfahrt-Mehrens, M.; van Mierlo, J.; Omar, N., Analysis of the effect of applying external mechanical pressure on next generation silicon alloy lithium-ion cells. *Electrochim Acta* **2019**, *306*, 387-395.
- (38) Singer, J. P.; Sämann, C.; Gössl, T.; Birke, K. P., Pressure Monitoring Cell for Constrained Battery Electrodes. *Sensors* **2018**, *18* (11).
- (39) Perdew, J. P.; Burke, K.; Ernzerhof, M., Generalized Gradient Approximation Made Simple. *Phys Rev Lett* **1996**, *77* (18), 3865-3868.

- (40) Kresse, G.; Furthmüller, J., Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science* **1996**, 6 (1), 15-50.
- (41) Dion, M.; Rydberg, H.; Schröder, E.; Langreth, D. C.; Lundqvist, B. I., Van der Waals Density Functional for General Geometries. *Phys Rev Lett* **2004**, 92 (24), 246401.
- (42) Klimeš, J.; Bowler, D. R.; Michaelides, A., Van der Waals density functionals applied to solids. *Phys Rev B* **2011**, 83 (19), 195131.
- (43) Klimeš, J.; Bowler, D. R.; Michaelides, A., Chemical accuracy for the van der Waals density functional. *Journal of Physics: Condensed Matter* **2009**, 22 (2), 022201.
- (44) Guin, P. S.; Das, S.; Mandal, P. C., Electrochemical Reduction of Quinones in Different Media: A Review. *International Journal of Electrochemistry* **2011**, 2011, 816202.
- (45) Uchimiya, M.; Stone, A. T., Reversible redox chemistry of quinones: Impact on biogeochemical cycles. *Chemosphere* **2009**, 77 (4), 451-458.
- (46) Schweidler, S.; de Biasi, L.; Schiele, A.; Hartmann, P.; Brezesinski, T.; Janek, J., Volume Changes of Graphite Anodes Revisited: A Combined Operando X-ray Diffraction and In Situ Pressure Analysis Study. *The Journal of Physical Chemistry C* **2018**, 122 (16), 8829-8835.
- (47) West, K.; Jacobsen, T.; Zachau-Christiansen, B.; Atlung, S., Determination of the differential capacity of intercalation electrode materials by slow potential scans. *Electrochim Acta* **1983**, 28 (1), 97-107.
- (48) Aihara, J.-i., The Solid-State Spectra of Several Rhodizonates. *Bulletin of the Chemical Society of Japan* **1974**, 47 (11), 2899-2900.

- (49) Lee, M.; Hong, J.; Lopez, J.; Sun, Y.; Feng, D.; Lim, K.; Chueh, W. C.; Toney, M. F.; Cui, Y.; Bao, Z., High-performance sodium–organic battery by realizing four-sodium storage in disodium rhodizionate. *Nature Energy* **2017**, 2 (11), 861-868.
- (50) Bracewell, B. L.; Veigele, W. J., Elemental X-Ray Cross Sections at Selected Wavelengths. *Advances in X-Ray Analysis* **1971**, 15, 352-364.
- (51) Silberstein, K. E.; Pastore, J. P.; Zhou, W.; Potash, R. A.; Hernández-Burgos, K.; Lobkovsky, E. B.; Abruña, H. D., Electrochemical lithiation-induced polymorphism of anthraquinone derivatives observed by operando X-ray diffraction. *Phys Chem Chem Phys* **2015**, 17 (41), 27665-27671.
- (52) Mao, H.-K.; Chen, B.; Chen, J.; Li, K.; Lin, J.-F.; Yang, W.; Zheng, H., Recent advances in high-pressure science and technology. *Matter and Radiation at Extremes* **2016**, 1 (1), 59-75.
- (53) Liu, H.; Maginn, E., Effect of ion structure on conductivity in lithium-doped ionic liquid electrolytes: A molecular dynamics study. *The Journal of Chemical Physics* **2013**, 139 (11), 114508.
- (54) Kang, K.; Ceder, G., Factors that affect Li mobility in layered lithium transition metal oxides. *Phys Rev B* **2006**, 74 (9), 094105.

Chapter 5: Demonstration of a Stable Redox-Active Metal Rhodizonate Cathode Material Through Crystal Structure Modification

(Work in preparation)

5.1. Introduction

Since their commercialization by the Sony corporation in 1991, lithium-ion batteries (LIBs) have become the energy storage medium of choice for small scale applications.^{1, 2} Their roll in facilitating the progressive miniaturization and adoption of portable electronics cannot be overstated; however, as the landscape for energy storage evolves, and the need for large scale power storage becomes more pressing, the limitations of existing LIB chemistry become ever more apparent. In particular, the limited gravimetric energy density and high cost of existing cobalt-based cathode materials remain pressing issues in the ongoing efforts to electrify transportation and to adapt LIB technology to larger-scale applications such as grid scale load leveling and intermittent power storage for renewable energy.³⁻⁶

To address the above limitations new chemistries must be developed which offer significant improvements in overall gravimetric energy density (at comparable or lower cost) without sacrificing the safety and longevity necessary for practical application.⁷⁻¹⁰ One promising approach is to supplant conventional transition metal-based cathode materials with redox active organic species. Indeed, organic materials have long been attractive as alternatives to metal oxides due to their low cost, high synthetic tunability, low molecular weight, and ability to accommodate multi-electron redox.¹¹⁻¹⁴ The beneficial combination of low molecular weight and multi-electron redox allows organic materials to surpass the capacities achievable in conventional inorganic electrodes by a large margin; in some instances by more than a factor of five.^{15, 16}

While there exists an enormous library of organic materials which can undergo redox transitions including amines, carboxylates, imines, and organosulfur compounds,¹⁷⁻²² the majority of work on organic cathodes has focused on quinone-based materials due to their high capacities and comparatively high voltages relative to other functional groups (ca. 3 V vs. Li/Li⁺).²³⁻²⁵ Even the simplest example of a quinone active material, 1,4-benzoquinone, displays a theoretical capacity of 496 mA h g⁻¹ at a nominal discharge voltage of 2.7 V vs Li/Li⁺ due to a two electron reduction of the quinone to form the fully aromatic lithiated diol. Despite the promise of this class of electrode materials, small molecule organic cathodes, almost universally, display poor capacity retention during cycling, a phenomenon that is oft attributed to the high solubility of the redox active moiety in conventional organic electrolytes.²⁶⁻²⁸

Numerous approaches have been proposed to address the solubility issues associated with organic electrodes. One of the most popular of these approaches is the incorporation of the redox active species in a high molecular weight polymer, thereby dramatically reducing the solubility of the cathode material. Unfortunately, despite significant progress in the development of polymer electrodes,²⁹⁻³² the presence of a (typically redox inactive) structural backbone lowers the gravimetric capacities achievable with these materials when compared to those accessible to redox active small molecules. Another approach, which has received less attention, is to sequester the organic species into an insoluble organic salt, thereby suppressing its dissolution. Indeed, this approach is appealing as, in certain cases, it has proven successful at suppressing active material dissolution, while maintaining the high gravimetric capacity of the small molecule redoxmer.³³

The potential benefits of crystalline organic salts as cathode materials have motivated the search for organic anions that display high gravimetric capacities at reasonable voltages. Among the sea of possible organic anions, oxocarbon structures have proven attractive as they exhibit the

highest possible density of redox active ketones, and therefore the highest possible capacity, for a stable carbon structure.^{27, 34, 35} Indeed, the redox active cyclic oxocarobon cyclohexane-hexone (C_6O_6) can, in principle, deliver a gravimetric capacity of 956 mA h g^{-1} if all carbonyl groups are accessible.¹⁵ In practice it is difficult to access full six electron redox from this material, with most reports discussing the semi-reversible four electron redox afforded by the lithiated salt of this structure di-lithium rhodizonate ($Li_2C_6O_6$).^{36, 37} $Li_2C_6O_6$ is promising as an electrode material as it is insoluble in many common organic solvents, and its four redox active carbonyl groups deliver an initial capacity of 580 mA h g^{-1} .

Unfortunately, the cycling stability of di-lithium rhodizonate remains poor despite its high crystallinity, and generally low solubility in non-aqueous media. The origin of the observed capacity fade has been debated to some extent, with early work suggesting the dissolution of one of the lithiated discharge products into the organic electrolyte.³⁶ Other work, however, pointed towards a chemo-mechanical origin, wherein the redox active material delaminates from the electrode due to repeated structural changes during intercalation and deintercalation.³⁷ More recently, crystallographic deactivation of the redox active species during lithiation has been proposed as the origin of the poor cycling stability.³⁸ This deactivation mechanism is believed to be the result of a change in Li^+ coordination within the host rhodizonate lattice wherein Li^+ irreversibly occupies low energy tetrahedral sites. The effects of this change in binding geometry are two-fold: firstly, the high binding affinity of tetrahedral Li^+ increases the overpotentials associated with deintercalation, secondly, the lattice contraction of the crystal on lithiation reduces the mobility of Li^+ within the layered structure.³⁸

While these investigations have been moving towards a more complete understanding of the capacity fade mechanism in $Li_2C_6O_6$ they raise the question of whether there exists an approach

to circumvent the crystallographic instabilities associated with the di-lithium salt. Building on the results we obtained from our previous investigation we aimed to explore the effect of crystal structure modification on the electrochemical stability of materials containing the rhodizonate dianion. This report describes the electrochemical behavior of crystalline salts of rhodizonate with different metals. While a handful of other groups have reported on the behavior of rhodizonate in the presence of transition metal dianions (namely Fe^{III} and Cu^{II})^{39, 40} there remains a lack of understanding regarding the impact the cation has on the structure of the resulting $\text{M(II)C}_6\text{O}_6$ and, by extension, the electrochemical behaviors of these complexes.

Herein, we consider rhodizonate complexes exhibiting three different structural archetypes generated by the chelation of the anion with either an alkali metal (Li^+), a first-row transition metals (Ni^{II}), or a heavy metal (Pb^{II}). We demonstrate that rhodizonate complexes that coordinate the cation in a square planar geometry co-planar with the rhodizonate layer (Li^+ , and Ni^{II}) universally demonstrate poor electrochemical stability as evidenced by rapid capacity fade during galvanostatic cycling. In contrast rhodizonate complexes with large, highly coordinated divalent cations (Pb^{II}) display much greater electrochemical stability largely due to the occlusion of low energy tetrahedral Li^+ binding sites between rhodizonate layers. This work suggests that the electrochemical stability of rhodizonate complexes during repeated lithiation and delithiation is largely dictated by the Li^+ coordination environment, and that by controlling this coordination chemistry, stable redox active materials can be obtained.

5.2. Experimental

5.2.1 Synthesis of MC_6O_6

All chemicals were used as received without further purification. Metal rhodizonates were synthesized by an aqueous precipitation reaction between rhodizonic acid (99% Alpha Aesar) and the corresponding metal carbonate ($Li_2CO_3 > 99\%$, $NiCO_3 > 99\%$, $PbCO_3 > 99\%$, Sigma Aldrich). In a typical synthesis rhodizonic acid powder (100 mg) was combined with the desired metal carbonate in a stoichiometric mixture (1:1 molar ratio between metal and rhodizonic acid for Ni and Pb, and 2:1 molar ratio for Li) and ground together in an agate mortar and pestle until visually homogeneous. This mixture was then allowed to react in a minimal volume of water (ca. 1 mL) with vigorous stirring for 12 hours followed by separation of the solid hydrated MC_6O_6 by centrifugation (3200 rpm, 15 min). The use of small liquid volumes is critical, as at higher solvent volumes rhodizonic acid is susceptible to ring scission to form the five membered croconate ion.^{41, 42} The as obtained powder was then rinsed three times with acetone (ca. 10 mL for each rinse cycle) and dried in a vacuum oven at 120 °C for 12 hours to remove lattice water.

5.2.2 Characterization of as-synthesized material

As-synthesized metal rhodizonates were initially characterized using powder X-Ray diffraction, Fourier transform infrared spectroscopy (FTIR), and Raman spectroscopy. Powder X-ray diffraction (PXRD) was carried out on a Bruker D8 Advance XRD system equipped with a $Cu\ \alpha$ source and a 2D Eiger2 R 500K detector. Bulk powder samples were analyzed between 10° and 60° 2 θ in the typically coupled Bragg–Brentano geometry. FTIR spectra were collected using a Perkin Elmer Frontier FT-IR Spectrometer within a wavenumber range of 500 to 4000 cm^{-1} . Raman spectra for bulk powders were obtained as described previously⁴³ using excitation from a

He/Ne laser (Meredith Instruments) at 632.8 nm. Spectra were accumulated from 30 individual acquisitions of 1 s each. A 50 μm slit width was used, providing an estimated spectral resolution of approximately 6–7 cm^{-1} .

The morphologies of as-synthesized metal rhodizonates were interrogated with scanning electron microscopy (SEM) using an FEI Quanta FEG 450 ESEM. Inductively couple plasma optical emission spectroscopy (ICP-OES) measurements were conducted on a PerkinElmer Optima 8300, while C, H, and N elemental analysis was conducted on an Exeter Analytical Model CE440 CHN analyzer. As synthesized samples were submitted for analysis under an inert atmosphere to prevent hydration from atmospheric moisture. Lithiated samples were prepared for ICP-OES analysis electrochemically in CR2032 coin cells under a constant reducing current (C/10) until the materials registered a potential of 1.5 V vs. Li/Li^+ (see 2.3 and 2.4 for detailed descriptions of electrode composition and cell fabrication). Lithiated cathodes were then removed from their cells and rinsed with ca. 10 mL of DME to remove residual electrolyte. The active material was then divested from the Al mesh by sonication in DME, followed by drying under vacuum. Samples were submitted for analysis under and Ar atmosphere to prevent hydration or oxidation of the lithiated sample.

5.2.3 Electrode Fabrication

In a typical preparation the as-synthesized metal rhodizonate (60 wt%) was mixed with conductive carbon (Super P Li, Timcal, 30 wt%) and PTFE (polytetrafluoroethylene, 5 μm mesh, Sigma Aldrich, 10 wt%) and ground together in an agate mortar and pestle until visually homogeneous. This active mixture was then pressed into an Al mesh disc (1.27 cm diameter) using a hydraulic press and pellet dye at a pressure of 1500 PSI for a duration of 5 minutes. Following pressurization, electrodes were recovered, and excess active material removed using scotch tape.

As-prepared electrodes were then dried in a vacuum oven for a minimum of 5 hours at 80 °C to remove any residual water before being transferred into an Ar filled glovebox (H_2O , $\text{O}_2 < 2$ ppm).

5.2.4 Electrochemical Measurements

Initial electrochemical characterizations of $\text{M(II)C}_6\text{O}_6$ electrodes were conducted in CR2032 coin cells (MTI Corporation, Richmond, CA). Battery stacks were assembled in an argon-filled glovebox (O_2 and $\text{H}_2\text{O} < 2$ ppm). The cells were constructed as follows: the positive electrode was placed at the base of the cell followed by a polyethylene/polypropylene separator (Celgard), which was wet with 40 μL of electrolyte. The electrolyte consisted of a 1 M solution of lithium bis(trifluoromethanesulfonyl)imide (LiTFSI, > 99%, Sigma Aldrich) in a 1:1 mixture of 1,3-dioxolane (DOL, 99.5%, anhydrous Sigma Aldrich) and 1,2-dimethoxyethane (DME, > 99% Sigma Aldrich) by volume. A lithium anode was then prepared as follows: a 14 mm diameter circular electrode was punched from lithium metal foil (0.75 mm thickness, 99.9%, Alfa Aesar). The native oxide layer was removed from both sides of the electrode by scraping the surface with a razor blade. The electrode was then smoothed with a stainless-steel roller to yield a flat, lustrous surface. The anode was placed onto the separator to serve as the counter/reference electrode. A stainless-steel disc and wave spring were then added onto the lithium disc to complete the stack, and the cells were sealed using a hydraulic coin cell crimper (MTI Corporation, Richmond, CA).

Cells were cycled voltammetrically between 1.5 and 3.5 V vs Li/Li^+ at a scan rate of 200 $\mu\text{V s}^{-1}$ using a CH Instruments electrochemical workstation (CHI, Model 6002E, Austin, TX). Galvanostatic cycling was conducted using a Neware BTS4000 cycler with cutoff voltages at 1.5 and 3.5 V vs Li/Li^+ at a C/10 rate determined from the theoretical capacities of each rhodizonate material assuming complete $4e^-$ redox. Electrochemical impedance spectroscopy (EIS) was

performed on a BioLogic Science Instruments impedance analyzer (SP-150) between 1 MHz and 10 mHz with an amplitude of 10 mV.

5.2.5 Computational Details

Structural optimizations were carried out using the Vienna ab initio simulation package (VASP 5.4.4) under the generalized gradient approximation (GGA) using Perdew–Burke–Ernzerhof (PBE) pseudopotentials, as described elsewhere.⁴⁴ All computations were conducted with a plane-wave basis and a projector-augmented wave (PAW) method.⁴⁵ van der Waals (VDW) contributions to the total energy were considered by applying a VDW-DF correction to the conventional Kohn–Sham DFT energy, as discussed elsewhere.^{46–48} Structures were optimized with a reduced unit cell consisting of two formula units of MC₆O₆. Energy calculations were performed with $7 \times 7 \times 7$ Γ -centered k-point mesh with an energy cutoff of 520 eV and convergence criterion of 2 meV $\text{\AA}^{-1}$. Ground state structures were visualized, and simulated diffraction patterns computed using VESTA.⁴⁹

Vibrational mode assignments were, in part, informed by DFT simulation of gas phase metal rhodizonate species. Ground state molecular structures and vibrational frequencies (both infrared and Raman) were computed in Gaussian 16.⁵⁰ Optimizations were carried out within the B3LYP density functional using the LANL2DZ basis set for metal atoms and the TZVP basis set for carbon and oxygen as described in previous vibrational analyses of transition metal complexes.^{51–54} Simulated infrared and Raman spectra for optimized geometries were constructed using GausSum.⁵⁵

5.3. Results

Table 5.1. Composition of as-synthesized metal rhodizonates. Theoretical atomic percentages are calculated from the expected stoichiometry of each material, while the experimental values were obtained from ICP-OES.

Stoichiometry	M at. %	M at. %	C at. %	C at. %
	Expected	Found	Expected	Found
Li ₂ C ₆ O ₆	7.63	8.05 ± 1.2	39.61	38.2 ± 0.6
NiC ₆ O ₆	25.9	25.8 ± 0.3	31.8	30.9 ± 0.3
PbC ₆ O ₆	55.2	54.5 ± 0.8	19.2	18.5 ± 0.7

The atomic compositions of as-synthesized metal rhodizonates were initially probed using inductively coupled plasma optical emission spectroscopy (ICP-OES) the results of which are presented in Table 5.1. For each complex the experimentally obtained atomic percentages of metal (lithium, nickel, or lead) and carbon align well with those expected from a one-to-one coupling of the divalent metal cation to the rhodizonate dianion (or a two-to-one coupling of Li⁺ with rhodizonate). Similarly, following dehydration, the C, H, and N elemental analyses for each material (Table 5.2) report very low atomic concentrations of hydrogen (< 0.1 at. %) and negligible atomic concentrations of nitrogen as expected for the MC₆O₆ complex in its fully dehydrated state.

Table 5.2. Results of C, H, and N analysis of as-synthesized metal rhodizonates.

Stoichiometry	C at. %	C at. %	H at. %	H at. %	N at. %	N at. %
	Expected	Found	Expected	Found	Expected	Found
Li ₂ C ₆ O ₆	7.63	8.05 ± 1.2	0	0.1 ± 0.1	0	0.1 ± 0.1
NiC ₆ O ₆	25.9	25.8 ± 0.3	0	0.3 ± 0.2	0	0.2 ± 0.1
PbC ₆ O ₆	55.2	54.5 ± 0.8	0	0.2 ± 0.1	0	0.2 ± 0.1

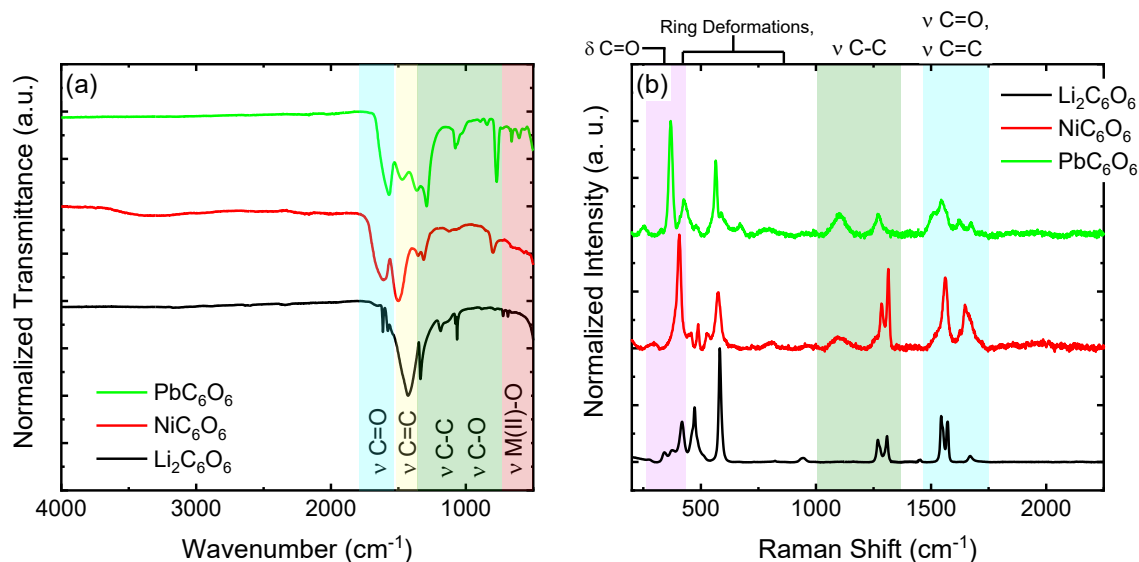


Figure 5.1. Vibrational spectra collected from as-synthesized metal rhodizonates. a) Presents the FTIR spectra while b) presents the Raman spectra obtained from lithium (black trace) nickel (red trace) and lead (green trace) rhodizonate. Spectral regions of interest are demarcated by colored bands along with tentative assignments for the vibrational modes associated with each region.

Figure 5.1a displays FTIR spectra obtained from each of the as synthesized metal rhodizonates with tentative assignments for the different vibrational modes associated with each structure. Regardless of the cation, the three spectra display similar spectral characteristics with strong absorbances between 1350 and 1650 cm^{-1} , moderate absorbance between 750 and 1350 cm^{-1} , and weak features below 750 cm^{-1} . Features within the first region are attributed to both the C=O and C=C stretching modes, aligning with previous work on quinone based molecules and other cyclic oxocarbons including croconic acid.^{56,57} While the C=O stretching modes of nickel and lead rhodizonates can be clearly distinguished (peak absorbances at ca. 1630 and 1580 cm^{-1} respectively), those of lithium rhodizonate cannot be easily identified due to overlap with the alkene stretch at lower frequency (ca. 1450 cm^{-1}). Regardless, the FTIR spectrum obtained from the di-lithium salt matches closely with the original report of its synthesis.³⁶

At lower frequencies, between 750 and 1350 cm^{-1} , several less intense spectral features are present, likely associated with the C-C carbonyl stretching modes and C-O stretching modes, all of which are expected from the rhodizonate dianion. At even lower frequencies faint features can be seen at ca. 600 cm^{-1} which may be manifestations of M-O stretching modes. Apart from the frequencies associated with the metal rhodizonate complex there is very little spectral intensity in the range of 3000-3500 cm^{-1} for any of the samples (a slight feature is present in the FTIR spectrum of NiC_6O_6) suggesting that following the drying process outlined in the experimental section the waters of hydration have been removed from the synthesized rhodizonate structures.

The vibrational mode assignments obtained from FTIR spectroscopy can also be deconvoluted in the Raman spectra of each rhodizonate complex (Figure 5.1b) along with several low frequency (400-600 cm^{-1}) vibrational modes associated with ring deformations.⁵⁸ Due to the overlap of ring deformation modes and carbonyl bending modes a confident deconvolution of the M-O stretching mode could not be obtained. Nevertheless, the vibrational mode assignments displayed in Figure 5.1 align both with previous experimental reports of other oxocarbons and quinones as well as with DFT simulation (Figure 5.2).

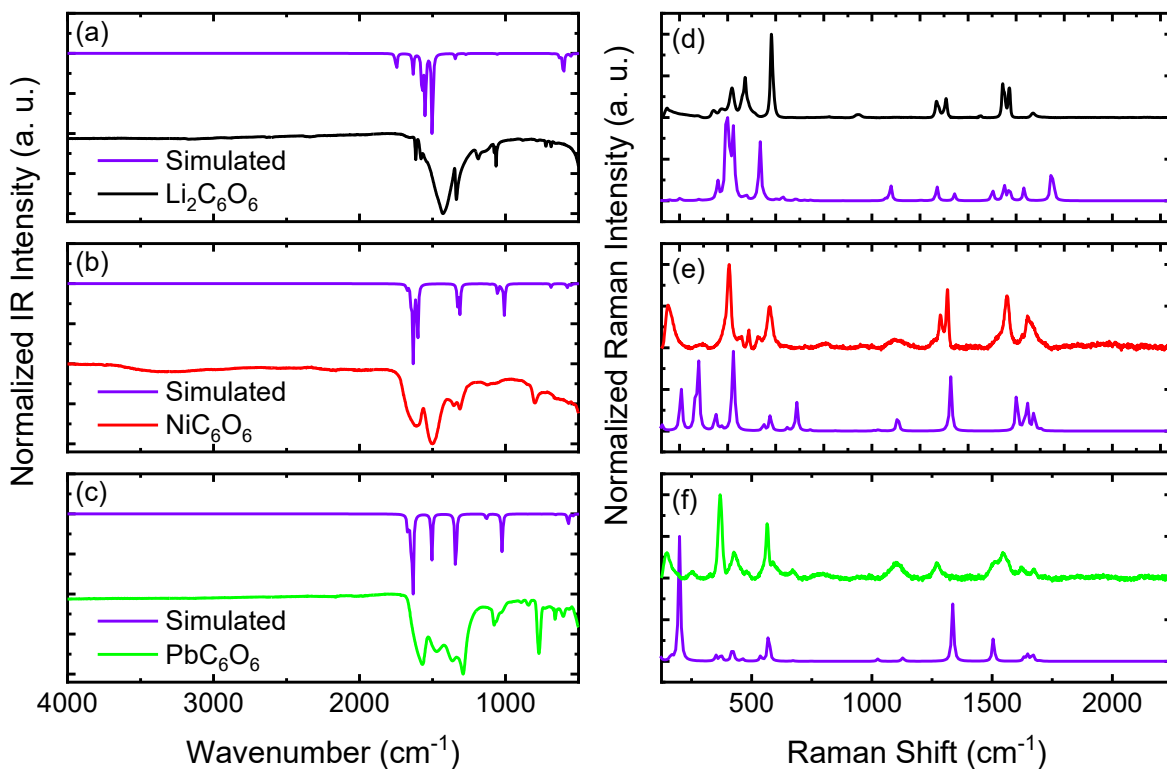


Figure 5.2. Simulated IR absorption spectra (a-c) and Raman spectra (d-f) of metal rhodizonate complexes plotted against experimental data.

To gain further insight into the structures of the as synthesized metal rhodizonates powders X-ray diffraction was conducted on each sample, the results of which are presented in Figure 5.3. The strong Bragg peaks displayed by each of the samples reflect a high degree of crystallinity in the salt precipitate, similar to other reports of both divalent and monovalent rhodizonate complexes.⁵⁹⁻⁶¹ Likewise, the significant variation in the powder diffraction patterns obtained from these materials suggest that the structure of rhodizonate complexes can vary dramatically as a function of the metal cation. Fortunately, regardless of these cation dependent structural changes, all materials display no trace of the metal carbonate precursors suggesting good conversion to the desired product (Figure 5.4).

Despite several reports describing the powder diffraction patterns of various rhodizonate complexes,⁶⁰⁻⁶⁴ the difficulties associated with producing single crystals of these materials, makes high quality structural data of many of them scarce. Regardless the diffraction patterns displayed in Figure 5.3 bear close resemblance to previous investigations of this class of materials. In particular, the experimental diffraction pattern for di-lithium rhodizonate is identical to what has been previously reported for this material which is suspected to form a layered structure comprised of hexagonally close packed arrays of rhodizonate ions wherein Li^+ occupies pseudo-square-planar sites within these rhodizonate layers (right hand side of Figure 5.3).³⁶⁻³⁸

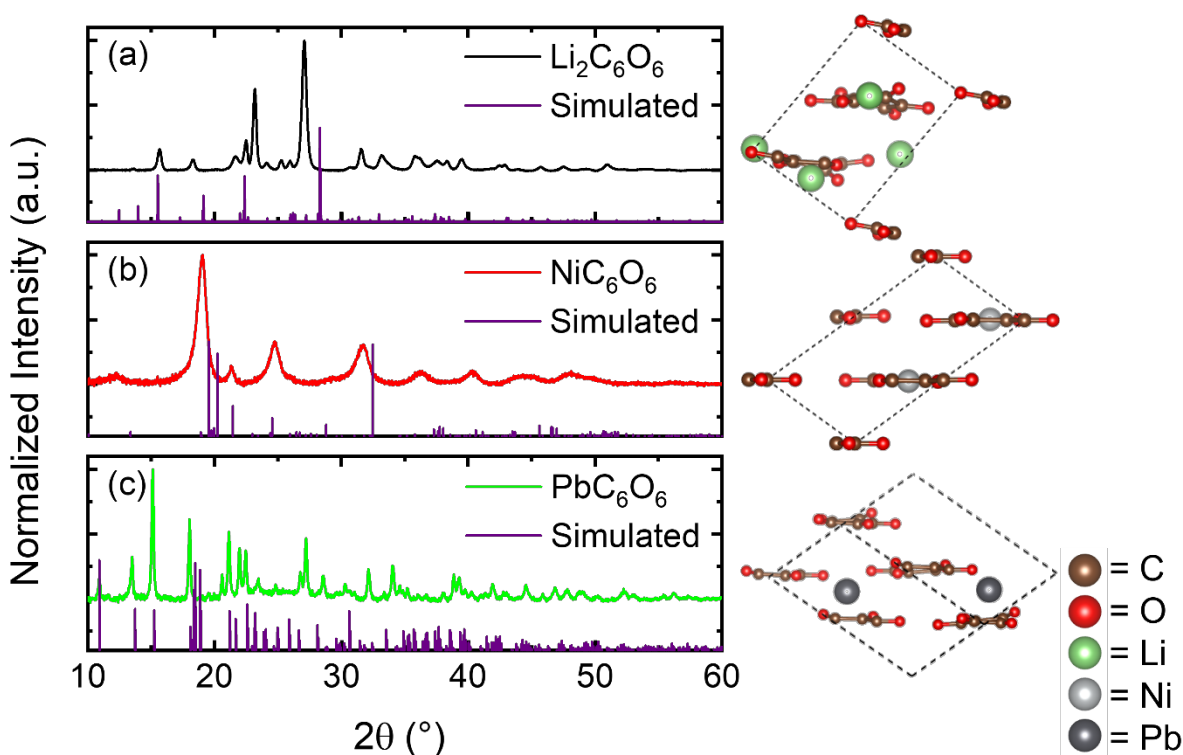


Figure 5.3. Powder diffraction patterns collected from as-synthesized a) $\text{Li}_2\text{C}_6\text{O}_6$ b) NiC_6O_6 and c) PbC_6O_6 . Simulated diffraction patterns obtained from DFT optimized structures are presented below each diffractogram as purple traces.

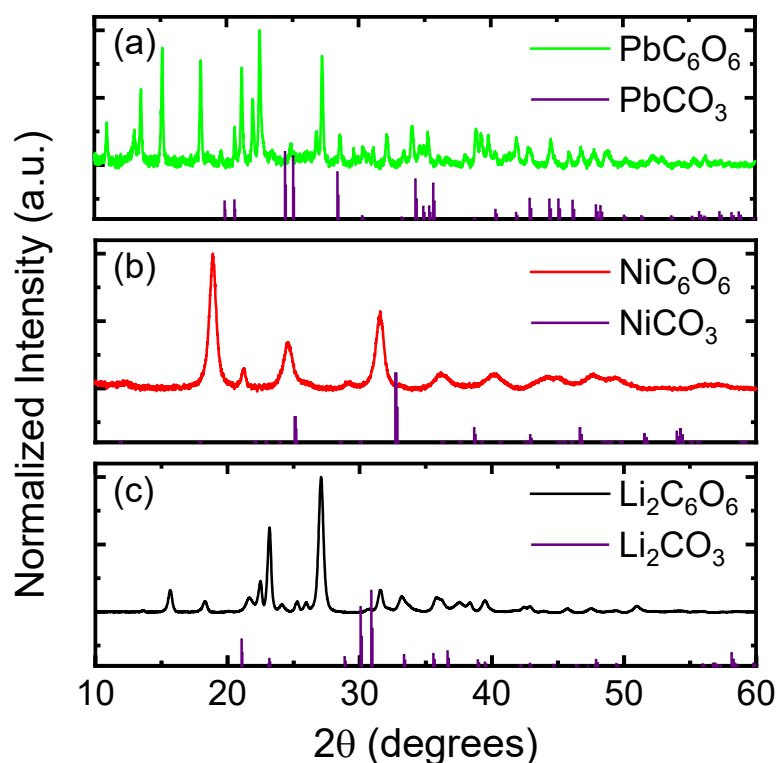


Figure 5.4. Comparison of the powder X-Ray diffraction patterns obtained from the as-synthesized metal rhodizonates to the standard patterns associated with their carbonate precursors.

The PXRD pattern of NiC_6O_6 presented here (as well as the PXRD patterns of other first row transition metal rhodizonates synthesized with this method, Figure 5.5) is very similar to structures that have been obtained from powders of hydrated $\text{NiC}_6\text{O}_6 \cdot n\text{H}_2\text{O}$.⁶⁰ To confirm that the as synthesized material is indeed structurally distinct from its hydrated counterpart, PXRD patterns were collected for each and are displayed in Figure 5.6. On dehydration there is a significant shift of most reflections towards higher 2θ signifying a lattice contraction due to the removal of water, although no large-scale structural reorganization is observed. Unfortunately, while the powder diffraction pattern of transition metal rhodizonates have been reported, there have not yet been unambiguous reports of their structure. Likewise, despite the popularity of the forensic sodium

rhodizonate test (used as a qualitative indicator of trace lead)⁶⁵ the single crystal structure of the lead rhodizonate has not been reported.

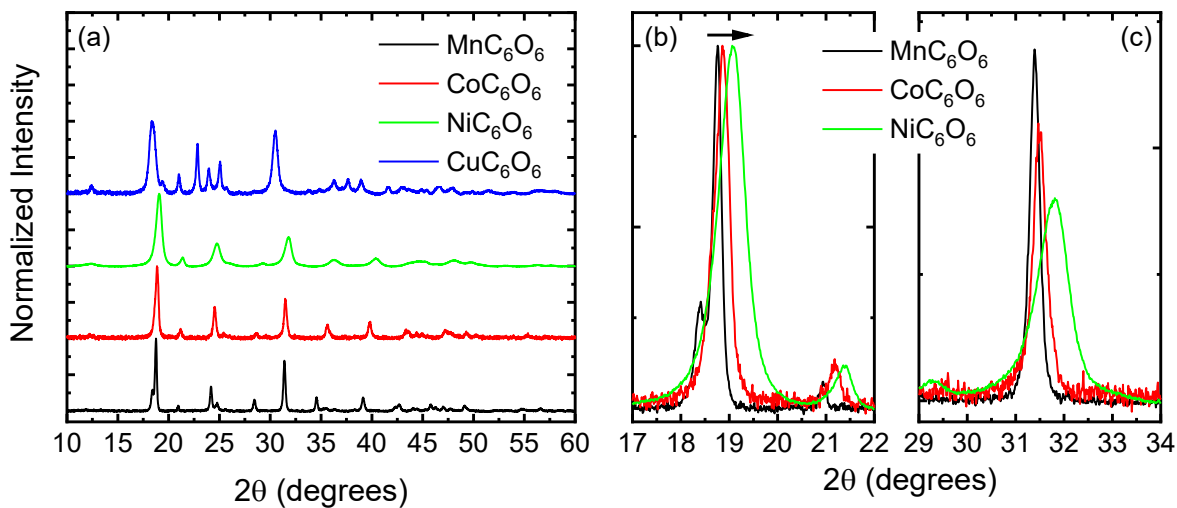


Figure 5.5. a) Powder X-ray Diffractograms obtained from as-synthesized first-row transition metal rhodizonate complexes. b) and c) present close-up regions of Mn, Co, and Ni rhodizonates displaying the progressive shift in lattice parameter moving across the d-block.

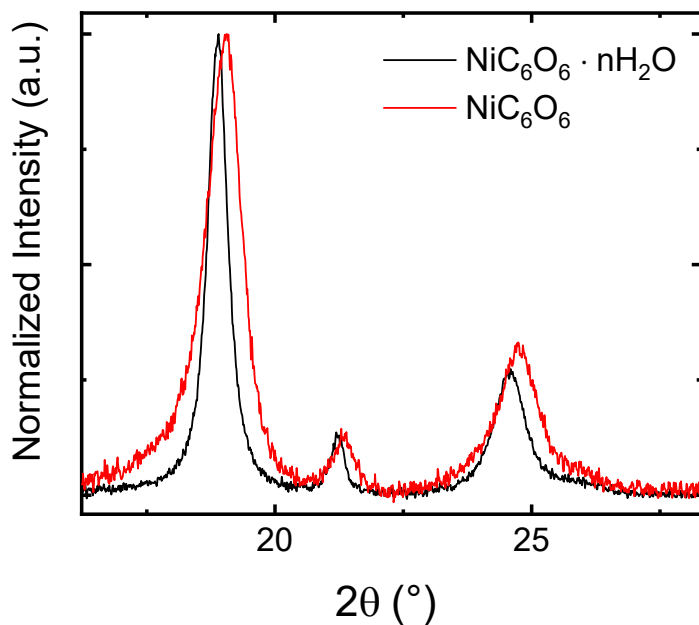


Figure 5.6. Powder diffraction patterns obtained from hydrated and dehydrated nickel rhodizonate powders.

While uncertainty remains about the structures of Ni and Pb rhodizonates it is likely that they are also composed of layered networks of rhodizonate ions. To explore this possibility the ground state structures of each of these materials (including lithium rhodizonate) were simulated using the Vienna *ab initio* Simulation Package (VASP). Within these simulations several different layering geometries and metal binding sites were considered, using the root mean square deviation between experimental and simulated diffraction patterns as a metric for the feasibility of computationally generated structures. In each instance the geometries that display the smallest deviation from experiment consist of hexagonally close packed arrays of rhodizonate ions which stack vertically in a canted ABAB pattern. Importantly, in all computed structures the optimized ground state displays an underestimation of the VDW layer spacing (demarcated by asterisks in Figures 5.3a-c) as has been seen in the simulation of other carbonaceous layered structures using vdW-DF2, or similar, functionals.⁶⁶

Interestingly, the predicted metal coordination within these structures varies with the cation. Both lithium and nickel ions are predicted to coordinate in pseudo square planar geometries within the rhodizonate layer, while Pb is predicted to coordinate between adjacent layers (right hand side of Figure 5.3). This change in coordination is likely due to the large size of the Pb^{2+} (ionic radius ≈ 133 pm) ion relative to Ni^{2+} (83 pm) or Li^+ (90 pm) as large ions may significantly disrupt vdw interactions between adjacent rhodizonate ions.⁶⁷ The purple traces in Figures 5.3a-c present the simulated diffraction patterns computed from VASP optimized ground state structures compared against the experimental patterns of lithium, nickel, and lead rhodizonates respectively.

In addition to their crystal structures, the morphologies of as-synthesized metal rhodizonates were interrogated with scanning electron microscopy (Figure 5.7). Both NiC_6O_6 and PbC_6O_6 display secondary particle sizes on the order of 10 μm with primary particles between 100

and 500 nm. The morphology of the primary particles in both instances is plate-like with thicknesses between 50 and 200 nm. In contrast $\text{Li}_2\text{C}_6\text{O}_6$ yields faceted particles roughly 1 μm across, similar to the morphologies reported by other groups.^{36, 37}

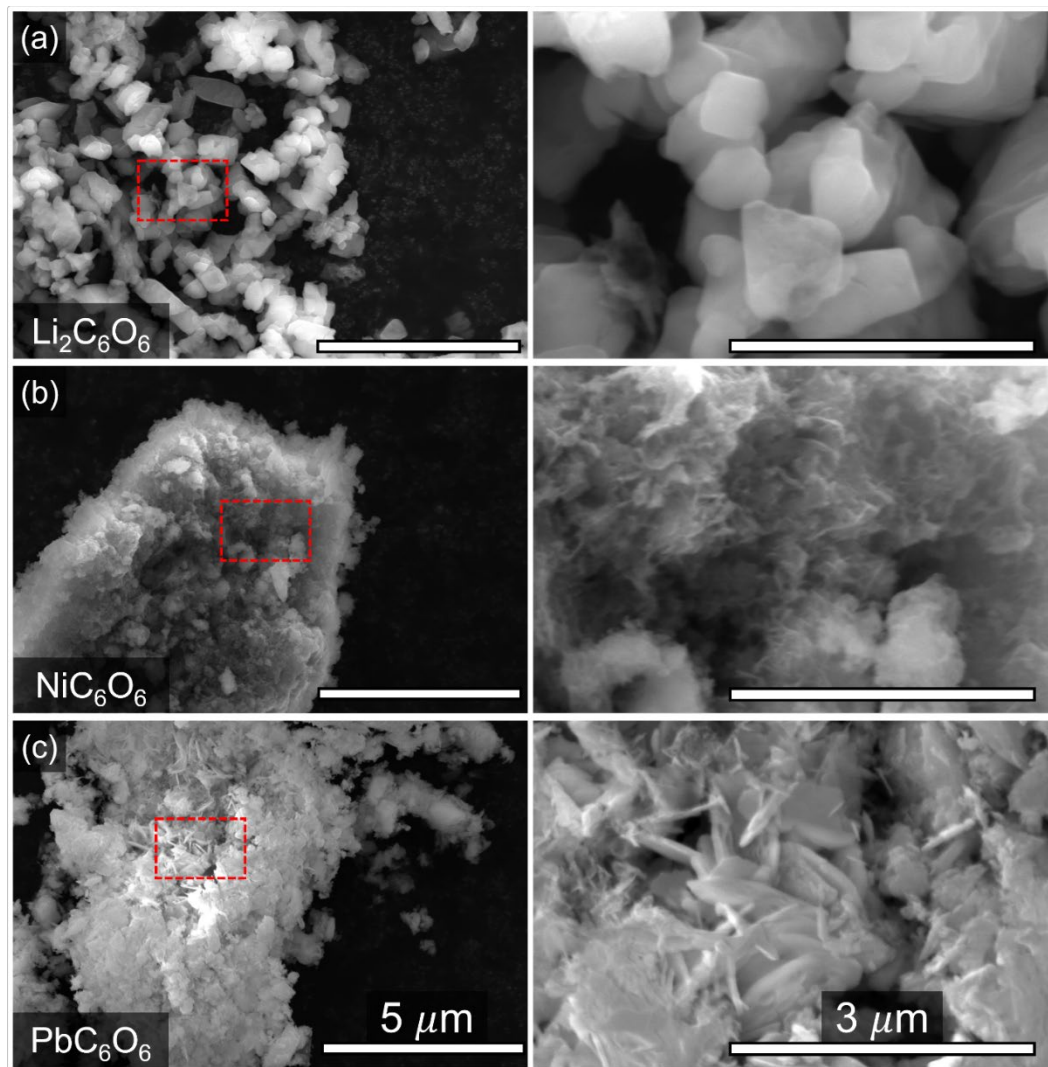


Figure 5.7. SEM micrographs obtained from as-synthesized a) $\text{Li}_2\text{C}_6\text{O}_6$, b) NiC_6O_6 , and c) PbC_6O_6 powders. Figures on the right are magnifications of the red regions demarcated in a-c describing the primary particle structure of as-synthesized materials.

To probe the electrochemical behaviors of each of the as synthesized rhodizonates, solid electrodes were fabricated by pressing the active material, along with conductive carbon and a polymer binder into an aluminum mesh. This solid-state preparation was used to avoid any

deleterious reactions between the redox active material and common solvents used for solution casting. Electrodes were initially characterized with cyclic voltammetry. Figures 5.8a-5.8c present the first 5 voltammetric cycles obtained from $\text{Li}_2\text{C}_6\text{O}_6$, NiC_6O_6 , and PbC_6O_6 respectively (the first cycle is differentiated by the dashed trace). In all cases the electrodes were cycled between 1.5 and 3.5 V vs Li/Li^+ at a scan rate of 0.2 mV s^{-1} using 1M LiTFSI in DOL:DME as the electrolyte. Regardless of the cation, all materials displayed oxidative peak current densities between 100 and 200 mA g^{-1} during the first cycle, reflecting similar initial electrochemical activity from all samples. While the activities displayed by the various active materials are similar, the overall voltammetric responses are very different.

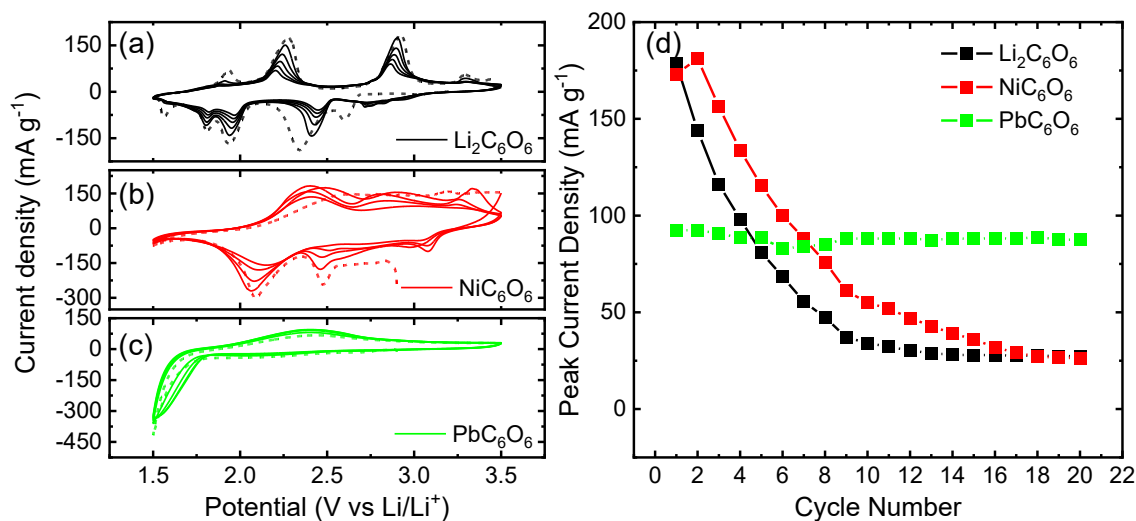


Figure 5.8. Voltammetric traces obtained from the first five cycles of a) $\text{Li}_2\text{C}_6\text{O}_6$, b) NiC_6O_6 and c) PbC_6O_6 electrodes at a scan rate of $200 \mu\text{V s}^{-1}$ with switching potentials at 1.5 and 3.5 V vs Li/Li^+ using 1M LiTFSI in 1:1 DOL : DME as electrolyte. d) Oxidative peak current densities obtained from the voltammograms presented in a-c plotted against cycle number. The potentials chosen for the oxidative waves were 2.25 V, 2.4 V, and 2.4 V vs Li/Li^+ for $\text{Li}_2\text{C}_6\text{O}_6$, NiC_6O_6 and PbC_6O_6 respectively.

Both $\text{Li}_2\text{C}_6\text{O}_6$ and NiC_6O_6 display multiple oxidative and reductive waves, associated with distinct electron transfer events.⁶⁸ The voltammetric behavior of PbC_6O_6 differs dramatically from these, as it displays only a single electron transfer event (as evidenced by the single oxidative and

reductive wave observed in Figure 5.8C). In the case of $\text{Li}_2\text{C}_6\text{O}_6$ the two primary redox events with half wave potentials at ca. 2.10 and 2.67 V vs Li/Li^+ have been previously reported and are the result of two sequential two electron transfers each of which converts a pair of ortho-ketones to the corresponding lithiated enediol.³⁶⁻³⁸ The voltametric response of NiC_6O_6 is similar in aspect to that of $\text{Li}_2\text{C}_6\text{O}_6$, albeit with significant broadening of the redox features. Whereas the lithium complex displays two distinct electron transfer events, the nickel complex displays three distinct redox processes with half wave potentials at ca. 2.20, 2.65 and 3.18 V vs Li/Li^+ respectively. While these events are likely due to a similar sequential lithiation as that observed in $\text{Li}_2\text{C}_6\text{O}_6$ the significant peak broadening makes a complete assignment of individual redox events difficult.

In contrast to both $\text{Li}_2\text{C}_6\text{O}_6$ and NiC_6O_6 , PbC_6O_6 displays a single broad redox feature within the same cycling window. This single redox feature, which possesses a half wave potential of ca. 1.98 V vs Li/Li^+ , displays a very high first cycle cathodic current density, which decreases in intensity and become significantly more stable on subsequent cycles. Interestingly, the peak current density achieved during the initial reductive sweep of PbC_6O_6 electrodes appears nearly double that of either lithium or nickel rhodizonate. The high redox activity of this first reductive sweep is tentatively assigned to the utilization of all four available ketones, with later cycles employing only two of these ketones for repeated intercalation and deintercalation.

In addition to its different electrochemical characteristics, the stabilities of the redox processes occurring in PbC_6O_6 are significantly greater than those associated with either $\text{Li}_2\text{C}_6\text{O}_6$, or NiC_6O_6 . Figure 5.8d displays the peak current densities associated with oxidative waves at 2.25 V, 2.4 V, and 2.4 V vs Li/Li^+ for $\text{Li}_2\text{C}_6\text{O}_6$, NiC_6O_6 and PbC_6O_6 respectively. While both $\text{Li}_2\text{C}_6\text{O}_6$, and NiC_6O_6 , display a rapid fade of their peak current densities with cycling, PbC_6O_6 displays comparatively stable peak current densities even after twenty cycles. This suggests that, while

PbC₆O₆ may not be utilizing all four available ketones for redox, it does not suffer the same electrochemically driven deactivation that affects the lithium and nickel analogues.

While the above voltammetry was collected between 1.5 and 3.5 V vs Li/Li⁺ (to avoid electrochemical decomposition of the rhodizonate ion or the electrolyte) cycling within a wider voltage window was also conducted with the interest of accessing redox couples associated with the metal (particularly the Ni²⁺/Ni³⁺/Ni⁴⁺ couples at potentials greater than 4 V). Figure 5.9 displays voltammograms collected from each material when cycled between 0.5 and 4.5 V vs Li/Li⁺. Despite this increased cycling window, none of the rhodizonates explored here display significant oxidative waves that might indicate additional contributions from transition metal redox. Interestingly, however, while Li₂C₆O₆ does not display additional redox features, both NiC₆O₆ and PbC₆O₆ exhibit additional reductive features below 1.5 V vs Li/Li⁺. These features were found to be irreversible and are likely associated with the formation and subsequent lithiation of zero-valent metal. Regardless of the origins of these additional redox features, their poor stability precludes reversible Li⁺ storage within this voltage window, and subsequent electrochemical measurements were conducted within the 1.5 to 3.5 V cycling window.

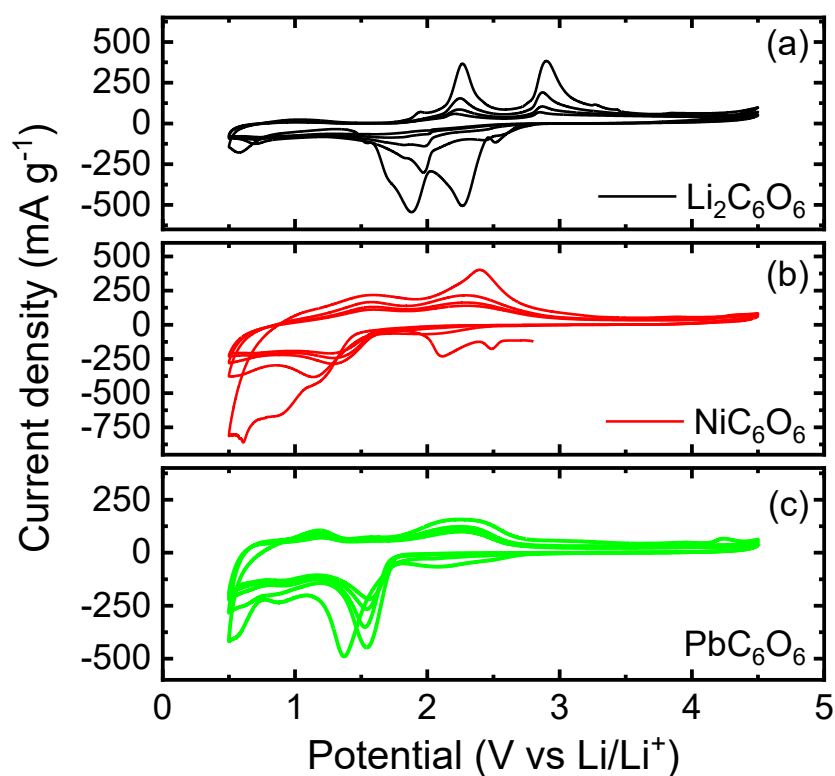


Figure 5.9. First five voltametric traces obtained from a) lithium b) nickel and c) lead rhodizonates when cycled between 0.5 and 4.5 V vs Li/Li⁺.

Table 5.3. Stoichiometries, molecular weights, and theoretical gravimetric capacities for target M-rhodizonate complexes.

Stoichiometry	MW (g mol ⁻¹)	Specific Capacity (mA h g ⁻¹)
Li ₂ C ₆ O ₆	181.94	589.2
NiC ₆ O ₆	226.75	472.8
PbC ₆ O ₆	375.26	285.7

To further characterize the electrochemical stabilities of the different metal rhodizonate material, electrodes were cycled galvanostatically, the results of which are displayed in Figure 5.10. Figure 5.10a displays the first two galvanostatic cycles obtained for each material, cycling at a C/10 rate calculated from the theoretical capacity of each material (Table 5.3). The dotted lines

in each figure demarcate the initial charge/discharge cycle. Each material displays clear redox plateaus associated with the intercalation and deintercalation of Li^+ at potentials very near those obtained voltammetrically. The computed differential capacity traces (obtained from the second charge/discharge cycle, Figure 5.10b) obtained for each material support the correlation between galvanostatic cycling and voltammetry, demonstrating that the potentials associated with intercalation and deintercalation are consistent between the two measurements.

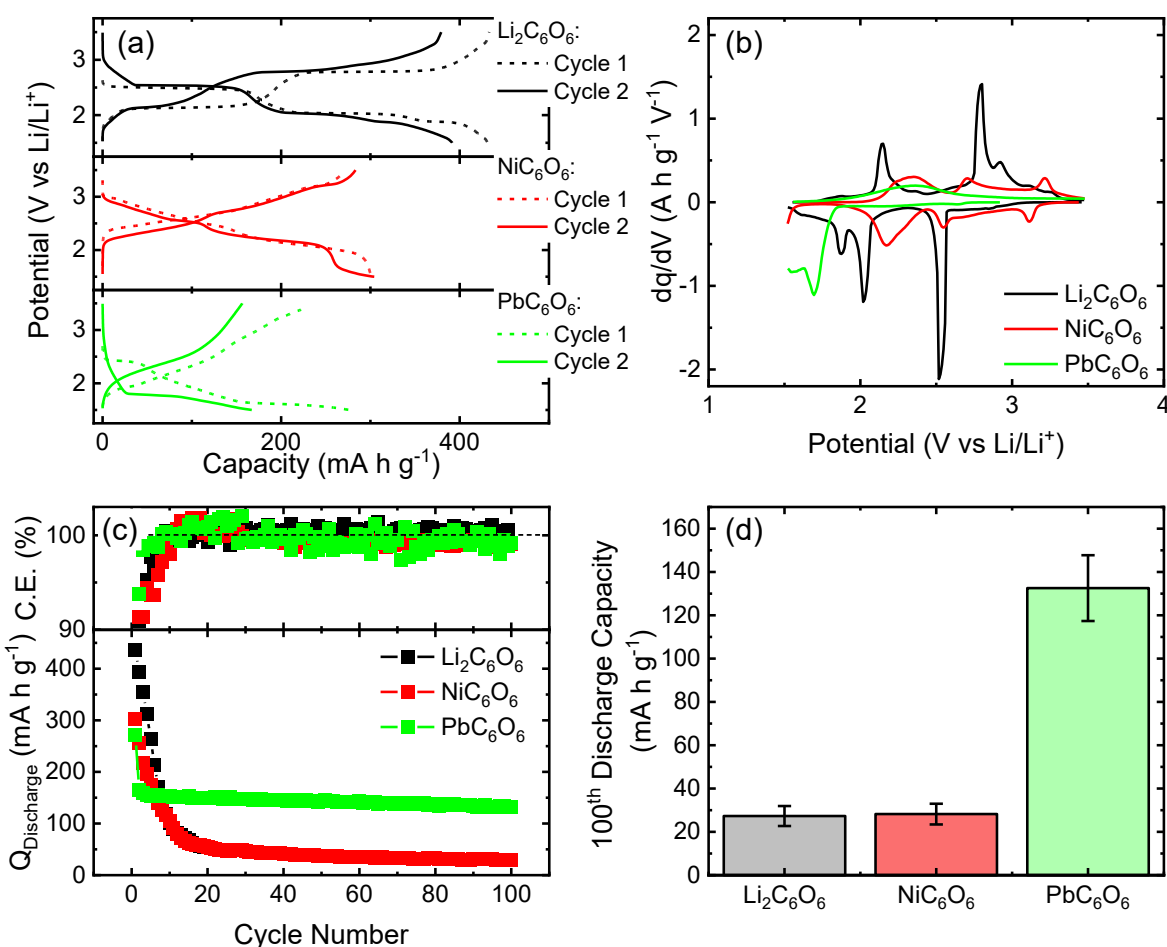


Figure 5.10. a) Galvanostatic charge/discharge profiles collected from the second cycle of lithium (black trace) nickel (red trace) and lead (green trace) rhodizonate electrodes. b) Differential capacity traces computed from the charge/discharge profiles presented in (a). c) Discharge capacities and Coulombic efficiencies collected from extended cycling of lithium, nickel, and lead rhodizonate electrodes. d) Discharge capacities obtained from each electrode after 100 charge/discharge cycles.

Figure 5.10c presents the capacity retention and Coulombic efficiencies of $\text{Li}_2\text{C}_6\text{O}_6$, NiC_6O_6 , and PbC_6O_6 as functions of cycle number. Both lithium and nickel rhodizonates initially display capacities very near their theoretical values. This capacity is short lived, however, as these electrodes show severe capacity fade within the first 15 cycles, broadly reflecting the progressive electrochemical deactivation obtained voltammetrically. The coulombic efficiency associated with these cycles is also poor, starting around 90% and approaching 100% as the capacity fades to only the capacitive charge storage associated with the conductive carbon additive. In contrast PbC_6O_6 displays a capacity near its theoretical on the first discharge which does not fully decay on subsequent cycles. While all further cycles display capacities approximately half that of theoretical the stability of this capacity is significantly greater than that of either nickel or lithium rhodizonate. Importantly, this initially high capacity at early cycles aligns directly with the results obtained voltammetrically wherein PbC_6O_6 displays higher redox activity during its initial delithiation followed by comparatively stable cycling during subsequent lithiation and delithiation. This cycling stability is reflected in Figure 5.10d which displays the discharge capacity associated with the 100th cycle of each electrode. As demonstrated in 4C both lithium and nickel rhodizonates display effectively no capacity at 100 cycles whereas lead rhodizonate displays a capacity roughly equivalent to its 2nd discharge capacity after 100 cycles.

Table 5.4. Atomic compositions and computed stoichiometries obtained from $\text{Li}_2\text{C}_6\text{O}_6$, NiC_6O_6 , and PbC_6O_6 electrodes in both their lithiated and subsequently delithiated states following one, five, and ten cycles. ΔLi was calculated by taking the difference in lithium stoichiometry between the lithiated and delithiated states.

	Active Material	[Li] Charged (at. %)	[Li] Discharged (at. %)	Stoichiometry		
				Charged	Discharged	ΔLi
Pristine Electrode	$\text{Li}_2\text{C}_6\text{O}_6$	12.4 ± 0.4	4.7 ± 0.2	$\text{Li}_{6.1}\text{C}_6\text{O}_6$	$\text{Li}_{2.1}\text{C}_6\text{O}_6$	4.0
	NiC_6O_6	6.4 ± 0.8	0.0	$\text{Li}_{3.9}\text{NiC}_6\text{O}_6$	$\text{Ni}_{1.0}\text{C}_6\text{O}_6$	3.9
	PbC_6O_6	4.5 ± 0.6	0.0	$\text{Li}_{4.8}\text{PbC}_6\text{O}_6$	$\text{Pb}_{1.0}\text{C}_6\text{O}_6$	4.8
Cycle 1	$\text{Li}_2\text{C}_6\text{O}_6$	--	--	$\text{Li}_{6.1}\text{C}_6\text{O}_6$	--	--
	NiC_6O_6	6.4 ± 0.8	1.65	$\text{Li}_{3.9}\text{NiC}_6\text{O}_6$	$\text{Li}_{1.2}\text{NiC}_6\text{O}_6$	2.7
	PbC_6O_6	4.5 ± 0.6	2.05	$\text{Li}_{4.8}\text{PbC}_6\text{O}_6$	$\text{Li}_{2.1}\text{PbC}_6\text{O}_6$	2.7
Cycle 5	$\text{Li}_2\text{C}_6\text{O}_6$	6.7	4.5	$\text{Li}_{3.1}\text{C}_6\text{O}_6$	$\text{Li}_{2.0}\text{C}_6\text{O}_6$	1.1
	NiC_6O_6	4.1	2.3	$\text{Li}_{3.3}\text{NiC}_6\text{O}_6$	$\text{Li}_{1.7}\text{NiC}_6\text{O}_6$	1.6
	PbC_6O_6	4.4	1.9	$\text{Li}_{4.4}\text{PbC}_6\text{O}_6$	$\text{Li}_{2.0}\text{PbC}_6\text{O}_6$	2.4
Cycle 10	$\text{Li}_2\text{C}_6\text{O}_6$	--	--	$\text{Li}_x\text{C}_6\text{O}_6$	$\text{Li}_x\text{C}_6\text{O}_6$	--
	NiC_6O_6	--	--	$\text{Li}_y\text{NiC}_6\text{O}_6$	$\text{Li}_y\text{NiC}_6\text{O}_6$	--
	PbC_6O_6	4.7	2.0	$\text{Li}_{4.3}\text{PbC}_6\text{O}_6$	$\text{Li}_{2.0}\text{PbC}_6\text{O}_6$	2.3

To verify that the capacities presented in Figure 5.10 are associated with lithiation of the redox active material, rather than any capacitive processes which would yield additional capacity, ICP-OES was conducted on lithiated rhodizonate electrodes. Electrodes were discharged at constant current (C/10) until they registered a potential of 1.5 V vs Li/Li^+ , at which point the cells were removed from the potentiostat and the cathodes recovered, rinsed and submitted for elemental analysis. Samples were also investigated in both their charged and discharged states after one, five, and ten charge/discharge cycles to elucidate any permanent retention of Li^+ within the rhodizonate structures on repeated cycling. The results of this analysis are presented in Table 5.4 and suggest that upon lithiation the stoichiometry of the as obtained material matches the expected result of intercalating 4 Li^+ per formula unit, likely occupying the four available ketones of the rhodizonate anion. This supports the conclusion that the capacity observed from galvanostatic measurements is due to the intercalation of Li^+ rather than capacitive charge transfer at electrode interfaces.

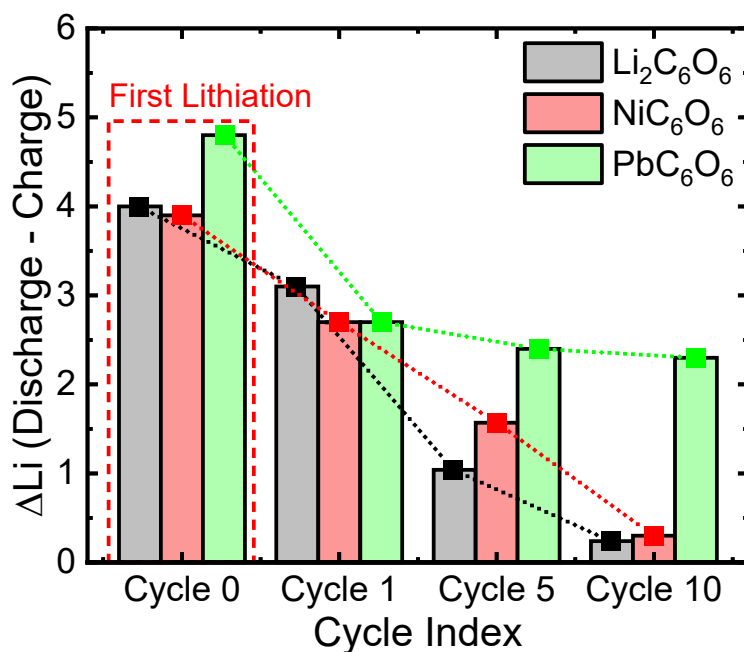


Figure 5.11. Change in lithium stoichiometry between subsequent discharge and charge cycles for Li (grey), Ni (red) and Pb (green) rhodizonates. ΔLi was calculated by taking the difference between the lithium stoichiometries in the discharged and subsequent charged states for each electrode. Cycle 0 provides the change in lithium stoichiometry for each sample during the first discharge cycle (wherein ΔLi was computed by subtracting the lithium content of the pristine electrode from electrodes collected after discharging to 1.5 V).

Additionally, the ratio of Li^+ to M(II) obtained from elemental analysis was used to compute the total change in the lithium stoichiometry for each electrode during repeated lithiation and delithiation (Figure 5.11). Interestingly, while all electrodes display the expected 4 Li^+ intercalation during their first discharge (yielding an overall ΔLi of ca. 4) the amount of lithium reversibly intercalated/deintercalated for each electrode decreases on subsequent charge/discharge cycles. Both PbC_6O_6 and NiC_6O_6 exhibit delithiation of ca. 2.7 Li^+ after the first discharge indicating significant irreversible retention of Li^+ within the structure following the initial discharge (need to rerun cycle 1 $\text{Li}_2\text{C}_6\text{O}_6$, also waiting on cycle 10 Li and Ni samples). Similarly, by cycle 5 both lithium and nickel rhodizonates can only reversibly shuttle 1.1 and 1.6 Li^+

respectively, signifying a dramatic decrease in the capability of these electrodes to uptake and release lithium. In contrast, lead rhodizonate exhibits reversible intercalation and deintercalation of ca. 2 Li^+ even at 10 cycles. This aligns well with the electrochemical behavior obtained from PbC_6O_6 electrodes and supports the conclusion that the available capacity is due to stable redox active sites that are not available in lithium or nickel rhodizonate after repeated lithiation and delithiation.

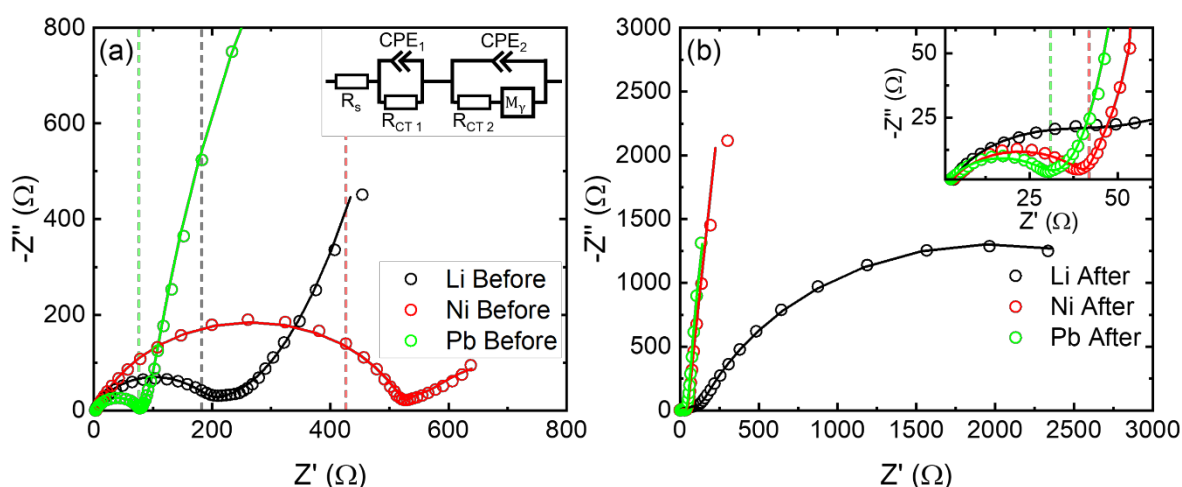


Figure 5.12. Electrochemical impedance spectra collected from MC_6O_6 electrodes both a) before and b) after 20 galvanostatic charge/discharge cycles. Open circles denote experimental data, while solid lines depict fits to experiment using the equivalent circuit depicted as an inset in a). Vertical dashed lines mark a constant frequency of 25 Hz for each trace.

To understand the impact the cation has on the charge transfer characteristics of metal rhodizonate electrodes, electrochemical impedance spectroscopy (EIS) was conducted on electrodes at their open circuit potentials both before and after 20 galvanostatic charge discharge cycles, the results of which are displayed in Figures 5.12a and b respectively. EIS spectra were fit to an equivalent circuit (Inset of Figure 5.12a) to extract charge transfer impedances. The equivalent circuit in this case consists of two charge transfer regions, one associated with the

$\text{Li}_m/\text{electrolyte}$ interface and one associated with the $\text{MC}_6\text{O}_6/\text{electrolyte}$ interface. Dotted vertical lines in each figure demarcate a frequency of 25 Hz.

Prior to cycling all electrodes exhibit charge transfer impedances below 600 Ω , similar to previous reports on rhodizonate based electrodes.^{36, 69} Interestingly, PbC_6O_6 displays the lowest overall charge transfer impedance among the electrodes tested followed by lithium and nickel rhodizonates respectively. The low charge transfer impedance may be due, in part to the large size of Pb^{II} within the rhodizonate structure, which may yield larger channels for lithium transport within the structure. Following galvanostatic cycling lead rhodizonate again displays the lowest overall charge transfer impedance, followed by nickel and then lithium rhodizonate. As reported elsewhere the impedance of $\text{Li}_2\text{C}_6\text{O}_6$ increases dramatically upon cycling do to a crystallographic deactivation that renders the electrode effectively inert.³⁸ While the charge transfer associated with $\text{Li}_2\text{C}_6\text{O}_6$ increases by nearly two orders of magnitude, the charge transfer impedances associated with nickel and lead rhodizonates both decrease upon cycling. The low charge transfer impedances associated with these electrode materials may be the result of improved electronic conductivity relative to $\text{Li}_2\text{C}_6\text{O}_6$.

5.4. Discussion

Taken together, these results demonstrate that the structures and electrochemical Li^+ storage behaviors of metal rhodizonates are directly influenced by the identity of the cation. In particular, we showed that both lithium and nickel rhodizonates (as well as other first row transition metal rhodizonates) display poor electrochemical stability, evidenced by rapid capacity fade during the first 15 charge/discharge cycles. Lead rhodizonate, however, exhibits good electrochemical stability relative to its nickel and lithium counterparts. Previous work has already discussed the origin of the poor electrochemical stability of $\text{Li}_2\text{C}_6\text{O}_6$ and has identified the

coordination of Li^+ in low energy tetrahedral binding sites during lithiation as the primary mode of active material failure as described briefly in section 5.1. The results presented here are in broad agreement with this conclusion. In particular, elemental analysis of cycled $\text{Li}_2\text{C}_6\text{O}_6$ electrodes demonstrates the irreversible retention of Li^+ with cycling, signaling the progressive decomposition of the active material.

The deleterious impact of tetrahedral lithium coordination on the electrochemical stability of rhodizonate structures helps to explain the equivalently poor electrochemical stability of NiC_6O_6 . Within this structure Ni^{2+} coordinates in much the same way that Li^+ does in $\text{Li}_2\text{C}_6\text{O}_6$ by occupying square planar sites within the rhodizonate plane. This square planar coordination leaves the interlayer space free to coordinate Li^+ in its energetically preferred tetrahedral geometry. Thus, during discharge incoming Li^+ occupies the interlayer space likely resulting in similar crystallographic changes to the active material (a lithiation induced lattice contraction) yielding irreversible Li^+ retention within the structure. This argument is supported by elemental analysis which shows that on repeated cycling nickel rhodizonate has irreversibly retained Li^+ which is no longer redox active.

In contrast to both lithium and nickel rhodizonates, PbC_6O_6 coordinates Pb^{2+} within the interlayer space, rather than within the rhodizonate layer (likely due to the large size of the Pb^{2+} cation). This occupation of interlayer sites allows Pb^{2+} to occlude domains that would otherwise irreversibly coordinate Li^+ in tight tetrahedral binding sites. Similarly, it may also serve to suppress the contraction of the crystal lattice during lithiation. While it is difficult to unambiguously deconvolute the impact of these two different stabilization mechanisms, the electrochemical behavior and elemental analysis of PbC_6O_6 electrodes provide insights into the most likely origin of the observed cycling stability. Importantly, during its initial lithiation PbC_6O_6 delivers a

capacity very near theoretical, whereas on subsequent charging and discharging cycles it yields roughly half that. Similarly, the results of elemental analysis suggest that the electrode irreversibly intercalates two Li^+ per formula unit yielding an overall stoichiometry of $\text{Li}_2\text{PbC}_6\text{O}_6$. Subsequent charging and discharging is then accomplished between this stoichiometry and the fully lithiated stoichiometry of $\text{Li}_4\text{PbC}_6\text{O}_6$ (Table 5.4). This suggests that Li^+ is still binding irreversibly within the structure, but that the occlusion of some of the preferred tetrahedral binding sites by Pb^{2+} likely forces Li^+ to coordinate to the energetically less favorable, but electrochemically more accessible, square planar sites within the rhodizonate layers. This analysis is supported by the proposed crystal structure of PbC_6O_6 which maintains two available tetrahedral binding sites per formula unit within the interlayer space as well as two available square planar binding sites per formula unit within the rhodizonate layers.

The results presented herein demonstrate that the occlusion of tetrahedral Li^+ binding sites by a stable chelating cation can facilitate greater electrochemical stability. Likewise, they show that the transition from monovalent to divalent cations only significantly impacts the electrochemical stability if this transition changes the number of available tetrahedral sites for Li^+ binding in the rhodizonate molecule. The occlusion of these sites can be accomplished by chelation of the rhodizonate dianion to larger cations, whose physical size precludes them from occupying electrochemically reversible square planar sites. This suggests that coordination of rhodizonate to a large monovalent cation such as Rb^+ may block even more of these deleterious Li^+ sites within the interlayer space. Similarly, the exploration of other binding environments for the rhodizonate dianion, including those with organic cationic species, may prove useful in developing a stable redox active organic cathode material.

5.5. Conclusions

Herein we explore the electrochemical behavior of a series of metal rhodizonate complexes with the aim of developing an improved understanding of the factors that dictate the electrochemical stability of this class of materials. We demonstrate that highly crystalline metal rhodizonates can be synthesized through a straightforward aqueous precipitation reaction similar to that used to obtain $\text{Li}_2\text{C}_6\text{O}_6$. We also show that the identity of the cation strongly dictates both the voltametric and galvanostatic responses of metal rhodizonate electrodes. In particular, while $\text{Li}_2\text{C}_6\text{O}_6$ and NiC_6O_6 display poor overall cycling stability, PbC_6O_6 can accommodate stable intercalation and deintercalation during extended cycling, albeit at roughly half of its theoretical capacity. Further we show, unambiguously, that the available capacity obtained from galvanostatic cycling is due to intercalation of Li^+ into the rhodizonate structure rather than capacitive or pseudocapacitive charge storage.

Importantly we demonstrate the impact cation coordination has on the electrochemical behavior of these materials. The occupation, or lack thereof, of low energy interplanar tetrahedral sites largely dictates the overall reversibility and capacity retention of electrodes containing rhodizonate ions. Specifically, both lithium and nickel rhodizonates leave these low energy sites open, as the structural cation occupies a pseudo square planar geometry within the rhodizonate layer. This allows the irreversible coordination of Li^+ within the interplanar space leading to the rapid deactivation of the redox active material. Per contra, Pb(II) occupies a significant number of these tetrahedral sites, forcing Li^+ to occupy the less energetically favorable, but electrochemically more reversible square planar sites within the rhodizonate layer, thereby affording stable electrochemical cycling. This result suggests that careful tuning of the rhodizonate coordination

environment may be sufficient to produce stable, high-capacity organic, or organometallic electrode materials for reversible Li⁺ based energy storage.

5.6. References:

- (1) Goodenough, J. B.; Park, K. S., The Li-Ion Rechargeable Battery: A Perspective. *J Am Chem Soc* **2013**, *135* (4), 1167-1176.
- (2) Manthiram, A., Materials Challenges and Opportunities of Lithium Ion Batteries. *Journal of Physical Chemistry Letters* **2011**, *2* (3), 176-184.
- (3) Manthiram, A., An Outlook on Lithium Ion Battery Technology. *Acs Central Sci* **2017**, *3* (10), 1063-1069.
- (4) Chen, T.; Jin, Y.; Lv, H.; Yang, A.; Liu, M.; Chen, B.; Xie, Y.; Chen, Q., Applications of Lithium-Ion Batteries in Grid-Scale Energy Storage Systems. *Transactions of Tianjin University* **2020**, *26* (3), 208-217.
- (5) Schmuck, R.; Wagner, R.; Hörpel, G.; Placke, T.; Winter, M., Performance and cost of materials for lithium-based rechargeable automotive batteries. *Nature Energy* **2018**, *3* (4), 267-278.
- (6) Dunn, B.; Kamath, H.; Tarascon, J. M., Electrical Energy Storage for the Grid: A Battery of Choices. *Science* **2011**, *334* (6058), 928-935.
- (7) Nitta, N.; Wu, F.; Lee, J. T.; Yushin, G., Li-ion battery materials: present and future. *Materials Today* **2015**, *18* (5), 252-264.
- (8) Manthiram, A., A reflection on lithium-ion battery cathode chemistry. *Nature Communications* **2020**, *11* (1), 1550.

- (9) Gannett, C. N.; Melecio-Zambrano, L.; Theibault, M. J.; Peterson, B. M.; Fors, B. P.; Abruña, H. D., Organic electrode materials for fast-rate, high-power battery applications. *Materials Reports: Energy* **2021**, *1* (1), 100008.
- (10) Choi, J. W.; Aurbach, D., Promise and reality of post-lithium-ion batteries with high energy densities. *Nat Rev Mater* **2016**, *1* (4).
- (11) Schon, T. B.; McAllister, B. T.; Li, P.-F.; Seferos, D. S., The rise of organic electrode materials for energy storage. *Chem Soc Rev* **2016**, *45* (22), 6345-6404.
- (12) Lu, Y.; Chen, J., Prospects of organic electrode materials for practical lithium batteries. *Nature Reviews Chemistry* **2020**, *4* (3), 127-142.
- (13) Liang, Y.; Yao, Y., Positioning Organic Electrode Materials in the Battery Landscape. *Joule* **2018**, *2* (9), 1690-1706.
- (14) Yin, X.; Sarkar, S.; Shi, S.; Huang, Q.-A.; Zhao, H.; Yan, L.; Zhao, Y.; Zhang, J., Recent Progress in Advanced Organic Electrode Materials for Sodium-Ion Batteries: Synthesis, Mechanisms, Challenges and Perspectives. *Advanced Functional Materials* **2020**, *30* (11), 1908445.
- (15) Lu, Y.; Hou, X.; Miao, L.; Li, L.; Shi, R.; Liu, L.; Chen, J., Cyclohexanone with Ultrahigh Capacity as Cathode Materials for Lithium-Ion Batteries. *Angewandte Chemie International Edition* **2019**, *58* (21), 7020-7024.
- (16) Chen, Z.; Su, H.; Sun, P.; Bai, P.; Yang, J.; Li, M.; Deng, Y.; Liu, Y.; Geng, Y.; Xu, Y., A nitroaromatic cathode with an ultrahigh energy density based on six-electron reaction per nitro group for lithium batteries. *Proceedings of the National Academy of Sciences* **2022**, *119* (6), e2116775119.

- (17) Liang, Y.; Tao, Z.; Chen, J., Organic Electrode Materials for Rechargeable Lithium Batteries. *Adv Energy Mater* **2012**, 2 (7), 742-769.
- (18) Wu, X.; Ma, J.; Hu, Y.-S.; Li, H.; Chen, L., Nano-sized carboxylates as anode materials for rechargeable lithium-ion batteries. *Journal of Energy Chemistry* **2014**, 23 (3), 269-273.
- (19) Sun, T.; Li, Z.-J.; Yang, X.; Wang, S.; Zhu, Y.-H.; Zhang, X.-B., Imine-rich poly (o-phenylenediamine) as high-capacity trifunctional organic electrode for alkali-ion batteries. *CCS Chemistry* **2019**, 1 (4), 365-372.
- (20) Luo, C.; Borodin, O.; Ji, X.; Hou, S.; Gaskell, K. J.; Fan, X.; Chen, J.; Deng, T.; Wang, R.; Jiang, J.; Wang, C., Azo compounds as a family of organic electrode materials for alkali-ion batteries. *Proceedings of the National Academy of Sciences* **2018**, 115 (9), 2004.
- (21) Shadike, Z.; Tan, S.; Wang, Q.-C.; Lin, R.; Hu, E.; Qu, D.; Yang, X.-Q., Review on organosulfur materials for rechargeable lithium batteries. *Materials Horizons* **2021**, 8 (2), 471-500.
- (22) Wang, D.-Y.; Guo, W.; Fu, Y., Organosulfides: An Emerging Class of Cathode Materials for Rechargeable Lithium Batteries. *Accounts of Chemical Research* **2019**, 52 (8), 2290-2300.
- (23) Wu, Y.; Zeng, R.; Nan, J.; Shu, D.; Qiu, Y.; Chou, S.-L., Quinone Electrode Materials for Rechargeable Lithium/Sodium Ion Batteries. *Adv Energy Mater* **2017**, 7 (24), 1700278.

- (24) Bitenc, J.; Pavčnik, T.; Košir, U.; Pirnat, K., Quinone Based Materials as Renewable High Energy Density Cathode Materials for Rechargeable Magnesium Batteries. *Materials* **2020**, *13* (3).
- (25) Guin, P. S.; Das, S.; Mandal, P. C., Electrochemical Reduction of Quinones in Different Media: A Review. *International Journal of Electrochemistry* **2011**, *2011*, 816202.
- (26) Bhosale, M. E.; Chae, S.; Kim, J. M.; Choi, J.-Y., Organic small molecules and polymers as an electrode material for rechargeable lithium ion batteries. *J Mater Chem A* **2018**, *6* (41), 19885-19911.
- (27) Zhao, Q.; Wang, J.; Lu, Y.; Li, Y.; Liang, G.; Chen, J., Oxocarbon Salts for Fast Rechargeable Batteries. *Angewandte Chemie International Edition* **2016**, *55* (40), 12528-12532.
- (28) Belanger, R. L.; Commarieu, B.; Paoletta, A.; Daigle, J.-C.; Bessette, S.; Vijh, A.; Claverie, J. P.; Zaghib, K., Diffusion Control of Organic Cathode Materials in Lithium Metal Battery. *Sci Rep-Uk* **2019**, *9* (1), 1213.
- (29) Petronico, A.; Bassett, K. L.; Nicolau, B. G.; Gewirth, A. A.; Nuzzo, R. G., Toward a Four-Electron Redox Quinone Polymer for High Capacity Lithium Ion Storage. *Adv Energy Mater* **2018**, *8* (5), 1700960.
- (30) Song, Z.; Qian, Y.; Gordin, M. L.; Tang, D.; Xu, T.; Otani, M.; Zhan, H.; Zhou, H.; Wang, D., Polyanthraquinone as a Reliable Organic Electrode for Stable and Fast Lithium Storage. *Angewandte Chemie International Edition* **2015**, *54* (47), 13947-13951.
- (31) Sieuw, L.; Ernould, B.; Gohy, J.-F.; Vlad, A., On the improved electrochemistry of hybrid conducting-redox polymer electrodes. *Sci Rep-Uk* **2017**, *7* (1), 4847.

- (32) Kapaev, R. R.; Zhidkov, I. S.; Kurmaev, E. Z.; Stevenson, K. J.; Troshin, P. A., Hexaazatriphenylene-based polymer cathode for fast and stable lithium-, sodium- and potassium-ion batteries. *J Mater Chem A* **2019**, 7 (39), 22596-22603.
- (33) Chen, H.; Armand, M.; Courty, M.; Jiang, M.; Grey, C. P.; Dolhem, F.; Tarascon, J.-M.; Poizot, P., Lithium Salt of Tetrahydroxybenzoquinone: Toward the Development of a Sustainable Li-Ion Battery. *J Am Chem Soc* **2009**, 131 (25), 8984-8988.
- (34) Doane, L. M.; Fatiadi, A. J., Electrochemical oxidation of several oxocarbon salts in N,N-dimethylformamide. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry* **1982**, 135 (2), 193-209.
- (35) Zhang, Z.; Ma, J.; Zhang, R.; Rajagopalan, R.; Tang, Y.; Ren, Y.; Wang, H., Oxocarbons Electrode Materials for Alkali Ion Batteries: Challenges, Strategies and Development. *Batteries & Supercaps* **2021**, 4 (12), 1791-1802.
- (36) Chen, H.; Armand, M.; Demailly, G.; Dolhem, F.; Poizot, P.; Tarascon, J.-M., From Biomass to a Renewable LiXC₆O₆ Organic Electrode for Sustainable Li-Ion Batteries. *ChemSusChem* **2008**, 1 (4), 348-355.
- (37) Kim, H.; Seo, D.-H.; Yoon, G.; Goddard, W. A.; Lee, Y. S.; Yoon, W.-S.; Kang, K., The Reaction Mechanism and Capacity Degradation Model in Lithium Insertion Organic Cathodes, Li₂C₆O₆, Using Combined Experimental and First Principle Studies. *The Journal of Physical Chemistry Letters* **2014**, 5 (17), 3086-3092.
- (38) Madsen, K. E.; Shin, M.; Gewirth, A. A., Pressure-Dependent Electrochemical Behavior of Di-Lithium Rhodizonate Cathodes. *Chem Mater* **2021**, 33 (14), 5738-5747.

- (39) Wu, D.; Li, H.; Li, R.; Hu, Y.; Hu, X., In situ growth of copper rhodizonate complexes on reduced graphene oxide for high-performance organic lithium-ion batteries. *Chem Commun* **2018**, 54 (81), 11415-11418.
- (40) Lu, C.; Dong, C.; Wu, H.; Ni, D.; Sun, W.; Wang, Z.; Sun, K., Achieving high capacity hybrid-cathode FeF₃@Li₂C₆O₆/rGO based on morphology control synthesis and interface engineering. *Chem Commun* **2018**, 54 (26), 3235-3238.
- (41) Fatiadi, A. J., Synthesis of 1,3-(dicyanomethylene)croconate salts. New bond-delocalized dianion, "Croconate Violet". *J Am Chem Soc* **1978**, 100 (8), 2586-2587.
- (42) Milán-Garcés, E. A.; Georgopoulos, S. L.; Barros Santos, P.; Sato, F.; Sant'Ana, A. C.; Andrade, G. F. S.; de Oliveira, L. F. C., Probing ring contraction and decarboxylation of Rhodizonate and the influence of Cu (II) using surface-enhanced Raman Scattering. *Journal of Raman Spectroscopy* **2020**, 51 (2), 256-263.
- (43) Schultz, Z. D.; Feng, Z. V.; Biggin, M. E.; Gewirth, A. A., Vibrational Spectroscopic and Mass Spectrometric Studies of the Interaction of Bis(3-sulfopropyl)-disulfide with Cu Surfaces. *J Electrochem Soc* **2006**, 153 (2), C97.
- (44) Perdew, J. P.; Burke, K.; Ernzerhof, M., Generalized Gradient Approximation Made Simple. *Phys Rev Lett* **1996**, 77 (18), 3865-3868.
- (45) Kresse, G.; Furthmüller, J., Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science* **1996**, 6 (1), 15-50.
- (46) Dion, M.; Rydberg, H.; Schröder, E.; Langreth, D. C.; Lundqvist, B. I., Van der Waals Density Functional for General Geometries. *Phys Rev Lett* **2004**, 92 (24), 246401.

- (47) Klimeš, J.; Bowler, D. R.; Michaelides, A., Chemical accuracy for the van der Waals density functional. *Journal of Physics: Condensed Matter* **2009**, 22 (2), 022201.
- (48) Klimeš, J.; Bowler, D. R.; Michaelides, A., Van der Waals density functionals applied to solids. *Phys Rev B* **2011**, 83 (19), 195131.
- (49) Momma, K.; Izumi, F., VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data. *J Appl Crystallogr* **2011**, 44 (6), 1272-1276.
- (50) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery Jr., J. A.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J. *Gaussian 16 Rev. C.01*, Wallingford, CT, 2016.
- (51) Assefa, M. K.; Devera, J. L.; Brathwaite, A. D.; Mosley, J. D.; Duncan, M. A., Vibrational scaling factors for transition metal carbonyls. *Chemical Physics Letters* **2015**, 640, 175-179.

- (52) Hay, P. J.; Wadt, W. R., Ab initio effective core potentials for molecular calculations. Potentials for K to Au including the outermost core orbitals. *The Journal of Chemical Physics* **1985**, 82 (1), 299-310.
- (53) Schäfer, A.; Horn, H.; Ahlrichs, R., Fully optimized contracted Gaussian basis sets for atoms Li to Kr. *The Journal of Chemical Physics* **1992**, 97 (4), 2571-2577.
- (54) Yang, Y.; Weaver, M. N.; Merz, K. M., Assessment of the “6-31+G** + LANL2DZ” Mixed Basis Set Coupled with Density Functional Theory Methods and the Effective Core Potential: Prediction of Heats of Formation and Ionization Potentials for First-Row-Transition-Metal Complexes. *The Journal of Physical Chemistry A* **2009**, 113 (36), 9843-9851.
- (55) O'Boyle, N. M.; Tenderholt, A. L.; Langner, K. M., cclib: A library for package-independent computational chemistry algorithms. *Journal of Computational Chemistry* **2008**, 29 (5), 839-845.
- (56) Charney, E.; Becker, E. D., Molecular Vibrations of Quinones. II. Infrared Spectra (Solution and Vapor) of p-Benzoquinone and p-Benzoquinone-d₄. *The Journal of Chemical Physics* **1965**, 42 (3), 910-913.
- (57) Campos-vallette, M.; Figueroa, K. A.; Puebla, C., Molecular Vibrational Constants and Chemical Bonding in the Cyclic Oxocarbon Dianions C_nO_{n-2} (n=3, 4 and 5). *Spectroscopy Letters* **1988**, 21 (4), 303-312.
- (58) Stammreich, H.; Sans, T. T., Molecular Vibrations of Quinones. IV. Raman Spectra of p-Benzoquinone and Its Centrosymmetrically Substituted Isotopic Derivatives and Assignment of Observed Frequencies. *The Journal of Chemical Physics* **1965**, 42 (3), 920-931.

- (59) Preisler, P. W.; Berger, L., Preparation of Tetrahydroxyquinone and Rhodizonic Acid Salts from the Product of the Oxidation of Inositol with Nitric Acid*. *J Am Chem Soc* **1942**, *64* (1), 67-69.
- (60) de Souza, H.; Faria, L. F. O.; Scaldini, F. M.; Diniz, R.; Edwards, H. G. M.; de Oliveira, L. F. C., Rhodizionate complexes with transition metal ions: A new vision from an old building block. *Journal of Molecular Structure* **2021**, *1238*, 130407.
- (61) Dinnebier, R. E.; Nuss, H.; Jansen, M., Disodium rhodizionate: a powder diffraction study. *Acta Crystallographica Section E* **2005**, *61* (10), m2148-m2150.
- (62) Aihara, J.-i., The Solid-State Spectra of Several Rhodizonates. *Bulletin of the Chemical Society of Japan* **1974**, *47* (11), 2899-2900.
- (63) Wang, X.; Zhang, P.; Tang, X.; Guan, J.; Lin, X.; Wang, Y.; Dong, X.; Yue, B.; Yan, J.; Li, K.; Zheng, H.; Mao, H.-k., Structure and Electrical Performance of Na₂C₆O₆ under High Pressure. *The Journal of Physical Chemistry C* **2019**, *123* (28), 17163-17169.
- (64) Braga, D.; Cojazzi, G.; Maini, L.; Grepioni, F., Reversible solid-state interconversion of rhodizonic acid H₂C₆O₆ into H₆C₆O₈ and the solid-state structure of the rhodizionate dianion C₆O₆²⁻ (aromatic or non-aromatic?). *New Journal of Chemistry* **2001**, *25* (10), 1221-1223.
- (65) Kobus, H.; Wainwright, K.; International, A.; Testing, A. S. f.; Materials, *An Update on the Use of the Sodium Rhodizionate Test for the Detection of Lead Originating from Firearm Discharges*. ASTM International: 1996.
- (66) Lebedeva, I. V.; Lebedev, A. V.; Popov, A. M.; Knizhnik, A. A., Comparison of performance of van der Waals-corrected exchange-correlation functionals for interlayer

- interaction in graphene and hexagonal boron nitride. *Computational Materials Science* **2017**, *128*, 45-58.
- (67) Hume-Rothery, W.; Raynor, G. V., Atomic and Ionic Radii.-I. Numerical Values and their Bearing on the Compressibilities of Metals. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science* **1938**, *26* (173), 129.
- (68) Bard, A. J.; Faulkner, L. R., Fundamentals and applications. *Electrochemical methods* **2001**, *2* (482), 580-632.
- (69) Lee, M.; Hong, J.; Lopez, J.; Sun, Y.; Feng, D.; Lim, K.; Chueh, W. C.; Toney, M. F.; Cui, Y.; Bao, Z., High-performance sodium–organic battery by realizing four-sodium storage in disodium rhodizonate. *Nature Energy* **2017**, *2* (11), 861-868.

Appendix A: Double Layer Model for Packing Density Approximation by XPS

To estimate the molecular packing density of organo-silanes on LMNO a simple double layer model was used to approximate the composition of the oxide surface. Within this model the surface is approximated by a semi-infinite, smooth substrate (LMNO) which is uniformly covered by an attenuating overlayer (silane) of known thickness. The model assumes that emitted photoelectrons are solely inelastically scattered such that they can be modelled effectively by the Beer-Lambert relation (eq. A.1).

$$I = I_0 \exp\left(\frac{-d}{\lambda \cos\theta}\right) \quad \text{eq. A.1}$$

Where I is the attenuated signal intensity, I_0 is the un-attenuated photoelectron signal, d is the thickness of the attenuating layer, λ is the inelastic mean free path, and θ is the detector angle measured normal to the substrate. The absolute intensity can then be normalized by the areal density of surface species to yield the following equivalence.

$$I_{area,n} = \frac{I_{0,n} \exp\left(\frac{-d_n}{\lambda_n \cos\theta}\right)}{\rho_{area,n}} \quad \text{eq. A.2}$$

$$\rho_{area,n} = \left(6.022 \times 10^{23} \left(\frac{\eta_n \rho_n}{MW_n}\right)\right)^{\frac{2}{3}} \quad \text{eq. A.3}$$

Where $\rho_{area,n}$ is the theoretical maximum two dimensional atomic density of species n , d_n is the length of the attenuating overlayer (the length of the silane molecule for the purpose of this investigation), $I_{0,n}$ is the unattenuated photoemission intensity of species n , and λ_n is the inelastic mean free path of species n at the characteristic photoemission energy. Eqn. A.2 presents the signal exhibited by a single monolayer of species n attenuated by an overlayer of a known thickness. The

theoretical atomic density of species n is computed from the bulk density as shown in eqn. A.3 where η_n is the number of atoms of species n per formula unit, ρ_n is the bulk density of species n , and MW_n is the molecular weight of species n . For this investigation the areal atomic density of Mn was calculated from the bulk properties of LMNO, and the areal atomic density of Si was approximated for a perfectly packed layer of silicic acid. This approximation represents the absolute highest packing density as limited by the radius of gyration of silicic acid molecules at the surface.

Under this approximation the theoretical maximum atomic surface density of Si and Mn atoms are 5.03×10^{14} atoms cm^{-2} and 7.85×10^{14} atoms cm^{-2} respectively. Assuming the composition of the modified surface is a semi-infinite LMNO substrate comprising a single continuous layer attenuated by a Si containing overlayer the signal contribution from each species can be determined by integration over the thickness of each layer. In this frame of reference $z = 0$ refers to the top of the silane monolayer exposed to vacuum, $z = d_{Si}$ is the depth of the siloxane head which delineates the organic silane layer from the inorganic LMNO layer. Here we approximate this structure as a perfect Gibbs dividing surface with no interdiffusion between the silane and the oxide substrate. Thus, the theoretical total photoelectron yield from each layer can be expressed as follows:

$$I_{area,n} = \frac{I_{0,n}}{\rho_{area,n}} \int_{Z_1}^{Z_2} I_{0,n} \exp\left(\frac{-t}{\lambda_n \cos\theta}\right) dt \quad \text{eq. A.4}$$

Where Z_1 and Z_2 represent the boundary limits of the layer containing species n . From here the theoretical ratio between the Mn and Si signal intensities can be expressed as follows:

$$\left(\frac{I_{area,Si}}{I_{area,Mn}}\right)_{Theoretical} = \frac{I_{0,Si}\rho_{area,Mn} \int_0^{d_{Si}} \exp\left(\frac{-t}{\lambda_{Si}\cos\theta}\right) dt}{I_{0,Mn}\rho_{area,Si} \int_{d_{Si}}^{\infty} \exp\left(\frac{-t}{\lambda_{Mn}\cos\theta}\right) dt} \quad \text{eq. A.5}$$

$$\left(\frac{I_{area,Si}}{I_{area,Mn}}\right)_{Theoretical} = \frac{I_{0,Si}\rho_{area,Mn} \left(1 - \exp\left(\frac{-d_{Si}}{\lambda_{Si}\cos\theta}\right)\right)}{I_{0,Mn}\rho_{area,Si} \left(\exp\left(\frac{-d_{Si}}{\lambda_{Mn}\cos\theta}\right)\right)} \quad \text{eq. A.6}$$

This expression can be used to obtain the theoretical ratio between the total number of Si and Mn atoms per unit area as follows:

$$N_{area,n} = \frac{I_{area,n}}{s_n} \quad \text{eq. A.7}$$

$$\left(\frac{N_{area,Si}}{N_{area,Mn}}\right)_{Theoretical} = \frac{s_{Mn}\rho_{area,Mn} \left(1 - \exp\left(\frac{-d_{Si}}{\lambda_{Si}\cos\theta}\right)\right)}{s_{Si}\rho_{area,Si} \left(\exp\left(\frac{-d_{Si}}{\lambda_{Mn}\cos\theta}\right)\right)} \quad \text{eq. A.8}$$

Where $N_{area,n}$ is the total number of atoms of species n per unit area and, s_n relative sensitivity factor of species n at its characteristic photoemission energy. Finally, the surface packing density was computed by comparing the experimentally obtained number ratio to that predicted by theory.

$$\rho_{Packing} = \frac{\left(\frac{N_{area,Si}}{N_{area,Mn}}\right)_{Experimental}}{\left(\frac{N_{area,Si}}{N_{area,Mn}}\right)_{Theoretical}} \rho_{area,Si} \quad \text{eq. A.9}$$

This approximation assumes that a monolayer of silicic acid represents the absolute maximum packing density possible at the surface, and that the computed number ratio is linearly correlated to the overall packing density.

Inelastic mean free paths were computed using the TPP-2M model and are displayed below. The total length of the overlayer d_{Si} was computed from silane structures optimized using an MMFF molecular modeling approximation.

Table A.1: Parameters used to compute inelastic mean free paths for emitted photoelectrons

	Si	LMNO
Valence electrons / molecule	32	80
MW (g mol ⁻¹)	96.113	182.69
ρ (g cm ⁻³)	1.8	4.45
Band Gap Energy	7	7
σ_n	0.328	2.659
λ_{Si} (cm, KE = 844 eV)	3.94E-07	--
λ_{Mn} (cm, KE = 1385 eV)	--	2.02E-07
λ_{Ni} (cm, KE = 632 eV)	--	1.11E-07

Uncertainty in the measurement was approximated by the standard fitting error incurred while computing integrated peak areas. Monte-Carlo error approximation was performed to determine the standard deviation in peak fitting parameters, the root mean square of which determined the total error in the experimental number ratio as follows:

$$\sigma_{fitting} \approx \left(\frac{N_{area,Si}}{N_{area,Mn}} \right)_{experimental} \sqrt{\left(\frac{\sigma_{Si}}{N_P} \right)^2 + \left(\frac{\sigma_{Mn}}{N_{Mn}} \right)^2 - 2 \left(\frac{\sigma_{Si} \sigma_{Mn}}{N_{Si} N_{Mn}} \right)} \quad \text{eq. A.10}$$

where σ is the standard deviation. Thus, the overall error in packing density is computed as

$$\sigma_{\rho_{packing}} = \frac{\rho_{area,Si}}{\left(\frac{N_{area,Si}}{N_{area,Mn}} \right)_{Theoretical}} \sigma_{fitting} \quad \text{eq. A.11}$$

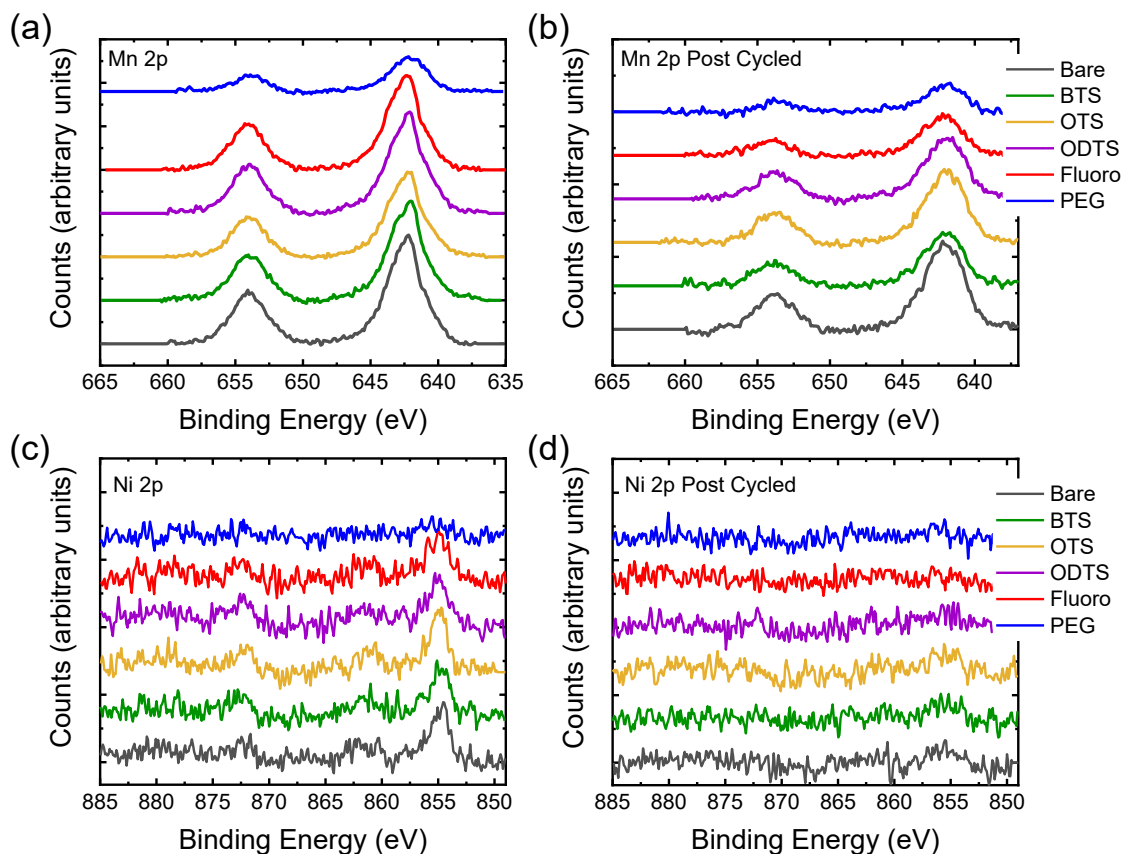


Figure A.1. Mn XPS spectra collected from uncycled (a) and post cycled (b) electrodes. Ni 2p spectra collected from uncycled (c) and post-cycled (d) electrodes.

Figure A.1 presents XPS spectra in the Mn and Ni 2p regions obtained from uncycled electrodes as well as electrodes recovered after 100 charge/discharge cycles. Both regions demonstrate diminished signal intensity after cycling consistent with the formation of the CEI during cell operation, attenuating photoelectron emission from the LMNO substrate.^{1,2} Mn spectra exhibit a change in peak shape for the Mn 2p_{1/2} feature owing to a reduction in signal intensity from Mn³⁺. This is consistent with previous post-mortem investigations of LMNO which suggest that electroactive Mn³⁺ near the surface is susceptible to disproportionation and dissolution into the electrolyte more so than electro-inactive Mn⁴⁺.³ The low signal-to-noise ratio found in the Ni 2p spectra makes quantitative analysis of spectral features difficult however some salient features are apparent. In the uncycled electrodes (Figure A.1c) spectral features are preserved after

modification by the silane monolayer as evidenced by the similarity in the observed peak binding energies. Unfortunately, after cycling the Ni 2p signal is almost totally attenuated by the CEI covering the electrode surface making additional characterization of the Ni oxidation state impossible.

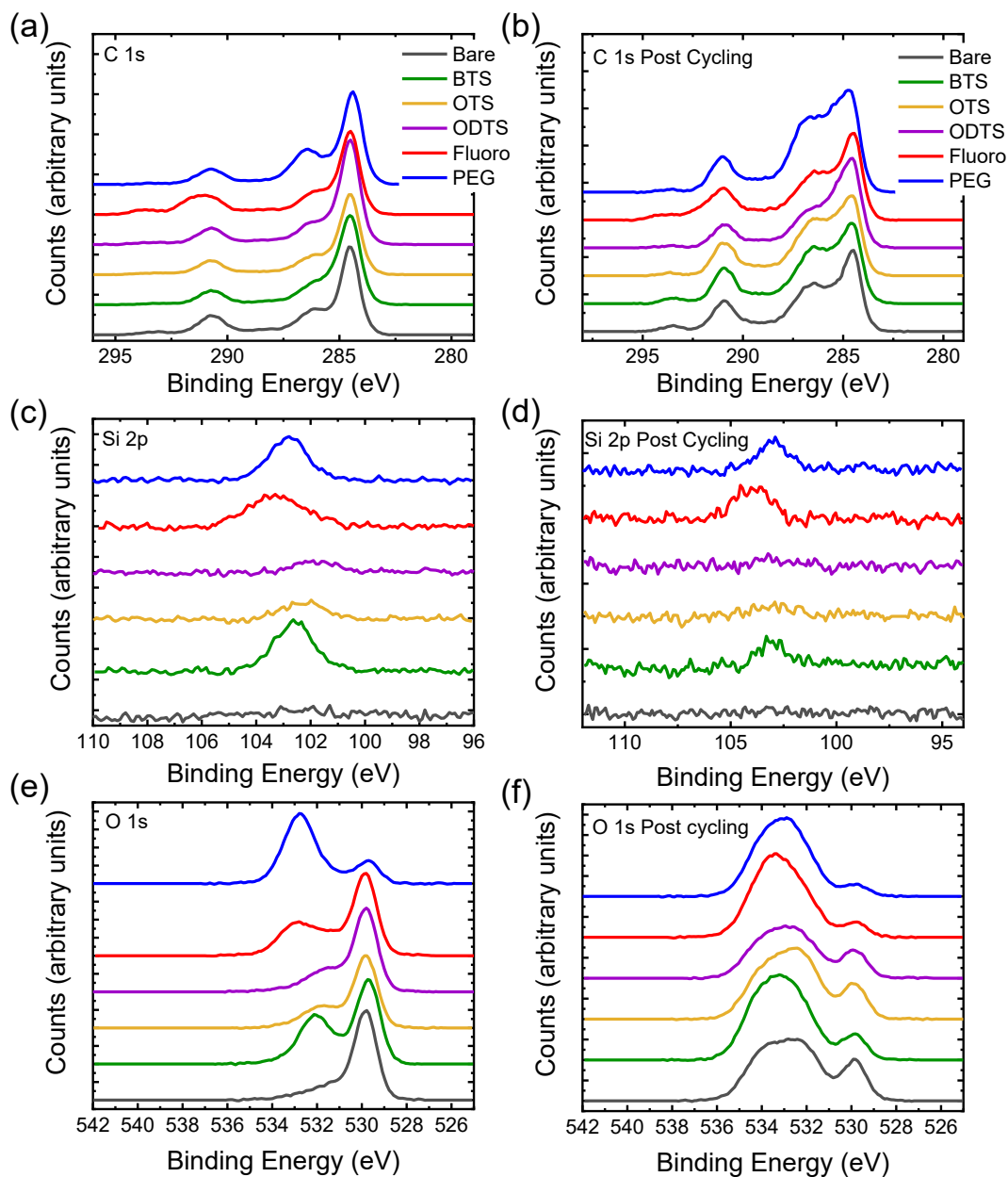


Figure A.2. C 1s XPS spectra from (a) uncycled and (b) cycled electrodes, Si 2p XPS spectra from (c) uncycled and (d) cycled electrodes and O 1s XPS spectra from (e) uncycled and (f) cycled electrodes.

Figure A.2 presents XPS spectra obtained from pristine electrodes and those recovered after 100 charge/discharge cycles. All spectra are energy calibrated to the sp^2 carbon peak associated with the Super P conductive additive at 284.5 eV. Regions interrogated here are associated with surface species present on the LMNO substrate including the silane monolayer, and common decomposition products from the electrolyte. The deposition of an interfacial layer is evident from the decreased signal intensity in the Si 2p region, as well as the increased signal intensity from ether and carbonate-based species (C 1s region 287 eV, O 1s region 533 eV).² The retention of the Si 2p signal suggests that despite the decomposition of electrolyte at the interphase, and the repeated deformation of the substrate lattice, the monolayer remains adhered to the surface throughout 100 galvanostatic cycles. Unfortunately, the reduction in signal intensity cannot be deconvoluted to determine if its origin is attenuation by the CEI or lower concentrations of silane at the surface.

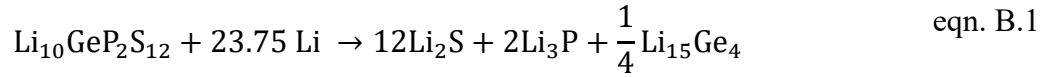
References:

- (1) Haasch, R. T., X-Ray Photoelectron Spectroscopy (XPS) and Auger Electron Spectroscopy (AES). In *Practical Materials Characterization*, M., S., Ed. Springer, New York, NY: 2014; pp 93-132.
- (2) Schulz, N.; Hausbrand, R.; Wittich, C.; Dimesso, L.; Jaegermann, W., XPS-Surface Analysis of SEI Layers on Li-Ion Cathodes: Part II. SEI-Composition and Formation inside Composite Electrodes. *J Electrochem Soc* **2018**, *165* (5), A833-A846.
- (3) Bhandari, A.; Bhattacharya, J., Review-Manganese Dissolution from Spinel Cathode: Few Unanswered Questions. *J Electrochem Soc* **2017**, *164* (2), A106-A127.

Appendix B: Critical Stress Approximation from Tomographic Data

Critical Stress Approximation

The critical stress imposed by individual decomposition sites prior to pellet fracture was approximated to determine the mechanical stability of LGPS when undergoing electrochemical decomposition. As discussed in the main text, this analysis approximates decomposition domains as oblate spheroids whose lateral and vertical radii were based on the experimentally observed critical dimensions prior to crack formation. The volumetric expansion of LGPS upon reduction was approximated by considering the stoichiometry of the following decomposition reaction:



The total volume of each of the species formed in the above reaction for each unit volume of LGPS consumed can be expressed as follows:

$$V_i = \rho_{\text{LGPS}} \cdot \frac{\text{MW}_i}{\text{MW}_{\text{LGPS}}} \cdot \frac{n_i}{n_{\text{LGPS}}} \cdot \frac{1}{\rho_i} \quad \text{eqn. B.2}$$

where V_i is volume of species i formed from the consumption of each unit volume of LGPS, ρ_i is the density of species i , MW_i is the molar mass of species i , and n_i is the stoichiometry of species i determined by eqn. B.1. The total volume of material formed from the decomposition of a given volume of LGPS can thus be described as:

$$V_{\text{decomposition}} = V_{\text{LGPS}} \sum_i V_i = V_{\text{LGPS}} \cdot V_n = 1.174 \cdot V_{\text{LGPS}} \quad \text{eqn. B.3}$$

Where V_n is the normalized volumetric expansion coefficient of LGPS during reduction. Using the stoichiometry prescribed by Eqn. B.1 and the material properties displayed in Table B.1 a volumetric expansion coefficient of 1.174 was obtained for the reduction of LGPS.

Table B.1. Values used to compute V_n

Species	ρ (g cm ⁻³)	MW (g mol ⁻¹)	n
Li ₁₀ GeP ₂ S ₁₂	1.5	588.8	1
Li ₂ S	1.66 ^[4]	45.9	12
Li ₃ P	1.43 ^[5]	51.8	2
Li ₁₅ Ge ₄	1.76 ^[6]	394.7	1/4

The volumetric expansion coefficient calculated above was used to describe the outward expansion of the inclusion within a simplified elastic medium modeling the interface between the lithium electrode and the LGPS pellet. Specifically, the LGPS/Li⁰ interface is treated as a semi-infinite boundary between two non-interacting phases in which an elliptical inclusion exists. The inclusion was centered about the interface between the two phases reflecting the behavior of the real system observed experimentally. With the exception of the inclusion itself, each phase is treated as a continuous domain of uniform composition such that at points far from the inclusion the absolute stress approaches 0. Stresses are treated by considering the radial expansion of the inclusion in x, y, and z as pseudo-linear such that the following equivalency is maintained:

$$\sigma = E\epsilon \quad \text{eqn. B.4}$$

Where σ is unidirectional stress, ϵ is the unidirectional strain (computed from the linear expansion of the decomposed material relative to LGPS), and E is the Youngs modulus (values of which are provided in table B.2). The LGPS/Li⁰ boundary is taken to be $z = 0$ such that the Youngs modulus can be described as follows:

$$\begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} = \begin{cases} \begin{bmatrix} E_{Li} \\ E_{Li} \\ E_{Li} \end{bmatrix}, & z > 0 \\ \begin{bmatrix} E_{LGPS} \\ E_{LGPS} \\ E_{Li} \end{bmatrix}, & z \leq 0 \end{cases} \quad \text{eqn. B.5}$$

Under this approximation it is assumed that stress development in the vertical dimension is governed entirely by the mechanical properties of Li, and that deformation of the bounding Li matrix will deform if stresses in the vertical dimension become sufficiently high.

The geometry of the inclusion was modeled as an oblate spheroid whose shape obeys the following equivalence:

$$\frac{x^2 + y^2}{a^2} + \frac{z^2}{c^2} = 1 \quad \text{eqn. B.6}$$

where a and c are the lateral and vertical radii respectively, taken directly from the experimentally determined critical dimensions displayed in the main text in Table 3.2.

Table B.2. Experimentally reported values used for critical stress approximation

Component	ρ (g cm ⁻³)	E (GPa)	REF
Li	0.534	4.9	1
LGPS	1.5	10.5	2

Strain vectors ($\vec{\epsilon}_{x,y,z}$) for each point ($p_{x,y,z}$) within the sampling window, were computed under the assumption that the volumetric expansion of the inclusion occurred isotropically centered about the origin. The isolated x, y, and z components of the strain vector can be readily expressed in spherical coordinates as:

$$\epsilon_x = \frac{(\sqrt[3]{V_n} - 1) L_{x,y,z} \sin(\phi) \cos(\theta)}{R_{x,y,z}} \quad \text{eqn. B.7}$$

$$\epsilon_y = \frac{(\sqrt[3]{V_n} - 1) L_{x,y,z} \sin(\phi) \sin(\theta)}{R_{x,y,z}} \quad \text{eqn. B.8}$$

$$\epsilon_z = \frac{(\sqrt[3]{V_n} - 1) L_{x,y,z} \cos(\phi)}{R_{x,y,z}} \quad \text{eqn. B.9}$$

where $R_{x,y,z}$ is the distance between $p_{x,y,z}$ and the origin, V_n is the normalized volumetric expansion coefficient determined from Eqn. B.2, and $L_{x,y,z}$ is the path length within the inclusion of the line segment between the origin and $p_{x,y,z}$ determined by:

$$L_{x,y,z} = \sqrt{\left(\frac{\sin^2 \phi}{a^2} + \frac{\cos^2 \phi}{c^2}\right)^{-1}} \quad \text{eqn. B.10}$$

Once the instantaneous strain vectors have been computed for every point within the sampling window, the instantaneous stress at each point can be computed as follows:

$$\sigma_{x,y,z} = \vec{E} \cdot \vec{\epsilon}_{x,y,z} \quad \text{eqn. B.11}$$

The results of this analysis, displayed within the x, z-plane for clarity, are presented in Figure B.1 wherein a series of different sized inclusions were considered approaching the critical dimensions observed experimentally (bottom-most figure). As observed experimentally, the locus of stress development occurs at the center of the inclusion, within the LGPS matrix. The critical stress at this point reaches approximately 0.8 GPa as the inclusion reaches its critical dimensions. Error was determined by considering the experimental uncertainty in the critical dimensions of the inclusion size to yield the final value of 0.8 ± 0.2 GPa.

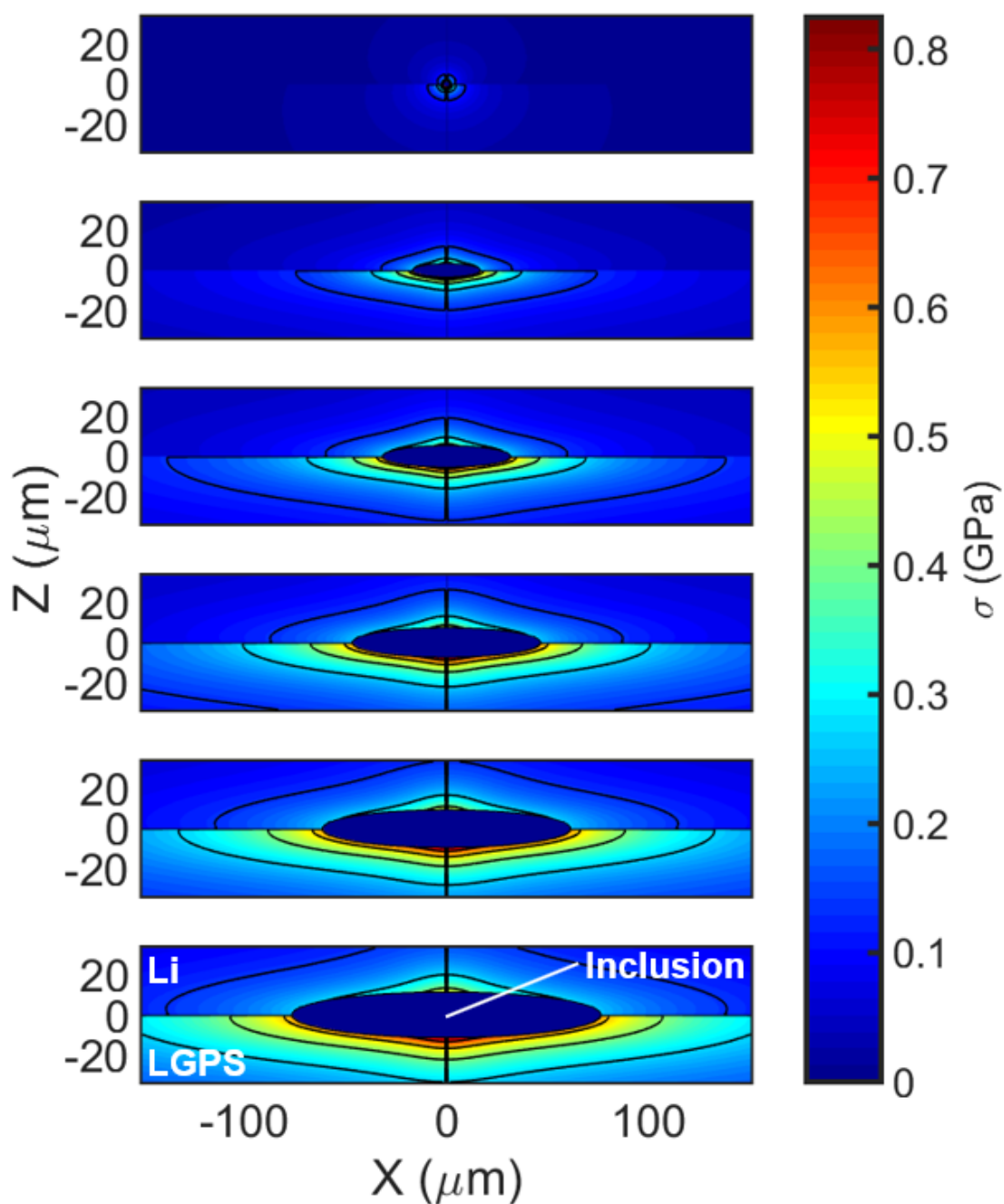


Figure B.1. Computed instantaneous stress values in the x, z-plane for a model system considering two noninteracting phases with different elastic moduli in which an ellipsoid inclusion is progressively growing. Inclusion size is increased to the nominal critical size (height: 19 μm , diameter: 72 μm) determined from experiment in the bottom-most image.

References:

- (1) Samsonov, G. V., *Handbook of the Physicochemical Properties of the Elements*. Springer Science & Business Media: 2012.
- (2) Weber, D. A.; Senyshyn, A.; Weldert, K. S.; Wenzel, S.; Zhang, W.; Kaiser, R.; Berendts, S.; Janek, J.; Zeier, W. G., Structural Insights and 3D Diffusion Pathways within the Lithium Superionic Conductor $\text{Li}_{10}\text{GeP}_2\text{S}_{12}$. *Chem Mater* **2016**, 28 (16), 5905-5915.

Appendix C: Pressure Sensing Electrochemical Cell (Design and Data Acquisition)

Cell Design:

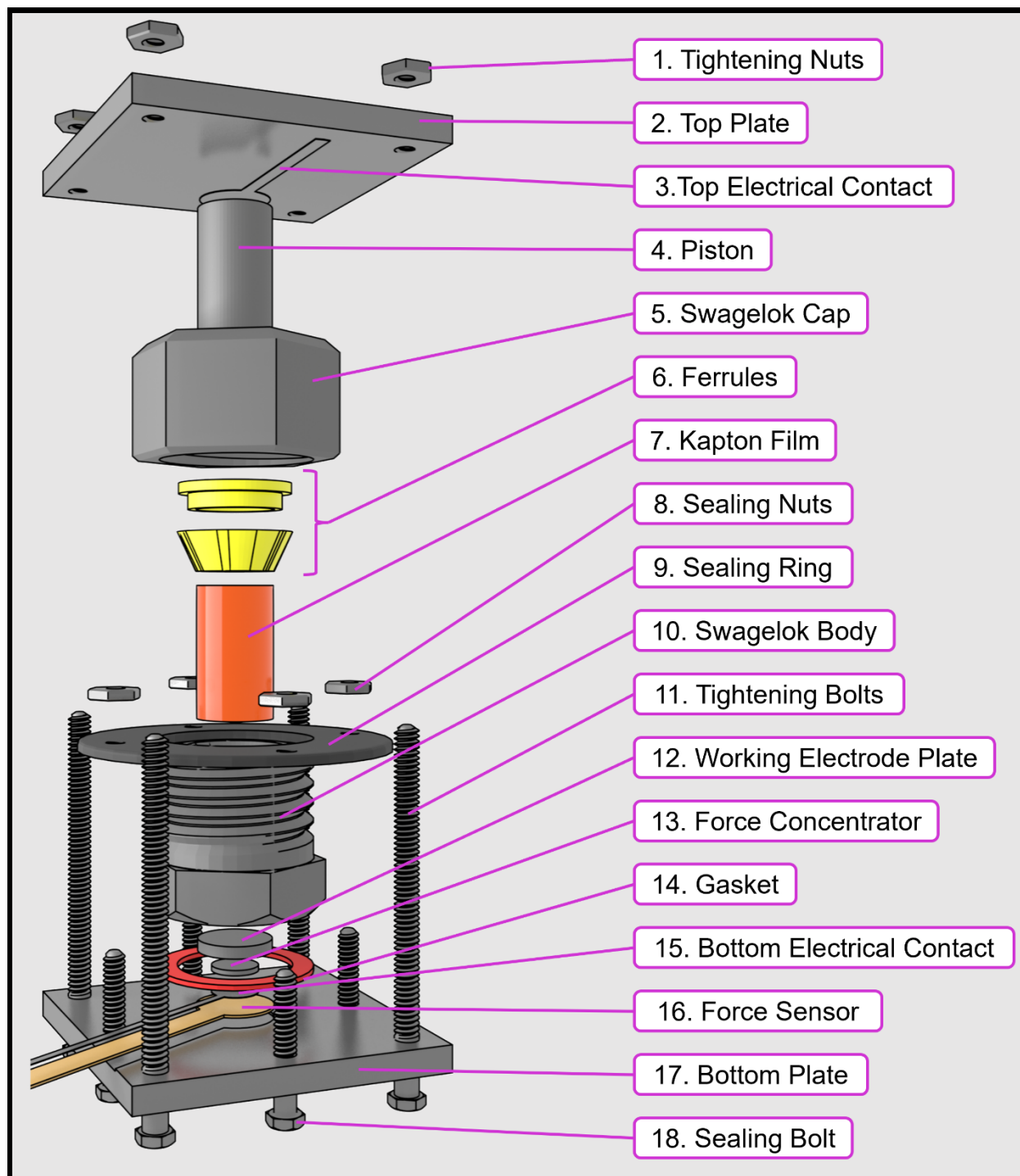


Figure C1. Exploded view of the custom electrochemical cell used to both apply and monitor the transverse mechanical load across an electrode stack.

This section details the construction and operation of the pressure sensing electrochemical cell described in chapter 4 of this thesis. The cell utilized A201 solid state force sensors (FlexiForce, Tekscan) to measure the transverse mechanical load exerted across the electrode stack comprising the battery configuration under investigation (in the case of the work detailed in chapter 4 the battery system of interest involved the organic electrode material dilithium rhodizonate, $\text{Li}_2\text{C}_6\text{O}_6$). The cell was constructed as a combination of a Swagelok housing as the upper half and electrode container terminating in an aluminum force plate onto which the force sensor was placed. A blown apart schematic of the cell design is provided in Figure C1 to complement the schematic provided in chapter 4. Using this construction, the force could be quantified by the resistance of the force sensor. To facilitate data acquisition the force sensor was integrated into a simple data monitoring system using, at its heart, an ARDUINO UNO to relay the force reading from the sensor to the computer.

A circuit diagram of the data acquisition system is provided in Figure C2. The resistance across the force sensor could easily be obtained from a simple voltage bridge, comparing it against a known predetermined resistance. Unfortunately, the resistance of the sensor varied many orders of magnitude, precluding the use of a single reference resistor. Instead, a resistor array was fabricated gating each bridge with a bipolar junction transistor. The gating was controlled by the ARDUINO board directly to maintain adequate accuracy and linearity in the sensor response. In addition to this gated resistor array a temperature sensor was also included to help deconvolute electrochemically induced changes in force from thermally induced changes in force. Indeed, it was noticed that changes in the electrode stack force of up to 20 N could be realized simply from thermal fluctuations in lab environment (the magnitude of these thermal changes were found to be roughly $\pm 1.5\text{ }^\circ\text{C}$). While *operando* force changes recorded during cell

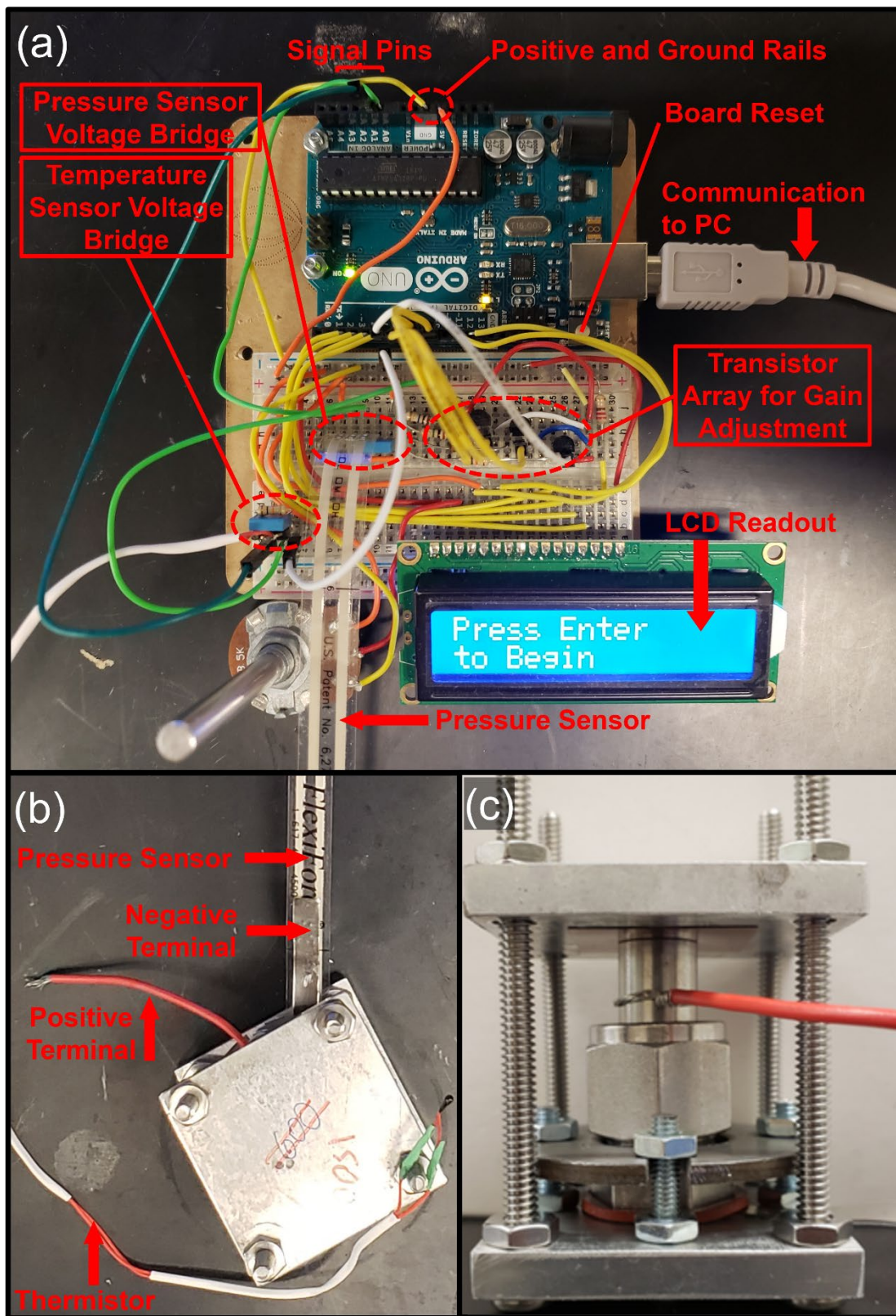


Figure C3. Photograph of the data acquisition system used to log pressure data collected by the cell depicted in Figure C1.

Figure C3 presents a photograph of the circuit setup described in Figure C2, pointing out the locations of key components of the data acquisition system. The ARDUINO board uses a USB connection to the computer for both serial line data communication and power, thus it must be actively connected to an external PC to allow data collection. The system features an LCD readout which provides both the real time force measurement as well as the resistor value being used to collect data to aid in troubleshooting. Likewise, the force sensor can be easily swapped if necessary as it simply plugs directly into the breadboard to facilitate communication with the data acquisition board. A total of two signal pins are used to collect both the force and temperature data. Similarly, while the data acquisition rate (and the accompanying baud rate of the device) is very low (as the galvanostatic cycling measurements occur over several days) it is possible to increase this rate for higher speed data acquisition depending on the application.

Practical Operation of the Pressure Sensing Cell:

As mentioned above the data acquisition system uses a serial line USB connection to both communicate data to the PC and to power the circuit. Thus, the board must be connected to a PC via USB for operation. Likewise, the board relies on an external python script to log the data output from the force sensor (the ARDUINO board does not store any data itself; it only writes time dependent data along the serial connection to be logged on the PC) so the PC to which the board is connected must have python (or preferably anaconda) installed to allow the acquisition program to run. Once these conditions have been met the program can be started by running the Windows batch file described above (Stress.bat). Please note that it may be necessary to change the target directories within the batch file if the python script (Sercom_5.py) has been moved to a different location. On running the batch file, the user will be prompted with the terminal window displayed in Figure C4 and note (if the serial connection is established correctly) that the LCD connected to the acquisition board will display “Press Enter to Begin.”

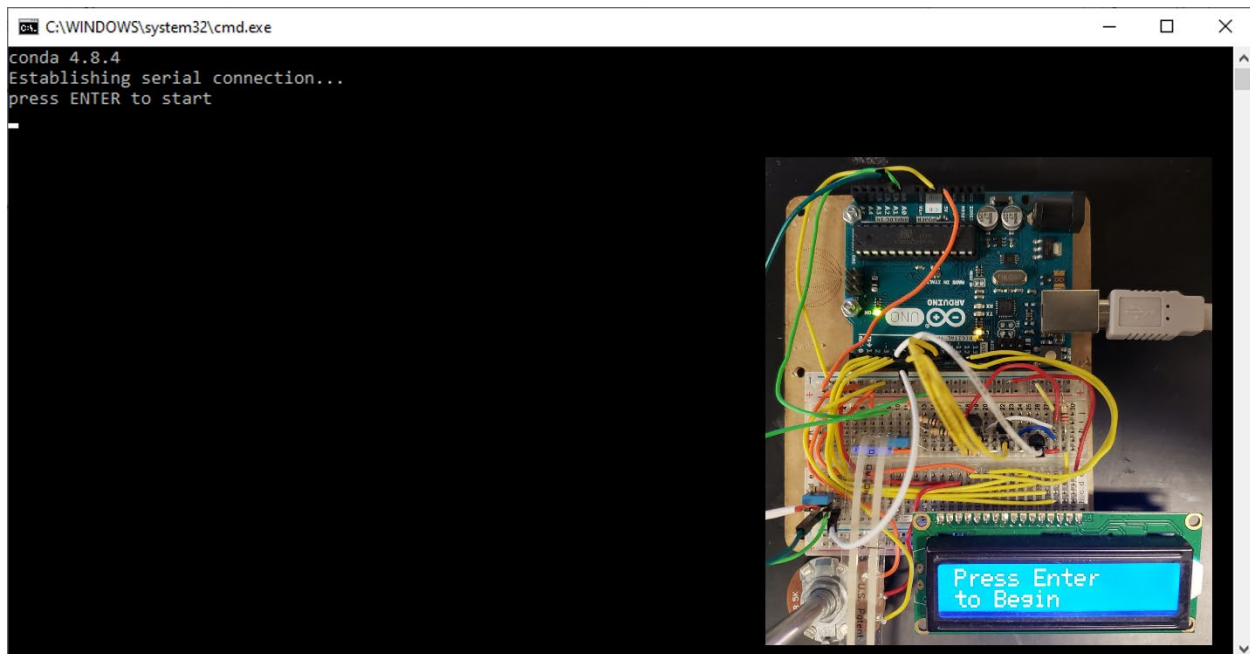


Figure C4. Command line prompt generated on running the windows batch file along with a photograph of the data acquisition board during startup.

In the event that serial communication cannot be established the command line will display an error message and will end the program execution. The most common problem in this instance is the that the python script is addressed to an incorrect COM port. The issue can be resolved by opening device manager to verify which COM port the serial communication is indexed to and entering the correct port into the python script during initialization of the serial communication. Another, less common, issue is that the ARDUINO board has been inadvertently reprogrammed. This issue can be quickly resolved by simply reuploading the working program to the board using the ARDUINO IDE. Once communication has been established and the correct prompt has been issued by the command line, the user need only press the “Enter” key to begin data accumulation. On pressing enter a serial line command is issued to the ARDUINO board which will update the LCD screen with the current force value and resistor number being used to probe the force sensor as depicted in Figure C5.

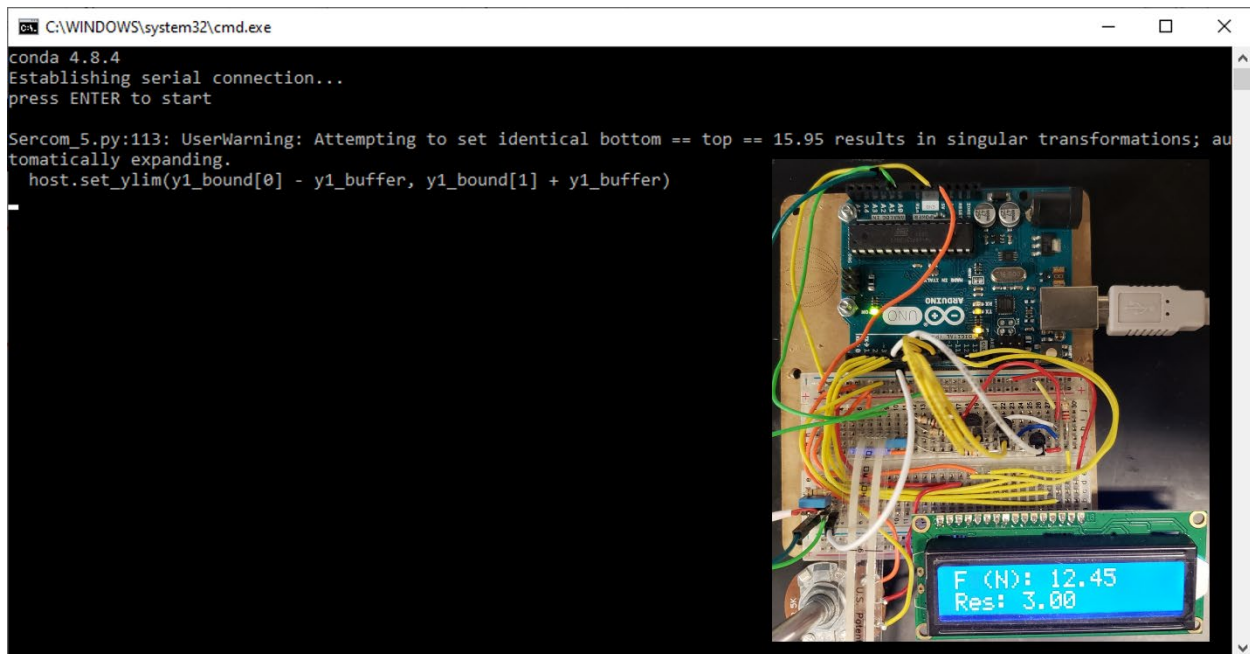


Figure C5. Command line state and board state while acquiring data.

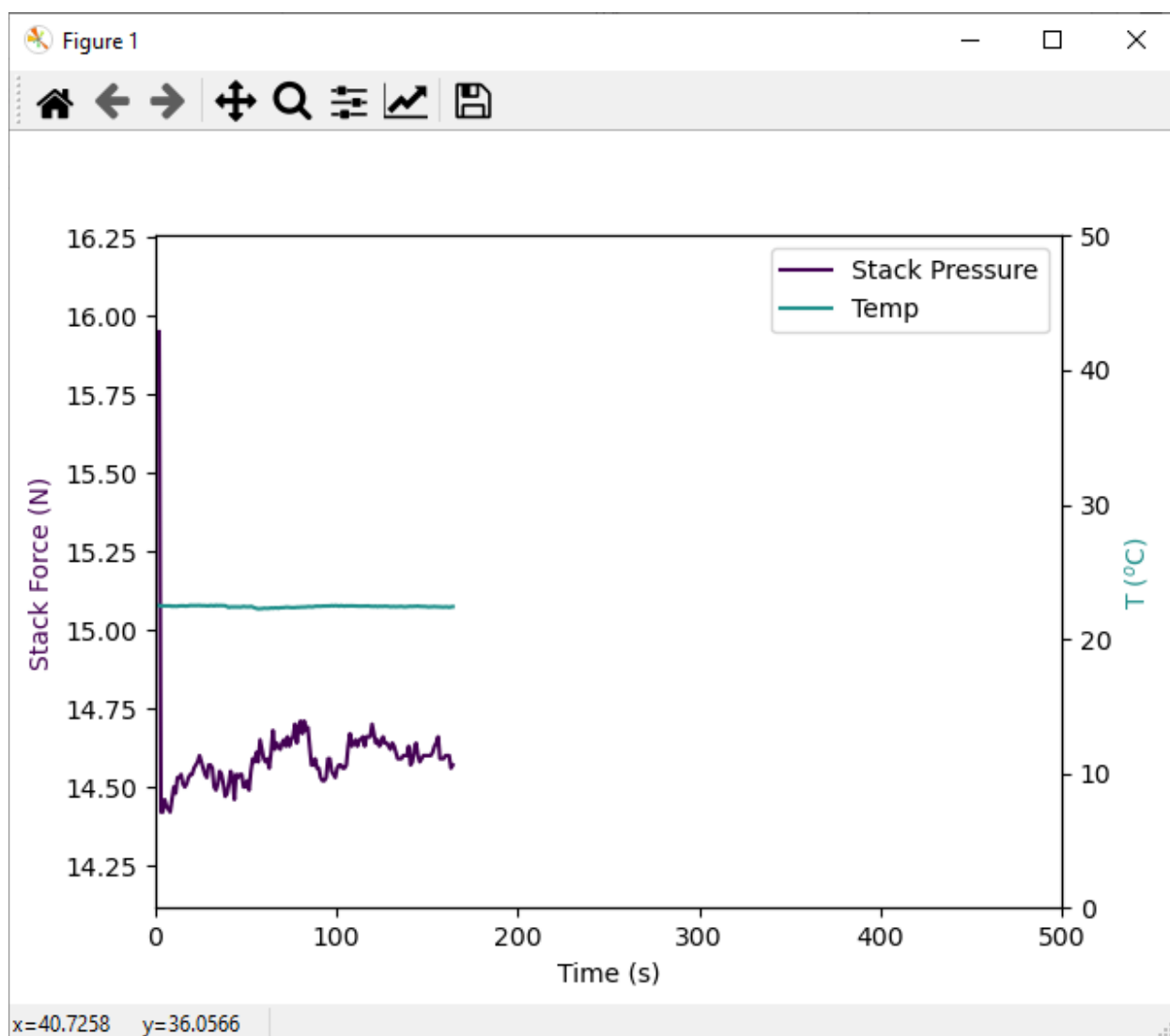


Figure C6. Exemplary data displayed to the PC during data accumulation.

While data acquisition is running the PC will display a live view of the data obtained from the cell. A matplotlib figure window is used to present both the electrode stack pressure and the ambient temperature registered by the cell as it cycles. The abscissa of the plot update automatically to allow for easy visualization of live data during collection. In addition to this live view the data acquisition script (Sercom_5.py) will periodically (once every 10 minutes) save the data to a text file (by default the text file is titled “out.txt”) in the same directory that Sercom_5.py is located. The output file contains the time points, force values, and temperatures

logged by the cell during cycling. Following collection, the data can be analyzed using Stress.py, provided above. This program takes the raw output files from both the potentiostat (at time of publication only CHInstruments are supported) and the force sensor to visualize the electrochemical dependence of the observed force changes.

Serial Communication Script (Python):

```
#!/usr/bin/env python3
#!/export MPLBACKEND=TkAgg
import serial
import subprocess
import numpy as np
import pandas as pd
import os
import csv
import time
import math
import matplotlib as mpl
import matplotlib.pyplot as plt
import serial.tools.list_ports as port_list

plt.ion()
plt.show()

#print(serial.tools.list_ports.comports(include_links=False))
print("Establishing serial connection...")
try:
    arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=.1)
except:
    print("Serial connection could not be established")
    exit()

arduino.flushInput()
arduino.flushOutput()

time.sleep(5)

print("press ENTER to start")
input()
arduino.write(b'5.0')

Out = pd.DataFrame(columns = [0, 1, 2])
test = pd.DataFrame(columns = [0])

x = Out.loc[:, 0].to_numpy(copy = False)
y1 = Out.loc[:, 1].to_numpy(copy = False)
y2 = test.loc[:, 0].to_numpy(copy = False)

fig = plt.figure()
host = fig.add_subplot(111)
par1 = host.twinx()

host.set_xlim(0, 100)
par1.set_ylim(0, 50)

host.set_xlabel("Time (s)")
host.set_ylabel("Stack Force (N)")
par1.set_ylabel("T ($^{o}$C)")

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
```

```

p1, = host.plot(x, y1, color=color1, label="Stack Pressure")
p2, = par1.plot(x, y2, color=color2, label="Temp")

lms = [p1, p2]
host.legend(handles=lms, loc='best')

host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())

start = time.time()

a = 0.003354017
b = 0.00025617244
c = 0.0000021400943
d = -0.000000072405219

try:
    while True:
        data = arduino.readline()[:-2] #the last bit gets rid of the new-
line chars
        if data:
            try:
                data = data.decode("utf-8")
                data = data.split(",")
                data = [float(i) for i in data]

                TData = data[2]/10000
                T =
(1/(a+b*math.log(TData)+c*math.log(TData)**2+d*math.log(TData)**3))-273.15

                Out = Out.append({0 : data[0]/1000, 1 : data[1], 2 :
T}, ignore_index = True)

                test = test.append({0 : T}, ignore_index = True)

                x = Out.loc[:, 0].to_numpy(copy = False)
                y1 = Out.loc[:, 1].to_numpy(copy = False)
                y2 = test.loc[:, 0].to_numpy(copy = False)

                x_bound = [min(x), max(x)]

                y1_bound = [min(y1), max(y1)]
                y1_buffer = (max(y1)-min(y1)) * 0.2

                y2_bound = [min(y2), max(y2)]
                y2_buffer = (max(y2)-min(y2)) * 0.2

                p1.set_xdata(x)
                p1.set_ydata(y1)

                p2.set_xdata(x)
                p2.set_ydata(y2)

                fig.canvas.flush_events()

                if x_bound[1] > host.get_xlim()[1]:

```

```

        host.set_xlim(0, host.get_xlim()[1] * 5)

        if y1_bound[0] < host.get_ylim()[0] or y1_bound[1] >
host.get_ylim()[1]:
            host.set_ylim(y1_bound[0] - y1_buffer,
y1_bound[1] + y1_buffer)

        if y2_bound[0] < par1.get_ylim()[0] or y2_bound[1] >
par1.get_ylim()[1]:
            par1.set_ylim(y2_bound[0] - y2_buffer,
y2_bound[1] + y2_buffer)
    except:
        print("invalid data point")

        if (time.time() - start) > 600:
            Out.to_csv('Out.txt', header=False, index=False)
            start = time.time()

except KeyboardInterrupt:
    arduino.write(b'5.0')
    Out.to_csv('Out.txt', header=False, index=False)

x = Out.loc[:, 0].to_numpy(copy = False)
y1 = Out.loc[:, 1].to_numpy(copy = False)
y2 = test.loc[:, 0].to_numpy(copy = False)

p1.set_xdata(x)
p1.set_ydata(y1)
p2.set_xdata(x)
p2.set_ydata(y2)
plt.draw()
plt.pause(.00001)

print("press ENTER to terminate execution")
input()

```

Board Operating Script (Arduino):

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int analogPin = 0;
int analogPin2 = 1;

float state = 0;
float raw = 0;
float raw2 = 0;
float Vin = 5.0;
float Vout = 0;
float Vout2 = 0;

float R1 = 10000;
float R2 = 170000;
float R3 = 1000000;
float R4 = 10000000;
float RT = 10000;

float RU = 0;
float RUT = 0;

float RN = 4;
float RN2 = 4;
float RTemp = R4;
// Decreasing nav lowers the averaging and allows for faster data acquisition
// A data point is logged every nav*10 milliseconds
float nav = 100;
float ct = 0;
float sum = 0;
float sum2 = 0;

float t = 0;
float tzero = 0;

float buffer = 0;

void setup() {
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);

  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);

  lcd.begin(16, 2);
  lcd.print("Hi There");
  delay(1000);
```



```

Serial.begin(9600);
while(Serial.available() > 0){
    Serial.read();
}
lcd.setCursor(0, 0);
lcd.print("Press Enter");

lcd.setCursor(0, 1);
lcd.print("to Begin");
while(Serial.available() == 0){
}
delay(1000);
tzero = millis();
state = Serial.available();
}

void loop() {
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting begins with 0):
    if (ct < nav){
        raw = analogRead(analogPin)*Vin;
        raw2 = analogRead(analogPin2)*Vin;

        Vout = raw/1024.0;
        Vout2 = raw2/1024.0;

        RU = RTemp*(Vin/Vout-1);
        RUT = RT*(Vin/Vout2-1);

        sum = sum + RU;
        sum2 = sum2 + RUT;
        ct = ct + 1;
        delay(10);
    }

    else{
        RU = sum / nav;
        RN2 = RN;
        if (RU > 1.2 * R3){
            digitalWrite(6, LOW);
            digitalWrite(7, LOW);
            digitalWrite(8, LOW);
            digitalWrite(9, HIGH);
            RN = 4;
            RTemp = R4;
        }
        else if (RU <= 1.2 * R3 && RU > 1.2 * R2){
            digitalWrite(6, LOW);
            digitalWrite(7, LOW);
            digitalWrite(8, HIGH);
            digitalWrite(9, LOW);
            RN = 3;
            RTemp = R3;
        }
        else if (RU <= 1.2 * R2 && RU > 1.2 * R1){
            digitalWrite(6, LOW);
            digitalWrite(7, HIGH);

```

```

        digitalWrite(8, LOW);
        digitalWrite(9, LOW);
        RN = 2;
        RTemp = R2;
    }
    else if (RU <= 1.2 * R1){
        digitalWrite(6, HIGH);
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);
        RN = 1;
        RTemp = R1;
    }
    if (RN2 == RN){
        //raw = analogRead(analogPin)*Vin;
        //Vout = raw/1024.0;
        //RU = RTemp*(Vin/Vout-1);
        t = millis() - tzero;
        RU = sum/nav;
        RU = 10000000/RU;
        RUT = sum2/nav;

        lcd.setCursor(0, 0);
        lcd.print("F (N): ");
        lcd.print(RU);
        Serial.print(t);
        Serial.print(", ");
        Serial.print(RU);
        Serial.print(", ");
        Serial.println(RUT);
        lcd.setCursor(0, 1);
        lcd.print("Res: ");
        lcd.print(RN);
    }
    ct = 0;
    sum = 0;
    sum2 = 0;

}

if (Serial.available() > state){
    exit(0);
}
}

```

Windows .bat file

```
@ECHO OFF
set
PATH=%PATH%;C:\Users\kmadsen2\anaconda3;C:\Users\kmadsen2\anaconda3\Scripts\
conda --version
cd ../anaconda3/envs/

IF not exist arduino\nul (
conda create --name arduino python=3.7
ECHO Environment Loaded
)

cd ../pkgs
IF not exist pyserial-3.4-py_2\nul (
conda install -c conda-forge pyserial
ECHO Serial module Loaded
)

call activate arduino
cd ../Codes

python Sercom_5.py

ECHO success
pause
```

Data Processing Script (Python):

```
import numpy as np
import pandas as pd
import math
import xlswriter
from bisect import bisect_left
import statistics
import os
import csv
import xlrd
import time, sys
#matplotlib.use('TkAgg')
%matplotlib qt
import matplotlib as mpl
from mpl_toolkits.mplot3d.axes3d import Axes3D
from tkinter import Tk
from tkinter.filedialog import askopenfilename
import matplotlib.pyplot as plt
import matplotlib.pylab as pl
import matplotlib.font_manager
from IPython.display import clear_output

font = {'weight' : 'normal',
        'size'   : 20}

mpl.rc('font', **font)
mpl.rcParams['axes.linewidth'] = 2
plt.close("all")
plt.ion()
plt.show()

cwd = os.getcwd()

def Moving_Avg(Data, window):
    New_Data = Data.copy()
    for i in range(0, len(New_Data)):
        if (i - window) >= 0 and (i + window) <= len(New_Data):
            New_Data[i, 0] = statistics.mean(New_Data[i-window: i+window, 0])
            New_Data[i, 1] = statistics.mean(New_Data[i-window: i+window, 1])
        elif (i - window) < 0 and (i + window) <= len(New_Data):
            New_Data[i, 0] = statistics.mean(New_Data[0 : i + window, 0])
            New_Data[i, 1] = statistics.mean(New_Data[0 : i + window, 1])

        elif (i - window) >= 0 and (i + window) > len(New_Data):
            New_Data[i, 0] = statistics.mean(New_Data[i-window: len(Data),
0])
            New_Data[i, 1] = statistics.mean(New_Data[i-window: len(Data),
1])

        elif (i - window) < 0 and (i + window) > len(New_Data):
            New_Data[i, 0] = New_Data[i, 0]
            New_Data[i, 1] = New_Data[i, 1]
    return New_Data

def Reducing_Avg(Data, n):
    DRav = []
```

```

ttemp = 0
ptemp = 0
ct = 0

for i in range(0, len(Data)-n):
    if ct < n:
        ttemp = ttemp + Data[i, 0]
        ptemp = ptemp + Data[i, 1]
        ct = ct + 1
    else:
        DRav.append([ttemp/n, ptemp/n])
        ttemp = 0
        ptemp = 0
        ct = 0

DRav = np.asarray(DRav)
return DRav

def Write_To_Excel(big_out, Names, name):
    writer = pd.ExcelWriter(name + '.xlsx', engine = 'xlsxwriter')

    for i in range(0, len(big_out)):
        if isinstance(big_out[i], np.ndarray):
            tmp = pd.DataFrame(big_out[i])
            tmp.to_excel(writer, sheet_name = Names[i])

        elif isinstance(big_out[i], list):
            L = []
            mat = []
            for j in range(0, len(big_out[i])):
                L.append(len(big_out[i][j]))
            L = max(L)

            for j in range(0, len(big_out[i])):
                tmp = big_out[i][j].copy()
                tmp = np.pad(tmp, ((0, L-len(tmp)), (0,0)), 'constant',
constant_values=np.nan)
                if j == 0:
                    mat = tmp
                else:
                    mat = np.hstack((mat, tmp))
            mat = pd.DataFrame(mat)
            mat.to_excel(writer, sheet_name = Names[i])

    writer.save()
    writer.close()
    return

def Write_To_Text(big_out, Names):
    for i in range(0, len(big_out)):
        if isinstance(big_out[i], np.ndarray):
            tmp = big_out[i]
            np.savetxt(Names[i] + '.txt', tmp, delimiter = ',')

        elif isinstance(big_out[i], list):
            L = []

```

```

        mat = []
        for j in range(0, len(big_out[i])):
            L.append(len(big_out[i][j]))
        L = max(L)

        for j in range(0, len(big_out[i])):
            tmp = big_out[i][j].copy()
            tmp = np.pad(tmp, ((0, L-len(tmp)), (0,0)), 'constant',
constant_values=np.nan)
            if j == 0:
                mat = tmp
            else:
                mat = np.hstack((mat, tmp))
        np.savetxt(Names[i] + '.txt', mat, delimiter = ',')

    return

def update_progress(progress, label):
    bar_length = 20
    if isinstance(progress, int):
        progress = float(progress)
    if not isinstance(progress, float):
        progress = 0
    if progress < 0:
        progress = 0
    if progress >= 1:
        progress = 1

    block = int(round(bar_length * progress))

    clear_output(wait = True)
    text = label + "Progress: [{0}] {1:.1f}%".format( "#" * block + "-" *
(bar_length - block), progress * 100)
    print(text)

def Numeric_Average(Data, n, n2):
    DRav = Reducing_Avg(Data, n)
    print('DRav Complete')

    D_Dav = []
    for i in range(int(n2/2), len(DRav)-int(n2/2)):
        D_Dav.append([DRav[i, 0], (DRav[i+int(n2/2), 1] - DRav[i-int(n2/2),
1])/(DRav[i+int(n2/2), 0] - DRav[i-int(n2/2), 0])])
    D_Dav = np.asarray(D_Dav)
    print('D_Dav Complete')

    return DRav, D_Dav

def Fast_Fourier(Data, hfreq, lfreq, tol, Pmax):
    f = Data[:, 1]
    t = Data[:, 0]
    dt = t[5] - t[4]
    n = len(t)

    bkg = []
    for i in range(0, n):
        val = f[1] + (i/n)*(f[-1]-f[1])

```

```

        bkg.append(val)

f = f - bkg
fhat = np.fft.fft(f, n)
PSD = fhat*np.conj(fhat) / n
freq = (1/(dt*n))*np.arange(n)
L = np.arange(1, np.floor(n/2), dtype = 'int')

if hfreq > 0:
    indices = (freq < hfreq)
if lfreq > 0:
    indices = (freq > lfreq)
if Pmax > 0:
    indices = (PSD > Pmax)

PSDclean = PSD
fhat = indices*fhat
DRav = np.fft.ifft(fhat)
DRav = DRav.real
DRav = DRav + bkg
DRav = np.append(np.c_[t], np.c_[DRav], 1)

dx = []
for i in range(0, len(DRav)):
    if abs(Data[i, 1]-DRav[i, 1]) > tol:
        dx.append(i)

DRav = np.delete(DRav, dx, 0)
DRav = Reducing_Avg(DRav, 1)
print('DRav Complete')

D_Dav = []
for i in range(1, len(DRav)-2):
    D_Dav.append([(DRav[i, 0], (DRav[i+1, 1] - DRav[i-1, 1])/(DRav[i+1, 0]
- DRav[i-1, 0]))])

D_Dav = np.asarray(D_Dav)
print('D_Dav Complete')
return DRav, D_Dav

def P_Bkg_Fit(Data, n):
    y = np.c_[Data[:, 1].copy()]
    x = np.c_[Data[:, 0].copy()]
    x_mat = np.ones((len(x), 1))
    for i in range(0, n):
        x_mat = np.hstack((np.power(x, i+1), x_mat))
    x_prime = np.linalg.pinv(x_mat)
    Cnst = np.transpose(np.dot(x_prime, y))
    x_mat = np.squeeze(np.asarray(x_mat))
    Cnst = np.squeeze(np.asarray(Cnst))
    Bkg = np.matmul(x_mat, Cnst) # - Cnst[-1]
    return Bkg

def read_echem_data(filename):
    file_name, file_extension = os.path.splitext(filename)
    if file_extension == '.txt':

```

```

EChem = pd.read_csv(efilename, names = [0, 1, 2, 3], engine='python')
if EChem.loc[1, 0] == "Chronopotentiometry":
    d_type = "GC Data"
    print("GC Data")
    SIndex = int(EChem[EChem[0]=="Time/sec"].index.values)
    EChem.columns = EChem.loc[SIndex, :]
    EChem.drop(EChem.loc[0:SIndex].index, inplace=True)
    EChem.reset_index(inplace = True, drop = True)
    EDat = EChem[['Time/sec', ' Potential/V']].copy()
    EDat.columns = ['Time/s', 'Potential/V']
    EDat = EDat.astype(float)
    EDat.loc[:, 'Time/s'] = EDat.loc[:, 'Time/s']/3600
    #EDat.plot(x = 'Time/s', y = 'Potential/V', kind = 'line')

elif EChem.loc[1, 0] == "Cyclic Voltammetry":
    d_type = "CV Data"
    SIndex = int(EChem[EChem[0]=="Potential/V"].index.values)
    EChem.columns = EChem.loc[SIndex, :]
    EChem.drop(EChem.loc[0:SIndex].index, inplace=True)
    EChem.reset_index(inplace = True, drop = True)
    if ' Time/s' in EChem.columns:
        EDat = EChem[[' Time/s', 'Potential/V', ' Current/A']].copy()
        EDat.columns = ['Time/s', 'Potential/V', 'Current/A']

    else:
        EDat = EChem[['Potential/V', ' Current/A']].copy()
        tme = []
        SR = .0002
        dV = float(EChem.loc[1, 'Potential/V']) - float(EChem.loc[0,
'Potential/V'])
        for i in range(0, len(EChem)):
            tme.append(i*dV/SR)
        EDat[' Time/s'] = tme
        EDat.columns = ['Potential/V', 'Current/A', 'Time/s']
        EDat = EDat['Time/s', 'Potential/V', 'Current/A']
        EDat = EDat.astype(float)
        EDat.loc[:, 'Time/s'] = EDat.loc[:, 'Time/s']/3600
        #EDat.plot(x = 'Time/s', y = 'Current/A', kind = 'line')
elif EChem.loc[1, 0] == "Open Circuit Potential - Time":
    d_type = "OCP Data"
    print("OCP Data")
    SIndex = int(EChem[EChem[0]=="Time/sec"].index.values)
    EChem.columns = EChem.loc[SIndex, :]
    EChem.drop(EChem.loc[0:SIndex].index, inplace=True)
    EChem.reset_index(inplace = True, drop = True)
    EDat = EChem[['Time/sec', ' Potential/V']].copy()
    EDat.columns = ['Time/s', 'Potential/V']
    EDat = EDat.astype(float)
    EDat.loc[:, 'Time/s'] = EDat.loc[:, 'Time/s']/3600
elif file_extension == '.xlsx':
    wb = xlrd.open_workbook(filename)
    sheets = wb.sheet_names()

    Data = pd.DataFrame()
    for i in range(1, len(sheets) - 1):
        sheet = wb.sheet_by_index(i)
        hdrs = sheet.row_values(0)

```



```

        l = []
        l.append([hdrs.index(i) for i in hdrs if 'Test_Time' in i][0])
        l.append([hdrs.index(i) for i in hdrs if 'Current' in i][0])
        l.append([hdrs.index(i) for i in hdrs if 'Voltage' in i][0])

        Data = Data.append(pd.read_excel(filename, sheet_name =
sheets[i], usecols = 1))
        Data = Data[Data.iloc[:, 1] != 0]
        wb.release_resources()
        EDat = Data.iloc[:, [0, 2]]
        EDat.iloc[:, 0] = EDat.iloc[:, 0]/3600
        d_type = "GC Data"

    EDat = EDat.to_numpy()
    return d_type, EDat

def process_gc(EDat, C_Rate):
    indices = []
    for i in range(1, len(EDat)):
        if EDat[i-1, 0] == EDat[i, 0] or EDat[i-1, 1] == EDat[i, 1]:
            indices.append(i)
    EDat = np.delete(EDat, indices, axis=0)

    hV = max(EDat[:, 1])
    lV = min(EDat[:, 1])
    mid_point = lV + (hV - lV)/2

    diff = EDat[:, 1] - mid_point
    diff = np.c_[diff]

    xings = []
    xings.append(0)
    for i in range(1, len(diff)-1):
        if diff[i-1]*diff[i] < 0: # and i > (int(xings[-1]) + 5):
            xings.append(i)
    xings = np.c_[xings]

    T_index = []

    for i in range(1, len(xings)):
        temp = EDat[int(xings[i-1]):int(xings[i]), 1]
        drex = np.argmax(abs(temp - mid_point))
        T_index.append(drex + int(xings[i-1]))

    if T_index[0] != 0:
        np.insert(T_index, 0, 0)

    T_index = np.c_[T_index]

    if EDat[int(T_index[1]), 1] < mid_point:
        start_character = 'Discharge'
    elif EDat[int(T_index[1]), 1] > mid_point:
        start_character = 'Charge'

    Charge = []
    Discharge = []

```

```

DCharge = []
DDischarge = []
step = 1
for i in range(1, len(T_index)):
    if start_character == 'Discharge':
        if (i - 1) % 2 == 0:
            Discharge.append(EDat[int(T_index[i-1]) : int(T_index[i]),
:].copy())
        else:
            Charge.append(EDat[int(T_index[i-1]) : int(T_index[i]),
:].copy())
    elif start_character == 'Charge':
        if (i - 1) % 2 == 0:
            Charge.append(EDat[int(T_index[i-1]) : int(T_index[i]),
:].copy())
        else:
            Discharge.append(EDat[int(T_index[i-1]) : int(T_index[i]),
:].copy())

DCapacity = []
CCapacity = []
for i in range(0, len(Discharge)):
    Discharge[i][:, 0] = Discharge[i][:, 0] - Discharge[i][0, 0]
    Discharge[i][:, 0] = Discharge[i][:, 0]*C_Rate
    DCapacity.append(Discharge[i][-1, 0])
for i in range(0, len(Charge)):
    Charge[i][:, 0] = Charge[i][:, 0] - Charge[i][0, 0]
    Charge[i][:, 0] = Charge[i][:, 0]*C_Rate
    CCapacity.append(Charge[i][-1, 0])

DCapacity = np.asarray(np.c_[DCapacity])
CCapacity = np.asarray(np.c_[CCapacity])

L = max([len(DCapacity), len(CCapacity)])
DCapacity = np.pad(DCapacity, ((0, L-len(DCapacity)), (0,0)), 'constant',
constant_values=np.nan)
CCapacity = np.pad(CCapacity, ((0, L-len(CCapacity)), (0,0)), 'constant',
constant_values=np.nan)
Capacities = np.hstack((CCapacity, DCapacity))

for i in range(0, len(Discharge)):
    S_Dat = Moving_Avg(Discharge[i], 5)
    tmp = []
    tme = []
    step = 1
    s_index = 0
    e_index = step
    for j in range(0, len(S_Dat)):
        if e_index < len(S_Dat) and (S_Dat[e_index, 1] - S_Dat[s_index,
1]) != 0:
            tmp.append((S_Dat[e_index, 0] - S_Dat[s_index,
0])/(S_Dat[e_index, 1] - S_Dat[s_index, 1]))
            tme.append((S_Dat[e_index, 1] + S_Dat[s_index, 1])/2)
            s_index = e_index
            e_index = e_index + step

```

```

        elif e_index < len(S_Dat) and (S_Dat[e_index, 1] - S_Dat[s_index,
1]) == 0:
            e_index = e_index + step
            elif e_index >= len(S_Dat):
                break
            tmp = np.c_[tmp]
            tme = np.c_[tme]
            tmp = np.append(tme, tmp, 1)
            tmp.shape
            DDischarge.append(tmp)
        for i in range(0, len(Charge)):
            S_Dat = Moving_Avg(Charge[i], 5)
            tmp = []
            tme = []
            s_index = 0
            e_index = step
            for j in range(0, len(S_Dat)):
                if e_index < len(S_Dat) and (S_Dat[e_index, 1] - S_Dat[s_index,
1]) != 0:
                    tmp.append((S_Dat[e_index, 0] - S_Dat[s_index,
0])/(S_Dat[e_index, 1] - S_Dat[s_index, 1]))
                    tme.append((S_Dat[e_index, 1] + S_Dat[s_index, 1])/2)
                    s_index = e_index
                    e_index = e_index + step
                elif e_index < len(S_Dat) and (S_Dat[e_index, 1] -
S_Dat[s_index, 1]) == 0:
                    e_index = e_index + step
                elif e_index >= len(S_Dat):
                    break
            tmp = np.c_[tmp]
            tme = np.c_[tme]
            tmp = np.append(tme, tmp, 1)
            tmp.shape
            DCharge.append(tmp)
        return EDat, T_index, start_character, Charge, Discharge, DCharge,
DDischarge, Capacities

def segment(data, index):
    s_index = []
    s_index.append(0)
    segments = []
    div = 1
    for i in range(div, len(data)-div):
        s1 = data[i, index] - data[i-div, index]
        s2 = data[i + div, index] - data[i, index]
        diff = s1*s2
        if diff < 0:
            s_index.append(i)
    s_index.append(len(data))
    s_index = np.asarray(s_index)

    for i in range(0, len(s_index)-1):
        segments.append(data[s_index[i] : s_index[i + 1]-1, :])

    return s_index, segments

def VI_find(DRav, D_Dav, EDat, d_type):

```

```

if d_type == "GC Data" or d_type == "OCP Data":

    V = []
    for i in range(0, len(DRav)):
        dx = np.argmin(abs(EDat[:, 0]-DRav[i, 0]))
        V.append(EDat[dx, 1])
    V = np.c_[V]
    DRav = np.append(DRav, V, 1)
    DRav.shape

    V = []
    for i in range(0, len(D_Dav)):
        dx = np.argmin(abs(EDat[:, 0]-D_Dav[i, 0]))
        V.append(EDat[dx, 1])
    V = np.c_[V]
    D_Dav = np.append(D_Dav, V, 1)
    D_Dav.shape

elif d_type == "CV Data":

    V = []
    C = []
    for i in range(0, len(DRav)):
        dx = np.argmin(abs(EDat[:, 0]-DRav[i, 0]))
        V.append(EDat[dx, 1])
        C.append(EDat[dx, 2])
    V = np.c_[V]
    C = np.c_[C]
    DRav = np.append(DRav, V, 1)
    DRav.shape
    DRav = np.append(DRav, C, 1)
    DRav.shape

    V = []
    C = []
    for i in range(0, len(D_Dav)):
        dx = np.argmin(abs(EDat[:, 0]-D_Dav[i, 0]))
        V.append(EDat[dx, 1])
        C.append(EDat[dx, 2])
    V = np.c_[V]
    C = np.c_[C]
    D_Dav = np.append(D_Dav, V, 1)
    D_Dav.shape
    D_Dav = np.append(D_Dav, C, 1)
    D_Dav.shape

return DRav, D_Dav

```

```

def Segment_Force(EDat, T_index, DRav, D_Dav, D_Dav_V, start_character):
    F_Charge = []
    F_Discharge = []
    dF_Charge = []
    dF_Discharge = []
    dFV_Charge = []
    dFV_Discharge = []

```

```

if start_character == 'Discharge':
    for i in range(0, len(Discharge)):
        t_find_1 = EDat[T_index[2*i], 0]
        t_find_2 = EDat[T_index[2*i + 1], 0]
        start = np.argmin(abs(DRav[:, 0] - t_find_1))
        end = np.argmin(abs(DRav[:, 0] - t_find_2))
        F_Discharge.append(DRav[start : end, :].copy())

    for i in range(1, len(Charge) + 1):
        t_find_1 = EDat[T_index[2*i - 1], 0]
        t_find_2 = EDat[T_index[2*i], 0]
        start = np.argmin(abs(DRav[:, 0] - t_find_1))
        end = np.argmin(abs(DRav[:, 0] - t_find_2))
        F_Charge.append(DRav[start : end, :].copy())

    for i in range(0, len(Discharge)):
        t_find_1 = EDat[T_index[2*i], 0]
        t_find_2 = EDat[T_index[2*i + 1], 0]
        start = np.argmin(abs(D_Dav[:, 0] - t_find_1))
        end = np.argmin(abs(D_Dav[:, 0] - t_find_2))
        dF_Discharge.append(D_Dav[start : end, :].copy())

    for i in range(1, len(Charge) + 1):
        t_find_1 = EDat[T_index[2*i - 1], 0]
        t_find_2 = EDat[T_index[2*i], 0]
        start = np.argmin(abs(D_Dav[:, 0] - t_find_1))
        end = np.argmin(abs(D_Dav[:, 0] - t_find_2))
        dF_Charge.append(D_Dav[start : end, :].copy())

    for i in range(0, len(Discharge)):
        t_find_1 = EDat[T_index[2*i], 0]
        t_find_2 = EDat[T_index[2*i + 1], 0]
        start = np.argmin(abs(D_Dav_V[:, 0] - t_find_1))
        end = np.argmin(abs(D_Dav_V[:, 0] - t_find_2))
        dFV_Discharge.append(D_Dav_V[start : end, :].copy())

    for i in range(1, len(Charge) + 1):
        t_find_1 = EDat[T_index[2*i - 1], 0]
        t_find_2 = EDat[T_index[2*i], 0]
        start = np.argmin(abs(D_Dav_V[:, 0] - t_find_1))
        end = np.argmin(abs(D_Dav_V[:, 0] - t_find_2))
        dFV_Charge.append(D_Dav_V[start : end, :].copy())

elif start_character == 'Charge':
    for i in range(1, len(Discharge) + 1):
        t_find_1 = EDat[T_index[2*i - 1], 0]
        t_find_2 = EDat[T_index[2*i], 0]
        start = np.argmin(abs(DRav[:, 0] - t_find_1))
        end = np.argmin(abs(DRav[:, 0] - t_find_2))
        F_Discharge.append(DRav[start : end, :].copy())

    for i in range(0, len(Charge)):
        t_find_1 = EDat[T_index[2*i], 0]
        t_find_2 = EDat[T_index[2*i + 1], 0]
        start = np.argmin(abs(DRav[:, 0] - t_find_1))
        end = np.argmin(abs(DRav[:, 0] - t_find_2))

```

```

F_Charge.append(D_Rav[start : end, :].copy())

for i in range(1, len(Discharge) + 1):
    t_find_1 = EDat[T_index[2*i - 1], 0]
    t_find_2 = EDat[T_index[2*i], 0]
    start = np.argmin(abs(D_Dav[:, 0] - t_find_1))
    end = np.argmin(abs(D_Dav[:, 0] - t_find_2))
    dF_Discharge.append(D_Dav[start : end, :].copy())

for i in range(0, len(Charge)):
    t_find_1 = EDat[T_index[2*i], 0]
    t_find_2 = EDat[T_index[2*i + 1], 0]
    start = np.argmin(abs(D_Dav[:, 0] - t_find_1))
    end = np.argmin(abs(D_Dav[:, 0] - t_find_2))
    dF_Charge.append(D_Dav[start : end, :].copy())

for i in range(1, len(Discharge) + 1):
    t_find_1 = EDat[T_index[2*i - 1], 0]
    t_find_2 = EDat[T_index[2*i], 0]
    start = np.argmin(abs(D_Dav_V[:, 0] - t_find_1))
    end = np.argmin(abs(D_Dav_V[:, 0] - t_find_2))
    dFV_Discharge.append(D_Dav_V[start : end, :].copy())

for i in range(0, len(Charge)):
    t_find_1 = EDat[T_index[2*i], 0]
    t_find_2 = EDat[T_index[2*i + 1], 0]
    start = np.argmin(abs(D_Dav_V[:, 0] - t_find_1))
    end = np.argmin(abs(D_Dav_V[:, 0] - t_find_2))
    dFV_Charge.append(D_Dav_V[start : end, :].copy())

for i in range(0, len(F_Discharge)):
    F_Discharge[i][:, 0] = F_Discharge[i][:, 0] - F_Discharge[i][0, 0]
    F_Discharge[i][:, 0] = F_Discharge[i][:, 0]*C_Rate

for i in range(0, len(F_Charge)):
    F_Charge[i][:, 0] = F_Charge[i][:, 0] - F_Charge[i][0, 0]
    F_Charge[i][:, 0] = F_Charge[i][:, 0]*C_Rate

for i in range(0, len(dF_Discharge)):
    dF_Discharge[i][:, 0] = dF_Discharge[i][:, 0] - dF_Discharge[i][0, 0]
    dF_Discharge[i][:, 0] = dF_Discharge[i][:, 0]*C_Rate

for i in range(0, len(dF_Charge)):
    dF_Charge[i][:, 0] = dF_Charge[i][:, 0] - dF_Charge[i][0, 0]
    dF_Charge[i][:, 0] = dF_Charge[i][:, 0]*C_Rate
return F_Charge, F_Discharge, dF_Charge, dF_Discharge, dFV_Charge,
dFV_Discharge

New_Data = True
EData = True
Writ_Data = True
D_Format = 'Excel' #'txt'
Refinement = 'Numeric Average' #'FFT' #'Numeric Average'
N_Av = 50
dN_Av = 50
Scan_Rate = 0.0002 #V / s
C_Rate = 100 #mA h / g

```

```

Bkg_Sub = True
nbkg = 3

#Process stress data
if New_Data:
    root = Tk()
    filename = askopenfilename()
    print(filename)
    root.destroy()
    if EData:
        root = Tk()
        efilename = askopenfilename()
        print(efilename)
        root.destroy()

    parts = filename.split("/")
    path = ""
    for i in range(0, len(parts)-1):
        path = path + parts[i] + "/"
    name = parts[-1]
    name = name.split(".")
    name = name[0]

    Raw_Data = np.genfromtxt(filename, delimiter = ',')
    num_rows, num_cols = Raw_Data.shape

    dex = []
    for i in range(1, len(Raw_Data)):
        if Raw_Data[i, 0] < Raw_Data[i-1, 0] or Raw_Data[i, 1] <
0.5*Raw_Data[i-1, 1] or Raw_Data[i, 1] > 2*Raw_Data[i-1, 1]:
            dex.append(i)

    Raw_Data = np.delete(Raw_Data, dex, 0)

    if num_cols > 2:
        T_Data = np.asarray(Raw_Data[:, (0, 2)].copy())
        T_Data = Reducing_Avg(T_Data, N_Av)
        hfreq = -1
        lfreq = -1
        tol = 5
        Pmax = 4

        #T_Data, D_TData = Fast_Fourier(T_Data, hfreq, lfreq, tol, Pmax)
        T_Data[:, 0] = T_Data[:, 0]/3600

    Data = Raw_Data[:, (0, 1)].copy()
    Data[:, 0] = Data[:, 0]/3600

if Refinement == 'Numeric Average':
    DRav, D_Dav = Numeric_Average(Data, N_Av, dN_Av)

elif Refinement == 'FFT':
    hfreq = 10
    lfreq = -1
    tol = 5
    Pmax = 1

```

```

print('running fft')
DRav, D_Dav = Fast_Fourier(Data, hfreq, lfreq, tol, Pmax)

#Process EChem Data
if EData:
    d_type, EDat = read_echem_data(efilename)
    if EDat[-1, 0] < DRav[-1, 0]:
        n = len(DRav) - np.where(DRav[:, 0] > EDat[-1, 0])[0][0]
        DRav = DRav[:-n, :]
    if EDat[-1, 0] < D_Dav[-1, 0]:
        n = len(D_Dav) - np.where(D_Dav[:, 0] > EDat[-1, 0])[0][0]
        D_Dav = D_Dav[:-n, :]
    if EDat[-1, 0] < T_Data[-1, 0]:
        n = len(T_Data) - np.where(T_Data[:, 0] > EDat[-1, 0])[0][0]
        T_Data = T_Data[:-n, :]

    if Bkg_Sub:
        Bkg = P_Bkg_Fit(DRav[:, (0, 1)], nbkg)
        DRav[:, 1] = DRav[:, 1] - Bkg

    print('running VI Find')
    DRav, D_Dav = VI_find(DRav, D_Dav, EDat, d_type)

    D_Dav_V = []
    for i in range(1, len(DRav)-1):
        if abs((DRav[i+1, 2] - DRav[i-1, 2])) > 0:
            D_Dav_V.append([DRav[i, 0], DRav[i, 2], (DRav[i+1, 1] - DRav[i-1,
1])/ (DRav[i+1, 2] - DRav[i-1, 2])])
    D_Dav_V = np.asarray(D_Dav_V)

    if d_type == "GC Data":
        EDat, T_index, start_character, Charge, Discharge, DCharge,
DDischarge, Capacities = process_gc(EDat, C_Rate)
        F_Charge, F_Discharge, dF_Charge, dF_Discharge, dFV_Charge,
dFV_Discharge = Segment_Force(EDat, T_index, DRav, D_Dav, D_Dav_V,
start_character)

        colors = pl.cm.jet(np.linspace(0,1,len(Discharge)))
        fig3 = plt.figure()
        ax7 = plt.subplot(1,3,(1,2))

        for i in range(0, len(Discharge)):
            ax7.plot(Discharge[i][:, 0], Discharge[i][:, 1], color=colors[i],
lw = 2)
        for i in range(0, len(Charge)):
            ax7.plot(Charge[i][:, 0], Charge[i][:, 1], color=colors[i], lw =
2)

        ax8 = plt.subplot(1,3,sharey=ax7)
        for i in range(0, len(DDischarge)):
            ax8.plot(DDischarge[i][:, 1], DDischarge[i][:, 0],
color=colors[i], lw = 2)
        for i in range(0, len(DCharge)):
            ax8.plot(DCharge[i][:, 1], DCharge[i][:, 0], color=colors[i], lw
= 2)
        plt.xlim(-3000, 3000)

```



```

        sm = plt.cm.ScalarMappable(cmap='jet', norm=plt.Normalize(vmin=1,
vmax=max([len(Charge), len(Discharge)])))
        sm._A = []
        cb1 = plt.colorbar(sm)
        cb1.set_label('Cycle Number')

        ax7.set_xlabel('Capacity (mA h $g^{-1}$)', fontsize=22)
        ax7.set_ylabel('Cell Potential (V)', fontsize=22)

        ax8.set_xlabel('dQ/dV', fontsize=22)

    elif d_type == "CV Data":
        fig3 = plt.figure()
        ax3 = plt.axes()
        CN, Cycles = segment(EDat, 1)
        ax3.plot(EDat[:, 1], EDat[:, 2]*1000)
        ax3.set_xlabel('Time (h)', fontsize=22)
        ax3.set_ylabel('Current (mA)', fontsize=22)
    elif d_type == "OCP Data":
        fig3 = plt.figure()
        ax3 = plt.axes()
        ax3.plot(EDat[:, 0], EDat[:, 2]*1000)
        ax3.set_xlabel('Time (h)', fontsize=22)
        ax3.set_ylabel('Current (mA)', fontsize=22)
    plt.draw()
    plt.pause(.00001)

if Bkg_Sub and not ('Bkg' in locals()) :
    Bkg = P_Bkg_Fit(DRav[:, (0, 1)], nbkg)
    DRav[:, 1] = DRav[:, 1] - Bkg

#Plotting
fig0 = plt.figure()
ax1 = fig0.add_subplot(111)
ax2 = ax1.twinx()

#host.set_xlim(0, 35)
#host.set_ylim(150, 250)
#par1.set_ylim(-.001, .0025)

ax1.set_xlabel("Time (h)")
ax1.set_ylabel("Stack Force (N)")
ax2.set_ylabel("Cell Temperature (C)")
ax2.set_ylim(0, 30)

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
color3 = plt.cm.viridis(0.2)

p1, = ax1.plot(Data[:, 0], Data[:, 1], color=color1, label="Raw Data")
p2, = ax1.plot(DRav[:, 0], DRav[:, 1], label="DRav")
p3, = ax2.plot(T_Data[:, 0], T_Data[:, 1], color=color2, label="Temp")

lns = [p1, p2, p3]

ax1.legend(handles=lns, loc='best')

```

```

ax1.yaxis.label.set_color(p1.get_color())
ax2.yaxis.label.set_color(p3.get_color())

plt.figure()
ax0 = plt.axes()
ax0.plot(D_Dav[:, 0], D_Dav[:, 1])
ax0.set_xlabel('Time (h)', fontsize=22)
ax0.set_ylabel('dP/dt', fontsize=22)
plt.draw()
plt.pause(.00001)

if EData:
    fig = plt.figure()
    host = fig.add_subplot(111)

    par1 = host.twinx()

    #host.set_xlim(0, 35)
    #host.set_ylim(150, 250)
    #par1.set_ylim(-.001, .0025)

    host.set_xlabel("Time (h)")
    host.set_ylabel("Stack Pressure (Pa)")
    par1.set_ylabel("Current (A)")

    color1 = plt.cm.viridis(0)
    color2 = plt.cm.viridis(0.5)

    p1, = host.plot(DRav[:, 0], DRav[:, 1], color=color1, label="Stack
Pressure")
    if d_type == "GC Data":
        p2, = par1.plot(EDat[:, 0], EDat[:, 1], color=color2, label="Cell
Potential (V)")
    if d_type == "OCP Data":
        p2, = par1.plot(EDat[:, 0], EDat[:, 1], color=color2, label="Cell
Potential (V)")
    if d_type == "CV Data":
        p2, = par1.plot(EDat[:, 0], EDat[:, 1], color=color2, label="Current
(A)")

    lns = [p1, p2]
    host.legend(handles=lns, loc='best')

    host.yaxis.label.set_color(p1.get_color())
    par1.yaxis.label.set_color(p2.get_color())

if Writ_Data:
    big_out = []
    big_out.append(DRav)
    big_out.append(D_Dav)
    Names = ['DRav', 'D_Dav']
    if EData:
        if d_type == "GC Data":
            big_out.append(Capacities)

```

```

big_out.append(Charge)
big_out.append(Discharge)
big_out.append(DCharge)
big_out.append(DDischarge)
big_out.append(F_Charge)
big_out.append(F_Discharge)
big_out.append(dF_Charge)
big_out.append(dF_Discharge)
big_out.append(dFV_Charge)
big_out.append(dFV_Discharge)
Names = ['DRav', 'D_Dav', 'Capacities', 'Charge', 'Discharge',
'DCharge', 'DDischarge', 'F_Charge', 'F_Discharge', 'dF_Charge',
'dF_Discharge', 'dFV_Charge', 'dFV_Discharge']

print("Saving Files to " + path)
os.chdir(path)
if not os.path.isdir(name + " Processed"):
    os.mkdir(name + " Processed")
os.chdir(name + " Processed")
if D_Format == 'Excel':
    Write_To_Excel(big_out, Names, name)
elif D_Format == 'txt':
    Write_To_Text(big_out, Names)
os.chdir(cwd)

```

Appendix D: Preliminary Results for *Operando* Neutron Depth Profiling on All-Solid-State and Interlayer Modified Cells

Neutron depth profiling is a sensitive method to probe the spatial distribution of lithium within a thin sample by leveraging the large neutron absorption cross section of ^6Li (940 barns at a neutron velocity of 2200 m/s) relative to other light atoms. Previous work has successfully used NDP to study mechanistic aspects of lithium electrodeposition from non-aqueous electrolytes onto copper electrodes (a schematic of the experiment is presented in Figure D1).¹ The work discussed in this chapter is preparatory work in collaboration with Professor Anne Co to conduct NDP on all-solid-state and interlayer modified solid state cells (this work was conceived as an extension of the work described in chapter 3). The bulk of this work is aimed at developing a coin-cell-based system that affords suitable electrochemical behavior with the inclusion of a thin copper window to access the Li/LGPS interface while cycling in a vacuum.

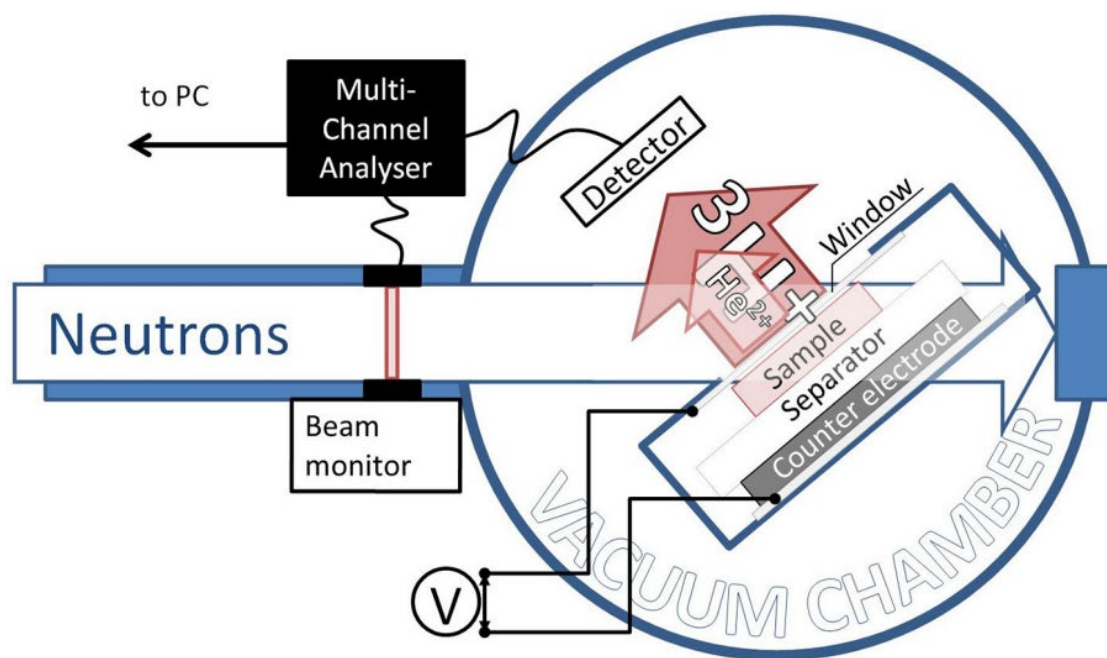


Figure D1. Schematic setup for *operando* neutron depth profiling of a lithium-ion cell.¹

The vacuum stability of fabricated cells was tested using a small vacuum chamber constructed from a Parr autoclave. The enclosure featured a pressure gauge to monitor the strength of the vacuum and a port for electrical feed throughs to connect to the cell (Figure D2). Two different cell geometries were considered (Figures D2c and D2d); the first was a traditional all-solid-state configuration wherein LGPS solid electrolyte pellets directly contacted metallic lithium on one side and Cu foil on the other. Experimentally this assembly was constructed by simply pressing the LGPS pellet along with the Cu window placed at the base of the pellet dye (pellets were typically pressed at room temperature at 1500 PSI for 5 minutes under an inert atmosphere). Pressing the pellet in this manner afforded a very stable direct contact between the metal electrode and the solid electrolyte. The second cell consisted of the same interlayer construction as detailed in chapter 3, wherein a highly concentrated liquid electrolyte (ca. 20 μ L, 7 molal LiTFSI in DOL:DME diluted 5 parts electrolyte to 1 part TTE) was applied to each side of the pellet. In the case of these interlayer cells, the Cu foil window was applied to the pellet after the application of the liquid electrolyte by simply placing it onto the wetted interface.

Regardless of the construction the solid electrolyte stack was then assembled into a coin cell whose top cap had been drilled out using a 6 mm bit to provide access to the Cu window (Figure D2b). Assembled cells were then removed from the glovebox and transported under Argon to the vacuum chamber where they were quickly hooked up to the potentiostat before the atmosphere was removed from the chamber. Unfortunately, despite careful preparation of the electrode stack, and cycling both under argon and in vacuum, cells assembled using Cu foil as the working electrode performed extremely badly when compared to the performance of the typical lithium-lithium symmetric cell (Figures D3a and D3b).

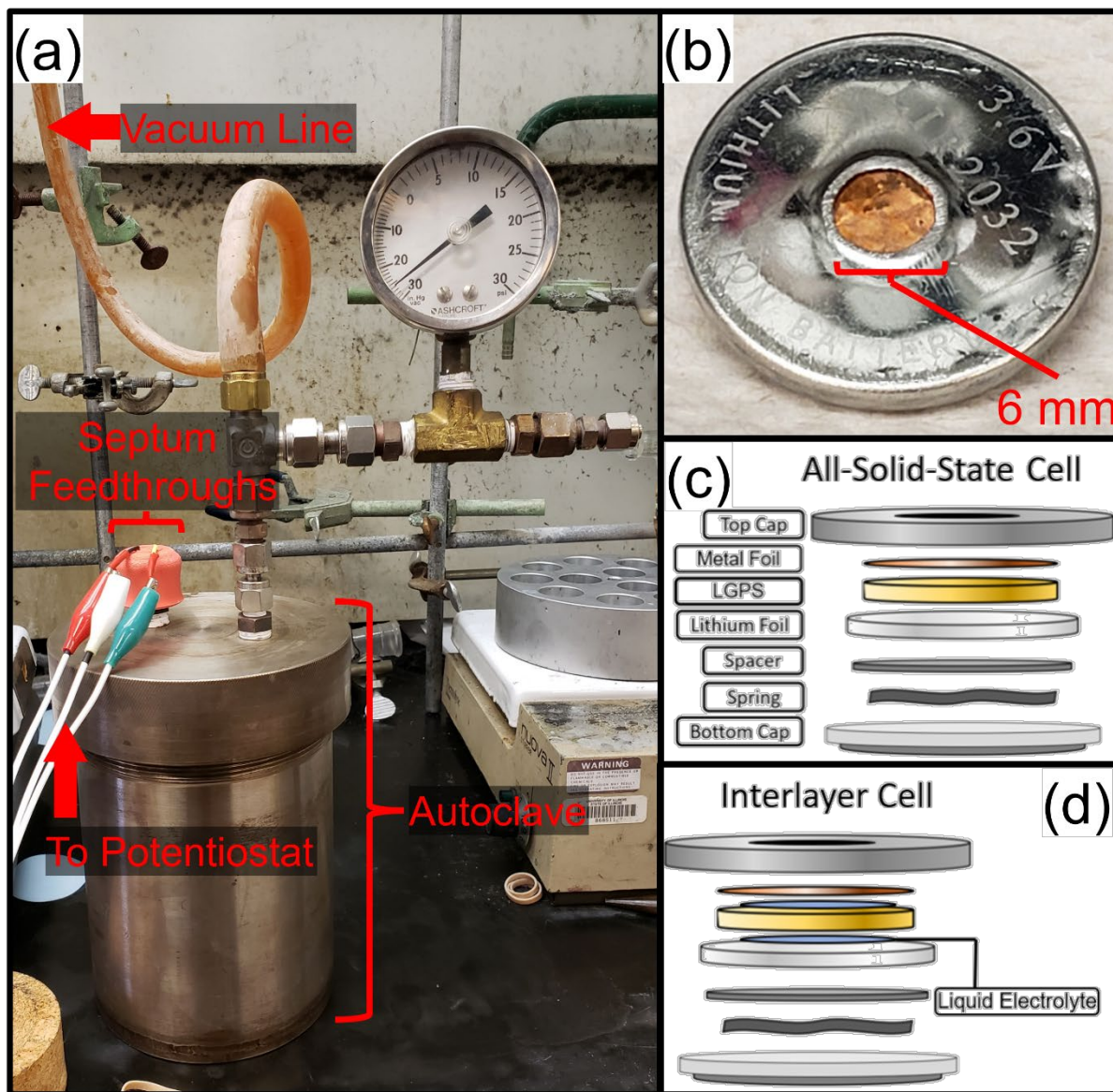


Figure D2. A) Experimental setup used to test the vacuum compatibility of modified coin cells for NDP. B) Photograph of a modified coin cell employing a 6 μm thick Cu window through which neutrons will travel. C and D) schematic depictions of the cell geometries both with and without a liquid interlayer that were tested inside the vacuum chamber.

The poor cycling performance was thought to be the result of either poor contact between the solid electrolyte and the Cu window or of some deleterious chemical instability of the SE in contact with Cu. To address the latter point a variety of metal windows were tested, taking care to interrogate metals which display sufficiently low triton absorption cross sections to make them

feasible as window materials. The results of these tests are presented in Figure D3b, and demonstrate that none of the first row transition metals tested yielded adequate electrochemical performance.

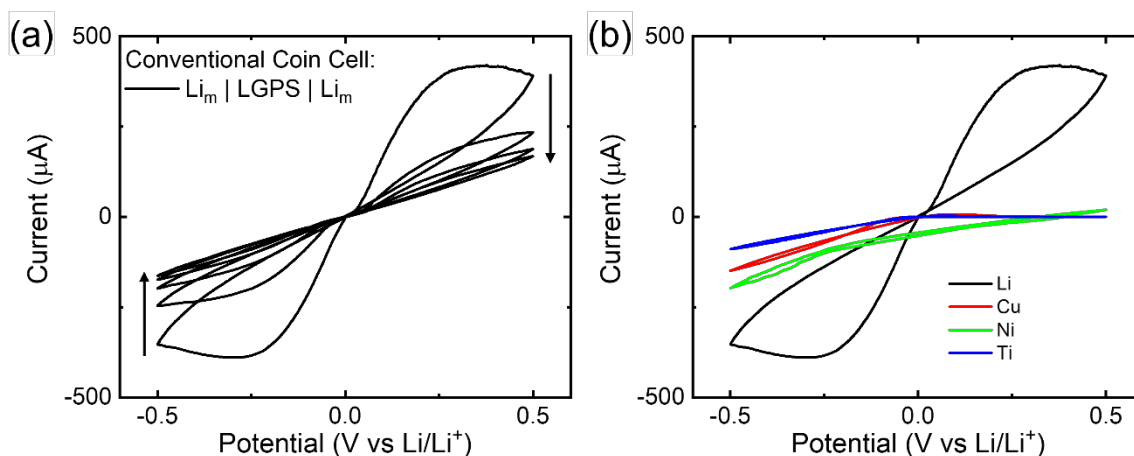


Figure D3. A) Cyclic voltammetry of a conventional lithium-lithium symmetric cell using LGPS as the solid electrolyte collected between -0.5 and 0.5 V at a scan rate of $200 \mu\text{V s}^{-1}$. B) Cyclic voltammetry of various metal foils used as the working electrode in conventional coin cells when paired with LGPS as the electrolyte.

To circumvent whatever instability was causing the poor electrochemical behavior of these systems a thin (ca 100 nm) gold layer was deposited on the $8 \mu\text{m}$ Cu foil to serve as a better electrode contact to the LGPS. While Au is fairly absorbing towards tritium nuclei such a thin layer is not expected to greatly interfere with neutron transmission through the window. This supposition is supported by ballistic trajectory simulations conducted using the “Stopping and Range of Ions in Matter,” (SRIM) package developed by James Ziegler.¹ Figure D4 shows the result of ion bombardment of a thin film solid comprised of an $8 \mu\text{m}$ Cu foil both in the absence (D4a) and presence (D4b) of a 100 nm gold coating. The ion trajectories and penetration depths of the two films towards outgoing triton particles are effectively identical, indicating that the Au coating does not markedly impact the performance of the foil as a window.

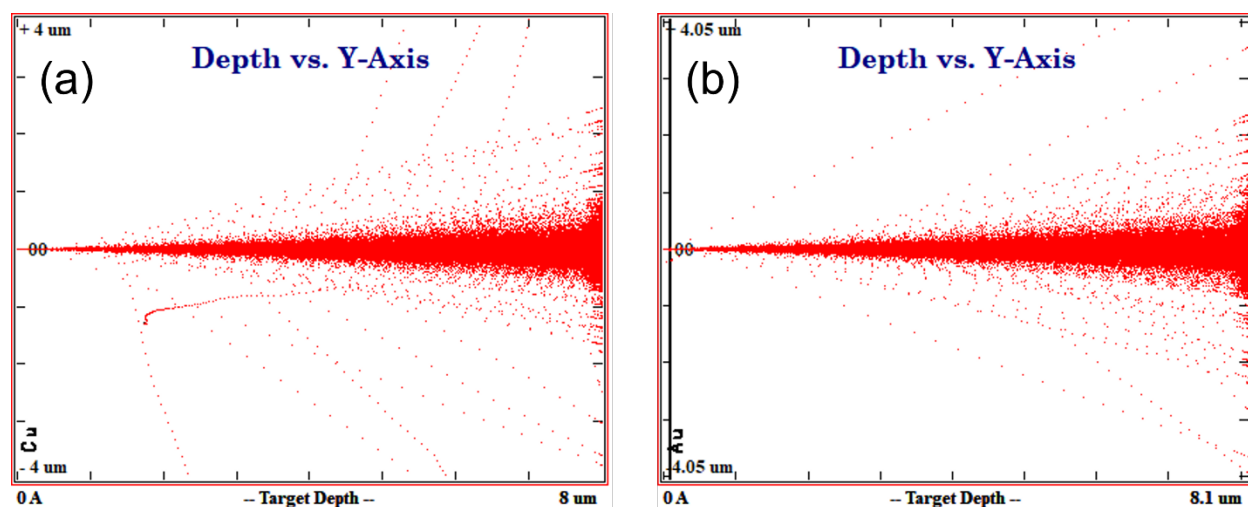


Figure D4. Ion trajectories of tritium nuclei in an 8 μm Cu foil both in the absence (a) and presence (b) of a 100 nm gold coating. Simulations were conducted on 5000 individual bombardments using SRIM with an incident triton energy of 2727 keV and an incident angle of zero degrees.

The application of a thin gold coating to the window material resulted in a significant improvement in the performance of the LGPS solid electrolyte. This improvement is presented in Figure D5 which shows the appearance of oxidative features consistent with lithium stripping from the working electrode, whereas these features are absent for a bare Cu foil. Similarly, the Au coated window allowed the cells to be tested under vacuum as presented in Figure D6. Irrespective of any interface modification cells cycled in vacuum using the Cu@Au windows demonstrate adequate stability with electrochemical behaviors similar to those of cells cycled under an Ar atmosphere. Likewise cells modified with a liquid interlayer display generally improved stability during voltametric cycling, reflecting the behaviors obtained from the same system during *operando* x-ray imaging.

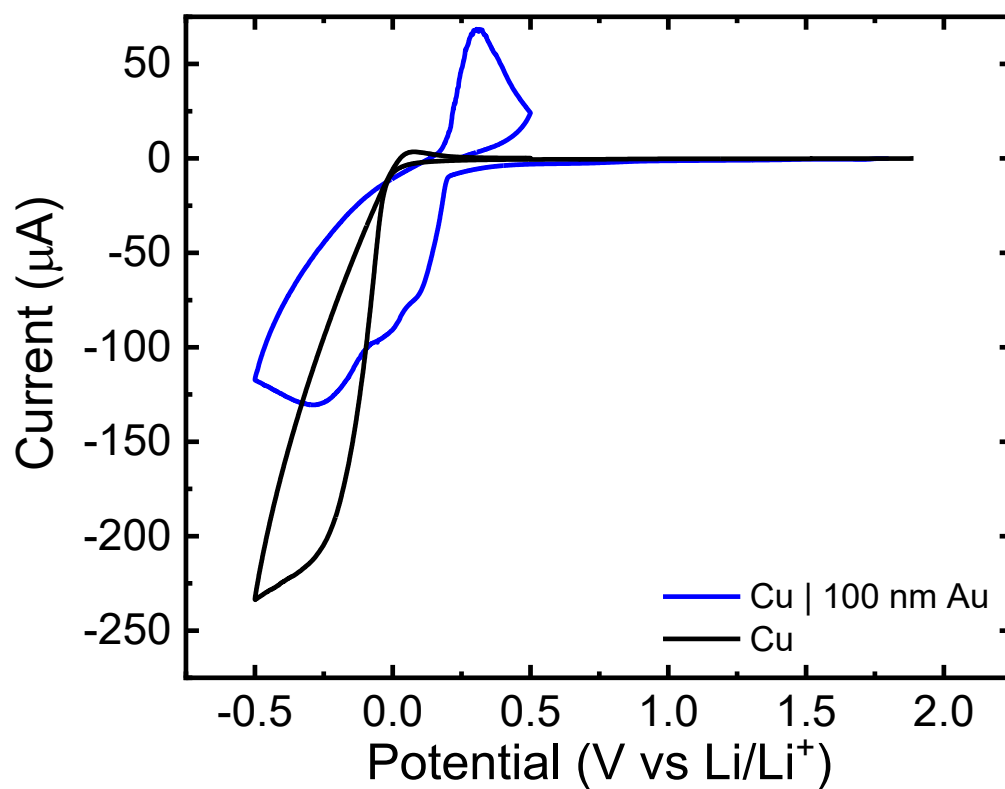


Figure D5. Comparison between the voltametric response of pure Cu windows (black trace) to Cu windows modified with a thin gold coating (blue trace).

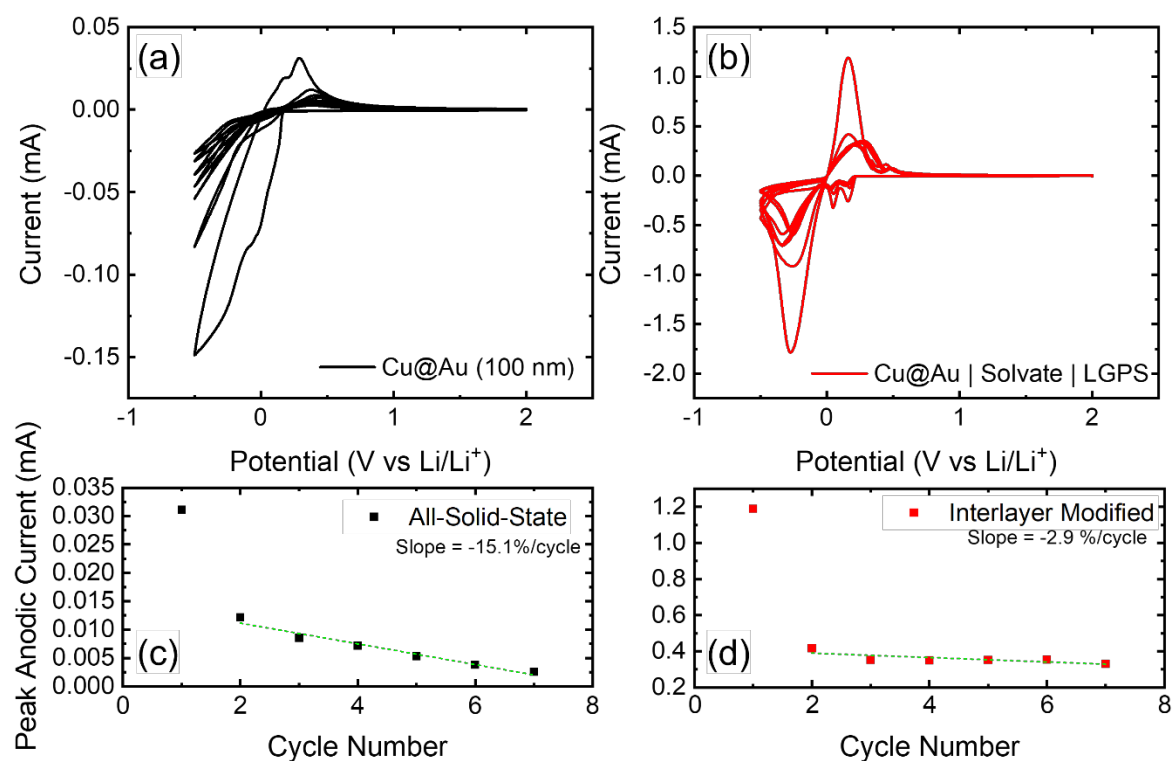


Figure D6. Cyclic voltammetry obtained from (a) bare and (b) interlayer modified LGPS pellets. (c) and (d) present the oxidative peak current intensities obtained from the unmodified and interlayer modified cells respectively.

References:

- (1) Verhallen, T. W.; Lv, S.; Wagemaker, M., Operando Neutron Depth Profiling to Determine the Spatial Distribution of Li in Li-ion Batteries. *Frontiers in Energy Research* **2018**, *6*.

Appendix E: VASP Utilities for Variable Pressure Simulations (Bash Scripts)

```
#!/bin/bash
nstep=15
SPress=0
EPress=750
div="0.001"
z=0

for i in $( seq 0 $nstep )
do
echo "step $i"
mkdir "Step $i"
cp POSCAR "Step $i"
cp KPOINTS "Step $i"
cp POTCAR "Step $i"
cp vdw_kernel.bindat "Step $i"
cp INCAR "Step $i"

cd "Step $i"
val=$((SPress + i * (EPress - SPress) / nstep))
val=$(bc<<<"$val * $div")
echo $val
echo "PSTRESS = $val" >> INCAR
cd ..

if (( i > 0 ))
then
j=$((i-1))
cd "Step $j"
cp CONTCAR ../"Step $i"
cp WAVECAR ../"Step $i"
cd ../"Step $i"
mv CONTCAR POSCAR
mpirun -np 4 vasp_std
cd ..

else
cd "Step $i"
mpirun -np 4 vasp_std
cd ..
fi

n=0
cd "Step $i"
until grep -q "reached required accuracy - stopping structural energy
minimisation" OUTCAR
do
n=$((n+1))
echo "Running Step $i Again $n"
rm POSCAR
mv CONTCAR POSCAR
mpirun -np 4 vasp_std
done

cd ..
done
```

```

#!/bin/bash

nstep=16
>Energies
mkdir CONTCARS

for i in $( seq 0 $nstep )
do
echo "Step $i" >> Energies
tac "Step $i"/OUTCAR | grep -m 1 'energy(sigma->0) =' >>Energies
echo "-----" >> Energies
-----" >> Energies

mkdir CONTCARS/"Step $i"
cp "Step $i"/OUTCAR CONTCARS/"Step $i"

done

cp Energies CONTCARS

>Energies2
for i in $( seq 0 $nstep )
do
TXT1=$(grep 'energy(sigma->0) =' "Step $i"/OUTCAR | tail -1)
TXT2=$(grep 'volume of cell :' "Step $i"/OUTCAR | tail -1)
TXT3=$(grep 'external pressure =' "Step $i"/OUTCAR | tail -1)
IFS=' '
read -ra ADDR <<< "$TXT1"
echo -e "$i\t${ADDR[-1]}" >> Energies2
read -ra ADDR <<< "$TXT2"
echo -e "$i\t${ADDR[-1]}" >> Cell_Volumes
read -ra ADDR <<< "$TXT3"
echo -e "$i\t$TXT3" >> P_Ext
cp Energies2 CONTCARS
cp Cell_Volumes CONTCARS
cp P_Ext CONTCARS
done
gnuplot <<- EOF
    set xlabel "Step #"
    set ylabel "Energy (eV/Cell)"
    set terminal png size 800,600
    set term png font "arial,24"
    set output "EMap.png"
    plot "Energies2" lt 7 lc 7 ps 1.5 lw 3 with linespoints
EOF

gnuplot <<- EOF
    set xlabel "Step #"
    set ylabel "Energy (eV/Cell)"
    set terminal png size 800,600
    set term png font "arial,24"
    set output "V_Map.png"
    plot "Cell_Volumes" lt 7 lc 7 ps 1.5 lw 3 with linespoints
EO

```

Appendix F: Computational Utilities for Tomographic Reconstructions

Centering Code (Python):

```
#!/usr/bin/env python
"""
Code developed in Cycle 2018-3 to calculate the center of rotation
automatically both a 180 and 350 degree spin.
"""

import os
import sys
import cv2
import copy
import time
import h5py
import glob
import psutil
import tomopy
import matplotlib
import numpy as np
from PIL import Image
import multiprocessing
import dxchange.writer
matplotlib.use('tkAgg')
from scipy import ndimage
import matplotlib.pyplot as plt
import tomopy.util.dtype as dtype
from tomopy.util.misc import fft2
from joblib import Parallel, delayed
from tomopy.misc.morph import downsample
#from skimage.filters import unsharp_mask
import tomopyScriptUtilities.BeamHardening_Transmission as bh

# USER SETTINGS
#####

#file information
main_folder = '/scs/people/kmadsen2/Tomography 2019'
data_key = '000' #'ALL' if all folders within main_folder should be
processed, otherwise specify part of dataset name
cfile = '/scs/people/kmadsen2/Tomography LGPS_11_OCP_000.txt' #where to save
calculated centers of rotation
magnification = 5 #specify magnification if it isn't in the HDF5 file

#Recon settings
pad_images = 'edge' #None, zero, edge
pad_fraction_width = 0.5 # fraction of image width to pad on each side, if
pad_images is not None
remove_pad = 1.0 # remove fraction of padded region to make output files
smaller?
algorithm = 'gridrec' # check tomopy docs for options
recon_filter = 'parzen' # check tomopy docs for options. this is for gridrec

#crop image columns - None if no cropping required
start_col = None
```

```

end_col = None

#automatic center of rotation at search row
find_center_automatically = False #using Nghia Vo's method
center_search_row = 260 #row at which to perform the center of rotation
diagnosis
center_location = 'left' #right or left of image midplane, if spinning 360
hist_thresh = 0.01 #used to increase contrast by stretching histogram
contrast_thresh = 0.03 #contrast threshold to set on sinogram row that is
being matched

#Parameters for Nghia Vo's automatic center of rotation method
smin = -1400 #Minimum coarse search radius. Reference to the horizontal
center of the sinogram.
smax = -900 #Maximum coarse search radius. Reference to the horizontal center
of the sinogram.
sstep = 1 #Step size for coarse search radius
ratio = .7 #The ratio between the FOV of the camera and the size of object.
It's used to generate the mask.
drop = 20 #Drop lines around vertical center of the mask.
outputsino = True #output sinogram images for troubleshooting
smooth = True #Whether to apply additional smoothing or not.
srad = 10 #Fine search radius.
step = 0.2 #Step of fine searching.

#write images around desired center of rotation
write_cor_images = True
center_guess = 476.5 #if None, get center from automatated calculation
search_margin = 20 # +/- number of pixels to search from search_margin
search_step = 0.5 #step size of search

#I/O region (for normalizing when doing conversion to path length)
I0_normalize = True
I0_box_size = 40 # pixels
I0_box_loc = 'upper right' # 'lower/upper/center left/right/middle'. Skips
first two rows due to bad pixel values.

# Phase retrieval parameters
phase_retrieval = False
margin = 0 # Num slices at ends to not write (due to phase retrieval
boundary effects)
z = 5 # propagation distance (cm)
eng = 62 # Flux-weighted mean energy of white beam (keV)
alpha = 1e-3 #6e-4 # Smaller number is smoother, larger number is less phase
retrieval correction

# Max number of CPUs for computation and I/O operations
ncore=multiprocessing.cpu_count()-1

# END USER SETTINGS
#####

'''FUNCTION DEFINITIONS BELOW'''

#Read in HDF5 projection data from tomography scan

```

```

def fread_projection_images(data_filename,row_slice=
slice(None,None,1),column_slice=slice(None,None,1)):
    '''Loads the flyscan data from HDF5 files.
    '''
    t0 = time.time()
    with h5py.File(data_filename,'r') as data_hdf:
        data = data_hdf['Prefiltered_images'][:,row_slice,column_slice]
    print('\tElapsed reading data: %i sec\n' % (time.time()-t0))
    return data.astype(np.float32)

#Read in bright and dark fields
def
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slic
e= slice(None,None,1),column_slice=slice(None,None,1),bright_ratio=1):
    '''Loads the bright and dark images.
    '''
    t0 = time.time()
    if bright_filename and 'tif' in bright_filename:
        bright_data_im = Image.open(bright_filename)
        bright_data_pre = np.array(bright_data_im)
    elif bright_filename and 'hdf5' in bright_filename:
        with h5py.File(bright_filename,'r') as bright_hdf:
            bright_data_pre = bright_hdf['Prefiltered_images'][...]
    else:
        with h5py.File(data_filename,'r') as data_hdf:
            bright_data_pre = data_hdf['Bright_field'][...]
    bright_data = bright_data_pre[row_slice,column_slice]
    #remove high intensity pixels in the top-left corner from bright field
    #bright_data = tomopy.misc.corr.remove_outlier1d(bright_data,dif=1000)

    if dark_filename and 'tif' in dark_filename:
        dark_data_im = Image.open(dark_filename)
        dark_data_pre = np.array(dark_data_im)
    elif dark_filename and 'hdf5' in dark_filename:
        with h5py.File(dark_filename,'r') as dark_hdf:
            dark_data_pre = dark_hdf['Prefiltered_images'][...]
    else:
        with h5py.File(data_filename,'r') as data_hdf:
            dark_data_pre = data_hdf['Dark_field'][...]
    dark_data = dark_data_pre[row_slice,column_slice]
    #remove high intensity pixels from dark image
    #dark_data = tomopy.misc.corr.remove_outlier1d(dark_data,dif=1000)
    print('\tElapsed reading bright/dark: %i sec\n' % (time.time()-t0))
    return (bright_data.astype(np.float32) * bright_ratio,
dark_data.astype(np.float32))

#Find IO region
def fIO_slices():
    '''Gives slice objects for the rows and columns for the IO region.
    '''
    #Find the shape of the underlying data
    dataset_shape = fget_dataset_shape(data_filename)
    M = dataset_shape[-2]
    N = dataset_shape[-1]
    mid_row = int((M+IO_box_size)/2.)
    mid_col = int((N+IO_box_size)/2.)
    #Make the location string lower case and split it

```

```

I0_box_split=I0_box_loc.lower().split()
#Figure out the vertical location
slice_M = None
slice_N = None

if I0_box_split[0] == 'lower':
    slice_M = slice(-I0_box_size,None,1)
#avoid very top row due to bad pixel values in top left corner
elif I0_box_split[0] == 'upper':
    slice_M = slice(2,I0_box_size,1)
elif I0_box_split[0] == 'center':
    slice_M = slice(mid_row-I0_box_size,mid_row,1)
else:
    raise ValueError()
#Figure out the horizontal location
if I0_box_split[1] == 'right':
    slice_N = slice(-I0_box_size,None,1)
elif I0_box_split[1] == 'left':
    slice_N = slice(0,I0_box_size,1)
elif I0_box_split[1] == 'middle':
    slice_N = slice(mid_col-I0_box_size,mid_col,1)
else:
    raise ValueError()
return (slice_M,slice_N)

#Calculate I0 region for each projection
def fI0_calculation(row_slices,column_slices,bright_ratio=1):
    '''Computes the I0 region.
    ...
    #Read in the projection data
    proj_data =
fread_projection_images(data_filename,row_slices,column_slices)
    #Read in the bright and dark image subsets
    bright_data,dark_data =
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slices,column_slices, bright_ratio)
    #Perform bright and dark corrections
    proj_data = tomopy.normalize(proj_data,bright_data,dark_data)
    #Average: use 64 bit for accumulator, then down-convert to 32 bit
    return_vals = np.mean(proj_data,axis=(-2,-
1),dtype=np.float64).astype(np.float32)
    #Norm by the maximum value
    return return_vals / np.max(return_vals)

#normalize by I0 region
def fnormalize_image_brightness(projection_data,I0_values):
    '''Uses a small region outside the sample to normalize for shifts
    in image intensity.
    ...
    t0=time.time()
    print("Normalizing by I0 box ...")
    projection_data /= I0_values[:,None,None]
    print(("Elapsed normalizing image by I0: {0:6.3f}
sec".format(time.time() - t0)))
    return projection_data

#Get shape of tomography data

```



```

def fget_dataset_shape(data_filename,row_slice=
slice(None,None,1),column_slice=slice(None,None,1)):
    with h5py.File(data_filename,'r') as data_hdf:
        return data_hdf['Prefiltered_images'].shape

#remove stripes from projection data
def fremove_stripes(dataset,method='sf'):
    '''Method for removing stripes from the dataset.
    '''
    t0=time.time()
    print("Removing stripes/rings...")
    if method == 'fw':
        dataset =
tomopy.prep.stripe.remove_stripe_fw(dataset,level=8,wname='sym16',sigma=1,pad
=True,ncore=ncore) # Fourier Wavelet method
    elif method == 'ti':
        dataset = tomopy.prep.stripe.remove_stripe_ti(dataset) # Titarenko
algorithm
    else:
        dataset = tomopy.prep.stripe.remove_stripe_sf(dataset, 10,
ncore=ncore) # Smoothing filter stripe removal
        print(("Elapsed removing chunk stripes: {0:6.3f}
sec".format(time.time() - t0)))
    return dataset

#phase retrieval algorithm, if using
def fapply_phase_retrieval(dataset):
    t0=time.time()
    print("Apply phase retrieval...")
    dataset =
tomopy.prep.phase.retrieve_phase(dataset,pixel_size=pixel_size,dist=z,energy=
eng,alpha=alpha,pad=True,ncore=ncore)
    print(("Elapsed in phase retrieval: {0:6.3f} sec".format(time.time() -
t0)))
    return dataset

#re-register data for 360 deg scans onto 0-180 deg
def
freregister_data(dataset,theta,rotation_center,center_location,delta_theta):
    #If we have > 180 degrees of projections, reregister
    if np.max(theta) - np.min(theta) > np.pi:
        print("\nRe-registering projections for span > 180 degrees...")
        #Find starting point for angles > 180 degrees, account for roundoff
error
        index_180 = np.argmax(theta >= (np.pi - delta_theta / 10.0)) #Returns
first argument where this is True
        #Make new theta array
        new_theta = theta[:index_180]
        #Verify that the angles before and after 180 degrees match closely
        if
(np.allclose(new_theta+np.pi,theta[index_180:2*index_180],delta_theta /
10.0)):
            print("Angles before and after 180 degrees match.")
        else:
            raise ValueError()
        #Round the center of rotation to the nearest half pixel.
        rotation_center = np.round(rotation_center * 2.0) / 2.0

```

```

        #Find new width of array
        old_image_center = (dataset.shape[2] - 1.0) / 2.0          #In between
middle two pixels
        overlap_width = float(dataset.shape[2]) - 2.0*
np.abs(old_image_center - rotation_center)
        new_width = dataset.shape[2] + int(2.0 * np.abs(old_image_center -
rotation_center))
        new_center = float(new_width - 1) / 2.0
        #Make new dataset array
        data_rereg =
tomopy.sino_360_t0_180(dataset,overlap_width,center_location)
        data_size = float((dataset.nbytes)/1e9)
        del(dataset)
        rereg_size = float((data_rereg.nbytes)/1e9)
        return(data_rereg,new_theta,new_center)
    else:
        print("No reregistration of angles needed.")
        return(dataset.copy(),theta.copy(),rotation_center.copy())

#pad images before reconstruction
def fpad_image(dataset,center_value,pad_fraction_width):
    '''Pads images with the desired values.
    '''
    if pad_images is None:
        return (dataset,center_value)

    t0=time.time()
    prev_width = dataset.shape[2]
    print("\nPadding array...", dataset.shape)
    pad_pixels=int(np.ceil(prev_width*pad_fraction_width))
    if pad_images == 'constant':
        padded_dataset = tomopy.pad(dataset,npad = pad_pixels, axis = 2,
mode='constant',ncore=ncore,constant_values=1.0)
    elif pad_images == 'edge':
        padded_dataset = tomopy.pad(dataset,npad = pad_pixels,axis = 2,
mode='edge',ncore=ncore)
    else:
        raise ValueError("pad_images = "+pad_images+" not understood!")
    print("\tNew size:",padded_dataset.shape)
    print("\tElapsed %i sec\n" % (time.time()-t0))
    #Center offset
    center_offset = (padded_dataset.shape[2]-dataset.shape[2])/2
    pad_size = float((padded_dataset.nbytes)/1e9)
    return (padded_dataset,center_value + center_offset)

#write out the tiffstack of reconstructed images
def fwrite_tiff_stack(recon_dataset,smin,total_rows):
    #Write 32-bit tiff stack out
    t0=time.time()
    #Leave out margin rows in case we did phase retrieval
    start_write_row = 1 if (smin and margin) else 0
    last_write_row = -1 if (recon_dataset.shape[0] - total_rows and margin)
else 0
    start_write_row += start_row + smin
    last_write_row += start_row + smin + recon_dataset.shape[0]
    print(("Writing slices {0:d}-{1:d} to {2:s}...".format(
        start_write_row, last_write_row - 1,output_path.tiffstack)))

```

```

    for i in range(start_write_row,last_write_row):
        j = i - start_row - smin

dxchange.write_tiff(recon_dataset[j,...],fname=output_path_tiffstack+'img_{0:
04d}'.format(i),overwrite=True)
    print('\tElapsed %i sec\n' % (time.time()-t0))

#reconstruct projections
def freconstruct_projections(dataset,theta,center,final_size):
    #Tomographic reconstruction
    t0=time.time()
    print("\nReconstructing...")
    data_recon = tomopy.recon(dataset,
theta,center=center,algorithm=algorithm,ncore=ncore,filter_name=recon_filter)
    print('\tElapsed %i sec\n' % (time.time()-t0))
    del(dataset,theta,center)
    #Crop the image to the desired size
    #print(("Cropping volume from size {0:d} to size
{1:d}".format(data_recon.shape[1],final_size)))
    crop_margin = int((data_recon.shape[1] - final_size) / 2.0)
    if crop_margin > 0:
        data_recon =
data_recon[:,crop_margin:crop_margin+final_size,crop_margin:crop_margin+final
_size]
        #Rearrange output (Dan notices that the images are flipped vertically!)
        data_recon = np.fliplr(data_recon)
        #print(("Chunk output stats: ", data_recon.shape))
        #print(("Nans:
",np.sum(np.isnan(data_recon)), '/', np.product(data_recon.shape)))
        #print(("Infs:
",np.sum(np.isinf(data_recon)), '/', np.product(data_recon.shape)))
        #Mask NaNs and INFs from the output data.
        data_recon[np.isnan(data_recon)]=0

data_recon[np.isinf(data_recon)]=np.nanmax(data_recon)*np.sign(data_recon[np.
isinf(data_recon)])
    print('\tTotal elapsed to reconstruct subset = {0:3.3f}
s.'.format(time.time() - t0))
    return data_recon

#remove outliers with a median filter
def remove_outliers(data,dif,size):
    out = tomopy.misc.corr.remove_outlier1d(data,dif=dif,size=size,axis=1)
    return out

#write out images with different center of rotation guesses
def write_center_images(data,theta,guess,cor,step_theta,path):
    old_image_center = (data.shape[2] - 1.0) / 2.0 #In between middle two
pixels
    overlap_width = float(data.shape[2]) - 2.0* np.abs(old_image_center -
cor)
    desired_size = data.shape[2] + int(2.0 * np.abs(old_image_center - cor))
    reg_data, reg_theta, reg_center =
frregister_data(data,theta,guess,center_location,step_theta)
    del(data)
    #Mask huge values, NaNs and INFs from the filtered data
    maxv=1e5

```

```

reg_data[np.isnan(reg_data)]=0

reg_data[np.isinf(reg_data)]=np.nanmax(reg_data)*np.sign(reg_data[np.isinf(re
g_data)])

reg_data[np.abs(reg_data)>maxv]=maxv*np.sign(reg_data[np.abs(reg_data)>maxv])
(pad_data, pad_center) =
fpad_image(reg_data,reg_center,pad_fraction_width=pad_fraction_width)
del(reg_data)
data_recon =
freconstruct_projections(pad_data,theta,pad_center,desired_size)
del(pad_data)
data_recon = tomopy.remove_ring(data_recon.copy(),ncore=ncore)
dxchange.write_tiff(data_recon[0,...],fname =
path+'%0.1f'%guess,overwrite=True)
return

# Nghia Vo's code from TomoPy v.1.3.0 copied below with minor edits that
allow for COR diagnosing #####

def find_center_vo(tomo,in180,ind=None, smin=-50, smax=50, sstep = 5,
outputsino = True, srاد=6, step=0.5, ratio=0.5, drop=20, smooth=False):
    """
    Find rotation axis location using Nghia Vo's method. :cite:`Vo:14`.

    Parameters
    -----
    tomo : ndarray
        3D tomographic data.
    ind : int, optional
        Index of the slice to be used for reconstruction.
    smin, smax : int, optional
        Coarse search radius. Reference to the horizontal center of the
    sinogram.
    sstep: int, optional
        Step size used in coarse search
    srاد : float, optional
        Fine search radius.
    step : float, optional
        Step of fine searching.
    outputsino: bool, optional
        Output sinograms into file path for diagnosing center of rotation
    ratio : float, optional
        The ratio between the FOV of the camera and the size of object.
        It's used to generate the mask.
    drop : int, optional
        Drop lines around vertical center of the mask.
    smooth : bool, optional
        Whether to apply additional smoothing or not.

    Returns
    -----
    float
        Rotation axis location.
    """
    tomo = dtype.as_float32(tomo)

```

```

if ind is None:
    ind = tomo.shape[1] // 2
    _tomo_pre = tomo[:, ind, :]
    _tomo = copy.deepcopy(_tomo_pre)

    #improve contrast by tightening up histogram
    hist = np.histogram(_tomo_pre,bins=100)
    histg = abs(np.gradient(hist[0]))
    maxval = np.max(histg)
    l = len(histg)
    #set thresholds based on gradient peaks
    threshl = hist[1][np.argmax(histg>hist_thresh*maxval)]
    lh = len(histg[int(l/2):])
    histgr = np.flip(histg)
    threshh = hist[1][int(l-(np.argmax(histgr>hist_thresh*maxval)))]
    plt.figure(1,figsize=(11,8.5))
    plt.title('Original Histogram')
    plt.plot(hist[1][1:],hist[0],'k-',lw=2)
    plt.xlabel('Intensity')
    plt.ylabel('Counts')
    plt.axvline(threshl,ls='--',color='b',label = 'Lower threshold')
    plt.axvline(threshh,ls='--',color = 'g', label = 'Upper threshold')
    plt.ylim(0,1.1*np.max(hist[0]))
    plt.legend(loc='best')
    #plt.show()
    #exit()
    plt.savefig(file_path+'SinoHist.png')
    plt.close(1)

    _tomo[_tomo_pre<threshl]= threshl
    _tomo[_tomo_pre>threshh] = threshh

    # Reduce noise by smooth filters. Use different filters for coarse and
    fine search
    if smooth:
        _tomo_cs_pre = ndimage.filters.gaussian_filter(_tomo,(3,3))
        _tomo_fs_pre = ndimage.filters.gaussian_filter(_tomo,(2,2))
        #use unsharp mask to accentuate features
        lap_cs = cv2.Laplacian(_tomo_cs_pre,cv2.CV_32F)
        lap_fs = cv2.Laplacian(_tomo_fs_pre,cv2.CV_32F)
        strength = 2
        _tomo_cs = _tomo_cs_pre - strength*lap_cs
        _tomo_fs = _tomo_fs_pre - strength*lap_fs
    else:
        _tomo_cs = _tomo
        _tomo_fs = _tomo
    #_tomo_cs = ndimage.filters.gaussian_filter(_tomo, (3, 1)) if smooth else
    _tomo
    #_tomo_fs = ndimage.filters.median_filter(_tomo, (2, 2)) if smooth else
    _tomo

    # Coarse and fine searches for finding the rotation center.
    if _tomo.shape[0] * _tomo.shape[1] > 4e6: # If data is large (>2kx2k)
        _tomo_coarse = downsample(np.expand_dims(_tomo_cs,1), level=2)[: , 0,
:]
        sc=4

```

```

        init_cen = _search_coarse(_tomo_coarse,inl80,sc,smin / 4.0, smax /
4.0, sstep, ratio, drop,outputsino)
        fine_cen = _search_fine(_tomo_fs, srاد, step, init_cen*4, ratio,
drop)
    else:
        sc=1
        init_cen = _search_coarse(_tomo_cs,inl80,sc, smin, smax,sstep, ratio,
drop,outputsino)
        fine_cen = _search_fine(_tomo_fs, srاد, step, init_cen, ratio, drop)
    return fine_cen

def _search_coarse(sino,inl80,sc,smin, smax, sstep, ratio, drop,outputsino):
    """
    Coarse search for finding the rotation center.
    """
    (Nrow, Ncol) = sino.shape
    centerfliplr = (Ncol - 1.0) / 2.0

    # Copy the sinogram and flip left right, the purpose is to
    # make a full [0;2Pi] sinogram
    _copy_sino = np.fliplr(sino[1:])

    # This image is used for compensating the shift of sinogram 2
    temp_img = np.zeros((Nrow - 1, Ncol), dtype='float32')
    temp_img[:, :] = np.flipud(sino)[1:]

    #plot sinograms
    """
    plt.figure(1)
    plt.imshow(sino,cmap='gray')
    plt.figure(2)
    plt.imshow(_copy_sino,cmap='gray')
    plt.figure(3)
    plt.imshow(temp_img,cmap='gray')
    plt.show()
    """

    # Start coarse search
    listshift = np.arange(smin, smax + 1,sstep).astype(int)
    listmetric = np.zeros(len(listshift), dtype='float32')
    mask = _create_mask(2 * Nrow - 1, Ncol, 0.5 * ratio * Ncol, drop)
    sino_sino = np.vstack((sino, _copy_sino))
    abs_fft2_sino = np.empty_like(sino_sino)
    for (num,i) in enumerate(listshift):
        _sino = sino_sino[len(sino):]
        _sino[...] = np.roll(_copy_sino, i, axis=1)
        if i >= 0:
            _sino[:, 0:i] = temp_img[:, 0:i]
        else:
            _sino[:, i:] = temp_img[:, i:]
        fft2sino = np.fft.fftshift(fft2(sino_sino))
        np.abs(fft2sino, out=abs_fft2_sino)
        abs_fft2_sino *= mask
        listmetric[num] = abs_fft2_sino.sum()
    if outputsino ==True:

```

```

dxchange.write_tiff(sino_sino, fname=file_path+'Sinograms_coarse/img_%i'%num,
overwrite=True)

#dxchange.write_tiff(abs_fft2_sino, fname=file_path+'FFT/img_%i'%num, overwrite
=True)
    minpos = np.argmin(listmetric)
    normetric = (listmetric-listmetric.min())/(listmetric.max()-
listmetric.min())
    out = start_col + sc*(centerfliplr + listshift/2)
    cent = start_col + sc*(centerfliplr + listshift[minpos]/2)
    fig = plt.figure(1, figsize=(8.5, 11))
    plt.plot(out, normetric, 'ok--', label='center=%0.2f'%cent)
    plt.xlabel('center loc. [px]')
    plt.ylabel('metric [a.u.]')
    plt.legend(loc='best')
    fig.savefig(file_path+'VO_coarse.png')
    plt.close(fig)
    return centerfliplr + listshift[minpos] / 2.0

def _search_fine(sino, srad, step, init_cen, ratio, drop):
    """
    Fine search for finding the rotation center.
    """
    Nrow, Ncol = sino.shape
    centerfliplr = (Ncol + 1.0) / 2.0 - 1.0
    # Use to shift the sinogram 2 to the raw CoR.
    shiftsino = np.int16(2 * (init_cen - centerfliplr))
    _copy_sino = np.roll(np.fliplr(sino[1:]), shiftsino, axis=1)
    if init_cen <= centerfliplr:
        lefttake = np.int16(np.ceil(srad + 1))
        righttake = np.int16(np.floor(2 * init_cen - srad - 1))
    else:
        lefttake = np.int16(np.ceil(init_cen - (Ncol - 1 - init_cen) + srad +
1))
        righttake = np.int16(np.floor(Ncol - 1 - srad - 1))
    Ncoll = righttake - lefttake + 1
    mask = _create_mask(2 * Nrow - 1, Ncoll, 0.5 * ratio * Ncol, drop)
    numshift = np.int16((2 * srad) / step) + 1
    listshift = np.linspace(-srad, srad, num=numshift)
    listmetric = np.zeros(len(listshift), dtype='float32')
    factor1 = np.mean(sino[-1, lefttake:righttake])
    factor2 = np.mean(_copy_sino[0, lefttake:righttake])
    _copy_sino = _copy_sino * factor1 / factor2
    num1 = 0
    for (num, i) in enumerate(listshift):
        _sino = ndimage.interpolation.shift(
            _copy_sino, (0, i), prefilter=False)
        sinojoin = np.vstack((sino, _sino))
        listmetric[num] = np.sum(np.abs(np.fft.fftshift(np.fft.fft2(sinojoin[:,
lefttake:righttake + 1])) * mask)
            if outputsino == True:
                dxchange.write_tiff(sinojoin[:, lefttake:righttake +
1], fname=file_path+'Sinograms_fine/img_%i'%num, overwrite=True)
        minpos = np.argmin(listmetric)
        normetric = (listmetric-listmetric.min())/(listmetric.max()-
listmetric.min())

```

```

out = start_col + init_cen + listshift/2
cent = start_col + init_cen + listshift[minpos]/2
fig = plt.figure(1,figsize=(8.5,11))
plt.plot(out,normmetric,'og--',label='center=%0.2f'%cent)
plt.xlabel('center loc. [px]')
plt.ylabel('metric [a.u.]')
plt.legend(loc='best')
fig.savefig(file_path+'VO_fine.png')
plt.close(fig)
return init_cen + listshift[minpos] / 2.0

def _create_mask(nrow, ncol, radius, drop):
    du = 1.0 / ncol
    dv = (nrow - 1.0) / (nrow * 2.0 * np.pi)
    centerrow = np.int16(np.ceil(nrow / 2) - 1)
    centercol = np.int16(np.ceil(ncol / 2) - 1)
    mask = np.zeros((nrow, ncol), dtype='float32')
    for i in range(nrow):
        num1 = np.round(((i - centerrow) * dv / radius) / du)
        (p1, p2) = np.int16(np.clip(np.sort(
            (-int(num1) + centercol, num1 + centercol)), 0, ncol - 1))
        mask[i, p1:p2 + 1] = np.ones(p2 - p1 + 1, dtype='float32')
    if drop < centerrow:
        mask[centerrow - drop:centerrow + drop + 1,
            :] = np.zeros((2 * drop + 1, ncol), dtype='float32')
    mask[:, centercol-1:centercol+2] = np.zeros((nrow, 3), dtype='float32')
    return mask

#calculate correct amount of images to load at once for system
def calculate_chunksize_cor(data,ims):
    #automated way to calculate the chunk size
    mem = psutil.virtual_memory()
    avmem = mem.available/(1e9) #system memory that is available and not
    swap, in GB
    #assume that we want 1/4 of the system memory to be free for other things
    and can therefore use 3/4 of it
    threshold = avmem*(3/4.)
    #calculate about how much RAM is going to be needed for the whole recon
    set_size = (4*np.size(data))/(1e9) #dataset gets converted to 32 bit i.e.
    4 bytes per bit
    pad_size = (set_size*pad_fraction_width)*2+set_size #size after padding
    occurs
    recon_size = pad_size - (set_size*(pad_fraction_width*remove_pad))*2
    #size of reconstruction
    #memory required if nothing gets deleted in time on any of the usable
    cores
    memreq = (set_size*2+pad_size+recon_size)*ncore*ims
    chunkfrac = threshold/memreq
    #find chunksize that can be supported in system memory using n cores
    rchunk = int(chunkfrac*ims)
    print(rchunk,ims)
    return rchunk

#####
#####
# Main function to do tomoPy reconstruction. Input parameters are defined at
the top of the script.

```



```

# This version will load ALL the images and work on chunks of specified size
in RAM.
# Chunks are overlapped by some 'margin' to avoid edge phase retrieval
errors.
### MAIN
#####
#####

def ftomopy_script():

    print(("TomoPy version = " + tomopy.__version__))

    #read in necessary properties from HDF5 file
    with h5py.File(data_filename,'r') as data_hdf:
        bright_exp =
data_hdf['measurement']['instrument']['detector']['brightfield_exposure_time'
][...] #bright field exposure time, units of seconds
        exp =
data_hdf['measurement']['instrument']['detector']['exposure_time'][...]
#images exposure time, units of seconds
        bright_ratio = exp/bright_exp #multiply bright field intensity values
by this ratio
        #load in magnification
        try:
            mag =
data_hdf['measurement']['instrument']['detector']['magnification'][...]
        except:
            try:
                mag =
data_hdf['measurement']['instrument']['detector']['setup']['Magnification'][.
..]
            except KeyError:
                mag = magnification
        cam_model =
str(data_hdf['measurement']['instrument']['detector']['model'][...])
        #retrieve pixel sizes from camera model name
        if cam_model == 'Grasshopper3 GS3-U3-123S6M':
            pixel_size = 3.45e-4/mag
        elif cam_model == 'Grasshopper3 GS3-U3-23S6M':
            pixel_size = 5.86e-4/mag
        #figure out theta range
        start_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_Start'][...])
        end_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_End'][...])
        step_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_Step'][...])
        #Span of angles for reconstruction; units radians
        theta = np.arange(start_theta,end_theta+step_theta,step_theta)

        #set slices to reconstruct
        row_slice = center_search_row
        column_slice = slice(None,None,1)

```

```

    #Import one projection image
    projection_data_full =
fread_projection_images(data_filename,row_slice,column_slice)
    #expand dimensions so that data has 3 axes
    projection_data_full = np.expand_dims(projection_data_full,axis=1)

    #Load the bright and dark images. Do over the same subset of the image
as the original data.
    bright_median, dark_median =
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slic
e,column_slice,bright_ratio)

    #Normalize the projection images by the bright and dark fields
    projection_data_full =
tomopy.normalize(projection_data_full,bright_median,dark_median)

    #Find the normalization values
    if I0_normalize:
        I0_row_slice,I0_column_slice = fI0_slices()
        I0_values = fI0_calculation(I0_row_slice,
I0_column_slice,bright_ratio)
        #Normalize by the I0 region.
        projection_data_full =
fnormalize_image_brightness(projection_data_full,I0_values)

#     plt.figure(1)
#     plt.imshow(projection_data_full[:,0,:],cmap='gray') #plot sinogram
#     plt.title('After Normalization')
#     plt.colorbar()
#     plt.show()

    #Remove stripes
    projection_data_full = fremove_stripes(projection_data_full)
    #Take log
    projection_data_full = tomopy.minus_log(projection_data_full)
    #Apply phase retrieval, if desired
    if phase_retrieval:
        projection_data_full = fapply_phase_retrieval(projection_data_full)

    '''
    plt.figure(1)
    plt.imshow(sub_data[:,0,:],cmap='gray',vmin=0,vmax=1) #plot sinogram
    plt.title('Slice for COR Calculation')
    plt.colorbar()
    plt.show()
    '''

    if find_center_automatically:
        '''
        Find center automatically using TomoPy Nghia Vo's method
        '''
        print("Diagnosing the center of rotation automatically...")
        in180 = np.argmax(theta >= (np.pi - step_theta / 10.0)) #Returns
first argument where this is True
        shift = 0
        sino_roll =
np.roll(projection_data_full[... ,start_col:end_col],shift,axis=0)

```

```

sino = sino_roll[:inl80+1,...]
meani = np.abs(np.mean(sino[-5:,0,...],axis=0))
diff = meani.max()-meani.min()
#look at the gradient at the boundary

while diff<contrast_thresh:
    shift+=100
    sino_roll =
np.roll(projection_data_full[...,start_col:end_col],shift,axis=0)
    sino = sino_roll[:inl80+1,...]
    meani = np.abs(np.mean(sino[-5:,0,...],axis=0))
    diff = meani.max()-meani.min()
    print("final shift is %i"%shift)
    cor = find_center_vo(sino,inl80,ind=0, smin=smin, smax=smax, sstep =
sstep, outputsino = outputsino, srاد=srad, step=step,
        ratio=ratio, drop=drop, smooth=smooth) + start_col #adjust
for start_col

    print("Center of rotation is at %0.2f px"%cor)

#create text file with centers of rotation
if os.path.isfile(cfile):
    with open(cfile) as f:
        modlines = []
        for line in f:
            if os.path.basename(data_filename) in line:
                modlines.append("%s %0.3f\n"
%(os.path.basename(data_filename),cor))
            else:
                modlines.append(line)
        if not [k for k in modlines if
os.path.basename(data_filename) in k]:
            modlines.append("%s %0.3f\n"
%(os.path.basename(data_filename),cor))
        with open(cfile,'w') as f:
            for line in modlines:
                f.write(line)
    else:
        with open(cfile, 'w+') as f:
            f.write("%s %0.3f\n" %(os.path.basename(data_filename),cor))

write_center_images(projection_data_full,theta,cor,cor,step_theta,path=file_p
ath+'/AutoCenter_Vo_')

if write_cor_images:
    print("Writing images for center of rotation check ...")
    if find_center_automatically is False:
        rot_guess = np.arange(center_guess - search_margin, center_guess
+ search_margin,search_step)
        guess_cor = center_guess
    else:
        rot_guess = np.arange(cor - search_margin, cor +
search_margin,search_step)
        guess_cor = cor
    rchunk = calculate_chunksize_cor(projection_data_full,len(rot_guess))
    print(rchunk)

```

```

        for i in range(1, len(rot_guess), rchunk):
            sim = i-1
            eim = i+rchunk-1
            if eim>len(rot_guess):
                eim=len(rot_guess)
            rot_set = rot_guess[sim:eim]

Parallel(n_jobs=ncore) (delayed(write_center_images) (projection_data_full, thet
a, guess, guess_cor, step_theta, path=file_path+'/DiagnoseCenter/img_') for guess
in rot_set)
        del(projection_data_full)

### MAIN
#####

if __name__ == "__main__":
    start_time = time.time()
    all_folders = next(os.walk(main_folder))[1]
    if data_key == 'ALL':
        subfold = [os.path.join(main_folder, j) for j in all_folders]
    else:
        subfold = [os.path.join(main_folder, j) for j in all_folders if
data_key in j]
    for item in subfold:
        infold = next(os.walk(item))[1]
        done = [j for j in infold if 'DiagnoseCenter' in j]
        if done:
            print("Center has already been diagnosed for %s. Moving to next
folder..."%os.path.basename(item))
            pass
        else:
            data_filename = glob.glob(item+ '/*Prefiltered.hdf5')[0]
            file_path = os.path.dirname(data_filename)+'/'
            bright_filename = None
            dark_filename = None
            #Give slice objects with the image ROI
            p, r, c = fget_dataset_shape(data_filename, row_slice=
slice(None, None, 1), column_slice=slice(None, None, 1))
            if start_col is None:
                start_col = 0
            if end_col is None:
                end_col = c
            print("Starting center of rotation for %s"
%os.path.basename(item))
            ftomopy_script()
            print('Total elapsed time = %.1f min' % ((time.time() -
start_time)/60.))

```

Prefiltering Code:

```
#!/usr/bin/env python
```

```
"""
```

```
Tomography image stack preprocessor service. Processes HDF5 files from
tomography fly scans,
performs median filtering on repeated scans to remove zingers, and averages
the frames to
improve SNR. If there is only one repetition of data, zinger removal using
TomoPy will be performed.
If there are two repetitions, zingers will be replaced with the minimum value
at that pixel location.
The code also calculates required chunksize assuming that the user wants to
keep 1/3 of the
system's available RAM free. This is set by the variable "threshold", and can
be changed
```

```
This file belongs in the top-level directory as it takes its location as the
current working directory.
The script walks through all of the folders inside this top-level directory,
searches for HDF5 files, and checks
if a prefiltered file already exists. If it doesn't, then it will create one.
```

```
Alan Kastengren, XSD, APS
Katie Matusik, XSD, APS
```

```
Edits by Aniket Tekawade, April 16, 2019:
```

```
It was found that the multiprocessing Queue approach would randomly skip
writing projections to the HDF5 file.
Defining two separate queues for processing and writing is leading to
occasional loss in data.
Also, multiprocessing was disabled for tomopy.outlier (nproc = 1) since
overall processs is being parallelized.
Replaced with multiprocessing Pool, which I have found more robust. Also,
sequentialized processing and writing into one function "fprocess_images"
Did not compare processing time with the previous code, however this solves
the issue.
```

```
Edits by Aniket Tekawade, April 18, 2019:
Added RAM_limit and nprocs settings to override defaults.
```

```
"""
```

```
import time
import os
import sys
import multiprocessing
from multiprocessing import cpu_count, Pool
from itertools import product #added to test with functionality in starmap
import glob
import h5py
import numpy as np
import tomopy
import psutil
```

```

import functools

oneGB = 1024**3
nbytes_16bit = 2
RAM_limit = 2/3. # While loading data to RAM, use this fraction as threshold
so the rest of the RAM is available for other processes.

"""
try:
    nprocs=int(os.environ["NSLOTS"])
except KeyError:
    nprocs=multiprocessing.cpu_count()-10
print("Using ncore=%d"%ncore)
"""

#nprocs = 30
# Use nprocs = cpu_count() as default.
nprocs = cpu_count()
print("nprocs = %i"%nprocs)

'''Function definitions'''

# This function process a set of images at the same projection angle.
# E.g. if 5 repeatable datasets were obtained, this function process 5 images
at the same projection angle.

def Parallelize(ListIn, f, procs = -1, **kwargs):

    # This function packages the "starmap" function in multiprocessing for
    Python 3.3+ to allow multiple iterable inputs for the parallelized function.
    # ListIn: any python list such that each item in the list is a tuple of
    non-keyworded arguments passable to the function 'f' to be parallelized.
    # f: function to be parallelized. Signature must not contain any other
    non-keyworded arguments other than those passed as iterables.
    # Example:
    # def multiply(x, y, factor = 1.0):
    #     return factor*x*y
    # X = np.linspace(0,1,1000)
    # Y = np.linspace(1,2,1000)
    # XY = [ (x, Y[i]) for i, x in enumerate(X)] # List of tuples
    # Z = IP.Parallelize_MultiIn(XY, multiply, factor = 3.0, procs = 8)
    # Create as many positional arguments as required, but remember all
    must be packed into a list of tuples.

    if type(ListIn[0]) != tuple:
        ListIn = [(ListIn[i],) for i in range(len(ListIn))]

    reduced_argfunc = functools.partial(f, **kwargs)

    if procs == -1:
        procs = cpu_count()

    if procs == 1:
        OutList = [reduced_argfunc(*ListIn[iS]) for iS in
range(len(ListIn))]
    else:

```

```

print("Pooling...")
p = Pool(processes = procs)
print("procs = %i"%procs)
print("Initializing p.starmap...")
OutList = p.starmap(reduced_argfunc, ListIn)

'''with multiprocessing.Pool(processes = procs) as pool:
    Outlist = pool.starmap(reduced_argfunc, ListIn)'''
p.close()
p.join()

return OutList

def fprocess_images(image_data):

    #if there is only one image set, do zinger removal with TomoPy
    if image_data.shape[0]==1:
        out = tomopy.remove_outlier(image_data,zinger_level,ncore = 1) #
DO NOT CHANGE ncore here! ...
        # This function is kept serial so that the several instances of
this function can be parallelized. Trying to change ncore here will result in
child processes which are not supported in multiprocessing.
        out = out[0]
        #if there are two datasets, replace zingers by min value at each pixel
location
    elif image_data.shape[0]==2:
        #find minimum value between the two replicates at each pixel
location
        minval = np.fmin(image_data[0,...],image_data[1,...])
        med = np.median(image_data,axis=0) #Calculate median to locate
zingers
        for j in range(image_data.shape[0]):
            replicate = image_data[j,...]
            mask = replicate - med > zinger_level
            replicate[mask] = minval[mask] # Substitute with median
            image_data[j,...] = replicate # Put data back in place

        out =
np.mean(image_data,axis=0,dtype=np.float32).astype(np.uint16)
        #otherwise, replace zingers with median and average stack
    elif image_data.shape[0]>1:
        #Find the median for each pixel of the prefiltered image
        med = np.median(image_data,axis=0)
        #Loop through the image set
        for j in range(image_data.shape[0]):
            replicate = image_data[j,...]
            mask = replicate - med > zinger_level
            replicate[mask] = med[mask] # Substitute with median
            image_data[j,...] = replicate # Put data back in place

        out =
np.mean(image_data,axis=0,dtype=np.float32).astype(np.uint16)

    return out

```

```

def fprocess_hdf_stack(hdf_filenames,output_filename,chunksize):
    #take care of bright and dark fields first
    file_list = []
    bright_list = []
    dark_list = []
    print("Loading in bright and dark field data...")
    for fname in hdf_filenames:
        file_list.append(h5py.File(fname,'r'))
        #save all of the bright and dark fields as separate lists
        bright_list.extend(file_list[-1]['exchange']['data_white'][...])
        dark_list.extend(file_list[-1]['exchange']['data_dark'][...])
    bright_list = np.asarray(bright_list)
    dark_list = np.asarray(dark_list)
    #run the bright and dark images through median filtering to get a
single image out
    bright_field = fprocess_hdf_bright_dark(bright_list)
    dark_field = fprocess_hdf_bright_dark(dark_list)

    #put everything in output HDF5 file
    fout = h5py.File(output_filename,'a')
    fout.create_dataset('Bright_field',data = bright_field,dtype=np.uint16)
    fout.create_dataset('Dark_field',data = dark_field,dtype=np.uint16)
    print("Done writing bright and dark fields to HDF5 file.")

    #copy things over from one of the data HDF5 files
    with h5py.File(fname,'r') as fin:
        fin.copy('measurement',fout)
        #fix the filter names and other attributes that should be strings
        filt1 =
fout['measurement']['instrument']['filters']['Filter_1_Material'][...].ravel(
)
        filt2 =
fout['measurement']['instrument']['filters']['Filter_2_Material'][...].ravel(
)

        del fout['measurement']['instrument/filters/Filter_1_Material']
        del fout['measurement']['instrument/filters/Filter_2_Material']
        fout['measurement']['instrument']['filters']['Filter_1_Material']
= ''.join(chr(i) for i in [d for d in filt1 if d])
        fout['measurement']['instrument']['filters']['Filter_2_Material']
= ''.join(chr(i) for i in [d for d in filt2 if d])
        strings =
['firmware_version','manufacturer','model','software_version']
        for keys in strings:
            temp =
fout['measurement']['instrument']['detector'][keys][...].ravel()
            del fout['measurement']['instrument']['detector'][keys]
            fout['measurement']['instrument']['detector'][keys] =
''.join(chr(i) for i in [d for d in temp if d])
            if 'table' in fout['measurement'].keys():
                del fout['measurement']['table']
    print("Done copying attributes over to HDF5 file.")

    #process chunks of data so that we don't run out of memory
    data_shape = h5py.File(hdf_filenames[0],'r')['exchange']['data'].shape
    tosize = data_shape[0]
    fout.create_dataset('Prefiltered_images',data_shape,dtype=np.uint16)
    fout.close()

```



```

ints = range(totsize)

print("Loading %i images into memory..."%chunksize)

for i in range(0,totsize,chunksize):
    time0 = time.time()
    chunk = ints[i:i+chunksize]
    data_list = []
    #Make a list of the HDF5 datasets we are reading in

    for files in file_list:
        #shape is (angles, rows, columns)
        data_list.append(files['exchange']['data'][chunk,...])

    data_list = np.asarray(data_list)
    data_list = np.swapaxes(data_list, 0, 1)
    data_list = [data_list[iL] for iL in range(data_list.shape[0])]
    print("Elapsed time to load images %i-%i is %0.2f minutes."
%(chunk[0],chunk[-1],(time.time() - time0)/60))

    data_list = np.asarray(Parallelize(data_list, fprocess_images,
procs = nprocs))

    with h5py.File(output_filename,'a') as fout:
        fout['Prefiltered_images'][i:i+chunksize,...] = data_list

return

def fprocess_hdf_bright_dark(data):
    '''Processes an HDF5 file that contains a stack of images to be
    prefiltered.
    Useful for bright/dark files.
    '''
    med = np.median(data,axis=0)
    for j in range(data.shape[0]):
        replicate = data[j,...]
        mask = replicate - med > zinger_level
        replicate[mask] = med[mask] # Substitute with median
        data[j,...] = replicate # Put data back in place
    # Mean filter the stack now that the zingers are gone.
    # Keep as int16 to save space
    img = np.mean(data,axis=0,dtype=np.float32).astype(np.uint16)
    return img

if __name__ == '__main__':
    start_time = time.time()
    main_folder = os.getcwd()
    zinger_level=3000
    all_folders = next(os.walk('.')[1])

    #all_folders = glob.glob(main_folder+'/LGPS_5_OCP_000')
    for item in all_folders:
        if item == 'tomoPyScriptUtilities':continue
        raw_images_filenames = glob.glob(main_folder+'/' +item +
'/*.hdf5')

```

```

        #raw_images_filenames = glob.glob(item+'/*.hdf5')
        #check for prefiltered hdf5 file in folder
#
        sys.exit()
        if raw_images_filenames:
            here = [j for j in raw_images_filenames if
'Prefiltered.hdf5' in j]
            if here:
                print("Prefiltering has already been completed.
Prefiltered images are saved in %s."%os.path.basename(here[0]))
                pass
            else:
                output_dir = item+'/'
                if output_dir[-1] != '/':
                    output_dir=output_dir + '/'
                if raw_images_filenames:
                    print('Raw HDF5 files are in directory %s'
%item)

                    if len(raw_images_filenames)>1:
                        print('There are %i raw image files.
These will be averaged together.' %(len(raw_images_filenames)))
                    elif len(raw_images_filenames)==1:
                        print('There is %i raw image file.'
%(len(raw_images_filenames)))
                    else:
                        print('There are %i raw image files.'
%(len(raw_images_filenames)))
                tempname =
os.path.basename(raw_images_filenames[0]).split('.')[0]
#
                tempname_split = tempname.split('_')[:-1]
                tempname_split = tempname.split('_')[:] #THIS
HAS BEEN CHANGED FOR THE CELL TOMO - BAS 4/3/2019
                output_filename =
output_dir+'_'.join(tempname_split) + '_Prefiltered.hdf5'
                print("Data will be written to
%s"%output_filename)

                #get size of images assuming 16 bit to figure
out chunksize

                with h5py.File(raw_images_filenames[0],'r') as
f:

                    #m = projections, n = rows, o = columns
                    (m,n,o) = f['exchange']['data'].shape
                    size = (nbytes_16bit*m*n*o)/(oneGB)
                    totsize =
size*(len(raw_images_filenames)+1) # +1 accounts for one prefiltered set
                    mem = psutil.virtual_memory()
                    avmem = mem.available/(oneGB) #system
memory that is available and not swap
                    #assume that we want 1/3 of the system
memory to be free for other things and can therefore use 2/3 of it
                    threshold = avmem*RAM_limit
                    if totsize>threshold:
                        #back out how many images can be
processed with threshold memory

```

```

                                chunksize =
int((threshold*(oneGB))/(len(raw_images_filenames)+1)/(nbytes_16bit*n*o))
                                else:
                                    chunksize = int(m)

                                chunksize = int(chunksize/3)
                                print("Chunksize = %s" %chunksize)

                                #sys.exit()

fprocess_hdf_stack(raw_images_filenames,output_filename,chunksize)
                                else:
                                    print('Raw image files could not be found.
Please check file location and names.')
                                    print("Elapsed time is %0.2f minutes" %((time.time()
- start_time)/60))
                                else:
                                    print("No 'hdf5' files found in %s. Skipping directory."
%item)
                                    pass

```

Reconstruction Code (Python):

```
#!/usr/bin/env python
"""
Code developed in Cycle 2018-3 to feature additional attributes that are
being now saved to the HDF5 file.
The goal of the script is to input everything that the reconstruction would
need to run successfully
with as little user input as possible. This code also inputs the file name
and center of rotation from a text file.
"""

import os
import sys
import copy
import h5py
import time
import tomopy
import psutil
import matplotlib
import numpy as np
import multiprocessing
import dxchange.writer
matplotlib.use('tkAgg')
import matplotlib.pyplot as plt
from joblib import Parallel, delayed
import tomoPyScriptUtilities.BeamHardening_Transmission as bh

# USER SETTINGS
#####

# Lists of images to read
parent_directory = os.getcwd() + "/"
centers_file = parent_directory + "LGPS_10_CV_083.txt"
data_key = "083"
bright_filename = None
dark_filename = None

# Recon settings
pad_images = 'edge' #None, zero, edge
pad_fraction_width = 0.8 # fraction of image width to pad on each side, if
pad_images is not None
remove_pad = 1.0 # remove fraction of padded region to make output files
smaller?
algorithm = 'gridrec' # check tomopy docs for options
recon_filter = 'parzen' # check tomopy docs for options. this is for gridrec

#crop image in horizontal - None if no cropping required
start_col = None
end_col = None
row_start = 80
row_end = 330

#I0 region (for normalizing when doing conversion to path length)
I0_normalize = True
I0_box_size = 40 # pixels
```

```

I0_box_loc = 'center right' # 'lower/upper/center left/right/middle'. Skips
first two rows due to bad pixel values.

# Phase retrieval parameters
phase_retrieval = False
margin = 0 # Num slices at ends to not write (due to phase retrieval
boundary effects)
z = 5 # propagation distance (cm)
eng = 62 # Flux-weighted mean energy of white beam (keV)
alpha = 1e-3 # 6e-4 # Smaller number is smoother, larger number is less phase
retrieval correction

# Max number of CPUs for computation and I/O operations
try:
    ncore=int(os.environ["NSLOTS"])
except KeyError:
    ncore=multiprocessing.cpu_count()-1
print("Using ncore=%d"%ncore)
#ncore=multiprocessing.cpu_count()

#Transmission mode
beam_hardening = False
scint_material = bh.LuAG_scint #either LuAG, YAG, or LYSO
scintillator_thickness = 100.0
sample_material = bh.Fe_filter
magnification = 5

# END USER SETTINGS
#####

'''FUNCTION DEFINITIONS BELOW'''

#Read in HDF5 projection data from tomography scan
def fread_projection_images(data_filename,row_slice=
slice(None,None,1),column_slice=slice(None,None,1)):
    '''Loads the flyscan data from HDF5 files.
    '''
    t0 = time.time()
    with h5py.File(data_filename,'r') as data_hdf:
        data = data_hdf['Prefiltered_images'][:,row_slice,column_slice]
    #print('\tElapsed reading data: %i sec\n' % (time.time()-t0))
    return data.astype(np.float32)

#Read in bright and dark fields
def
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slic
e= slice(None,None,1),column_slice=slice(None,None,1),bright_ratio=1):
    '''Loads the bright and dark images.
    '''
    t0 = time.time()
    if bright_filename and 'tif' in bright_filename:
        bright_data_im = Image.open(bright_filename)
        bright_data_pre = np.array(bright_data_im)
    elif bright_filename and 'hdf5' in bright_filename:
        with h5py.File(bright_filename,'r') as bright_hdf:

```

```

        bright_data_pre = bright_hdf['Prefiltered_images'][...]
    else:
        with h5py.File(data_filename, 'r') as data_hdf:
            bright_data_pre = data_hdf['Bright_field'][...]
    bright_data = bright_data_pre[row_slice, column_slice]
    #remove high intensity pixels in the top-left corner from bright field
    #bright_data = tomopy.misc.corr.remove_outlier1d(bright_data, dif=1000)

    if dark_filename and 'tif' in dark_filename:
        dark_data_im = Image.open(dark_filename)
        dark_data_pre = np.array(dark_data_im)
    elif dark_filename and 'hdf5' in dark_filename:
        with h5py.File(dark_filename, 'r') as dark_hdf:
            dark_data_pre = dark_hdf['Prefiltered_images'][...]
    else:
        with h5py.File(data_filename, 'r') as data_hdf:
            dark_data_pre = data_hdf['Dark_field'][...]
    dark_data = dark_data_pre[row_slice, column_slice]
    #remove high intensity pixels from dark image
    #dark_data = tomopy.misc.corr.remove_outlier1d(dark_data, dif=1000)
    #print('\tElapsed reading bright/dark: %i sec\n' % (time.time()-t0))
    return (bright_data.astype(np.float32) * bright_ratio,
            dark_data.astype(np.float32))

#Find IO region
def fIO_slices():
    '''Gives slice objects for the rows and columns for the IO region.
    '''
    #Find the shape of the underlying data
    dataset_shape = fget_dataset_shape(data_filename)
    M = dataset_shape[-2]
    N = dataset_shape[-1]
    mid_row = int((M+IO_box_size)/2.)
    mid_col = int((N+IO_box_size)/2.)
    #Make the location string lower case and split it
    IO_box_split=IO_box_loc.lower().split()
    #Figure out the vertical location
    slice_M = None
    slice_N = None

    if IO_box_split[0] == 'lower':
        slice_M = slice(-IO_box_size, None, 1)
    #avoid very top row due to bad pixel values in top left corner
    elif IO_box_split[0] == 'upper':
        slice_M = slice(2, IO_box_size, 1)
    elif IO_box_split[0] == 'center':
        slice_M = slice(mid_row-IO_box_size, mid_row, 1)
    else:
        raise ValueError()
    #Figure out the horizontal location
    if IO_box_split[1] == 'right':
        slice_N = slice(-IO_box_size, None, 1)
    elif IO_box_split[1] == 'left':
        slice_N = slice(0, IO_box_size, 1)
    elif IO_box_split[1] == 'middle':
        slice_N = slice(mid_col-IO_box_size, mid_col, 1)
    else:

```

```

        raise ValueError()
    return (slice_M,slice_N)

#Calculate I0 region for each projection
def fI0_calculation(row_slices,column_slices,bright_ratio=1):
    '''Computes the I0 region.
    '''
    #Read in the projection data
    proj_data =
fread_projection_images(data_filename,row_slices,column_slices)
    #Read in the bright and dark image subsets
    bright_data,dark_data =
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slices,column_slices, bright_ratio)
    #Perform bright and dark corrections
    proj_data = tomopy.normalize(proj_data,bright_data,dark_data)
    #Average: use 64 bit for accumulator, then down-convert to 32 bit
    return_vals = np.mean(proj_data,axis=(-2,-
1),dtype=np.float64).astype(np.float32)
    #Norm by the maximum value
    return return_vals / np.max(return_vals)

#normalize by I0 region
def fnormalize_image_brightness(projection_data,I0_values):
    '''Uses a small region outside the sample to normalize for shifts
    in image intensity.
    '''
    t0=time.time()
    print("Normalizing by I0 box ...")
    projection_data /= I0_values[:,None,None]
    #print(("Elapsed normalizing image by I0: {0:6.3f}
sec".format(time.time() - t0)))
    return projection_data

#Get shape of tomography data
def fget_dataset_shape(data_filename,row_slice=
slice(None,None,1),column_slice=slice(None,None,1)):
    with h5py.File(data_filename,'r') as data_hdf:
        return data_hdf['Prefiltered_images'].shape

#Calculate beam hardening correction coefficients from input filter values
def fbeam_hardening_function(coeffs,dataset):
    '''Return data in pathlength of pixels.
    '''
    return (np.polyval(coeffs,np.log(dataset)).astype(np.float32))

#Apply beam hardening correction
def fapply_beam_hardening_projection(dataset):
    '''Applies beam hardening corrections.
    Uses joblib to speed this up.
    Send bigger chunks to joblib so we have less process overhead.
    If we send in smaller chunks, it doesn't work as well.
    '''
    t0=time.time()
    print("Apply beam hardening...")
    joblib_chunk =
int(np.ceil(float(dataset.shape[1])/float(multiprocessing.cpu_count() - 1)))

```

```

        slices = []
        for i in range(0,dataset.shape[1],joblib_chunk):
            slices.append(slice(i,i+joblib_chunk,1))
        result_list =
Parallel(n_jobs=ncore)(delayed(fbeam_hardening_function)(coeffs,dataset[:,s,:
].copy()) for s in slices)
        for i,s in enumerate(slices):
            dataset[:,s,:] = result_list[i]
            #print(("Elapsed in beam hardening: {0:6.3f} sec".format(time.time()
- t0)))
        return dataset

#remove stripes from projection data
def fremove_stripes(dataset,method='fw'):
    '''Method for removing stripes from the dataset.
    ...
    t0=time.time()
    print("Removing stripes/rings...")
    print("Using %s Cores" %ncore)
    if method == 'fw':
        print('Fourier Wavelet...')
        dataset =
tomopy.prep.stripe.remove_stripe_fw(dataset,level=8,wname='sym16',sigma=1,pad
=True,ncore=ncore-10) # Fourier Wavelet method
    elif method == 'ti':
        print('Titarenko Algorithm...')
        dataset =
tomopy.prep.stripe.remove_stripe_ti(dataset,ncore=ncore-10) # Titarenko
algorithm
    else:
        print('Smoothing filter...')
        dataset = tomopy.prep.stripe.remove_stripe_sf(dataset, 10,
ncore=ncore-10) # Smoothing filter stripe removal
        #print(("Elapsed removing chunk stripes: {0:6.3f}
sec".format(time.time() - t0)))
        print('Returning Dataset')
        return dataset

#phase retrieval algorithm, if using
def fapply_phase_retrieval(dataset):
    t0=time.time()
    #print("Apply phase retrieval...")
    dataset =
tomopy.prep.phase.retrieve_phase(dataset,pixel_size=pixel_size,dist=z,energy=
eng,alpha=alpha,\

    pad=True,ncore=ncore)
    #print(("Elapsed in phase retrieval: {0:6.3f} sec".format(time.time()
- t0)))
    return dataset

#re-register data for 360 deg scans onto 0-180 deg
def
freregister_data(dataset,theta,rotation_center,center_location,delta_theta):
    #If we have > 180 degrees of projections, reregister
    if np.max(theta) - np.min(theta) > np.pi:
        #print("\nRe-registering projections for span > 180 degrees...")

```



```

        # Find starting point for angles > 180 degrees, account for
roundoff error
        index_180 = np.argmax(theta >= (np.pi - delta_theta / 10.0))
#Returns first argument where this is True
        # Make new theta array
        new_theta = theta[:index_180]
        #Verify that the angles before and after 180 degrees match
closely
        if
(np.allclose(new_theta+np.pi,theta[index_180:2*index_180],delta_theta /
10.0)):
            print("Angles before and after 180 degrees match.")
        else:
            raise ValueError()
            #Round the center of rotation to the nearest half pixel.
            rotation_center = np.round(rotation_center * 2.0) / 2.0
            #print('Initial center of rotation = %0.2f' %rotation_center)
            # Find new width of array
            old_image_center = (dataset.shape[2] - 1.0) / 2.0      #In
between middle two pixels
            overlap_width = float(dataset.shape[2]) - 2.0*
np.abs(old_image_center - rotation_center)
            new_width = dataset.shape[2] + int(2.0 * np.abs(old_image_center
- rotation_center))
            new_center = float(new_width - 1) / 2.0
            #print(("Overlap between views 180 degrees apart = {0:0.1f}
pixels.".format(overlap_width)))
            #print(("New image width = {0:0.1f} pixels.".format(new_width)))
            #print(("New image center = {0:0.1f}
pixels.".format(new_center)))
            # Make new dataset array
            data_rereg =
tomopy.sino_360_t0_180(dataset,overlap_width,center_location)
            return(data_rereg,new_theta,new_center)
        else:
            print("No reregistration of angles needed.")
            return(dataset,theta,rotation_center)

#pad images before reconstruction
def fpad_image(dataset,center_value,pad_fraction_width):
    '''Pads images with the desired values.
    '''
    if pad_images is None:
        return (dataset,center_value)

    t0=time.time()
    prev_width = dataset.shape[2]
    print("\nPadding array...", dataset.shape)
    pad_pixels=int(np.ceil(prev_width*pad_fraction_width))
    if pad_images == 'constant':
        padded_dataset = tomopy.pad(dataset,npad = pad_pixels, axis = 2,
mode='constant',ncore=ncore,constant_values=1.0)
    elif pad_images == 'edge':
        padded_dataset = tomopy.pad(dataset,npad = pad_pixels,axis = 2,
mode='edge',ncore=ncore)
    else:
        raise ValueError("pad_images = "+pad_images+" not understood!")

```

```

        #print("\tNew size:", padded_dataset.shape)
        print('\tElapsed %i sec\n' % (time.time()-t0))
        # Center offset
        center_offset = (padded_dataset.shape[2]-dataset.shape[2])/2
        return (padded_dataset, center_value + center_offset)

#write out the tiffstack of reconstructed images
def fwrite_tiff_stack(recon_dataset, smin, total_rows):
    #Write 16 or 32-bit tiff stack out
    t0=time.time()
    #Leave out margin rows in case we did phase retrieval
    start_write_row = 1 if (smin and margin) else 0
    last_write_row = -1 if (recon_dataset.shape[0] - total_rows and margin)
else 0
    start_write_row += start_row + smin
    last_write_row += start_row + smin + recon_dataset.shape[0]
    print(("Writing slices {0:d}-{1:d} to {2:s}..." .format(start_write_row,
last_write_row - 1, output_path_tiffstack)))
    for i in range(start_write_row, last_write_row):
        j = i - start_row - smin

        dxchange.write_tiff(recon_dataset[j,...], fname=output_path_tiffstack+'i
mg_{0:04d}'.format(i), overwrite=True)
        #print('\tElapsed %i sec\n' % (time.time()-t0))

#reconstruct projections
def freconstruct_projections(dataset, theta, center, final_size):
    # Tomographic reconstruction
    t0=time.time()
    print("\nReconstructing...")
    data_recon = tomopy.recon(dataset,
theta, center=center, algorithm=algorithm, ncore=ncore, filter_name=recon_filter)
    print('\tElapsed %i sec\n' % (time.time()-t0))

    #Crop the image to the desired size
    #print(("Cropping volume from size {0:d} to size
{1:d}" .format(data_recon.shape[1], final_size)))
    crop_margin = int((data_recon.shape[1] - final_size) / 2.0)
    if crop_margin > 0:
        data_recon =
data_recon[:, crop_margin:crop_margin+final_size, crop_margin:crop_margin+final
_size]

    #Rearrange output (Dan notices that the images are flipped vertically!)
    data_recon = np.fliplr(data_recon)

    #print(("Chunk output stats: ", data_recon.shape))
    #print(("Nans:
", np.sum(np.isnan(data_recon)), '/', np.product(data_recon.shape)))
    #print(("Infs:
", np.sum(np.isinf(data_recon)), '/', np.product(data_recon.shape)))

    # Mask NaNs and INFs from the output data.
    data_recon[np.isnan(data_recon)]=0
    data_recon[np.isinf(data_recon)]=np.nanmax(data_recon)*np.sign(data_rec
on[np.isinf(data_recon)])

```

```

        print('\tTotal elapsed to reconstruct subset = {0:3.3f}
s.'.format(time.time() - t0))
        return data_recon

#remove outliers with a median filter
def remove_outliers(data,dif,size):
    out = tomopy.misc.corr.remove_outlier1d(data,dif=dif,size=size,axis=1)
    return out

#calculate correct amount of images to load at once for system
def calculate_chunksize(rowmin,rowmax):
    if rowmin is None:
        rowmin = 0
    if rowmax is None:
        rowmax = r
    #automated way to calculate the chunk size
    mem = psutil.virtual_memory()
    avmem = mem.available/(1e9) #system memory that is available and not
swap
    #assume that we want 1/4 of the system memory to be free for other
things and can therefore use 3/4 of it
    threshold = avmem*(3/4.)
    #calculate about how much RAM is going to be needed for the whole recon
    im_size = (4*p*c)/(1e9) #image gets converted to 32 bit i.e. 4 bytes
per bit
    pad_size_im = (im_size*pad_fraction_width)*2+im_size #size after
padding occurs
    set_size = (4*p*r*c)/(1e9) #dataset gets converted to 32 bit i.e. 4
bytes per bit
    pad_size = (set_size*pad_fraction_width)*2+set_size #size after padding
occurs
    #biggest memory bottleneck is the padded chunk getting copied on all
used cores
    chunkfrac = threshold/(pad_size*(ncore/2))
    rchunk = int(chunkfrac*r)
    return rowmin,rowmax,rchunk

#find the files on different computers
def find(name, path):
    for root, dirs, files in os.walk(path):
        if name in files:
            return root, os.path.join(root,name)
#####
#####
# Main function to do tomoPy reconstruction. Input parameters are defined at
the top of the script.
# This version will load ALL the images and work on chunks of specified size
in RAM.
# Chunks are overlapped by some 'margin' to avoid edge phase retrieval
errors.
### MAIN
#####
#####

def ftomopy_script():

    #read in necessary properties from HDF5 file

```

```

        with h5py.File(data_filename, 'r') as data_hdf:
            bright_exp =
data_hdf['measurement']['instrument']['detector']['brightfield_exposure_time'
][...] #bright field exposure time, units of seconds
            exp =
data_hdf['measurement']['instrument']['detector']['exposure_time'][...]
#images exposure time, units of seconds
            bright_ratio = exp/bright_exp #multiply bright field intensity
values by this ratio
            #load in A-hutch metal filter information
            try:
                filt1 =
str(data_hdf['measurement']['instrument']['filters']['Filter_1_Material'][...
])
            except KeyError:
                filt1 = []
            try:
                filt2 =
str(data_hdf['measurement']['instrument']['filters']['Filter_2_Material'][...
])
            except KeyError:
                filt2 = []
            #load in magnification
            #load in magnification
            try:
                mag =
data_hdf['measurement']['instrument']['detector']['magnification'][...]
            except:
                try:
                    mag =
data_hdf['measurement']['instrument']['detector']['setup']['Magnification'][.
..]
                except KeyError:
                    mag = magnification
            cam_model =
str(data_hdf['measurement']['instrument']['detector']['model'][...])
            #retrieve pixel sizes from camera model name
            if cam_model == 'Grasshopper3 GS3-U3-123S6M':
                pixel_size = 3.45e-4/mag
            elif cam_model == 'Grasshopper3 GS3-U3-23S6M':
                pixel_size = 5.86e-4/mag
            #figure out theta range
            start_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_Start'][...])
            end_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_End'][...])
            step_theta =
np.deg2rad(data_hdf['measurement']['instrument']['sample_position']['Sample_T
heta_Step'][...])
            #Span of angles for reconstruction; units radians
            theta = np.arange(start_theta, end_theta+step_theta, step_theta)

            #Handle beam hardening
            if beam_hardening:
                #import the filters that were used in data acquisition

```

```

bh_dir = dir(bh)
filters = {}
filters = {bh.Be_filter:750.0}
if filt1 != 'Open':
    mat1 = filt1.split('_')[0]
    thick1 = float(filt1.split('_')[-1][:-2])
    temp1 = [j for j in range(len(bh_dir)) if mat1+'_filter' in
bh_dir[j]][0]
    filters[getattr(bh,bh_dir[temp1])]=thick1

if filt2 != 'Open':
    mat2 = filt2.split('_')[0]
    thick2 = float(filt2.split('_')[-1][:-2])
    temp2 = [j for j in range(len(bh_dir)) if mat2+'_filter' in
bh_dir[j]][0]
    filters[getattr(bh,bh_dir[temp2])]=thick2

filtered_spectrum = bh.fapply_filters(filters)
plt.plot(bh.spectral_energies,filtered_spectrum)
plt.show()
#bh.sample_material = bh.Si_filter
global coeffs
coeffs =
bh.fcompute_lookup_coeffs(bh.spectral_energies,filtered_spectrum,sample_mater
ial,scint_material,scintillator_thickness) / (pixel_size * 1e4)
#print("\nCoefficients for fit for transmission to pathlength
conversion complete.")

#set slices to reconstruct
row_slice = slice(start_row, end_row+1, row_step)
column_slice = slice(start_col, end_col+1, 1)

#Import the projection images
projection_data_full =
fread_projection_images(data_filename,row_slice,column_slice)

#Load the bright and dark images. Do over the same subset of the image
as the original data.
bright_median, dark_median =
fread_bright_dark_images(data_filename,bright_filename,dark_filename,row_slic
e,column_slice,bright_ratio)

#Normalize the projection images by the bright and dark fields
projection_data_full =
tomopy.normalize(projection_data_full,bright_median,dark_median)

#Find the normalization values
if I0_normalize:
    I0_row_slice,I0_column_slice = fI0_slices()
    I0_values = fI0_calculation(I0_row_slice,
I0_column_slice,bright_ratio)
    #Normalize by the I0 region.
    projection_data_full =
fnormalize_image_brightness(projection_data_full,I0_values)

# plt.figure(1)
# plt.imshow(projection_data_full[:,0,:],cmap='gray') #plot sinogram

```

```

# plt.title('After Normalization')
# plt.colorbar()
# plt.show()

    print("Center of rotation is at %s" %rotation_center)

    #start reconstruction of dataset chunks
    ta = time.time()
    print('\nWorking on slices {0:d} to {1:d} of dataset ({2:d} slices
total)'.format(start_row,end_row,projection_data_full.shape[1]))

    #Stripe removal
    projection_data_full = fremove_stripes(projection_data_full)
    print('Stripe Removal Complete...')

    #Correct for beam hardening, if desired. Otherwise, convert to
pathlength
    if beam_hardening:
        projection_data_full =
fapply_beam_hardening_projection(projection_data_full)
    else:
        projection_data_full = tomopy.minus_log(projection_data_full)

    #If desired, perform phase retrieval
    if phase_retrieval:
        projection_data_full =
fapply_phase_retrieval(projection_data_full)

    #Take care of datasets where we do a 0-360 degree rotation to increase
FOV
    if (c-1)/2 > rotation_center:
        center_location = 'left'
    else:
        center_location = 'right'
        working_data_subset, working_theta, working_rotation_center =
freregister_data(projection_data_full,theta,rotation_center,center_location,s
tep_theta)
    del(projection_data_full)
    #print(("Rotation center = {:.2f}
pixels.".format(working_rotation_center)))
    newdata = float((working_data_subset.nbytes)/1e9)
    print("Size of chunked data is %0.2f GB" %newdata)

    #Mask huge values, NaNs and INFs from the filtered data
    maxv=1e5
    working_data_subset[np.isnan(working_data_subset)]=0
    working_data_subset[np.isinf(working_data_subset)]=np.nanmax(working_da
ta_subset)*np.sign(working_data_subset[np.isinf(working_data_subset)])
    working_data_subset[np.abs(working_data_subset)>maxv]=maxv*np.sign(work
ing_data_subset[np.abs(working_data_subset)>maxv])
    old_width = working_data_subset.shape[2]

    #Pad the array
    (working_data_subset, new_center) =
fpad_image(working_data_subset,working_rotation_center,pad_fraction_width)
    pad_size = float((working_data_subset.nbytes)/1e9)
    print("Padded array is %0.2f GB" %pad_size)

```

```

    print('Starting Reconstruction...')
    #Tomographic reconstruction
    data_recon =
freconstruct_projections(working_data_subset,working_theta,new_center,old_wid
th)
    recon_size = float((data_recon.nbytes)/1e9)
    #print("Recon array is %0.2f GB" %recon_size)
    print('Finished Reconstruction...')

    #Ring removal
    t0=time.time()
    print("\nRemoving rings...")
    data_recon = tomopy.remove_ring(data_recon.copy(),center_x =
rotation_center,center_y = rotation_center,ncore=ncore)
    #mask outer edges with circular mask
    data_recon = tomopy.circ_mask(data_recon, axis=0, ratio=0.95)
    #print('\tElapsed %i sec\n' % (time.time()-t0))

    #Write tiff stack out
    if output_path_tiffstack is not None:
        fwrite_tiff_stack(recon_dataset = data_recon,smin=0,total_rows =
r)

    #Clear memory to keep from blowing up the memory usage.
    print('Clearing Memory')
    del(working_data_subset,data_recon)
    return

# plt.figure()
# plt.imshow(projection_data_full[20,...])
# plt.title('After Beam Hardening')
# plt.colorbar()
# plt.show()

### MAIN
#####

if __name__ == "__main__":
    #find file name and center of rotation from text file
    with open(centers_file,'r') as f:
        modlines = []
        for line in f:
            modlines.append(line)
    if data_key == 'ALL':
        dat = copy.deepcopy(modlines)
    else:
        dat = [j for j in modlines if data_key in j]
    print(("Using TomoPy version = " + tomopy.__version__))
    for cond in dat:
        if cond.startswith("#"):
            pass
        else:
            start_time = time.time()
            file_path,data_filename = find(cond.split('
')[0],parent_directory)
            rotation_center = float(cond.split(' ')[1][:-2])

```

```

        #create location for saving data
        basename = os.path.basename(os.path.normpath(file_path))
        output_path_tiffstack = file_path+"/Recon_"+basename+"/" #
None for no TIFF output
        print("Working on dataset %s" %basename)
        #print("Images will be written to
%s"%output_path_tiffstack)
        #Give slice objects with the image ROI
        p,r,c = fget_dataset_shape(data_filename,row_slice=
slice(None,None,1),column_slice=slice(None,None,1))
        #Fraction of slices to load into RAM at once (smaller
number reduces I/O overhead)
        rowmin, rowmax, rchunk =
calculate_chunksize(row_start,row_end)
        rchunk = 5
        print("Chunk size is %i"%rchunk)
        row_step = 1 # Phase retrieval requires step of 1
        for rows in np.arange(rowmin,rowmax,rchunk):
            start_row = rows
            end_row = rows+rchunk-1
            if end_row>rowmax:
                end_row=rowmax
            if start_col is None:
                start_col = 0
            if end_col is None:
                end_col = c
            ftomopy_script()
        print('Total time for recon of %s = %.1f min' %
(basename,(time.time() - start_time)/60.))

```


Post Processing Code (Matlab):

```
clear all
close all
A = pwd;
set(0,'DefaultFigureWindowStyle','docked')
warning('off', 'images:imshow:magnificationMustBeFitForDockedFigure');

% User input dialogue to select desired operation
List = {'Void Analysis', 'Density Approximation', 'Pellet Reconstruction',...
        'Generate Video', 'Generate Single Image'};
[indx,tf] = listdlg('ListString',List);
switch indx
    case 1
        %% Case 1 Program Description
        % Script is designed to import processed image data from tomographic
        % reconstructions to determine total internal void volume and internal
        % pellet surface area. Some general considerations: the code computes void
        % space within a finite limit of pellet volume defined in plane by a 2D
        % mask, and defined in z by user inputs. The null background signal makes
        % total pellet analysis exceedingly difficult. While this approach misses
        % out on some data it characterizes a large chunk of pellet data.
        % Additionally it should be remembered that this program uses one mask to
        % compute void volume for each time point (folder of images) to maintain
        % consistency. If a subset of the data is to be analyzed it is strongly
        % recommended that a pre-existing mask be imported and used for analysis.
        % The general workflow is as follows: Read User Inputs and Find paths to
        % the data -> Import a cell array for each folder to prime computation ->
        % Apply Mask and compute void volume and surface derivative -> Repeat for
        % each folder in the directory containing the keyword -> Output final data
        % as a table with time stamps for image correlation

        disp('Executing Void Analysis');
        warn = 0;

        %% User Settings
        % Process a subset of data
        SubSet = 'y';
        % Use Pre-existing mask
        Msk = 'y';
        % Rough density calculation
        Density = 'n';
        %Surface Profilometry
        Profile = 'n';

        %Obtain subfolder directories for a given cell
        if SubSet == 'y'
            inpt = inputdlg({'Start Row', 'End Row', 'Keyword',...
                            'Start Folder', 'End Folder'});
        else
            inpt = inputdlg({'Start Row', 'End Row', 'Keyword'});
        end
        yn = input('Run Differential Analysis? (y/n)', 's');
        RowStart = str2num(inpt{1});
        RowEnd = str2num(inpt{2});
        NRows = RowEnd - RowStart;
```

```

disp('Kindly select parent directory for all timepoint folders');
Path = uigetdir();
SubPath = dir(Path);
SubPath = SubPath([SubPath(:).isdir] == 1);
SubPath = SubPath(~ismember({SubPath(:).name}, {'.', '..'}));
Folders = {SubPath.name};
N = ~cellfun('isempty', strfind(Folders, inpt{3}));
Folders = Folders(N);
if SubSet == 'y'
    num1 = find(~cellfun(@isempty, strfind(Folders, inpt{4})));
    num2 = find(~cellfun(@isempty, strfind(Folders, inpt{5})));
    Folders = Folders(num1 : num2);
end
FileNames = Folders;
Folders = fullfile(Path, Folders);
minpeakdistance = 0.2e-3;

%Obtain time values for each reconstruction as determined by
%HDF5timeelapsed.m. Data is stored in a table labeled Time Order.txt
cd(Path);
try
    times = importdata('Time Order.txt', '\t');
    times.textdata = times.textdata(2: end, 1);
catch
    disp('Cannot find time data');
end
cd(A);

%Obtain Directories to reconstructed image slices
for i = 1 : length(Folders)
    tempdir = dir(Folders{i});
    tempdir = tempdir([tempdir(:).isdir] == 1);
    tempdir = {tempdir.name};
    N = ~cellfun('isempty', strfind(tempdir, 'Recon'));
    tempdir = tempdir(N);
    Folders(i) = fullfile(Folders(i), tempdir);
end

%% Initialize Values and Begin Computation:

V = zeros(NRows, length(Folders));
DV = zeros(NRows, length(Folders));
Wbar = waitbar(0, strcat('Processing folder',{' '}, '1', {' '}, 'of', ...
    {' '}, num2str(length(Folders))));
ICheck = {};
DiffCheck = {};
hplot = {};

if Density == 'y'
    H = {};
end

%Loop through each flyscan
for i = 1 : length(Folders)

    %Path to Folder Directory
    cd(Folders{i});

```

```

%Loop through each image slice
for j = RowStart : 1 : RowEnd
    if j < 100
        imstr = strcat(num2str(0), num2str(0), num2str(j));
    else
        imstr = strcat(num2str(0), num2str(j));
    end
    Current = dir();
    Current = {Current.name};
    N = ~cellfun('isempty', strfind(Current, imstr));
    imstr = Current(N);
    I = imread(char(imstr));
    Av = mean(mean(I));
    dev = std(mean(I));
    I(I>Av + 10*dev) = Av;

    % Choose Elliptical Region from first image slice
    if i == 1 && j == RowStart
        if Msk == 'y'
            [MName, MDir] = uigetfile('*..*', 'All Files (*.*)');
            MName = fullfile(MDir, MName);
            Mask = imread(MName);
            imshow(I.*Mask, []);
        else
            f = imshow(I, []);
            el = imellipse;
            pause
            Mask = createMask(el);
        end
        %delete(h);
        %close;
    end

    %Adjust image size to be equal to Mask size if necessary
    if size(I)~=size(Mask)
        I = I(1:size(Mask, 1), 1:size(Mask, 2));
        warn = 1;
        index = Folders(i);
    end

    I = I .* Mask;
    I(I == 0) = nan;

    if yn == 'y'
        [DiffX, DiffY] = gradient(I);
        DiffX = abs(DiffX);
        DiffY = abs(DiffY);
        Diff = (DiffX + DiffY)/2;
        DiffAv = nanmean(nanmean(Diff));
        DiffStd = nanmean(nanstd(Diff));
        Diff(Diff < DiffAv + 3*DiffStd) = 0;
        Diff(isnan(Diff)) = 0;
        DV(j + 1 - RowStart, i) = nnz(Diff);
    end
    figure
    h = histogram(I, 100);

```

```

x = h.BinEdges;
x = x(1: end - 1);
x = double(x);
y = h.Values;
threshold = max(y)*2/3;
plot(x, y);
[pk, loc, w] = findpeaks(y, x, 'WidthReference','halfheight',...
    'MinPeakHeight', threshold, 'MinPeakDistance', minpeakdistance);
close;
if Density == 'y'
    H{i - RowStart + 1} = [x' y'];
end

I(I < (loc - w*5/2.355)) = 0;
V(j + 1 - RowStart, i) = sum(I(:) == 0);
waitbar((j + 1 - RowStart)/NRows, Wbar);
if i == 1
    hplot{j + 1 - RowStart} = [x' y'];
    if j == RowStart
        ICheck{1} = I;
        DiffCheck{1} = Diff;
    end
elseif i == length(Folders) && j == RowStart
    ICheck{2} = I;
    DiffCheck{2} = Diff;
end
end

waitbar(0, Wbar, strcat('Processing folder', {' '}, num2str(i+1), {' '},...
    'of', {' '}, num2str(length(Folders))));
end
cd(A);

%% Build Output Arrays and Plot Pertinent Data from Analysis
Fnum = [1 : length(Folders)];
VoidVolume = zeros(size(V, 2), 1);
for i = 1 : size(V, 2)
    VoidVolume(i) = sum(V(:, i));
end

SurfaceArea = zeros(size(DV, 2), 1);
for i = 1 : size(DV, 2)
    SurfaceArea(i) = sum(DV(:, i));
end

%Plot sll histograms from first file in queue
Three_D_Plot(hplot, 'Histograms from file 1', 1)

if Density == 'y'
    Three_D_Plot(H, 'Histograms from file 1', 1)
end

ylabel('Pixel Intensity');
xlabel('Vertical Slice Number');
zlabel('Counts');

```

```

%Plot cropped image and differential from first slice of first flyscan
figure
imshow(ICheck{1}, []);
figure
imshow(DiffCheck{1}, [])
colormap(bluewhitered);

%plot Cropped image and differential from first slice of last flyscan
figure
imshow(ICheck{2}, []);
figure
imshow(DiffCheck{2}, [])
colormap(bluewhitered);

figure
plot(Fnum, VoidVolume);
figure
plot(Fnum, SurfaceArea);
close(Wbar)

if warn == 1;
    disp(strcat('Adjusted image dimensions to fit mask ', {' '}, index));
end
FileNames = FileNames';

%% Import time data and write out final analysis table:
%Obtain times of analyzed data
if exist('times') == 1
    Time = [];
    for i = 1: length(FileNames)
        temp = strfind(times.textdata, FileNames{i});
        nmbr = find(~cellfun('isempty', temp));
        Time = [Time; times.data(nmbr)];
    end
Grad = gradient(VoidVolume, Time);
T = table(FileNames, Time, VoidVolume, SurfaceArea, Grad);
else
    T = table(FileNames, VoidVolume, SurfaceArea);
end

if SubSet == 'y'
    writetable(T, strcat(Path, '\Void_Analysis_Subset.txt'), 'delimiter',
'\t');
else
    writetable(T, strcat(Path, '\Void_Analysis.txt'), 'delimiter', '\t');
end

if Msk ~= 'y'
    imwrite(Mask, strcat(Path, '\Mask.tiff'), 'Compression', 'none');
end

case 2
%% Case 2 Program Description
% Script is used to compute most probable pixel intensity as a function of
% vertical position within the pellet. The code is set up in a very similar
% way to ProcessStack.m and PlotTomo.m and relies on histogram analysis to

```

```

% determine the most populated pixel intensity above void for each z
% position. General Work Flow: path to folder and import each z image ->
% apply mask to ommit statistical abberation from large amounts of void
% space -> compute a histogram for each z -> determine maximum of histogram
% and consider this position the intensity of highest population -> output
% data values for each folder.

```

```

disp('Executing Density Approximation');
MaskImage = '046';

```

```

Path = uigetdir();
inpt = inputdlg('Start Row', 'End Row','Normalization Row', 'Start Folder',
'End Folder', 'Keyword');
cd(Path);
Current = dir();
Current = Current([Current(:).isdir] == 1);
Current = Current(~ismember({Current(:).name}, {'.', '..'}));
Folders = {Current.name};
N = ~cellfun('isempty', strfind(Folders, inpt{6}));
Folders = Folders(N);
num1 = find(~cellfun(@isempty, strfind(Folders, inpt{4})));
num2 = find(~cellfun(@isempty, strfind(Folders, inpt{5})));
Folders = Folders(num1 : num2);
FileNames = Folders;
Folders = fullfile(Path, Folders);

```

```

for i = 1 : length(Folders)
    tempdir = dir(Folders{i});
    tempdir = tempdir([tempdir(:).isdir] == 1);
    tempdir = {tempdir.name};
    N = ~cellfun('isempty', strfind(tempdir, 'Recon'));
    tempdir = tempdir(N);
    Folders(i) = fullfile(Folders(i), tempdir);
end

```

```

RowStart = str2num(inpt{1});
RowEnd = str2num(inpt{2});
NormRow = str2num(inpt{3});
NRows = RowEnd - RowStart;

```

```

N = ~cellfun('isempty', strfind(Folders, MaskImage));
temp = Folders(N);
cd(char(temp));
Current = dir();
Names = {Current.name};
Names = Names(~ismember(Names, {'.', '..'}));
N = ~cellfun('isempty', strfind(Folders, MaskImage));

```

```

%% Open normalization row used to compute baseline histogram
if NormRow < 100
    imstr = strcat(num2str(0), num2str(0), num2str(NormRow));
else
    imstr = strcat(num2str(0), num2str(NormRow));
end

```

```

N = ~cellfun('isempty', strfind(Names, imstr));

```

```

imstr = Names(N);
I = imread(char(imstr));
Av = mean(mean(I));
dev = std(mean(I));
I(I>Av + 10*dev) = Av;
imshow(I, []);

% Create Mask From Positive Normalization Image

el = imellipse;
pause
Mask = createMask(el);

figure
low = {};
med = {};
hgh = {};
Wbar = waitbar(0, strcat('Processing folder',{' '}, '1', {' '}, 'of',...
    {' '}, num2str(length(Folders))));

%% Loop through Folders
for k = 1 : length(Folders)
    %Path to appropriate folder
    cd(Folders{k});
    Current = dir();

    if k == 1
    else
        waitbar(0, Wbar, strcat('Processing folder',{' '}, num2str(k), {' '},
'of',...
            {' '}, num2str(length(Folders))));
    end

    Names = {Current.name};
    Names = Names(~ismember(Names, {'.', '..'}));
    Names = fullfile(Folders(k), Names);

    cd(A);
    H = {};
    imlow = [];
    immed = [];
    imhgh = [];
    for i = RowStart: 1 : RowEnd
        waitbar((i + 1 - RowStart)/NRows, Wbar);
        if i < 100
            imstr = strcat(num2str(0), num2str(0), num2str(i));
        else
            imstr = strcat(num2str(0), num2str(i));
        end

        N = ~cellfun('isempty', strfind(Names, imstr));
        imstr = Names(N);
        I = imread(char(imstr));
        I(I == 0) = nan;
        %I(I<3e-4) = nan;

```

```

%I(I> .8e-3) = nan;

h = histogram(I, 500);
x = h.BinEdges;
x = x(1: end - 1);
x = double(x);
w = x(2)-x(1);
y = h.Values;

%smooth histogram
y = datasmooth(y, 7);
Window = find(x>-2e-4 & x<2e-4);

threshold = max(y)/2;
plot(x(Window), y(Window));
[pk, loc, w] = findpeaks(y(Window), x(Window),
'WidthReference','halfheight',...
'MinPeakHeight', threshold);

I = I - loc;
if size(I, 1) > size(Mask, 1)
    I = I(1:size(Mask, 1), 1:size(Mask, 2));
elseif size(I, 1) < size(Mask, 1)
    diff = size(Mask, 1) - size(I, 1);
    I = [I nan(size(Mask, 1)-diff, diff)];
    I = [I; nan(diff, size(Mask, 1))];
end
I = I.*Mask;
I(I == 0) = nan;

h = histogram(I, 500);
x = h.BinEdges;
x = x(1: end - 1);
x = double(x);
w = x(2)-x(1);
y = h.Values;
y = datasmooth(y, 7);

Window2 = find(x>3e-4 & x<8e-4);
if ~isempty(Window2)
    H{i - RowStart + 1} = [x(Window2)' y(Window2)'];
    SPW = .1e-3;
    Pks = peakfit(H{i - RowStart + 1}, 0, 0, 2, 6, 0, 25, [.4e-3 SPW
.6e-3 SPW], 3); %,
    Pks(:, 5) = Pks(:, 5)/w;

    Pos = Pks(:, 2);

    ltmp = Pks(Pos>-2e-4 & Pos<3e-4, 5); %Pos(Pos>-2e-4 & Pos<3e-4);
    %Pks(Pos>-2e-4 & Pos<3e-4, 5)];
    mtmp = Pks(Pos>3e-4 & Pos<4.75e-4, 5); %Pos(Pos>3e-4 & Pos<5e-4);
    %Pks(Pos>-2e-4 & Pos<3e-4, 5)];
    htmp = Pks(Pos>4.75e-4 & Pos<10e-4, 5); %Pos(Pos>5e-4 & Pos<10e-
4); %Pks(Pos>5e-4 & Pos<10e-4, 5)];
else
    ltmp = [];
    mtmp = [];

```



```

        htmp = [];
    end

    if ~isempty(ltmp)
        %ltmp(1, 1) = mean(ltmp(:, 1));
        ltmp(1) = sum(ltmp);
        ltmp = [i ltmp(1)];
        imlow = [imlow; ltmp];
    else
        %ltmp(1, 1) = 0;
        ltmp(1) = 0;
        ltmp = [i ltmp(1)];
        imlow = [imlow; ltmp];
    end

    if ~isempty(mtmp)
        %mtmp(1, 1) = mean(mtmp(:, 1));
        mtmp(1) = sum(mtmp);
        mtmp = [i mtmp(1)];
        immed = [immed; mtmp];
    else
        %mtmp(1, 1) = 0;
        mtmp(1) = 0;
        mtmp = [i mtmp(1)];
        immed = [immed; mtmp];
    end

    if ~isempty(htmp)
        %htmp(1, 1) = mean(htmp(:, 1));
        htmp(1) = sum(htmp);
        htmp = [i htmp(1)];
        imhgh = [imhgh; htmp];
    else
        %htmp(1, 1) = 0;
        htmp(1) = 0;
        htmp = [i htmp(1)];
        imhgh = [imhgh; htmp];
    end

    end

    low{k} = imlow;
    med{k} = immed;
    hgh{k} = imhgh;

end

%% Plot and output values
close(Wbar);
figure
plot(med{1}(:, 1), med{1}(:, 2));
hold on
plot(hgh{1}(:, 1), hgh{1}(:, 2));
plot(low{1}(:, 1), low{1}(:, 2));
Int_Map = [];
x = med{1}(:, 1);
y = [];
z = [];

```

```

for i = 1 : size(med, 2)
    z = [z; med{i}(:, 2)'];
    y = [y; i];
    tmp = [i.*ones(length(med{i}), 1) med{i}];
    Int_Map = [Int_Map; tmp];
end

```

```

high = [];
low = [];
for i = 1 : size(med, 2)
    high = [high; sum(med{i}(1:100, 2))];
    low = [low; sum(med{i}(100:end, 2))];
end

```

case 3

```

%% Case 3 Program Description
% Script imports reconstructed tomographic images and processes them to
% create a series of 3D maps which can be manipulated within the matlab
% workspace to more clearly interpret pellet morphology. General
% considerations: This script does not reconstruct the entire pellet volume
% in 3D rather it computes the pellet/electrode interfaces (that is the
% boundaries of the pellet) and rebuilds them in 3D. The resultant plots
% have the appearance of an entire pellet however the interior is hollow.
% Additionally, due to limitations both in computational resources and in
% Matlabs built in graphical interface raw data must be compressed for
% visualization. Course reconstructions (7 X 7 X 7 pixel compression or
% greater) can be developed in less than an hour and result in a reasonable
% interpretation of the image. the lowest compression recommended is (3 X 3
% X 3) which will require upwards of 16 hours to compute. It is recommended
% that for low data compression figures surface lines should be omitted for
% data clarity. The General workflow is as follows: Import single folders
% at a time -> Use histogram analysis to set void space to nan intensity ->
% scan vertically through image stack to find topmost and bottommost pixels
% -> define these pixels as the interface -> create surface using these
% pixels and output figure.

```

```

disp('Executing Pellet Reconstruction');
%User Inputs
%Process entire list of files
stack = 'y';
%Change to 'y' to define an elliptical analysis region
%Recommended for Low Density Reconstruction to avoid interference from
%image aberrations
elpyn = 'n';
%Change to 'y' to compute differential images
diffyn = 'n';
%Change to 'y' to export 3D scatter plots of data
scttr = 'n';
%Change to 'y' to build Volume Array (NOT RECOMMENDED)
volary = 'n';
%Change to 'y' to plot a smoothed image using gaussian smoothing algorithm
smth = 'n';
%Lets just see what happens
Detonate = 'y';
% Display total cell volume

```

```

DCV = 'y';
% Plot Top and Bottom Figures Separately
PTB = 'n';
%Plot medium density regions rather than high density regions
LD = 'n';

%Initialize Values
I = {};
Pnts = [];
Pnts2 = [];
Icomp = {};
IDiff = {};
IDiffComp = {};
elps = 0;
warn = 0;
if Detonate == 'y'
    Cfactor = 1;
else
    Cfactor = 3;
end
minpeakheight = 15000;
minpeakdistance = 0.5e-3;
minpixelnum = 50000;
topbuffer = [];

%Import Data From file
inpt = inputdlg({'Start Row', 'End Row', 'Search Row',...
    'Vertical Step Size', 'Keyword', 'Start Folder', 'End
Folder', 'Output Folder Title'});
RowStart = str2num(inpt{1});
RowEnd = str2num(inpt{2});
SearchRow = str2num(inpt{3});
StepSize = str2num(inpt{4});
SearchIndex = floor((SearchRow - RowStart)/StepSize);
numfiles = floor((RowEnd - RowStart)/StepSize);
Path = uigetdir();

if stack == 'y'
    SubPath = dir(Path);
    SubPath = SubPath([SubPath(:).isdir] == 1);
    SubPath = SubPath(~ismember({SubPath(:).name}, {'.', '..'}));
    Folders = {SubPath.name};
    N = ~cellfun('isempty', strfind(Folders, inpt{5}));
    Folders = Folders(N);
    Filenames = Folders;
    Folders = fullfile(Path, Folders);

    for i = 1 : length(Folders)
        tempdir = dir(Folders{i});
        tempdir = tempdir([tempdir(:).isdir] == 1);
        tempdir = {tempdir.name};
        N = ~cellfun('isempty', strfind(tempdir, 'Recon'));
        tempdir = tempdir(N);
        Folders(i) = fullfile(Folders(i), tempdir);
    end
end
end

```

```

if stack == 'y'
    fldrs = length(Folders);
elseif stack == 'n'
    fldrs = 1;
end

if isempty(inpt{6})
    start = 1;
else
    start = strfind(Folders, inpt{6});
    start = find(~cellfun(@isempty, start));
end

if isempty(inpt{7})
    finish = fldrs;
else
    finish = strfind(Folders, inpt{7});
    finish = find(~cellfun(@isempty, finish));
end

% If desired define an elliptical region of interest to analyze
if elpyn == 'y'
    cd(Folders{ceil(length(Folders)/2)});
    Current = dir();
    Current = {Current.name};
    N = ~cellfun('isempty', strfind(Current, num2str(SearchRow)));
    maskfile = Current(N);
    Mimage = imread(char(maskfile));
    elps = 1;
    f = imshow(Mimage, []);
    h = imellipse;
    pause
    Mask = createMask(h);
    pos = getPosition(h);
    delete(h);
end

%% Begin Image Analysis:
for p = start : finish
if stack == 'y'
    Wbar = waitbar(0, strcat('Importing Images for file', {' '}, num2str(p),
    ...
    {' '}, 'of', {' '}, num2str(fldrs)));
else
    Wbar = waitbar(0, 'Importing Images');
end

if stack == 'n'
    cd(Path);
elseif stack == 'y'
    cd(Folders{p});
end

for i = 0 : numfiles - 1
    waitbar(i/numfiles, Wbar);

```

```

        if RowStart + i*StepSize < 100
            imstr = strcat(num2str(0), num2str(0), num2str(RowStart +
i*StepSize));
        else
            imstr = strcat(num2str(0), num2str(RowStart + i*StepSize));
        end
        Current = dir();
        Current = {Current.name};
        N = ~cellfun('isempty', strfind(Current, imstr));
        imstr = Current(N);
        I{i + 1} = imread(char(imstr));
        I{i + 1}(I{i + 1} == 0) = nan;
    end

    centerrow = I{1}(floor(size(I{1}, 1)/2), :);
    F = find(isnan(centerrow));
    Fprime = gradient(F);
    ponts = find(Fprime~=1);
    cellstart = floor(F(ponts(1))/Cfactor);
    cellend = floor(F(ponts(end))/Cfactor);
    cellwidth = cellend - cellstart;
    cd(A);
    num = size(I, 2);

    %Remove Overly Bright Pixels
    for i = 1 : size(I, 2)
        Av = nanmean(nanmean(I{i}));
        S = nanmean(nanstd(I{i}));
        I{i}(I{i} > Av + 10*S) = Av + 10*S;
    end

    %Compute representative pixel distribution from search row image
    h = histogram(I{SearchIndex}, 500);
    x = h.BinEdges;
    x = x(1: end - 1);
    x = double(x);
    y = h.Values;
    threshold = max(y)*2/3;
    plot(x, y);
    [pk, loc, w] = findpeaks(y, x, 'WidthReference','halfheight',...
'NPeaks', 2, 'MinPeakDistance', minpeakdistance, 'MinPeakHeight',...
minpeakheight);
    close;

    if stack == 'y'
        waitbar(0, Wbar, strcat('Processing Images for file', {' '}, num2str(p),
...
                                {' '}, 'of', {' '}, num2str(fldrs)));
    elseif stack == 'n'
        waitbar(0, Wbar, 'Processing Images');
    end

    for k = 1 : size(I, 2)
        waitbar(k/size(I, 2), Wbar);

        %If a mask is desired apply mask to all images
        %adjust image size to fit mask when necessary

```

```

if elps == 1
    if size(I{k}) ~= size(Mask)
        I{k} = I{k}(1:size(Mask, 1), 1:size(Mask, 2));
        warn = 1;
    end
    I{k} = I{k} .* Mask;
    I{k}(I{k} == 0) = nan;
end

if LD ~= 'y'
    I{k}(I{k} > (loc(2) + w(2)*3/2.355)) = 0;
    I{k}(I{k} < (loc(2) - w(2)*1.5/2.355)) = 0;
    I{k}(isnan(I{k})) = 0;
else
    I{k}(I{k} > (loc(2) - w(2)*4/2.355)) = 0;
    I{k}(I{k} < (loc(1) + w(1)*4/2.355)) = 0;
    I{k}(isnan(I{k})) = 0;
end

if p == start && nnz(I{k}) < minpixelnum
    I{k}(I{k}>0) = 0;
    topbuffer = [topbuffer; k];
elseif nnz(I{k}) < minpixelnum
    I{k}(I{k}>0) = 0;
end

for bn = 1 : length(topbuffer)
    I{topbuffer(bn)}(I{k}>0) = 0;
end

%Compute numeric derivative for each slice
if diffyn == 'y'
    [DiffX, DiffY] = gradient(I{k});
    DiffX = abs(DiffX);
    DiffY = abs(DiffY);
    Diff = (DiffX + DiffY);
    DiffAv = nanmean(nanmean(Diff));
    DiffStd = nanmean(nanstd(Diff));
    Diff(Diff < DiffAv + 3*DiffStd) = 0;
    Diff(isnan(Diff)) = 0;
    IDiff{k} = Diff;
end

if Detonate ~= 'y'
    %Compress image by a factor of Cfactor
    [L, W] = size(I{k});
    L = floor(L/Cfactor);
    W = floor(W/Cfactor);
    I2Size = floor(size(I{k}, 1)/Cfactor);
    I2 = zeros(floor(size(I{k}, 1)/Cfactor), floor(size(I{k}, 2)/Cfactor));
    Icomp{k} = I{k}(1 : L*Cfactor, 1 : W*Cfactor);

    for i = 1 : I2Size - 1
        for j = 1 : I2Size - 1
            if LD ~= 'y'
                I2(i, j) = min(min(Icomp{k}(i*Cfactor : (i + 1)* Cfactor - 1,...
                    j*Cfactor : (j + 1)* Cfactor - 1)));
            end
        end
    end
end

```

```

        else
            I2(i, j) = min(min(Icomp{k}(i*Cfactor : (i+1)*Cfactor - 1,...
                                j*Cfactor : (j + 1)*Cfactor - 1)));
        end
    end
end

I2(:, end) = nan;
I2(end, :) = nan;
Icomp{k} = I2;

%{
if LD == 'y'
    Icomp{k}(Icomp{k} < 1.0e-4) = nan;
    Icomp{k}(Icomp{k} > 5.0e-4) = nan;
end
%}

if scctr == 'y'
    [rows, cols] = find(Icomp{k}~=0);
    z = repmat(k, length(rows), 1);
    Pnts = [Pnts; [rows cols z]];
end

if diffyn == 'y'
    D2 = zeros(floor(size(I{k}, 1)/Cfactor), floor(size(I{k},
2)/Cfactor));
    IDiffComp{k} = IDiff{k}(1 : L*Cfactor, 1 : W*Cfactor);
    for i = 1 : I2Size - 1
        for j = 1 : I2Size - 1
            D2(i, j) = max(max(IDiffComp{k}(i*Cfactor : (i + 1)* Cfactor
- 1,...
                                j*Cfactor : (j + 1)* Cfactor - 1)));
        end
    end
    IDiffComp{k} = D2;

    if scctr == 'y'
        [rows2, cols2] = find(IDiffComp{k}~=0);
        z2 = repmat(k, length(rows2), 1);
        Pnts2 = [Pnts2; [rows2 cols2 z2]];
    end
end

%Obtain Scatter Plot data for both normal and derivative compressed
%images
else
    %{
    if LD == 'y'
        I{k}(I{k} < 2.0e-4) = nan;
        I{k}(I{k} > 5.0e-4) = nan;
    end
    %}
end
end
end
%close(Wbar);

```

```

if scctr == 'y'
    %Scatter plot data computed above
    plot3(Pnts(:, 1), Pnts(:, 2), Pnts(:, 3), '.', 'MarkerSize', 2)
    zlim([-150 150]);
    figure
    plot3(Pnts2(:, 1), Pnts2(:, 2), Pnts2(:, 3), '.', 'MarkerSize', 2)
    zlim([-150 150]);
end

%Never Ever Use VolArray for big data!!!!
%inpt = input('plot slice array? (y/n)', 's');
if volary == 'y'

    for i = 1 : size(Icomp, 2)
        Icomp{i}(Icomp{i}==0) = nan;
        M1 = min(min(Icomp{i}));
        M2 = max(max(Icomp{i}));
        Icomp{i}(isnan(Icomp{i})) = 0;
        Icomp{i} = Icomp{i}/M2;

    end

    VolArray = Icomp{1};
    for i = 2 : size(Icomp, 2)
        VolArray(:, :, i) = Icomp{i};
    end
    figure
    s = slice(VolArray, [], [], 1 : size(Icomp, 2));
    alpha('color')
    set(s, 'EdgeColor', 'none');
    grid off
    zlim([-150 150]);

end

if Detonate == 'n'

    if LD ~= 'y'
        %Find top surface points
        top = [];
        ztop = zeros(size(Icomp{1}));
        surfmask = ones(size(Icomp{1}));
        for i = 1 : size(Icomp, 2)
            waitbar(i/size(Icomp, 2), Wbar, strcat('Building top surfaces for
file', {' ', num2str(p), ...
            {' ', 'of', {' ', num2str(fldrs))});
            Icomp{i}(isnan(Icomp{i})) = 0;
            temp = Icomp{i}.*surfmask;
            [rows, cols] = find(temp~=0);
            for j = 1 : length(rows)
                surfmask(rows(j),cols(j)) = 0;
            end
            z = repmat(size(Icomp, 2) - i + 1, length(rows), 1);
            top = [top; [rows cols z]];
        end

        %find bottom surface points

```



```

bottom = [];
zbot = zeros(size(Icomp{1}));
surfmask = ones(size(Icomp{1}));
for i = size(Icomp, 2) : -1 : 1
    waitbar((size(Icomp, 2)-i)/size(Icomp, 2), Wbar, strcat('Building
bottom surfaces for file', {' '}, num2str(p), ...
{' '}, 'of', {' '}, num2str(fldrs)));
    temp = Icomp{i}.*surfmask;
    [rows, cols] = find(temp~=0);
    for j = 1 : length(rows)
        surfmask(rows(j),cols(j)) = 0;
    end
    z = repmat(size(Icomp, 2) - i + 1 , length(rows), 1);
    bottom = [bottom; [rows cols z]];
end

%Build top Z matrix for surf plot
for i = 1 : size(top, 1)
    ztop(top(i, 1), top(i, 2)) = top(i, 3);
end

%Build bottom Z matrix for surf plot
for i = 1 : size(bottom, 1)
    zbot(bottom(i, 1), bottom(i, 2)) = bottom(i, 3);
end
ztop(ztop==0) = nan;
zbot(zbot==0) = nan;

else
    wtop = [];
    zwtop = zeros(size(Icomp{1}));
    surfmask = ones(size(Icomp{1}));
    for i = 1 : floor(size(Icomp, 2)/2)
        waitbar(i/floor(size(Icomp, 2)/2), Wbar, strcat('Building working
top surfaces for file', {' '}, num2str(p), ...
{' '}, 'of', {' '}, num2str(fldrs)));
        Icomp{i}(isnan(Icomp{i})) = 0;
        temp = Icomp{i}.*surfmask;
        [rows, cols] = find(temp~=0);
        for j = 1 : length(rows)
            surfmask(rows(j),cols(j)) = 0;
        end
        z = repmat(size(Icomp, 2) - i + 1, length(rows), 1);
        wtop = [wtop; [rows cols z]];
    end

    for i = 1 : size(wtop, 1)
        zwtop(wtop(i, 1), wtop(i, 2)) = wtop(i, 3);
    end

    wbottom = [];
    zwbot = zeros(size(Icomp{1}));
    surfmask = ones(size(Icomp{1}));
    for i = floor(size(Icomp, 2)/2) : -1 : 1
        waitbar((floor(size(Icomp, 2)/2)-i)/size(Icomp, 2), Wbar,
strcat('Building working bottom surfaces for file', {' '}, num2str(p), ...
{' '}, 'of', {' '}, num2str(fldrs)));

```

```

temp = Icomp{i}.*surfmask;
[rows, cols] = find(temp~=0);
for j = 1 : length(rows)
    surfmask(rows(j),cols(j)) = 0;
end
z = repmat(size(Icomp, 2) - i + 1 , length(rows), 1);
wbottom = [wbottom; [rows cols z]];
end

for i = 1 : size(wbottom, 1)
    zwbot(wbottom(i, 1), wbottom(i, 2)) = wbottom(i, 3);
end

ctop = [];
zctop = zeros(size(Icomp{1}));
surfmask = ones(size(Icomp{1}));
for i = floor(size(Icomp, 2)/2) + 1 : 1 : size(Icomp, 2)
    waitbar((i-floor(size(Icomp, 2)/2) - 1)/(size(Icomp, 2)-
floor(size(Icomp, 2)/2)-1), Wbar, strcat('Building counter top surfaces for
file', {' '}, num2str(p), ...
    {' '}, 'of', {' '}, num2str(fldrs)));
    Icomp{i}(isnan(Icomp{i})) = 0;
    temp = Icomp{i}.*surfmask;
    [rows, cols] = find(temp~=0);
    for j = 1 : length(rows)
        surfmask(rows(j),cols(j)) = 0;
    end
    z = repmat(size(Icomp, 2) - i + 1, length(rows), 1);
    ctop = [ctop; [rows cols z]];
end

for i = 1 : size(ctop, 1)
    zctop(ctop(i, 1), ctop(i, 2)) = ctop(i, 3);
end

cbottom = [];
zcbot = zeros(size(Icomp{1}));
surfmask = ones(size(Icomp{1}));
for i = size(Icomp, 2) : -1 : floor(size(Icomp, 2)/2) + 1
    waitbar((size(Icomp, 2) - i)/(size(Icomp, 2)-(floor(size(Icomp,
2)/2) + 1)), Wbar, strcat('Building counter bottom surfaces for file', {' '},
num2str(p), ...
    {' '}, 'of', {' '}, num2str(fldrs)));
    temp = Icomp{i}.*surfmask;
    [rows, cols] = find(temp~=0);
    for j = 1 : length(rows)
        surfmask(rows(j),cols(j)) = 0;
    end
    z = repmat(size(Icomp, 2) - i + 1 , length(rows), 1);
    cbottom = [cbottom; [rows cols z]];
end

for i = 1 : size(cbottom, 1)
    zcbot(cbottom(i, 1), cbottom(i, 2)) = cbottom(i, 3);
end

zwtop(zwtop==0) = nan;

```

```

    zwbot(zwbot==0) = nan;
    zctop(zctop==0) = nan;
    zcbot(zcbot==0) = nan;

    v1 = zwtop - zwbot;
    v2 = zctop - zcbot;
    v1(v1 < 5) = nan;
    v2(v2 < 5) = nan;
    v1(~isnan(v1)) = 1;
    v2(~isnan(v2)) = 1;
    zwtop = zwtop.*v1;
    zwbot = zwbot.*v1;
    zctop = zctop.*v2;
    zcbot = zcbot.*v2;

end

elseif Detonate == 'y'
    if LD ~= 'y'
        %Find top surface points
        top = [];
        ztop = zeros(size(I{1}));
        surfmask = ones(size(I{1}));
        for i = 1 : size(I, 2)
            waitbar(i/size(I, 2), Wbar, strcat('Building top surfaces for file',
{' '}, num2str(p), ...
{' '}, 'of', {' '}, num2str(fldrs)));
            I{i}(isnan(I{i})) = 0;
            temp = I{i}.*surfmask;
            [rows, cols] = find(temp~=0);
            for j = 1 : length(rows)
                surfmask(rows(j),cols(j)) = 0;
            end
            z = repmat(size(I, 2) - i + 1, length(rows), 1);
            top = [top; [rows cols z]];
        end

        %find bottom surface points
        bottom = [];
        zbot = zeros(size(I{1}));
        surfmask = ones(size(I{1}));
        for i = size(I, 2) : -1 : 1
            waitbar((size(I, 2) - i)/size(I, 2), Wbar, strcat('Building bottom
surfaces for file', {' '}, num2str(p), ...
{' '}, 'of', {' '}, num2str(fldrs)));
            temp = I{i}.*surfmask;
            [rows, cols] = find(temp~=0);
            for j = 1 : length(rows)
                surfmask(rows(j),cols(j)) = 0;
            end
            z = repmat(size(I, 2) - i + 1, length(rows), 1);
            bottom = [bottom; [rows cols z]];
        end

        %Build top Z matrix for surf plot
        for i = 1 : size(top, 1)

```

```

        ztop(top(i, 1), top(i, 2)) = top(i, 3);
    end

    %Build bottom Z matrix for surf plot
    for i = 1 : size(bottom, 1)
        zbot(bottom(i, 1), bottom(i, 2)) = bottom(i, 3);
    end
    ztop(ztop==0) = nan;
    zbot(zbot==0) = nan;

    else
        wtop = [];
        zwtop = zeros(size(I{1}));
        surfmask = ones(size(I{1}));
        for i = 1 : floor(size(I, 2)/2)
            waitbar(i/floor(size(I, 2)/2), Wbar, strcat('Building working top
surfaces for file', {' '}, num2str(p), ...
                {' '}, 'of', {' '}, num2str(fldrs)));
            I{i}(isnan(I{i})) = 0;
            temp = I{i}.*surfmask;
            [rows, cols] = find(temp~=0);
            for j = 1 : length(rows)
                surfmask(rows(j),cols(j)) = 0;
            end
            z = repmat(size(I, 2) - i + 1, length(rows), 1);
            wtop = [wtop; [rows cols z]];
        end

        for i = 1 : size(wtop, 1)
            zwtop(wtop(i, 1), wtop(i, 2)) = wtop(i, 3);
        end

        wbottom = [];
        zwbot = zeros(size(I{1}));
        surfmask = ones(size(I{1}));
        for i = floor(size(I, 2)/2) : -1 : 1
            waitbar((floor(size(I, 2)/2) - i)/size(I, 2), Wbar,
strcat('Building working bottom surfaces for file', {' '}, num2str(p), ...
                {' '}, 'of', {' '}, num2str(fldrs)));
            temp = I{i}.*surfmask;
            [rows, cols] = find(temp~=0);
            for j = 1 : length(rows)
                surfmask(rows(j),cols(j)) = 0;
            end
            z = repmat(size(I, 2) - i + 1, length(rows), 1);
            wbottom = [wbottom; [rows cols z]];
        end

        for i = 1 : size(wbottom, 1)
            zwbot(wbottom(i, 1), wbottom(i, 2)) = wbottom(i, 3);
        end

        ctop = [];
        zctop = zeros(size(I{1}));
        surfmask = ones(size(I{1}));
        for i = floor(size(I, 2)/2) + 1 : 1 : size(I, 2)

```

```

        waitbar((i-floor(size(I, 2)/2) - 1)/(size(I, 2)-floor(size(I,
2)/2)-1), Wbar, strcat('Building counter top surfaces for file', {' '},
num2str(p), ...
        {' '}, 'of', {' '}, num2str(fldrs)));
        I{i}(isnan(I{i})) = 0;
        temp = I{i}.*surfmask;
        [rows, cols] = find(temp~=0);
        for j = 1 : length(rows)
            surfmask(rows(j),cols(j)) = 0;
        end
        z = repmat(size(I, 2) - i + 1, length(rows), 1);
        ctop = [ctop; [rows cols z]];
    end

    for i = 1 : size(ctop, 1)
        zctop(ctop(i, 1), ctop(i, 2)) = ctop(i, 3);
    end

    cbottom = [];
    zcbot = zeros(size(I{1}));
    surfmask = ones(size(I{1}));
    for i = size(I, 2) : -1 : floor(size(I, 2)/2) + 1
        waitbar((size(I, 2) - i)/(size(I, 2)-(floor(size(I, 2)/2) + 1)),
Wbar, strcat('Building counter bottom surfaces for file', {' '}, num2str(p),
...
        {' '}, 'of', {' '}, num2str(fldrs)));
        temp = I{i}.*surfmask;
        [rows, cols] = find(temp~=0);
        for j = 1 : length(rows)
            surfmask(rows(j),cols(j)) = 0;
        end
        z = repmat(size(I, 2) - i + 1, length(rows), 1);
        cbottom = [cbottom; [rows cols z]];
    end

    for i = 1 : size(cbottom, 1)
        zcbot(cbottom(i, 1), cbottom(i, 2)) = cbottom(i, 3);
    end

    zwtop(zwtop==0) = nan;
    zwbot(zwbot==0) = nan;
    zctop(zctop==0) = nan;
    zcbot(zcbot==0) = nan;

    v1 = abs(zwtop - zwbot);
    v2 = abs(zctop - zcbot);
    v1(v1 < 10) = nan;
    v2(v2 < 10) = nan;
    v1(v1 > 30) = nan;
    v2(v2 > 30) = nan;

    v1(~isnan(v1)) = 1;
    v2(~isnan(v2)) = 1;
    zwtop = zwtop.*v1;
    zwbot = zwbot.*v1;
    zctop = zctop.*v2;
    zcbot = zcbot.*v2;

```

```

    end
end
close(Wbar);
%% Plotting Utilities
a = -1*size(I, 2);
b = 2*size(I, 2);
%Plot Surfaces
if PTB == 'y'
    figure
    subplot(1, 2, 1);
    surf(ztop, 'linestyle', 'none');
    xlim([cellstart cellend]);
    ylim([cellstart cellend]);
    zlim([a b]);
    daspect([1 1 2]);

    subplot(1, 2, 2);
    surf(zbot, 'linestyle', 'none');
    xlim([cellstart cellend]);
    ylim([cellstart cellend]);
    zlim([a b]);
    daspect([1 1 2]);
else
    if LD ~= 'y'
        figure
        h = surf(ztop, 'linestyle', 'none');
        hold on
        surf(zbot, 'linestyle', 'none');
        hold on
        xlim([cellstart cellend]);
        ylim([cellstart cellend]);
        zlim([a b]);
        daspect([1 1 2]);
    else
        figure
        h = surf(zwtop, 'linestyle', 'none');
        hold on
        surf(zwbot, 'linestyle', 'none');
        hold on
        surf(zctop, 'linestyle', 'none');
        hold on
        surf(zcbot, 'linestyle', 'none');
        xlim([cellstart cellend]);
        ylim([cellstart cellend]);
        zlim([a b]);
        daspect([1 1 2]);
    end
end
end
r=cellwidth/2;
teta=-pi:0.01:pi;
x = r*cos(teta) + cellstart + cellwidth/2;
y = r*sin(teta) + cellstart + cellwidth/2;
if DCV == 'y'
    plot3(x,y, repmat(a , 1, length(x)));
end

```

```

%colormap([flipud(gray); gray])
%set(gca, 'visible', 'off')

%Save this figure for later use
if stack == 'y'
cd(Path)
name = Filenames{p};
elseif stack == 'n'
cd(Path)
cd ..
cd ..
S = strsplit(Path, '\');
name = strcat(S{end}, '.fig');
end

if isempty(inpt{8})
    fname = 'Matlab_Figures';
else
    fname = inpt{8};
end

if ~exist(fname, 'dir')
    mkdir(fname);
end
cd(fname);
hgsave(gcf, name, '-v7.3');
cd(A);

%Smooth surfaces with gaussian masking
if smth == 'y'
    G = fspecial('gaussian', [3 3], 2);
    STop = imfilter(real(ztop), G, 'same');
    SBot = imfilter(real(zbot), G, 'same');

    %Plot smoothed surfaces
    figure
    surf(STop);
    hold on
    surf(SBot);
    xlim([cellstart cellend]);
    ylim([cellstart cellend]);
    a = -1*size(I, 2);
    b = 2*size(I, 2);
    zlim([a b]);
    daspect([1 1 2])
end
if stack == 'y'
    close all
end
end

clearvars -except bottom Cfactor G I Icomp IDiff IDiffComp Path Pnts Pnts2
SBot STop top zbot ztop cellstart cellend S zctop zcbot zwtop zwbot

```

```

case 4
%% Case 4 Program Description
% Script pulls images computed by Case 3, allows user manipulation of
% the sample view and outputs a video of time dependant changes occuring to
% the pellet surface morphology. Script also imports electrochemical data
% to match up with data time stamps. General Work Flow: Import relevant
% data -> initially plot pellet -> request user intervention to define
% desired image orientation -> apply orientation to each subsequent image
% -> output video.

disp('Executing Video Analysis');
%use existing ROIS
UE = 'y';
if UE == 'y'
    clearvars -except ROIS UE
else
    clear all
    UE = 'n';
end

close all
set(0,'DefaultFigureWindowStyle','docked')
warning('off', 'images:imshow:magnificationMustBeFitForDockedFigure');
A = pwd;

%Omit Surface Lines
OSL = 'y';
%Omit Axes
OA = 'n';
%set view from top and bottom
TB = 'n';
%Include Colorbar
CB = 'n';
% Plot Electrochemistry on Same Window
EC = 'n';
EPOS = 'bottom';

% Hard set Current limits
HS = 'y';
current = 10;
%If Reconstructing split top and bottom data
PTB = 'n';
%Rectangular region of interest
ROI = 'n';
% NI regions of interest
% Set NI to 0 to ignore command
NI = 0;
%Highlight the low density regions of material and superimpose that plot
%on top of the surface itself
LD = 'n';
Trode = 'W';
% Open one image but don't run movie
Terminate = 'n';
% Process Figures with scale bar and black background
% Not yet set up for multiple axes systems like PTB, NI, EC, or LD

```



```

ImProc = 'y';
ImProcROI = 'y';
cfactor = 1;
ctour = 'n';

%Import Electrochemistry
if EC == 'y'
[EchemName, EchemDir] = uigetfile('*..*', 'All Files (*.*)', 'MultiSelect',
'on');

for i = 1 : length(EchemName)
    EchemName(i) = fullfile(EchemDir, EchemName(i));
    C = importdata(EchemName{i}, ',');
    if ~isempty(strfind(EchemName{i}, 'OCP'))
        t1 = C.data(:, 1);
        V1 = C.data(:, 2);
        I1 = zeros(length(t1), 1);
    end
    if ~isempty(strfind(EchemName{i}, 'CVnarrow'))
        V2 = C.data(:, 1);
        I2 = C.data(:, 2);
        t2 = [1 : 5 : length(V2)*5] + t1(end);
    end
    if ~isempty(strfind(EchemName{i}, 'CVwide'))
        V3 = C.data(:, 1);
        I3 = C.data(:, 2);
        t3 = [1 : 5 : length(V3)*5] + t2(end);
    end
end
t = [t1; t2'; t3'];
t = t/3600;
V = [V1; V2; V3];
I = [I1; I2; I3];
I = I*1000000;
end

[VA, VADir] = uigetfile('*..*', 'All Files (*.*)');
VA = fullfile(VADir, VA);
Tbl = readtable(VA, 'Delimiter', '\t');
Tbl = table2array(Tbl(:, 2));

%Running Program as normal
if LD ~= 'y'
    Path = uigetdir();
    cd(Path);
    fls = dir;
    N = find(cell2mat({fls.isdir}) == 0);
    names = {fls.name};
    names = names(N);
    names = names(~cellfun('isempty', strfind(names, '.fig')));
    NumberOfFrames = size(names, 2);

    %% Open first image and request orientation from user

    openfig(names{40});

    if OSL == 'y'

```

```

surface = findobj(gcf, 'type', 'surface');
set(surface(1), 'linestyle', 'none');
set(surface(2), 'linestyle', 'none');
L = findall(gcf, 'type', 'line');
if ~isempty(L)
    for i = 1 : length(L)
        delete(L(i));
    end
end
end

if NI == 0

if EC == 'y'
    % Initialize
    figure1 = gcf;
    axes1 = gca;

    % Move 3D plot up

    if PTB == 'y'
        axes1 = get(gcf, 'children');
        set(axes1(2), 'Position', [0 0.25 0.5 0.70]);
        set(axes1(1), 'Position', [0.5 0.25 0.5 0.70]);
    else
        set(axes1, 'Position', [0.1 0.25 .8 0.70]);
    end

    %Create New Axis and Plot Echem Data

    if PTB == 'y'
        axes2 = axes('Parent', figure1, 'OuterPosition', [0 0.05 1
0.20]);
    else
        axes2 = axes('Parent', figure1, 'OuterPosition', [0.15 0.05 0.7
0.20]);
    end
    [axes2, line1, line2] = plotyy(t, I, t, V);

    % Set the desired plotting window for Echem based on available
    % tomographic reconstructions

    xlim(axes2(1), [Tbl(1)/3600 Tbl(end)/3600]);
    xlim(axes2(2), [Tbl(1)/3600 Tbl(end)/3600]);
    ylim(axes2(1), 'auto');
    if HS == 'y'
        y = current;
    else
        y = get(axes2(1), 'ylim');
        y = max(abs(y));
    end
    ylim(axes2(1), [-y y]);
    line(get(axes2(1), 'XLim'), [0 0], 'Color', [0 0 0], 'LineStyle', '--
');

```

```

% Make Echem plot look nice
line1.LineWidth = 2;
line1.Color = 'r';
line2.LineWidth = 2;
line2.Color = 'b';
set(axes2, {'ycolor'}, {'r'; 'b'});
set(axes2, 'FontSize', 24);
xlabel(axes2(1), 'Time (h)');
ylabel(axes2(1), 'Current (\muA)');
ylabel(axes2(2), 'Potential (V)');
set(axes2(1), 'YTick', [-y : y : y]);

% Plot a vertical line depicting current Echem state
hold(axes2(1), 'on')
line([Tb1(1)/3600 Tb1(1)/3600], get(axes2(1), 'YLim'), 'Color', [1 0 0],
'LineWidth', 2);

if strcmp(EPOS, 'side')
    camroll(axes2(1), -90)
    camroll(axes2(2), -90)
    set(axes2, 'OuterPosition', [0.15 0 0.1 1]);
    set(axes1, 'Position', [0 0 1 1]);
end

else
    figure1 = gcf;
    axes1 = gca;
end

if TB == 'y'
    if PTB == 'y'
        set(axes1(2), 'view', [180 90]);
        set(axes1(1), 'view', [0 -90]);
    else
        set(axes1, 'view', [180 90]);
    end
end

if ROI == 'y'
    if PTB == 'y'
        h2 = imrect(axes1(2));
        pause
        Pos = getPosition(h2);
        delete(h2)
        set(axes1(2), 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(axes1(2), 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
        set(axes1(1), 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(axes1(1), 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
    else
        h2 = imrect(axes1);
        pause
        Pos = getPosition(h2);
        delete(h2)
        set(axes1, 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(axes1, 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
    end
end

```

```

    end
end

set(axes1, 'visible', 'on');
set(gcf, 'Color', 'w');
axes1.Color = 'w';
axes1.Box = 'on';
axes1.LineWidth = 2.0;
axes1.FontSize = 24.0;
axes1.GridAlpha = 0.2;
axes1.XColor = 'k';
axes1.YColor = 'k';
axes1.ZColor = 'k';

surface = findall(axes1, 'type', 'surface');
hgt2 = 50*ceil(max(max(surface(2).ZData))/50);
%low = hgt - 100;
hgt = max(max(surface(2).ZData));
low = min(min(surface(2).ZData));
axes1.ZLim = [0 hgt2];
daspect(axes1, [1 1 1]);
brighten(-0.5);
xstart = 250*ceil(axes1.XLim(1)/250);
xend = 250*floor(axes1.XLim(2)/250);
axes1.XTick = [xstart: 250: xend];
axes1.YTick = [xstart: 250: xend];

c1 = colorbar(axes1);
c1.Label.String = 'Z-Height (\mum)';
c1.FontSize = 24.0;
c1.FontWeight = 'bold';
c1.Color = 'k';
set(c1, 'ylim', [low hgt]);

%User Sets Orientation of surface

pause

%% Obtain orientation information from image
if CB == 'y'
    c = colorbar;
    c.Label.String = 'Z-height (pixels)';
    set(gca, 'FontSize', 24);
end

if PTB == 'y'

    camerat1 = get(axes1(2), 'CameraTarget');
    camerat2 = get(axes1(2), 'CameraPosition');
    camerat3 = get(axes1(2), 'CameraViewAngle');
    camerat4 = get(axes1(2), 'CameraUpVector');

    camerab1 = get(axes1(1), 'CameraTarget');

```

```

camerab2 = get(axes1(1), 'CameraPosition');
camerab3 = get(axes1(1), 'CameraViewAngle');
camerab4 = get(axes1(1), 'CameraUpVector');

if OA == 'y'
    set(axes1(1), 'visible', 'off');
    set(axes1(2), 'visible', 'off');
    set(gcf, 'color', 'w');
end
else
    if ~isempty(axes1.XTickLabel)
        axes1.XTickLabel(end) = strcat(axes1.XTickLabel(end), '\mum');
    end
    if ~isempty(axes1.YTickLabel)
        axes1.YTickLabel(end) = strcat(axes1.YTickLabel(end), '\mum');
    end
    if ~isempty(axes1.ZTickLabel)
        axes1.ZTickLabel(end) = strcat(axes1.ZTickLabel(end), '\mum');
    end

    camera1 = get(gca, 'CameraTarget');
    camera2 = get(gca, 'CameraPosition');
    camera3 = get(gca, 'CameraViewAngle');
    camera4 = get(gca, 'CameraUpVector');

    if OA == 'y'
        set(gca, 'visible', 'off');
        set(gcf, 'color', 'w');
    end
end

end

else
    surface = findobj(gcf, 'type', 'surface');
    set(gca, 'view', [180 90]);
    temp1 = get(gca, 'XLim');
    temp2 = get(gca, 'YLim');
    x = surface(2).XData;
    y = surface(2).YData;
    z = surface(2).ZData;
    sub = max(max(z))+1;
    z = z - sub;
    figure;
    surf(x, y, z, 'LineStyle', 'None');
    if UE ~= 'y'
        h = {};
        ROIS = {};
    end

    set(gca, 'Xlim', temp1);
    set(gca, 'Ylim', temp2);
    set(gca, 'view', [180 90]);
    daspect([1 1 1]);

    t1s = ceil(sqrt(NI));
    if UE ~= 'y'
        for i = 1 : NI

```

```

        h{i} = imellipse(gca);
        pause
        ROIS{i} = getPosition(h{i});
    end
end

z = z + sub;

XPOS = 0;
YPOS = 0;
DIM = 1/tls;

for i = 1 : NI - 1
    XPOS = [XPOS; floor( i/tls)*DIM];
    YPOS = [YPOS; rem(i, tls)*DIM];
end

figure;
SAxes = {};

for i = 1 : NI
    SAxes{i} = axes('Parent', gcf, 'OuterPosition', [XPOS(i) - .035,
YPOS(i) - .035, DIM + .07, DIM + .07]);
    surf(x, y, z, 'LineStyle', 'None');
    set(gca, 'Xlim', [ROIS{i}(1) (ROIS{i}(1) + ROIS{i}(3))]);
    set(gca, 'Ylim', [ROIS{i}(2) (ROIS{i}(2) + ROIS{i}(3))]);
    set(gca, 'view', [180 90]);
    daspect([1 1 1]);
    set(gca, 'visible', 'off');
    set(gcf, 'color', 'w');

end

pause

v = get(SAxes{1}, 'view');

for i = 2 : size(SAxes, 2)
    set(SAxes{i}, 'view', v);
end

%Construct scale bars and text
sf = 3; %scaling factor from x units to microns
sze = 150; %scalebar size in microns
for i = 1 : NI
    axes(SAxes{i});
    xpos = [(ROIS{i}(1) + .05 * ROIS{i}(3)); (ROIS{i}(1) + .05 *
ROIS{i}(3) + sze/sf)];
    ypos = [(ROIS{i}(2) + .95 * ROIS{i}(3)); (ROIS{i}(2) + .95 *
ROIS{i}(3))];
    if v(2) == 90
        zpos = [(max(max(z)) + 10); (max(max(z)) + 10)];
        hold on
        line(xpos, ypos, zpos, 'LineWidth', 6, 'Color', 'k');
        text(ROIS{i}(1) + .05 * ROIS{i}(3) + sze/(2*sf), ROIS{i}(2) + .85
* ROIS{i}(3), zpos(2), strcat(num2str(sze), '\mum'), 'FontSize', 18,
'FontWeight', 'bold', 'Color', 'k', 'HorizontalAlignment', 'center');
    end
end

```

```

        hold off
    end
end
end

%Case for opening multiple superimposed images
else
    disp('Navigate to parent image directory');
    ParentPath = uigetdir();
    disp('Navigate to low density image directory');
    LDPath = uigetdir();

    %Obtain directories for parent images
    cd(ParentPath);
    fls = dir;
    N = find(cell2mat({fls.isdir}) == 0);
    ParentNames = {fls.name};
    ParentNames = ParentNames(N);
    ParentNames = ParentNames(~cellfun('isempty', strfind(ParentNames,
'.fig')));
    NumberOfFrames = size(ParentNames, 2);
    openfig(ParentNames{45});

    xlm = get(gca, 'XLim');
    ylm = get(gca, 'YLim');
    zlm = get(gca, 'ZLim');
    aspct = get(gca, 'PlotBoxAspectRatio');

    surfacel = findobj(gcf, 'type', 'surface');
    xpellet = {};
    ypellet = {};
    zpellet = {};
    subpellet = {};

    for i = 1 : size(surfacel, 1)
        xpellet{i} = surfacel(i).XData;
        ypellet{i} = surfacel(i).YData;
        zpellet{i} = surfacel(i).ZData;
        subpellet{i} = max(max(zpellet{i})) + 1;
    end

    close(gcf);

    %Obtain LD directories
    cd(LDPath);
    fls = dir;
    N = find(cell2mat({fls.isdir}) == 0);
    LDNames = {fls.name};
    LDNames = LDNames(N);
    LDNames = LDNames(~cellfun('isempty', strfind(LDNames, '.fig')));

    openfig(LDNames{45});
    surface2 = findobj(gcf, 'type', 'surface');
    xld = {};
    yld = {};
    zld = {};

```

```

subLD = {};

for i = 1 : size(surface2, 1)
    xld{i} = surface2(i).XData;
    yld{i} = surface2(i).YData;
    zld{i} = surface2(i).ZData;
    subLD{i} = max(max(zld{i})) + 1;
end
h = gcf;
figure
close(h);

if Trode == 'W'
    [Gmag, Gpos] = imgradient(zpellet{2}, 'Prewitt');
    zpellet{2}(abs(Gmag) > 30) = nan;
    if ROI ~= 'y'
        surf(xpellet{2}, ypellet{2}, zpellet{2}, 'linestyle', 'none');
    else
        surf(xpellet{2}, ypellet{2}, zpellet{2} - subpellet{2},
'linestyle', 'none');
    end
    ax1 = gca;

    set(ax1, 'XLim', xlm);
    set(ax1, 'YLim', ylm);
    set(ax1, 'ZLim', zlm);
    set(ax1, 'PlotBoxAspectRatio', aspct);

    ax2 = axes;
    hold on
    if ROI ~= 'y'
        surf(xld{3}, yld{3}, zld{3}, 'linestyle', 'none');
        surf(xld{4}, yld{4}, zld{4}, 'linestyle', 'none');
    else
        surf(xld{3}, yld{3}, zld{3} - subLD{4}, 'linestyle', 'none');
        surf(xld{4}, yld{4}, zld{4} - subLD{4}, 'linestyle', 'none');
    end

    Link = linkprop([ax1, ax2], {'CameraUpVector', 'CameraPosition', ...
                                'CameraTarget', 'CameraViewAngle', ...
                                'XLim', 'YLim', 'ZLim', 'View', ...
                                'PlotBoxAspectRatio'});

    colormap(ax1, 'gray');
    colormap(ax2, flipud('hot'));
    axes(ax1);
    cameratoolbar('ToggleSceneLight');
    axes(ax2);
    set(ax2, 'visible', 'off');

elseif Trode == 'C'
    [Gmag, Gpos] = imgradient(zpellet{1}, 'Prewitt');
    zpellet{1}(abs(Gmag) > 30) = nan;
    if ROI ~= 'y'
        surf(xpellet{1}, ypellet{1}, zpellet{1}, 'linestyle', 'none');
    else

```



```

        surf(xpellet{1}, ypellet{1}, zpellet{1} -
subpellet{1},'linestyle', 'none');
    end
    ax1 = gca;

    set(ax1, 'XLim', xlm);
    set(ax1, 'YLim', ylm);
    set(ax1, 'ZLim', zlm);
    set(ax1, 'PlotBoxAspectRatio', aspc);

    ax2 = axes;
    hold on
    if ROI ~= 'y'
        surf(xld{1}, yld{1}, zld{1},'linestyle', 'none');
        surf(xld{2}, yld{2}, zld{2},'linestyle', 'none');
    else
        surf(xld{1}, yld{1}, zld{1} - subLD{2},'linestyle', 'none');
        surf(xld{2}, yld{2}, zld{2} - subLD{2},'linestyle', 'none');
    end
    Link = linkprop([ax1, ax2],{'CameraUpVector', 'CameraPosition',...
                                'CameraTarget', 'CameraViewAngle',...
                                'XLim', 'YLim', 'ZLim', 'View',...
                                'PlotBoxAspectRatio'});

    colormap(ax1, 'gray');
    colormap(ax2, flipud('hot'));
    axes(ax1);
    cameratoolbar('ToggleSceneLight');
    axes(ax2);
    set(ax2, 'visible', 'off');
end

if ROI == 'y'
    view(2);
    h = imellipse;
    pause
    Pos = getPosition(h);
    xlm = [Pos(1) (Pos(1) + Pos(3))];
    ylm = [Pos(2) (Pos(2) + Pos(3))];
    h = gcf;
    figure
    close(h);

    if Trode == 'W'

        surf(xpellet{2}, ypellet{2}, zpellet{2}, 'linestyle', 'none');
        ax1 = gca;
        set(ax1, 'XLim', xlm);
        set(ax1, 'YLim', ylm);
        zlm = [min(min(zpellet{2})) - 10, max(max(zpellet{2}))+10];
        set(ax1, 'ZLim', zlm);

        ax2 = axes;
        hold on
        surf(xld{3}, yld{3}, zld{3},'linestyle', 'none');
        surf(xld{4}, yld{4}, zld{4},'linestyle', 'none');
        Link = linkprop([ax1, ax2],{'CameraUpVector',
'CameraPosition',...

```

```

        'CameraTarget', 'CameraViewAngle',...
        'XLim', 'YLim', 'ZLim', 'View',...
        'PlotBoxAspectRatio'}));

colormap(ax1, 'gray');
colormap(ax2, flipud('hot'));
axes(ax1);
cameratoolbar('ToggleSceneLight');
axes(ax2);
set(ax2, 'visible', 'off');

elseif Trode == 'C'

    surf(xpellet{1}, ypellet{1}, zpellet{1}, 'linestyle', 'none');
    ax1 = gca;
    set(ax1, 'XLim', xlm);
    set(ax1, 'YLim', ylm);
    zlm = [min(min(zpellet{1})) - 10, max(max(zpellet{1}))+10];
    set(ax1, 'ZLim', zlm);
    set(ax1, 'PlotBoxAspectRatio', aspct);
    ax2 = axes;
    hold on
    surf(xld{2}, yld{2}, zld{2}, 'linestyle', 'none');
    surf(xld{2}, yld{2}, zld{2}, 'linestyle', 'none');
    Link = linkprop([ax1, ax2], {'CameraUpVector',
'CameraPosition',...
        'CameraTarget', 'CameraViewAngle',...
        'XLim', 'YLim', 'ZLim', 'View',...
        'PlotBoxAspectRatio'}));

    colormap(ax1, 'gray');
    colormap(ax2, flipud('hot'));
    axes(ax1);
    cameratoolbar('ToggleSceneLight');
    axes(ax2);
    set(ax2, 'visible', 'off');

end
end

disp('User Defines Orientation');
pause

camera1 = get(gca, 'CameraTarget');
camera2 = get(gca, 'CameraPosition');
camera3 = get(gca, 'CameraViewAngle');
camera4 = get(gca, 'CameraUpVector');

if OA == 'y'
    set(ax1, 'visible', 'off');
    set(ax2, 'visible', 'off');
    set(gcf, 'color', 'w');
end

end

```

```

if ImProc == 'y'
    ax = gca;
    set(ax, 'visible', 'on');
    set(gcf, 'Color', 'w');
    ax.Color = 'w';
    ax.Box = 'on';
    ax.LineWidth = 2.0;
    ax.FontSize = 24.0;
    ax.GridAlpha = 0.2;
    ax.XColor = 'k';
    ax.YColor = 'k';
    ax.ZColor = 'k';
    %{
    xwidth = ax.XLim(2) - ax.XLim(1);
    xstart = ax.XLim(1) - xwidth;
    xend = ax.XLim(2) + xwidth;
    ax.XLim = [xstart xend];
    ax.YLim = [xstart xend];
    %}
    surface = findall(gca, 'type', 'surface');
    hgt2 = 50*ceil(max(max(surface(2).ZData))/50);
    %low = hgt - 100;
    hgt = max(max(surface(2).ZData));
    low = min(min(surface(2).ZData));
    ax.ZLim = [0 hgt2];
    daspect([1 1 1]);
    brighten(-0.5);

    if ctour == 'y'
        z = surface(2).ZData;
        hold on
        contour3(z, 'Color', 'k');
        hold off
    end

    if ImProcROI == 'y'
        % Hard set to look at working electrode
        z = surface(2).ZData;
        x = surface(2).XData;
        y = surface(2).YData;
        tmp = gcf;
        figure

        [posfile, posdir] = uigetfile('*..*', 'All Files (*.*)');

        if posfile ~= 0
            posdata = fullfile(posdir, posfile);
            Pos = importdata(posdata, ',', '');
        else

            sub = max(max(z)) + 10;
            z = z-sub;
            s = surf(x, y, z, 'LineStyle', 'None');
            daspect([1 1 1]);
            zlim('auto');
            view(2);
            h2 = imrect(gca);

```

```

        pause
        Pos = getPosition(h2);
        z = z + sub;
        delete(s);
        delete(h2);
        csvwrite('Pos.csv', Pos);
    end

    z = z(Pos(2) : (Pos(2) + Pos(3)), Pos(1) : (Pos(1) + Pos(3)));
    x = [Pos(2) : (Pos(2) + Pos(3))];
    y = [Pos(1) : (Pos(1) + Pos(3))];
    x = x - Pos(2);
    y = y - Pos(1);

    %figure
    surf(x, y, z, 'LineStyle', 'None');
    if ctour == 'y'
        hold on
        contour3(z, 20, 'Color', 'k');
    end
    daspect([1 1 1]);
    zlim([low hgt2]);
    caxis([low hgt]);
    view(3);
    set(gca, 'xlim', [min(x) max(x)]);
    set(gca, 'ylim', [min(y) max(y)]);

    set(gca, 'visible', 'on');
    set(gcf, 'Color', 'w');
    ax2 = gca;
    ax2.Color = 'w';
    ax2.Box = 'on';
    ax2.LineWidth = 2.0;
    ax2.GridAlpha = 0.2;
    ax2.XColor = 'k';
    ax2.YColor = 'k';
    ax2.ZColor = 'k';

end
% Note: setting the x and y axis ticks assumes no data compression
% Working on ensureing it can work with compressed data
%Put Proper Scaling Code Here

lines = findall(gcf, 'type', 'line');
if ~isempty(lines)
    for i = 1 : length(lines)
        delete(lines(i));
    end
end

xstart = 250*ceil(ax.XLim(1)/250);
xend = 250*floor(ax.XLim(2)/250);
ax.XTick = [xstart: 250: xend];
ax.YTick = [xstart: 250: xend];

```

```

if ImProcROI ~= 'y'

    %{
    x1 = [250, 1250, nan, 250, 250, nan, 0, 0];
    y1 = [250, 250, nan, 250, 1250, nan, 0, 0];
    z1 = [0, 0, nan, 0, 0, nan, 0, 0];
    %}

    x1 = [0, 1000, nan, 0, 0, nan, 0, 0];
    y1 = [0, 0, nan, 0, 1000, nan, 0, 0];
    z1 = [0, 0, nan, 0, 0, nan, 0, 0];

    h1 = line(x1, y1, z1);
    h1.Color = 'w';
    h1.LineWidth = 5;

    %{
    ht1 = text(250,625,0,[num2str(1), ' ', 'mm']);
    ht2 = text(625,250,0,[num2str(1), ' ', 'mm']);
    %}

    ht1 = text(0,500,0,[num2str(1), ' ', 'mm']);
    ht2 = text(500,0,0,[num2str(1), ' ', 'mm']);

    ht1.FontSize = 24.0;
    ht2.FontSize = 24.0;
    ht1.FontWeight = 'bold';
    ht2.FontWeight = 'bold';
    ht1.Color = 'w';
    ht2.Color = 'w';
    ht1.VerticalAlignment = 'top';
    ht2.VerticalAlignment = 'top';
    ht1.HorizontalAlignment = 'right';
    ht2.HorizontalAlignment = 'left';
else
    set(gca, 'FontSize', 24);
    brighten(-0.5);
end

c1 = colorbar;
c1.Label.String = 'Z-Height (\mum)';
c1.FontSize = 24.0;
c1.FontWeight = 'bold';
c1.Color = 'k';
set(c1, 'ylim', [low hgt]);

set(gca, 'CameraTarget', camera1);
set(gca, 'CameraPosition', camera2);
set(gca, 'CameraViewAngle', camera3);
set(gca, 'CameraUpVector', camera4);

disp('Reposition if Desired');
pause
drawnow
if ImProcROI == 'y'
    close(tmp);

```

```

ax = gca;
if ~isempty(ax.XTickLabel)
    ax.XTickLabel(end) = strcat(ax.XTickLabel(end), '\mum');
end
if ~isempty(ax.YTickLabel)
    ax.YTickLabel(end) = strcat(ax.YTickLabel(end), '\mum');
end
if ~isempty(ax.ZTickLabel)
    ax.ZTickLabel(end) = strcat(ax.ZTickLabel(end), '\mum');
end
else
    if ~isempty(ax.XTickLabel)
        ax.XTickLabel(end) = strcat(ax.XTickLabel(end), '\mum');
    end
    if ~isempty(ax.YTickLabel)
        ax.YTickLabel(end) = strcat(ax.YTickLabel(end), '\mum');
    end
    if ~isempty(ax.ZTickLabel)
        ax.ZTickLabel(end) = strcat(ax.ZTickLabel(end), '\mum');
    end
end

camera1 = get(gca, 'CameraTarget');
camera2 = get(gca, 'CameraPosition');
camera3 = get(gca, 'CameraViewAngle');
camera4 = get(gca, 'CameraUpVector');
end
if Terminate == 'y'
    cd(A);
    return
end

%{
drawnow;
h =(gcf;
Frames = getframe(h);
%}

%% Loop through each image in folder path
for i = 1 : NumberOfFrames
    if LD ~= 'y'
        openfig(names{i});

        if OSL == 'y'
            surface = findobj(gcf, 'type', 'surface');
            set(surface(1), 'linestyle', 'none');
            set(surface(2), 'linestyle', 'none');
        end

        if NI == 0
            if CB == 'y'
                c = colorbar;
                c.Label.String = 'Z-height (pixels)';
                set(gca, 'FontSize', 24);
            end

```

```

if PTB == 'y'
    axes1 = get(gcf, 'children');
    if ROI == 'y'
        set(axes1(2), 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(axes1(2), 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
        set(axes1(1), 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(axes1(1), 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
    end

    set(axes1(2), 'CameraTarget', camerat1);
    set(axes1(2), 'CameraPosition', camerat2);
    set(axes1(2), 'CameraViewAngle', camerat3);
    set(axes1(2), 'CameraUpVector', camerat4);

    set(axes1(1), 'CameraTarget', camerab1);
    set(axes1(1), 'CameraPosition', camerab2);
    set(axes1(1), 'CameraViewAngle', camerab3);
    set(axes1(1), 'CameraUpVector', camerab4);

    if OA == 'y'
        set(axes1(1), 'visible', 'off');
        set(axes1(2), 'visible', 'off');
        set(gcf, 'color', 'w');
    end
end

else
    if ROI == 'y'
        set(gca, 'xlim', [Pos(1) (Pos(1) + Pos(3))]);
        set(gca, 'ylim', [Pos(2) (Pos(2) + Pos(3))]);
    end

    set(gca, 'CameraTarget', camera1);
    set(gca, 'CameraPosition', camera2);
    set(gca, 'CameraViewAngle', camera3);
    set(gca, 'CameraUpVector', camera4);

    if OA == 'y'
        set(gca, 'visible', 'off');
        set(gcf, 'color', 'w');
    end
end

axes1 = gca;
if EC == 'y'
    % Initialize
    figure1 = gcf;
    axes1 = gca;

    % Move 3D plot up
    if PTB == 'y'
        axes1 = get(gcf, 'children');
        set(axes1(2), 'Position', [0 0.25 0.5 0.70]);
        set(axes1(1), 'Position', [0.5 0.25 0.5 0.70]);
    else
        set(axes1, 'Position', [0.1 0.25 0.8 0.70]);
    end
end

%Create New Axis and Plot Echem Data

```

```

        if PTB == 'y'
            axes2 = axes('Parent', figure1, 'OuterPosition', [0 0.05 1
0.20]);
        else
            axes2 = axes('Parent', figure1, 'OuterPosition', [.15 0.05 0.7
0.20]);
        end
        [axes2, line1, line2] = plotyy(t, I, t, V);

        % Set the desired plotting window for Echem based on available
        % tomographic reconstructions
        xlim(axes2(1), [Tb1(1)/3600 Tb1(end)/3600]);
        xlim(axes2(2), [Tb1(1)/3600 Tb1(end)/3600]);
        ylim(axes2(1), 'auto');
        if HS == 'y'
            y = current;
        else
            y = get(axes2(1), 'ylim');
            y = max(abs(y));
        end
        ylim(axes2(1), [-y y]);
        line(get(axes2(1), 'XLim'), [0 0], 'Color', [0 0 0], 'LineStyle', '--
');

        drawnow;
        % Make Echem plot look nice
        line1.LineWidth = 2;
        line1.Color = 'r';
        line2.LineWidth = 2;
        line2.Color = 'b';
        set(axes2, {'ycolor'}, {'r'; 'b'});
        set(axes2, 'FontSize', 20);
        xlabel(axes2(1), 'Time (h)');
        ylabel(axes2(1), 'Current (\muA)');
        ylabel(axes2(2), 'Potential (V)');
        set(axes2(1), 'YTick', [-y : y : y]);

        % Plot a vertical line depicting current Echem state
        hold(axes2(1), 'on')
        line([Tb1(i)/3600 Tb1(i)/3600], get(axes2(1), 'YLim'), 'Color', [1 0
0], 'LineWidth', 2);

        if strcmp(EPOS, 'side')
            camroll(axes3(1), -90)
            camroll(axes3(2), -90)
            set(axes3, 'OuterPosition', [0.15 0 0.1 1]);
            set(axes1, 'Position', [0 0 1 1]);
        end

    end

end
set(axes1, 'visible', 'on');
set(gcf, 'Color', 'w');
axes1.Color = 'w';
axes1.Box = 'on';
axes1.LineWidth = 2.0;
axes1.FontSize = 24.0;
axes1.GridAlpha = 0.2;

```



```

axes1.XColor = 'k';
axes1.YColor = 'k';
axes1.ZColor = 'k';

surface = findall(axes1, 'type', 'surface');
hgt2 = 50*ceil(max(max(surface(2).ZData))/50);
%low = hgt - 100;
hgt = max(max(surface(2).ZData));
low = min(min(surface(2).ZData));
axes1.ZLim = [0 hgt2];
daspect(axes1, [1 1 1]);
brighten(-0.5);
xstart = 250*ceil(axes1.XLim(1)/250);
xend = 250*floor(axes1.XLim(2)/250);
axes1.XTick = [xstart: 250: xend];
axes1.YTick = [xstart: 250: xend];

c1 = colorbar(axes1);
c1.Label.String = 'Z-Height (\mum)';
c1.FontSize = 24.0;
c1.FontWeight = 'bold';
c1.Color = 'k';
set(c1, 'ylim', [low hgt]);

if ~isempty(axes1.XTickLabel)
    axes1.XTickLabel(end) = strcat(axes1.XTickLabel(end), '\mum');
end
if ~isempty(axes1.YTickLabel)
    axes1.YTickLabel(end) = strcat(axes1.YTickLabel(end), '\mum');
end
if ~isempty(axes1.ZTickLabel)
    axes1.ZTickLabel(end) = strcat(axes1.ZTickLabel(end), '\mum');
end

if strcmp(EPOS, 'side')
    set(axes2, 'OuterPosition', [0.15 0 0.1 1]);
end

else
    surface = findobj(gcf, 'type', 'surface');
    x = surface(2).XData;
    y = surface(2).YData;
    z = surface(2).ZData;
    close(gcf);
    figure;
    for k = 1 : NI
        SAxes{k} = axes('Parent', gcf, 'OuterPosition', [XPOS(k) - .035,
YPOS(k) - .035, DIM + .07, DIM + .07]);
        surf(x, y, z, 'LineStyle', 'None');
        set(gca, 'Xlim', [ROIS{k}(1) (ROIS{k}(1) + ROIS{k}(3))]);
        set(gca, 'Ylim', [ROIS{k}(2) (ROIS{k}(2) + ROIS{k}(3))]);
        set(gca, 'view', [180 90]);
        daspect([1 1 1]);
        set(gca, 'visible', 'off');
    end
end

```

```

        set(gcf, 'color', 'w');
    end

    for p = 1 : size(SAxes, 2)
        set(SAxes{p}, 'view', v);
    end

    %Construct scale bars and text
    sf = 3; %scaling factor from x units to microns
    size = 150; %scalebar size in microns
    for p = 1 : NI
        axes(SAxes{p});
        xpos = [(ROIS{p}(1) + .05 * ROIS{p}(3)); (ROIS{p}(1) + .05 *
ROIS{p}(3) + size/sf)];
        ypos = [(ROIS{p}(2) + .95 * ROIS{p}(3)); (ROIS{p}(2) + .95 *
ROIS{p}(3))];
        if v(2) == 90
            zpos = [(max(max(z)) + 10); (max(max(z)) + 10)];
            hold on
            line(xpos, ypos, zpos, 'LineWidth', 6, 'Color', 'k');
            text(ROIS{p}(1) + .05 * ROIS{p}(3) + size/(2*sf), ROIS{p}(2) +
.85 * ROIS{p}(3), zpos(2), strcat(num2str(size), '\mum'), 'FontSize', 18,
'FontWeight', 'bold', 'Color', 'k', 'HorizontalAlignment', 'center');
            hold off
        end
    end
end

else
    cd(ParentPath);
    openfig(ParentNames{i});

    surfacel = findobj(gcf, 'type', 'surface');
    xpellet = {};
    ypellet = {};
    zpellet = {};

    for p = 1 : size(surfacel, 1)
        xpellet{p} = surfacel(p).XData;
        ypellet{p} = surfacel(p).YData;
        zpellet{p} = surfacel(p).ZData;
    end

    close(gcf);

    cd(LDPath);
    openfig(LDNames{i});
    surface2 = findobj(gcf, 'type', 'surface');
    xld = {};
    yld = {};
    zld = {};

    for p = 1 : size(surface2, 1)
        xld{p} = surface2(p).XData;
        yld{p} = surface2(p).YData;
        zld{p} = surface2(p).ZData;
    end
end

```

```

h = gcf;
figure
close(h);

if ROI ~= 'y'
if Trode == 'W'
    [Gmag, Gpos] = imgradient(zpellet{2}, 'Prewitt');
    zpellet{2}(abs(Gmag) > 30) = nan;
    surf(xpellet{2}, ypellet{2}, zpellet{2}, 'linestyle', 'none');
    ax1 = gca;

    set(ax1, 'XLim', xlm);
    set(ax1, 'YLim', ylm);
    set(ax1, 'ZLim', zlm);
    set(ax1, 'PlotBoxAspectRatio', aspct);

    ax2 = axes;
    hold on
    surf(xld{3}, yld{3}, zld{3}, 'linestyle', 'none');
    surf(xld{4}, yld{4}, zld{4}, 'linestyle', 'none');
    Link = linkprop([ax1, ax2], {'CameraUpVector', 'CameraPosition', ...
                                'CameraTarget', 'CameraViewAngle', ...
                                'XLim', 'YLim', 'ZLim', 'View', ...
                                'PlotBoxAspectRatio'});

    set(ax1, 'CameraTarget', camera1);
    set(ax1, 'CameraPosition', camera2);
    set(ax1, 'CameraViewAngle', camera3);
    set(ax1, 'CameraUpVector', camera4);

    colormap(ax1, 'gray');
    colormap(ax2, flipud('hot'));
    axes(ax1);
    cameratoolbar('ToggleSceneLight');
    axes(ax2);
    set(ax2, 'visible', 'off');

elseif Trode == 'C'
    [Gmag, Gpos] = imgradient(zpellet{1}, 'Prewitt');
    zpellet{1}(abs(Gmag) > 30) = nan;
    surf(xpellet{1}, ypellet{1}, zpellet{1}, 'linestyle', 'none');
    ax1 = gca;

    set(ax1, 'XLim', xlm);
    set(ax1, 'YLim', ylm);
    set(ax1, 'ZLim', zlm);
    set(ax1, 'PlotBoxAspectRatio', aspct);

    ax2 = axes;
    hold on
    surf(xld{1}, yld{1}, zld{1}, 'linestyle', 'none');
    surf(xld{2}, yld{2}, zld{2}, 'linestyle', 'none');
    Link = linkprop([ax1, ax2], {'CameraUpVector', 'CameraPosition', ...
                                'CameraTarget', 'CameraViewAngle', ...
                                'XLim', 'YLim', 'ZLim', 'View', ...

```

```

        'PlotBoxAspectRatio'}});

set(ax1, 'CameraTarget', camera1);
set(ax1, 'CameraPosition', camera2);
set(ax1, 'CameraViewAngle', camera3);
set(ax1, 'CameraUpVector', camera4);

colormap(ax1, 'gray');
colormap(ax2, flipud('hot'));
axes(ax1);
cameratoolbar('ToggleSceneLight');
axes(ax2);
set(ax2, 'visible', 'off');
end
else
    if Trode == 'W'
        [Gmag, Gpos] = imgradient(zpellet{2}, 'Prewitt');
        zpellet{2}(abs(Gmag) > 30) = nan;
        surf(xpellet{2}, ypellet{2}, zpellet{2}, 'linestyle', 'none');
        ax1 = gca;
        set(ax1, 'XLim', xlm);
        set(ax1, 'YLim', ylm);
        set(ax1, 'ZLim', zlm);

        ax2 = axes;
        hold on
        surf(xld{3}, yld{3}, zld{3}, 'linestyle', 'none');
        surf(xld{4}, yld{4}, zld{4}, 'linestyle', 'none');
        Link = linkprop([ax1, ax2], {'CameraUpVector',
'CameraPosition', ...,
                                'CameraTarget', 'CameraViewAngle', ...,
                                'XLim', 'YLim', 'ZLim', 'View', ...,
                                'PlotBoxAspectRatio'}});

        set(ax1, 'CameraTarget', camera1);
        set(ax1, 'CameraPosition', camera2);
        set(ax1, 'CameraViewAngle', camera3);
        set(ax1, 'CameraUpVector', camera4);

        colormap(ax1, 'gray');
        colormap(ax2, flipud('hot'));
        axes(ax1);
        cameratoolbar('ToggleSceneLight');
        axes(ax2);
        set(ax2, 'visible', 'off');

    elseif Trode == 'C'

        [Gmag, Gpos] = imgradient(zpellet{1}, 'Prewitt');
        zpellet{1}(abs(Gmag) > 30) = nan;
        surf(xpellet{1}, ypellet{1}, zpellet{1}, 'linestyle', 'none');
        ax1 = gca;
        set(ax1, 'XLim', xlm);
        set(ax1, 'YLim', ylm);
        set(ax1, 'ZLim', zlm);
        set(ax1, 'PlotBoxAspectRatio', aspct);
        ax2 = axes;

```

```

hold on
surf(xld{2}, yld{2}, zld{2}, 'linestyle', 'none');
surf(xld{2}, yld{2}, zld{2}, 'linestyle', 'none');
Link = linkprop([ax1, ax2], {'CameraUpVector',
'CameraPosition', ...,
                             'CameraTarget', 'CameraViewAngle', ...,
                             'XLim', 'YLim', 'ZLim', 'View', ...,
                             'PlotBoxAspectRatio'}});

set(ax1, 'CameraTarget', camera1);
set(ax1, 'CameraPosition', camera2);
set(ax1, 'CameraViewAngle', camera3);
set(ax1, 'CameraUpVector', camera4);

colormap(ax1, 'gray');
colormap(ax2, flipud('hot'));
axes(ax1);
cameratoolbar('ToggleSceneLight');
axes(ax2);
set(ax2, 'visible', 'off');

end
end
end

if ImProc == 'y'
ax = gca;
set(ax, 'visible', 'on');
set(gcf, 'Color', 'w');
ax.Color = 'w';
ax.Box = 'on';
ax.LineWidth = 2.0;
ax.FontSize = 24.0;
ax.GridAlpha = 0.2;
ax.XColor = 'k';
ax.YColor = 'k';
ax.ZColor = 'k';

%{
xwidth = ax.XLim(2) - ax.XLim(1);
xstart = ax.XLim(1) - xwidth;
xend = ax.XLim(2) + xwidth;
ax.XLim = [xstart xend];
ax.YLim = [xstart xend];
%}

ax.ZLim = [0 hgt2];
daspect([1 1 1]);
brighten(-0.5);

if ctour == 'y'
    z = surface(2).ZData;
    hold on
    contour3(z, 'Color', 'k');
    hold off
end
end

```

```

if ImProcROI == 'y'
    % Hard set to look at working electrode
    z = surface(2).ZData;
    x = surface(2).XData;
    y = surface(2).YData;
    tmp = gcf;

    z = z(Pos(2) : (Pos(2) + Pos(3)), Pos(1) : (Pos(1) + Pos(3)));
    x = [Pos(2) : (Pos(2) + Pos(3))];
    y = [Pos(1) : (Pos(1) + Pos(3))];
    x = x - Pos(2);
    y = y - Pos(1);

    figure
    surf(x, y, z, 'LineStyle', 'None');
    daspect([1 1 1]);
    zlim([low hgt2]);
    caxis([low hgt]);
    view(3);
    set(gca, 'xlim', [min(x) max(x)]);
    set(gca, 'ylim', [min(y) max(y)]);

    set(gca, 'visible', 'on');
    if ctour == 'y'
        hold on
        contour3(z, 20, 'Color', 'k');
    end
    set(gcf, 'Color', 'w');
    ax2 = gca;
    ax2.Color = 'w';
    ax2.Box = 'on';
    ax2.LineWidth = 2.0;
    ax2.GridAlpha = 0.2;
    ax2.XColor = 'k';
    ax2.YColor = 'k';
    ax2.ZColor = 'k';

end
% Note: setting the x and y axis ticks assumes no data compression
% Working on ensuring it can work with compressed data
%Put Proper Scaling Code Here

lines = findall(gcf, 'type', 'line');
if ~isempty(lines)
    for itemp = 1 : length(lines)
        delete(lines(itemp));
    end
end

xstart = 250*ceil(ax.XLim(1)/250);
xend = 250*floor(ax.XLim(2)/250);
ax.XTick = [xstart: 250: xend];
ax.YTick = [xstart: 250: xend];

if ImProcROI ~= 'y'
    %{

```

```

x1 = [0, 1000, nan, 0, 0, nan, 0, 0];
y1 = [0, 0, nan, 0, 1000, nan, 0, 0];
z1 = [0, 0, nan, 0, 0, nan, 0, 0];
%}

x1 = [250, 1250, nan, 250, 250, nan, 0, 0];
y1 = [250, 250, nan, 250, 1250, nan, 0, 0];
z1 = [0, 0, nan, 0, 0, nan, 0, 0];

h1 = line(x1, y1, z1);
h1.Color = 'w';
h1.LineWidth = 5;
%{
ht1 = text(0,500,0,[num2str(1), ' ', 'mm']);
ht2 = text(500,0,0,[num2str(1), ' ', 'mm']);
%}

ht1 = text(250,625,0,[num2str(1), ' ', 'mm']);
ht2 = text(625,250,0,[num2str(1), ' ', 'mm']);
ht1.FontSize = 24.0;
ht2.FontSize = 24.0;
ht1.FontWeight = 'bold';
ht2.FontWeight = 'bold';
ht1.Color = 'w';
ht2.Color = 'w';
ht1.VerticalAlignment = 'top';
ht2.VerticalAlignment = 'top';
ht1.HorizontalAlignment = 'right';
ht2.HorizontalAlignment = 'left';
else
    set(gca, 'FontSize', 24);
    brighten(-0.5);
end

c1 = colorbar;
c1.Label.String = 'Z-Height (\mum)';
c1.FontSize = 24.0;
c1.FontWeight = 'bold';
c1.Color = 'k';
set(c1, 'ylim', [low hgt]);

set(gca, 'CameraTarget', camera1);
set(gca, 'CameraPosition', camera2);
set(gca, 'CameraViewAngle', camera3);
set(gca, 'CameraUpVector', camera4);

if ImProcROI == 'y'
    close(tmp);
    ax = gca;
    drawnow
    if ~isempty(ax.XTickLabel)
        ax.XTickLabel(end) = strcat(ax.XTickLabel(end), '\mum');
    end
    if ~isempty(ax.YTickLabel)
        ax.YTickLabel(end) = strcat(ax.YTickLabel(end), '\mum');
    end
end

```

```

        end
        if ~isempty(ax.ZTickLabel)
            ax.ZTickLabel(end) = strcat(ax.ZTickLabel(end), '\mum');
        end
    else

        if ~isempty(ax.XTickLabel)
            ax.XTickLabel(end) = strcat(ax.XTickLabel(end), '\mum');
        end
        if ~isempty(ax.YTickLabel)
            ax.YTickLabel(end) = strcat(ax.YTickLabel(end), '\mum');
        end
        if ~isempty(ax.ZTickLabel)
            ax.ZTickLabel(end) = strcat(ax.ZTickLabel(end), '\mum');
        end

        end
        end
        drawnow;
        h = gcf;
        if i == 1
            Frames = getframe(h);
        else
            temp = getframe(h);
            Frames = [Frames temp];
        end
        close
    end

%% Write movie as .MP4
movie(gcf, Frames, 10, 1);
profile = 'MPEG-4';
cd ..
writerObj = VideoWriter('Surface Movie', profile);
set(writerObj, 'FrameRate', 3)
open(writerObj);

% Write out all the frames.
for i = 1 : NumberOfFrames
    writeVideo(writerObj, Frames(i));
end

close(writerObj);
cd(A);

clearvars -except Frames Path names I t V Tbl ROIS Pos x y z
case 5
%% Case 5 Program Description
% Program is written to accept a small number of 3D reconstructions
% obtained from Case 3 and plot them in a way that is visually
% understandable. Time dependent data at present is stacked vertically with
% time points placed adjacent to the corresponding image. Electrochemical
% data is also plotted alongside 3D reconstructions with lines indicating
% the time points corresponding to images. The actual placement of plots is
% somewhat persnickity so some user intervention after image generation
% will likely be necessary.

```



```

disp('Generating Single Image');
A = pwd;

% Set EC to 'n' to omit electrochemistry
EC = 'y';
ECPos = 'RHS';

HS = 'y';
current = 10;

ROI = 'y';

disp('Kindly Select Void Analysis Table');
[VA, VADir] = uigetfile('*..*', 'All Files (*.*)');
VA = fullfile(VADir, VA);
Tb = readtable(VA, 'Delimiter', '\t');
Tbl = table2array(Tb(:, 2));
Nms = table2array(Tb(:, 1));

if EC == 'y'
    disp('Kindly Select Electrochemical Data');
    [EchemName, EchemDir] = uigetfile('*..*', 'All Files (*.*)',
    'MultiSelect', 'on');

    for i = 1 : length(EchemName)
        EchemName(i) = fullfile(EchemDir, EchemName(i));
        C = importdata(EchemName{i}, ',');
        if ~isempty(strfind(EchemName{i}, 'OCP'))
            t1 = C.data(:, 1);
            V1 = C.data(:, 2);
            I1 = zeros(length(t1), 1);
        end
        if ~isempty(strfind(EchemName{i}, 'CVnarrow'))
            V2 = C.data(:, 1);
            I2 = C.data(:, 2);
            t2 = [1 : 5 : length(V2)*5] + t1(end);
        end
        if ~isempty(strfind(EchemName{i}, 'CVwide'))
            V3 = C.data(:, 1);
            I3 = C.data(:, 2);
            t3 = [1 : 5 : length(V3)*5] + t2(end);
        end
    end
    t = [t1; t2'; t3'];
    t = t/3600;
    V = [V1; V2; V3];
    I = [I1; I2; I3];
    I = I*1000000;
end

%Select figures to open
[Figname, Figdir] = uigetfile('*..*', 'All Files (*.*)', 'MultiSelect',
'on');
N = length(Figname);
cd(Figdir);
if ROI == 'y'

```

```

    openfig(Figname{end});
    tempsurf = findall(gca, 'type', 'surface');
    xtemp = tempsurf(2).XData;
    ytemp = tempsurf(2).YData;
    ztemp = tempsurf(2).ZData;
    sub = max(max(ztemp)) + 5;
    ztemp = ztemp - sub;
    h = gcf;
    figure
    close(h);
    surf(xtemp, ytemp, ztemp, 'LineStyle', 'none');
    view(2);
    daspect([1 1 1]);
    h = imellipse;
    pause
    Pos = getPosition(h);
    close all
    clearvars tempsurf xtemp ytemp ztemp sub
end

openfig(Figname{1});
box on
fig1 = gcf;
ax{1} = gca;
surfaces{1} = findall(ax{1}, 'type', 'surface');
L = findall(ax{1}, 'type', 'line');
delete(L);

x = surfaces{1}(1).XData;
y = surfaces{1}(1).YData;
z{1,1} = surfaces{1}(1).ZData;
z{1,2} = surfaces{1}(2).ZData;

if ROI == 'y'
    close(fig1);
    z2{1} = z{1, 2};
    z2{1} = z2{1}(Pos(2) : (Pos(2) + Pos(3)), Pos(1) : (Pos(1) + Pos(3)));
    x = [Pos(2) : (Pos(2) + Pos(3))];
    y = [Pos(1) : (Pos(1) + Pos(3))];
    x = x - Pos(2);
    y = y - Pos(1);

    surf(x, y, z2{1}, 'LineStyle', 'none');
    fig1 = gcf;
    ax{1} = gca;
    box('on');
    daspect([1 1 2]);
    ax{1}.XLim = [0 Pos(3)];
    ax{1}.YLim = [0 Pos(3)];
    ax{1}.ZLim = [100 180];
    ax{1}.XTickMode = 'auto';
    ax{1}.YTickMode = 'auto';
end

view(-45, 20);

```

```

sep = 900;
if ROI ~= 'y'
    zmax = (max(max(z{1, 2})));
    zmin = (-sep*(N-1));
    ax{1}.ZLim = [zmin zmax];
end

ax{1}.FontSize = 24;
ax{1}.LineWidth = 2.0;
ax{1}.GridAlpha = 0.2;
ax{1}.ZTickLabel = [];
ax{1}.TickLength = [0 0];

if ~isempty(ax{1}.XTickLabel)
    ax{1}.XTickLabel(end) = strcat(ax{1}.XTickLabel(end), '\mum');
end
if ~isempty(ax{1}.YTickLabel)
    ax{1}.YTickLabel(end) = strcat(ax{1}.YTickLabel(end), '\mum');
end
if ~isempty(ax{1}.ZTickLabel)
    ax{1}.ZTickLabel(end) = strcat(ax{1}.ZTickLabel(end), '\mum');
end

op = ax{1}.OuterPosition;
ti = ax{1}.TightInset;
bottom = op(2) + ti(2);
h = op(4) - ti(2) - ti(4);

if ROI ~= 'y'
    ax{1}.Position = [0 bottom 1 (h-.01)];
    daspect(ax{1}, [1 1 1]);
else
    ax{1}.Position = [0 (bottom + 1.05 - 1/N) 1 (h/N-0.05)];
end
for i = 2 : N
    openfig(Figname{i});
    temp = gca;
    surfaces{i} = findall(temp, 'type', 'surface');
    z{i, 1} = surfaces{i}(1).ZData;
    z{i, 2} = surfaces{i}(2).ZData;
    x = surfaces{i}(1).XData;
    y = surfaces{i}(1).YData;
    close(gcf);
    if ROI == 'y'
        z2{i} = z{i, 2};
        z2{i} = z2{i}(Pos(2) : (Pos(2) + Pos(3)), Pos(1) : (Pos(1) + Pos(3)));
        x = [Pos(2) : (Pos(2) + Pos(3))];
        y = [Pos(1) : (Pos(1) + Pos(3))];
        x = x - Pos(2);
        y = y - Pos(1);

    end

    ax{i} = axes('Parent', fig1);
    if ROI == 'y'
        surf(x, y, z2{i}, 'LineStyle', 'none');
        ax{i}.Position = [0 (bottom + 1.05 - i/N) 1 (h/N-0.05)];
    end
end

```

```

daspect(ax{i}, [1 1 2]);
ax{i}.View = ax{1}.View;
ax{i}.XLim = ax{1}.XLim;
ax{i}.YLim = ax{1}.YLim;
ax{i}.ZLim = ax{1}.ZLim;

ax{i}.FontSize = 24;
ax{i}.LineWidth = 2.0;
ax{i}.GridAlpha = 0.2;
ax{i}.ZTickLabel = [];
ax{i}.TickLength = [0 0];
ax{i}.CLim = ax{1}.CLim;
box('on');

if ~isempty(ax{i}.XTickLabel)
    ax{i}.XTickLabel(end) = strcat(ax{i}.XTickLabel(end), '\mum');
end
if ~isempty(ax{i}.YTickLabel)
    ax{i}.YTickLabel(end) = strcat(ax{i}.YTickLabel(end), '\mum');
end
if ~isempty(ax{i}.ZTickLabel)
    ax{i}.ZTickLabel(end) = strcat(ax{i}.ZTickLabel(end), '\mum');
end

else
    surf(ax{i}, x, y, z{i, 1}, 'LineStyle', 'none');
    hold on
    surf(ax{i}, x, y, z{i, 2}, 'LineStyle', 'none');
    hold off
    daspect(ax{i}, [1 1 1]);
    ax{i}.View = ax{1}.View;
    ax{i}.ZLim = ax{i-1}.ZLim + sep;
    ax{i}.XLim = ax{1}.XLim;
    ax{i}.YLim = ax{1}.YLim;
    ax{i}.Position = ax{1}.Position;
    ax{i}.Visible = 'off';
end

end

brighten(-0.5);
c = colorbar('east');
c.Label.String = 'Z-Height \mum';
c.FontSize = 24.0;
c.FontWeight = 'bold';
c.TickDirection = 'out';
c.LineWidth = 2;
c.YAxisLocation = 'right';
cpos = c.Position;
cpo(1) = 0.765;
c.Position = cpo;
if ROI == 'y'
    c.Position = [0.74 0.025 0.015 .95];
end

%Create Axis for Echem and plot vertically next to tomo stack
ECax = axes('Parent', fig1);
[ECax, line1, line2] = plotyy(t, I, t, V);
xlim(ECax(1), [Tb1(1)/3600 Tb1(end)/3600]);

```

```

xlim(ECax(2), [Tbl(1)/3600 Tbl(end)/3600]);
ylim(ECax(1), 'auto');

if HS == 'y'
    ymax = current;
else
    ymax = get(ECax(1), 'ylim');
    ymax = max(abs(ymax));
end
ylim(ECax(1), [-ymax ymax]);
line(get(ECax(1), 'XLim'), [0 0], 'Color', [0 0 0], 'LineStyle', '--');

% Make Echem plot look nice
ECax(1).LineWidth = 2.0;
ECax(2).LineWidth = 2.0;
line1.LineWidth = 2;
line1.Color = 'r';
line2.LineWidth = 2;
line2.Color = 'b';
set(ECax, {'ycolor'}, {'r'; 'b'});
set(ECax, 'FontSize', 24);
xlabel(ECax(1), 'Time (h)');
ylabel(ECax(1), 'Current (\muA)');
ylabel(ECax(2), 'Potential (V)');
set(ECax(1), 'YTick', [-y : y : y]);

camroll(ECax(1), -90)
camroll(ECax(2), -90)
if ROI ~= 'y'
    set(ECax, 'OuterPosition', [0.1 0 0.125 .975]);
else
    set(ECax, 'OuterPosition', [0.1 bottom 0.125 (h-0.015)]);
end
ECax(1).YTickMode = 'auto';
ECax(2).YTickMode = 'auto';

for i = 1 : length(Figname)
    axes(ECax(2));
    [pth, SearchName, ext] = fileparts(Figname{i});
    tpnt = Tbl(~cellfun('isempty', (strfind(Nms, SearchName))));
    hold(ECax(1), 'on')
    line([tpnt/3600 tpnt/3600], get(ECax(1), 'YLim'), 'Color', [1 0 0],
'LineWidth', 2);
    hold(ECax(1), 'off')
    axes(ax{i});

    if ROI ~= 'y'
        Txt{i} = text(0, max(y), 150, strcat('T = ', num2str(round(tpnt/3600,
1)),{' ' '}', 'h'), 'FontSize', 24.0, 'FontWeight', 'bold',
'HorizontalAlignment', 'right');
    else
        Txt{i} = text(0, max(y) + max(y)/5, 150, strcat('T = ',
num2str(round(tpnt/3600, 1)),{' ' '}', 'h'), 'FontSize', 24.0, 'FontWeight',
'bold', 'HorizontalAlignment', 'right');
    end
end
set(fig1, 'Color', 'w');

```

```
if ROI == 'y'  
    drawnow;  
    set(ECax, 'OuterPosition', [0.2 bottom 0.1 (h-0.015)]);  
end  
cd(A);  
  
end
```

Appendix G: Charge Density Visualization from VASP CHGCAR File (MATLAB):

```
% Initialize default settings
clear all; close all
Atom_List = matfile('Atom_List.mat').Atom_List;
HKL{1} = [1 0 0];
d{1} = 0.5;
Min_D = 20; Max_D = 1500;
Subdiv = 2;
r = 0.25;

% Read data from CHGCAR file
[Filename, filedir] = uigetfile('*..*', 'All Files (*.*)');
if Filename == 0; return; end
[filepath, name, ext] = fileparts(Filename);
Filename = fullfile(filedir, Filename);
[Chg, lat, atom_positions, atom_type, atom_num] = importCHGCAR(Filename);
Cell_Vol = norm(lat(:, 1))*norm(lat(:, 2))*norm(lat(:, 3));
n_cells = size(Chg, 1)*size(Chg, 2)*size(Chg, 3);
vol_cell = Cell_Vol/n_cells;

% Define general image variables
Atoms = {};
Unit_Cell = {};
HKL_Plane = {};
Chg_Cut = {}; Chg_Cut{1} = [];
Chg_Map = {};
flamp = {};

% Plot data using default settings
%To properly display color maps charge cuts must be displayed before atoms
f1 = figure('Color', 'w'); ax1=axes; axis equal; camlight headlight
ax1.Visible='off';
Unit_Cell = draw_unit_cell(f1, ax1, Unit_Cell, lat); view(HKL{1}*lat);
[leg] = draw_legend(f1, ax1, lat);
[Chg_Cut, Chg_Plane] = Build_Charge_Density_Plane(f1, ax1, Chg, HKL{1}, d{1},
lat, Chg_Cut, 1);
[Atoms, flamp] = draw_atoms(f1, ax1, Atoms, flamp, atom_positions, atom_type,
Atom_List, lat, atom_num, r);
[HKL_Plane] = Draw_HKL_Plane(f1, ax1, HKL_Plane, HKL{1}, d{1}, lat, 1);
Plot_GUI(f1, ax1, HKL, d, lat, Chg, HKL_Plane, Chg_Cut, Chg_Map, Atoms,
flamp, ...
Unit_Cell, atom_positions, atom_type, Atom_List, atom_num,
r, vol_cell);

function [CHG, lat, atom_positions, atom_type, atomNum] = importCHGCAR(fname)
lat = zeros(3,3);
atom_positions = [];
count = 0;
fid=fopen(fname);
N = 10^16;
while 1
tline = fgetl(fid);
if ~ischar(tline), break, end
count = count + 1;
% Lattice vectors
if count >=3 && count <= 5
```

```

        lat(count-2,:) = sscanf(tline, '%f %f %f');
    end

    if count == 6
        atom_type = sscanf(tline, '%c');
    end

    if count == 7
        atomNum = sscanf(tline, '%f %f %f');
        N = sum(atomNum);
    end

    if (count >= 9) && (count <= N+8)
        atom_positions = [atom_positions; sscanf(tline, '%f %f %f')'];
    end

    if count == N+10;
        CHGdim = sscanf(tline, '%f %f %f');
        break
    end
end

% Now get all CHG data
data = textscan(fid, '%f');
Nentry = CHGdim(1)*CHGdim(2)*CHGdim(3);
CHG = reshape(data{1}(1:Nentry), CHGdim');

fclose(fid);
end

function [Unit_Cell] = draw_unit_cell(f1, ax1, Unit_Cell, lat)
figure(f1); axes(ax1); hold on
pnts = [zeros(1, 3); eye(3, 3); mod(ones(3, 3)+eye(3, 3), 2); ones(1,
3)];
test_edge = [0, 0; 0, 1; 1, 0; 1, 1];
for i = 1 : 4
    for j = 1:3
        test = repmat(test_edge(i, :), 2, 1);
        if j == 1; test = [[0;1] test]; elseif j == 2; test = [test(:, 1)
[0; 1] test(:, 2)];
        else test = [test [0;1]]; end;
        test = test*lat;
        Unit_Cell{end+1} = plot3(test(:, 1), test(:, 2), test(:, 3),
'Color', 'k', 'LineWidth', 1.5, 'LineStyle', '--');
    end
end
end

function [leg] = draw_legend(f1, ax1, lat)
figure(f1); axes(ax1); hold on; leg={};
start = repmat(2*[-1 -1 -1]*lat/norm([-1 -1 -1]*lat), 3, 1);
dir = eye(3)*lat;
for i = 1 : 3; dir(:, i)=2*dir(:, i)/norm(dir(:, i)); end;
leg{end+1} = quiver3(start(:, 1), start(:, 2), start(:, 3), dir(:, 1),...
dir(:, 2), dir(:, 3), 'Color', 'k', 'LineWidth', 3);

dir = start+dir;

```



```

    leg{end+1} = text(dir(1,1), dir(1, 2), dir(1, 3), 'x', 'FontName',
'Arial', 'FontSize', 14);
    leg{end+1} = text(dir(2,1), dir(2, 2), dir(2, 3), 'y', 'FontName',
'Arial', 'FontSize', 14);
    leg{end+1} = text(dir(3,1), dir(3, 2), dir(3, 3), 'z', 'FontName',
'Arial', 'FontSize', 14);
end

function [HKL_Plane] = Draw_HKL_Plane(f1, ax1, HKL_Plane, HKL, d, lat, dex)
figure(f1); axes(ax1); hold on
test_edge = [0, 0; 0, 1; 1, 0; 1, 1];
Edge_Pnts = [];
temp = {};
for i = 1:4
    val = [];
    for j = 1:3
        mult = ones(1, 3);
        mult(j) = 0;
        mult2 = mult.*HKL;
        mult2(j) = [];
        if HKL(j)~=0
            val = -1*dot(mult2, test_edge(i, :)-d*mult2)/HKL(j);
            nw_pnt = test_edge(i, :)-d*mult2;
            nw_pnt = [nw_pnt(1:j-1) val nw_pnt(j:end);]+d*HKL;
            idx=sum(nw_pnt<0|nw_pnt>1);
            if idx==0
                Edge_Pnts = [Edge_Pnts; nw_pnt];
            end
        end
    end
end
Edge_Pnts = unique(Edge_Pnts,'rows');

i = 1;
if nnz(HKL)==1; val=2; else; val=1; end
while i <= size(Edge_Pnts, 1)
    j = i+1;
    while j < size(Edge_Pnts, 1)+1
        test = Edge_Pnts(i, :);
        if nnz(HKL)==1; n_zed = sum((Edge_Pnts(i, :)-Edge_Pnts(j,
:))==0);
        else; n_zed = sum((Edge_Pnts(i, :)+Edge_Pnts(j, :))==0); end;
        n_ones = sum((Edge_Pnts(i, :)+Edge_Pnts(j, :))==2);
        if n_zed>=val|n_ones>=val
            test = [test; Edge_Pnts(j, :)];
            test=test*lat;
            temp{end+1} = plot3(test(:, 1), test(:, 2), test(:, 3),
'LineWidth', 2.0, 'Color', 'r'); %[0.4941 0.1843 0.5569]);
        end
        j = j+1;
    end
    i = i+1;
end

HKL_Plane{dex} = temp;
end

```

```

function [Atoms, flamp] = draw_atoms(f1, ax1, Atoms, flamp, atom_positions,
atom_type, Atom_List, lat, atom_num, r)
    figure(f1); axes(ax1); hold on
    atom_positions = atom_positions*lat;
    atom_type = strsplit(atom_type);
    atom_type = atom_type(~cellfun('isempty',atom_type));
    atom_index = {};
    for i = 1:size(atom_type, 2)
        atom_index = [atom_index; repmat({atom_type{i}}, atom_num(i), 1)];
    end

    strt = 1;
    nd = atom_num(1);

    for i = 1 : size(atom_positions, 1)
        Clor = Atom_List{strcmp({Atom_List{:, 1}}, atom_index{i}), 2};
        [x, y, z] = sphere(15);
        x = r*x+atom_positions(i, 1);
        y = r*y+atom_positions(i, 2);
        z = r*z+atom_positions(i, 3);
        Atoms{end+1} = surf(x, y, z, 'FaceColor', Clor, 'EdgeColor', 'none',
'FaceLighting', 'gouraud', 'SpecularStrength', 0.5);
    end

    if size(flamp, 2)>0; delete(flamp{1}); delete(flamp{2}); end;
    flamp{1} = camlight('left'); flamp{2} = camlight('right');
end

function [coords, Pnt_Dens, Chg_Map]=draw_density_map(f1, ax1, Chg_Map,
Subdiv, Chg, Min_D, Max_D, lat, sHKL, sd);
    figure(f1); axes(ax1); hold on
    map_limits = round(([1 1 1]-sd*sHKL).*size(Chg));
    x = (1:Subdiv:map_limits(1))';
    y = (1:Subdiv:map_limits(2))';
    z = (1:Subdiv:map_limits(3))';
    coords = zeros(length(x)*length(y)*length(z), 3);
    Pnt_Dens = zeros(length(x)*length(y)*length(z), 1);
    dex = 1;
    for i = 1:length(x)
        for j = 1:length(y)
            for k = 1:length(z)
                coords(dex, :) = [x(i), y(j), z(k)];
                Pnt_Dens(dex) = Chg(x(i), y(j), z(k));
                dex=dex+1;
            end
        end
    end
end

[dex, ~] = find(Pnt_Dens<Min_D);
coords(dex, :) = []; Pnt_Dens(dex) = [];
[dex, ~] = find(Pnt_Dens>Max_D);
coords(dex, :) = []; Pnt_Dens(dex) = [];
coords = coords-1;
coords(:, 1) = coords(:, 1)/size(Chg, 1);
coords(:, 2) = coords(:, 2)/size(Chg, 2);
coords(:, 3) = coords(:, 3)/size(Chg, 3);

```

```

coords = coords*lat;
Ctemp = ceil(Pnt_Dens-min(Pnt_Dens))+1;
jetcustom = jet(max(Ctemp));
Chg_Map{end+1} = scatter3(coords(:, 1), coords(:, 2), coords(:, 3), [],
jetcustom(Ctemp, :));
end

function [Chg_Cut, Chg_Plane] = Build_Charge_Density_Plane(f1, ax1, Chg, HKL,
d, lat, Chg_Cut, dex)
figure(f1); axes(ax1); hold on
cart_cell = {}; sz = size(Chg);
cart_cell{1} = (1/sz(1):1/sz(1):1)'; cart_cell{2} = (1/sz(2):1/sz(2):1)';
cart_cell{3} = (1/sz(3):1/sz(3):1)';

v = [];
[~, z_positions] = find(HKL==0);
% Handle [1 0 0], [0 1 0], and [0 0 1]
if size(z_positions, 2)==2
    for i = 1:size(z_positions, 2)
        temp = zeros(1, 3); temp(z_positions(i))=1;
        v = [v;temp];
    end
    v1 = cart_cell{z_positions(1)}*v(1, :);
    v2 = cart_cell{z_positions(2)}*v(2, :);

    dist_1 = linspace(0, 1, size(v1, 1))'*v(1, :);
    dist_2 = linspace(0, 1, size(v2, 1))'*v(2, :);
    % Handle [1 1 0], [0 1 1], and [1 0 1]
elseif size(z_positions, 2)==1
    temp = zeros(1, 3); temp(z_positions)=1;
    v = [v;temp]; v = [v; cross(v(1, :), HKL)];
    v(1, :) = v(1, :)/norm(v(1, :)); v(2, :) = v(2, :)/norm(v(2, :));
    v1 = cart_cell{z_positions}*v(1, :);
    temp = sz.*HKL; [~, pos] = max(temp);
    v2 = cart_cell{pos}*v(2, :)*norm(HKL);
    v2 = v2 - (HKL~=0).*[mean(v2(:, 1)) mean(v2(:, 2)) mean(v2(:, 3))];
    if any(v2(:, 3)<0); v2 = -1*v2; disp('invert'); end;

    dist_1 = linspace(0, 1, size(v1, 1))'*v(1, :);
    dist_2 = linspace(0, 1, size(v2, 1))'*v(2, :)*norm(HKL);
    dist_2 = dist_2 - (HKL~=0).*[mean(dist_2(:, 1)) mean(dist_2(:, 2))
mean(dist_2(:, 3))];

%Handle general [a b c]
else
    L1 = norm(HKL.*[1 1 0]); L2 = norm(HKL);
    v = [v; cross(HKL, HKL.*[1 1 0])]; v = [v; cross(HKL, v(1, :))];
    v(1, :) = v(1, :)/norm(v(1, :)); v(2, :) = v(2, :)/norm(v(2, :));
    [~, pos] = max(sz);
    v1 = cart_cell{pos}*v(1, :)*L1; v1 = v1-[mean(v1(:, 1)) mean(v1(:,
2)) 0];
    v2 = cart_cell{pos}*v(2, :)*L2; v2 = v2-[mean(v2(:, 1)) mean(v2(:,
2)) mean(v2(:, 3))];

    dist_1 = linspace(0, 1, size(v1, 1))'*v(1, :)*L1;
    dist_1 = dist_1 - [mean(dist_1(:, 1)) mean(dist_1(:, 2)) 0];
    dist_2 = linspace(0, 1, size(v2, 1))'*v(2, :)*L2;

```

```

        dist_2 = dist_2 - [mean(dist_2(:, 1)) mean(dist_2(:, 2))
mean(dist_2(:, 3))];
    end
    %v1 = v1+d*HKL; v2 = v2+d*HKL;

    pnts = zeros(size(v1, 1), size(v2, 1), 3);
    Chg_Plane = zeros(size(v1, 1), size(v2, 1), 1);
    for i = 1:size(v1, 1)
        for j = 1:size(v2, 1)
            xyz = uint16((v1(i, :)+v2(j, :)+d*HKL).*sz);
            if (xyz>[0 0 0])&(xyz<=sz)
                Chg_Plane(i, j) = Chg(xyz(1), xyz(2), xyz(3));
            end
            pnts(i, j, :) = (dist_1(i, :)+dist_2(j, :)+d*HKL)*lat;
        end
    end
    if size(z_positions, 2)<2
        K = (1/9)*ones(3); Chg_Plane = conv2(Chg_Plane, K, 'same');
    end
    Chg_Plane(Chg_Plane==0)=nan;
    Chg_Cut{1, dex} = pcolor(Chg_Plane);
    Chg_Cut{1, dex}.XData = pnts(:, :, 1);
    Chg_Cut{1, dex}.YData = pnts(:, :, 2);
    Chg_Cut{1, dex}.ZData = pnts(:, :, 3);
    colormap jet; shading interp;
    Chg_Cut{1, dex}.AmbientStrength=1;
    Chg_Cut{1, dex}.SpecularStrength=0;
end

function Plot_GUI(f1, ax1, HKL, d, lat, Chg, HKL_Plane, Chg_Cut, Chg_Map,
Atoms, flamp, ...
    Unit_Cell, atom_positions, atom_type, Atom_List, atom_num,
r, vol_cell)
    figure(f1); axes(ax1); hold on
    fcontrol = figure('Position', [50 50 500 500], 'Name', 'Control Panel',
'numbertitle', 'off');
    uicontrol('Style', 'text', 'String', 'Charge Plane Parameters:',
'FontName', ...
        'Arial', 'FontSize', 14, 'Position', [20, 470, 260, 25]);
    uicontrol('Style', 'text', 'String', 'H', 'FontName', ...
        'Arial', 'FontSize', 12, 'Position', [40, 450, 20, 20]);
    uicontrol('Style', 'text', 'String', 'K', 'FontName', ...
        'Arial', 'FontSize', 12, 'Position', [140, 450, 20, 20]);
    uicontrol('Style', 'text', 'String', 'L', 'FontName', ...
        'Arial', 'FontSize', 12, 'Position', [240, 450, 20, 20]);
    HKL_Entries = {};
    HKL_Entries{1} = uicontrol('Style', 'edit', 'String', 1, ...
        'Position', [20, 430, 60, 20], 'FontName',
'Arial', ...
        'FontSize', 12, 'Callback', @change_HKL);
    HKL_Entries{2} = uicontrol('Style', 'edit', 'String', 0, ...
        'Position', [120, 430, 60, 20], 'FontName',
'Arial', ...
        'FontSize', 12, 'Callback', @change_HKL);
    HKL_Entries{3} = uicontrol('Style', 'edit', 'String', 0, ...
        'Position', [220, 430, 60, 20], 'FontName',
'Arial', ...

```

```

        'FontSize', 12, 'Callback', @change_HKL);
uicontrol('Style', 'text', 'String', 'd along HKL', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[20, 400, 180, 20]);

s1 = uicontrol('Style','slider', 'Value', 0.5, 'Position',[20, 380, 190,
20], ...
    'Callback', @s1_function);
t1 = uicontrol('Style', 'edit', 'String', num2str(get(s1, 'Value')), ...
    'Position',[220, 380, 60, 20], 'FontName', 'Arial', ...
    'FontSize', 12, 'Callback', @t1_function);

uicontrol('Style', 'text', 'String', 'Plane Color Max', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[20, 350, 180, 20]);

cstart{1} = caxis(ax1);
cmax = cstart{1}(2);
cmin = cstart{1}(1);

c1 = uicontrol('Style','slider', 'Min', cmin, 'Max', cmax, 'Value', ...
    cmax, 'Position',[20, 330, 190, 20], ...
    'Callback', @c1_func);
c_max = uicontrol('Style', 'edit', 'String', num2str(get(c1, 'Value')),
...
    'Position',[220, 330, 60, 20], 'FontName', 'Arial', ...
    'FontSize', 12, 'Callback', @cmax_func);

uicontrol('Style', 'text', 'String', 'Plane Color Min', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[20, 300, 180, 20]);

c2 = uicontrol('Style','slider', 'Min', cmin, 'Max', cmax, 'Value', ...
    cmin, 'Position',[20, 280, 190, 20], ...
    'Callback', @c2_func);
c_min = uicontrol('Style', 'edit', 'String', num2str(get(c2, 'Value')),
...
    'Position',[220, 280, 60, 20], 'FontName', 'Arial', ...
    'FontSize', 12, 'Callback', @cmin_func);

uicontrol('Style', 'text', 'String', 'Charge Map Parameters:',
'FontName', ...
    'Arial', 'FontSize', 14, 'Position',[20, 240, 260, 25]);
uicontrol('Style', 'text', 'String', 'H', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[40, 220, 20, 20]);
uicontrol('Style', 'text', 'String', 'K', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[140, 220, 20, 20]);
uicontrol('Style', 'text', 'String', 'L', 'FontName', ...
    'Arial', 'FontSize', 12, 'Position',[240, 220, 20, 20]);

Map_Entries = {};
Map_Entries{1} = uicontrol('Style', 'edit', 'String', 1, ...
    'Position',[20, 200, 60, 20], 'FontName',
'Arial', ...
    'FontSize', 12);
Map_Entries{2} = uicontrol('Style', 'edit', 'String', 0, ...
    'Position',[120, 200, 60, 20], 'FontName',
'Arial', ...
    'FontSize', 12);
Map_Entries{3} = uicontrol('Style', 'edit', 'String', 0, ...

```

```

        'Position',[220, 200, 60, 20], 'FontName',
'Arial', ...
        'FontSize', 12);
    uicontrol('Style','text','String','d along HKL','FontName',...
        'Arial','FontSize',12,'Position',[20, 170, 180, 20]);
    s2 = uicontrol('Style','slider','Value', 0.5, 'Position',[20, 150, 190,
20], ...
        'Callback', @s2_function);
    t2 = uicontrol('Style','edit','String', num2str(get(s2, 'Value')), ...
        'Position',[220, 150, 60, 20], 'FontName', 'Arial', ...
        'FontSize', 12, 'Callback', @t2_function);

    uicontrol('Style','text','String','Scatter Max','FontName',...
        'Arial','FontSize',12,'Position',[20, 120, 180, 20]);

    s_min = min(min(min(Chg))); s_max = max(max(max(Chg)));
    c3 = uicontrol('Style','slider','Min', s_min, 'Max', s_max, 'Value',
s_max, 'Position',[20, 100, 190, 20], ...
        'Callback', @c3_function);
    map_max = uicontrol('Style','edit','String', s_max, ...
        'Position',[220, 100, 60, 20], 'FontName', 'Arial', ...
        'FontSize', 12, 'Callback', @map_max_function);

    uicontrol('Style','text','String','Scatter Min','FontName',...
        'Arial','FontSize',12,'Position',[20, 70, 180, 20]);

    c4 = uicontrol('Style','slider','Min', s_min, 'Max', s_max, 'Value',
s_min, 'Position',[20, 50, 190, 20], ...
        'Callback', @c4_function);
    map_min = uicontrol('Style','edit','String', s_min, ...
        'Position',[220, 50, 60, 20], 'FontName', 'Arial', ...
        'FontSize', 12, 'Callback', @map_min_function);

    uicontrol('Style','text','String','Scatter Plot Subdivisions:',
'FontName',...
        'Arial','FontSize',12,'Position',[20, 20, 200, 20]);

    sd = uicontrol('Style','edit','String', 2, 'Position', [220, 20, 40,
20], ...
        'FontName', 'Arial', 'FontSize', 12, 'Callback',
@redraw_selected_plane);

    cb = {};
    cb{1} = uicontrol('Style','checkbox','String','Charge Cut','Value',
...
        1, 'Position', [300, 470, 200, 20], 'FontName', ...
        'Arial','FontSize',12);
    cb{2} = uicontrol('Style','checkbox','String','Project Charge Cut',
'Value', ...
        0, 'Position', [300, 450, 200, 20], 'FontName', ...
        'Arial','FontSize',12,'Callback',
@redraw_selected_plane);
    cb{3} = uicontrol('Style','checkbox','String','Retain Slices',
'Value', ...
        0, 'Position', [300, 430, 200, 20], 'FontName', ...
        'Arial','FontSize',12,'Callback',
@redraw_selected_plane);

```

```

    cb{4} = uicontrol('Style', 'checkbox', 'String', 'Charge Scatter Plot',
'Value', ...
                    0, 'Position', [300, 410, 200, 20], 'FontName', ...
                    'Arial', 'FontSize', 12, 'Callback', @cb4_function);

    uicontrol('Style', 'text', 'String', 'View Vector:', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [300, 380, 180, 20]);

    uicontrol('Style', 'text', 'String', 'H', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [320, 360, 20, 20]);
    uicontrol('Style', 'text', 'String', 'K', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [380, 360, 20, 20]);
    uicontrol('Style', 'text', 'String', 'L', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [440, 360, 20, 20]);

    vv = {};
    vv{1} = uicontrol('Style', 'edit', 'String', 1, ...
                      'Position', [310, 340, 40, 20], 'FontName',
'Arial', ...
                      'FontSize', 12, 'Callback', @change_view);
    vv{2} = uicontrol('Style', 'edit', 'String', 0, ...
                      'Position', [370, 340, 40, 20], 'FontName',
'Arial', ...
                      'FontSize', 12, 'Callback', @change_view);
    vv{3} = uicontrol('Style', 'edit', 'String', 0, ...
                      'Position', [430, 340, 40, 20], 'FontName',
'Arial', ...
                      'FontSize', 12, 'Callback', @change_view);

    uicontrol('Style', 'text', 'String', 'HKL List:', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [300, 310, 180, 20]);

    lb = uicontrol('Style', 'listbox', 'String', '1 0 0', 'Value', 1,
'FontName', ...
                  'Arial', 'FontSize', 12, 'Position', [300, 160, 180, 150],
'Callback', @lb_function);

    HKL_New = uicontrol('Style', 'pushbutton', 'String', 'New HKL',
'FontName', ...
                      'Arial', 'FontSize', 12, 'Position', [305, 130, 80, 20],
'Callback', @add_HKL);

    HKL_Remove = uicontrol('Style', 'pushbutton', 'String', 'Remove',
'FontName', ...
                          'Arial', 'FontSize', 12, 'Position', [395, 130, 80, 20],
'Callback', @remove_HKL);

    uicontrol('Style', 'text', 'String', 'X Lim:', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [300, 100, 60, 20]);
    uicontrol('Style', 'text', 'String', 'Y Lim:', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [300, 70, 60, 20]);
    uicontrol('Style', 'text', 'String', 'Z Lim:', 'FontName', ...
              'Arial', 'FontSize', 12, 'Position', [300, 40, 60, 20]);

    lm{1, 1} = uicontrol('Style', 'edit', 'String', ax1.XLim(1), ...
                        'Position', [360, 100, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);

```

```

    lm{1, 2} = uicontrol('Style','edit','String', ax1.YLim(1), ...
                        'Position',[360, 70, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);
    lm{1, 3} = uicontrol('Style','edit','String', ax1.ZLim(1), ...
                        'Position',[360, 40, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);

    lm{2, 1} = uicontrol('Style','edit','String', ax1.XLim(2), ...
                        'Position',[420, 100, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);
    lm{2, 2} = uicontrol('Style','edit','String', ax1.YLim(2), ...
                        'Position',[420, 70, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);
    lm{2, 3} = uicontrol('Style','edit','String', ax1.ZLim(2), ...
                        'Position',[420, 40, 50, 20], 'FontName',
'Arial', ...
                        'FontSize', 12, 'Callback', @change_limits);

    HKL_Remove = uicontrol('Style','pushbutton','String','Remove',
'FontName', ...
                        'Arial', 'FontSize', 12, 'Position',[395, 130,
80, 20], 'Callback', @remove_HKL);

    new_fig = uicontrol('Style','pushbutton','String','Output Charge Map',
'FontName', ...
                        'Arial', 'FontSize', 12, 'Position',[300, 10, 160, 20], 'Callback',
@output);

function s1_function(source, eventdata)
    set(t1, 'String', num2str(get(s1, 'Value')));
    move_plane;
end

function t1_function(source, eventdata)
    set(s1, 'Value', str2num(get(t1, 'String')));
    move_plane;
end

function s2_function(source, eventdata)
    set(t2, 'String', num2str(get(s2, 'Value')));
    redraw_selected_plane;
end

function t2_function(source, eventdata)
    set(t2, 'Value', str2num(get(t1, 'String')));
    redraw_selected_plane;
end

function c3_function(source, eventdata)
    set(map_max, 'String', num2str(get(c3, 'Value')));
    redraw_selected_plane;
end

```



```

function map_max_function(source, eventdata)
    set(c3, 'Value', str2num(get(map_max, 'String')));
    redraw_selected_plane;
end

function c4_function(source, eventdata)
    set(map_min, 'String', num2str(get(c4, 'Value')));
    redraw_selected_plane;
end

function map_min_function(source, eventdata)
    set(c4, 'Value', str2num(get(map_min, 'String')));
    redraw_selected_plane;
end

function move_plane(source, eventdata)
    dex = get(lb, 'Value');
    d{dex} = str2num(get(tl, 'String'));
    redraw_selected_plane;
end

function change_HKL(source, eventdata)
    dex = get(lb, 'Value'); HKL_List = get(lb, 'String');
    d{dex} = str2num(get(tl, 'String'));
    H = str2num(get(HKL_Entries{1}, 'String'));
    K = str2num(get(HKL_Entries{2}, 'String'));
    L = str2num(get(HKL_Entries{3}, 'String'));
    if all([H K L]==[0 0 0]); return; end;
    HKL{dex} = [H K L];

    if size(HKL_List, 1) == 1
        set(lb, 'String', strcat(num2str(H), {' '}, num2str(K), {' '},
num2str(L)));
    else
        HKL_List(dex) = strcat(num2str(H), {' '}, num2str(K), {' '},
num2str(L));
        set(lb, 'String', HKL_List);
    end
    redraw_selected_plane;
    cstart{dex} = Chg_Cut{dex}.CData;
end

function c1_funcnt(source, eventdata)
    dex = get(lb, 'Value');
    set(c_max, 'String', num2str(get(c1, 'Value')));
    val = [str2num(get(c_min, 'String')) str2num(get(c_max, 'String'))];
    caxis(ax1, val);
end

function cmax_funcnt(source, eventdata)
    dex = get(lb, 'Value');
    set(c1, 'Value', str2num(get(c_max, 'String')));
    val = [str2num(get(c_min, 'String')) str2num(get(c_max, 'String'))];
    caxis(ax1, val);
end

function c2_funcnt(source, eventdata)

```

```

    dex = get(lb, 'Value');
    set(c_min, 'String', num2str(get(c2, 'Value')));
    val = [str2num(get(c_min, 'String')) str2num(get(c_max, 'String'))];
    caxis(ax1, val);
end

function cmin_func(source, eventdata)
    dex = get(lb, 'Value');
    set(c2, 'Value', str2num(get(c_min, 'String')));
    val = [str2num(get(c_min, 'String')) str2num(get(c_max, 'String'))];
    caxis(ax1, val);
end

function change_view(source, eventdata)
    figure(f1);
    H = str2num(get(vv{1}, 'String'));
    K = str2num(get(vv{2}, 'String'));
    L = str2num(get(vv{3}, 'String'));
    if all([H K L]==[0 0 0])
        return
    end
    view([H K L]*lat);
end

function lb_function(source, eventdata)
    dex = get(lb, 'Value');
    set(t1, 'String', d{dex});
    set(s1, 'Value', d{dex});

    set(HKL_Entries{1}, 'String', HKL{dex}(1));
    set(HKL_Entries{2}, 'String', HKL{dex}(2));
    set(HKL_Entries{3}, 'String', HKL{dex}(3));
end

function cb4_function(source, eventdata)
    if size(Chg_Map, 2) > 0
        delete(Chg_Map{1}); Chg_Map = {};
    end
    redraw_selected_plane;
end

function add_HKL(source, eventdata)
    d{end+1} = str2num(get(t1, 'String'));
    HKL_List = get(lb, 'String');
    H = str2num(get(HKL_Entries{1}, 'String'));
    K = str2num(get(HKL_Entries{2}, 'String'));
    L = str2num(get(HKL_Entries{3}, 'String'));
    HKL{end+1} = [H K L];
    HKL_List = [HKL_List; strcat(num2str(H), {' '}, num2str(K), {' '},
num2str(L))];
    dex = size(HKL_List, 1);
    set(lb, 'String', HKL_List); set(lb, 'Value', dex);

    pos = size(HKL_List, 1);
    [Chg_Cut, Chg_Plane] = Build_Charge_Density_Plane(f1, ax1, Chg,
HKL{end}, d{end}, lat, Chg_Cut, pos);

```

```

        HKL_Plane = Draw_HKL_Plane(f1, ax1, HKL_Plane, HKL{end}, d{end}, lat,
pos);

        [Atoms, flamp] = draw_atoms(f1, ax1, Atoms, flamp, atom_positions,
atom_type, Atom_List, lat, atom_num, r);
        caxis(ax1, 'auto');

    end

    function redraw_selected_plane(source, eventdata)
        dex = get(lb, 'Value');
        if ~get(cb{3}, 'Value')
            for i = 1:size(HKL_Plane{dex}, 2); delete(HKL_Plane{dex}{i});
end;
            for i = 1:size(Atoms, 2); delete(Atoms{i}); end;
            delete(Chg_Cut{1, dex});
        end

        [Chg_Cut, Chg_Plane] = Build_Charge_Density_Plane(f1, ax1, Chg,
HKL{dex}, d{dex}, lat, Chg_Cut, dex);
        HKL_Plane = Draw_HKL_Plane(f1, ax1, HKL_Plane, HKL{dex}, d{dex}, lat,
dex);

        [Atoms, flamp] = draw_atoms(f1, ax1, Atoms, flamp, atom_positions,
atom_type, Atom_List, lat, atom_num, r);

        if get(cb{2}, 'Value')
            x = get(Chg_Cut{dex}, 'XData');
            y = get(Chg_Cut{dex}, 'YData');
            z = get(Chg_Cut{dex}, 'ZData');
            if HKL{dex}(2) ~= 0
                shft = (d{dex}-1)*HKL{dex}*lat - 1*HKL{dex};
            else
                shft = d{dex}*HKL{dex}*lat + 1*HKL{dex};
            end

            for i = 1:size(x, 1)
                for j = 1:size(x, 2)
                    x(i, j) = x(i, j) - shft(1);
                    y(i, j) = y(i, j) - shft(2);
                    z(i, j) = z(i, j) - shft(3);
                end
            end

            set(Chg_Cut{dex}, 'XData', x);
            set(Chg_Cut{dex}, 'YData', y);
            set(Chg_Cut{dex}, 'ZData', z);
        end

        if get(cb{4}, 'Value')
            H = str2num(get(Map_Entries{1}, 'String'));
            K = str2num(get(Map_Entries{2}, 'String'));
            L = str2num(get(Map_Entries{3}, 'String'));
            Subdiv = str2num(get(sd, 'String'));
            sHKL = [H K L];
            ds = 1-str2num(get(t2, 'String'));
            if size(Chg_Map, 2) > 0
                delete(Chg_Map{1}); Chg_Map = {};
            end
        end
    end
end

```

```

        [coords, Pnt_Dens, Chg_Map]=draw_density_map(f1, ax1, Chg_Map,
Subdiv, Chg, str2num(get(map_min, 'String')), str2num(get(map_max,
'String')), lat, sHKL, ds);
        end

        caxis(ax1, 'auto'); val = caxis(ax1);
        set(c1, 'Min', val(1)); set(c1, 'Max', val(2)); set(c1, 'Value',
val(2));
        set(c2, 'Min', val(1)); set(c2, 'Max', val(2)); set(c2, 'Value',
val(1));
        set(c_max, 'String', val(2)); set(c_min, 'String', val(1));
        end

function remove_HKL(source, eventdata)
    dex = get(lb, 'Value');
    for i = 1:size(HKL_Plane{dex}, 2); delete(HKL_Plane{dex}{i}); end;
    delete(Chg_Cut{1, dex});
    Chg_Cut{1, dex} = [];
    HKL_List = get(lb, 'String');
    HKL_List(dex) = [];
    set(lb, 'String', HKL_List); set(lb, 'Value', 1);
    HKL(dex) = []; d(dex) = []; caxis(ax1, 'auto');
end

function change_limits(source, eventdata)
    ax1.XLim = [str2num(get(lm{1, 1}, 'String')), str2num(get(lm{2, 1},
'String'))];
    ax1.YLim = [str2num(get(lm{1, 2}, 'String')), str2num(get(lm{2, 2},
'String'))];
    ax1.ZLim = [str2num(get(lm{1, 3}, 'String')), str2num(get(lm{2, 3},
'String'))];
end

function output(source, eventdata)
    dex = get(lb, 'Value');
    val = caxis(ax1);
    fnew = figure('Color', 'w'); axnew = axes('Color', 'none');
    axnew.YAxis.Visible = 'off'; axnew.XAxis.Visible = 'off';
    axnew.ZAxis.Visible = 'off'; axis equal;
    Chg_temp = {};
    [Chg_temp, ~] = Build_Charge_Density_Plane(fnew, axnew, Chg,
HKL{dex}, d{dex}, lat, Chg_temp, dex);

    caxis(axnew, val);

    c = colorbar;
    temp = linspace(val(1), val(2), 10);
    c.Ticks = temp;
    c.TickLabels = round(temp*vol_cell, 1);
    temp = cellstr(c.TickLabels);
    temp{1} = strcat('<', temp{1});
    temp{end} = strcat('>', temp{end});
    c.TickLabels = temp;
    c.FontName = 'Arial';
    c.FontSize = 32.0;
    c.Label.String= strcat('Charge Density (e^{-}', char(197), '^{-3})');
    temp = c.Position; temp2 = axnew.Position;

```

```

temp(1) = temp2(1) + temp2(3) - 0.1; temp(2) = temp2(2); temp(4) =
temp2(4);
temp(3) = 0.02; c.Position = temp;

H = str2num(get(vv{1}, 'String'));
K = str2num(get(vv{2}, 'String'));
L = str2num(get(vv{3}, 'String'));
if all([H K L]==[0 0 0])
    return
end
view([H K L]*lat);
axnew.XLim = [str2num(get(lm{1, 1}, 'String')), str2num(get(lm{2, 1},
'String'))];
axnew.YLim = [str2num(get(lm{1, 2}, 'String')), str2num(get(lm{2, 2},
'String'))];
axnew.ZLim = [str2num(get(lm{1, 3}, 'String')), str2num(get(lm{2, 3},
'String'))];

figure(f1); axis(ax1);
end
end

```

Users Guide:

The program provided above is designed to accept CHGCAR outputs from structural optimizations conducted in the Vienna *ab initio* Simulation Package (VASP) to aid in the visualization of electronic charge density of target structures. The program was developed and implemented in MATLAB 2021b and while it should be fully compatible with older or newer versions, its reliability in these versions cannot be guaranteed. The program can be executed through the MATLAB command line by running Charge_Density.m. Importantly, to properly display atoms with unique colors based on their atomic number the file Atom_List.m must be included in the same directory from which Charge_Density.m is called.

Upon running the program, the user is presented with a file dialogue to select the file they wish to use for visualization. This program only accepts CHGCAR files output from the Vienna *ab Initio* Simulation Package (VASP). This program was compiled and tested using CHGCAR outputs from VASP 5.4.4 and performance cannot be guaranteed for other versions of VASP although the CHGCAR output is not expected to change significantly. Once the user has provided the file they wish to use for visualization the program will pull both the crystal structure information and charge density matrix from the selected input file. By default, the program will then display the crystal structure in a separate figure window as displayed in Figure G1 (using $\text{Li}_2\text{C}_6\text{O}_6$ as an example structure). The default presentation includes the positions of all atoms within the unit cell (the colors of these atoms are tabulated in Atom_List.m and can be edited as desired) and presents the boundaries of the unit cell as dashed lines. Additionally, the program displays a charge density map taken computed halfway along the $[1\ 0\ 0]$ lattice direction. The map displays regions of high electron density as red and low electron density as blue. The specific location of this map is outlined in red on the periphery of the unit cell.

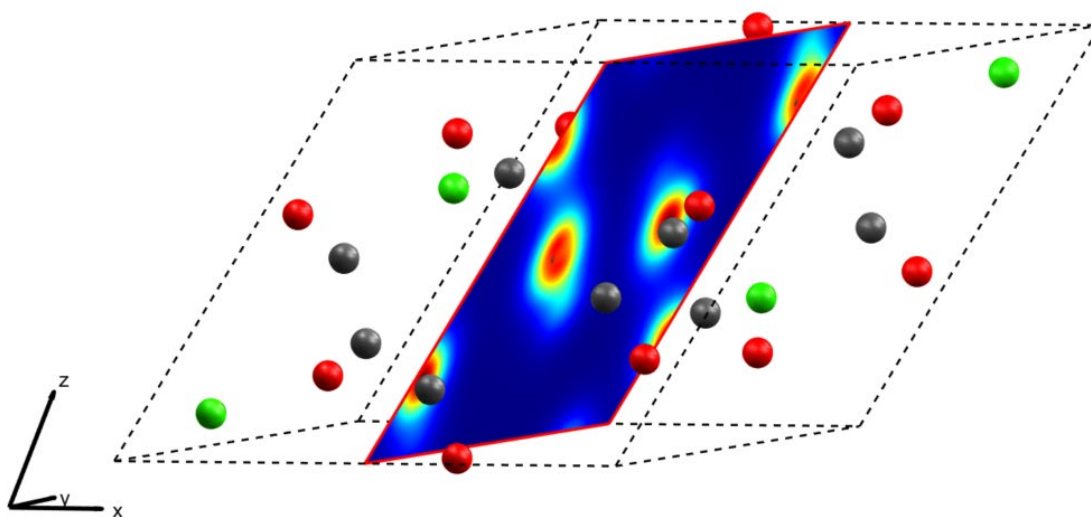


Figure G1. Example data set presented to the user upon loading a CHGCAR file into the programs memory. Atom positions are displayed as spheres of different color as dictated by Atom_List.mat and a planar charge density map is presented halfway along the $[1\ 0\ 0]$ lattice plane.

Along with this initial visualization, the program displays a control panel as pictured in Figure G2. The control panel gives the user the freedom to change the placement, color scale, and number of charge maps displayed along with other parameters used to view the electronic charge density of a given structure. What follows is a description of the various functions available from this control panel and their utility in visualizing electronic distributions in solids. It should be noted that, at time of publication, this program was able to process any CHGCAR file it was given including bulk structural optimizations as well as interfaces.

Possibly most important function of this program is the ability of the user to change the HKL index of the plane used to visualize charge density. The text boxes contained in region 1 of Figure G2 allow the user to define a given miller index along which the charge density of the crystal will be computed. Importantly, as the charge density information supplied by VASP has a limited spatial resolution the use of higher order miller indices ($[7\ 5\ 8]$ for instance) may lead to

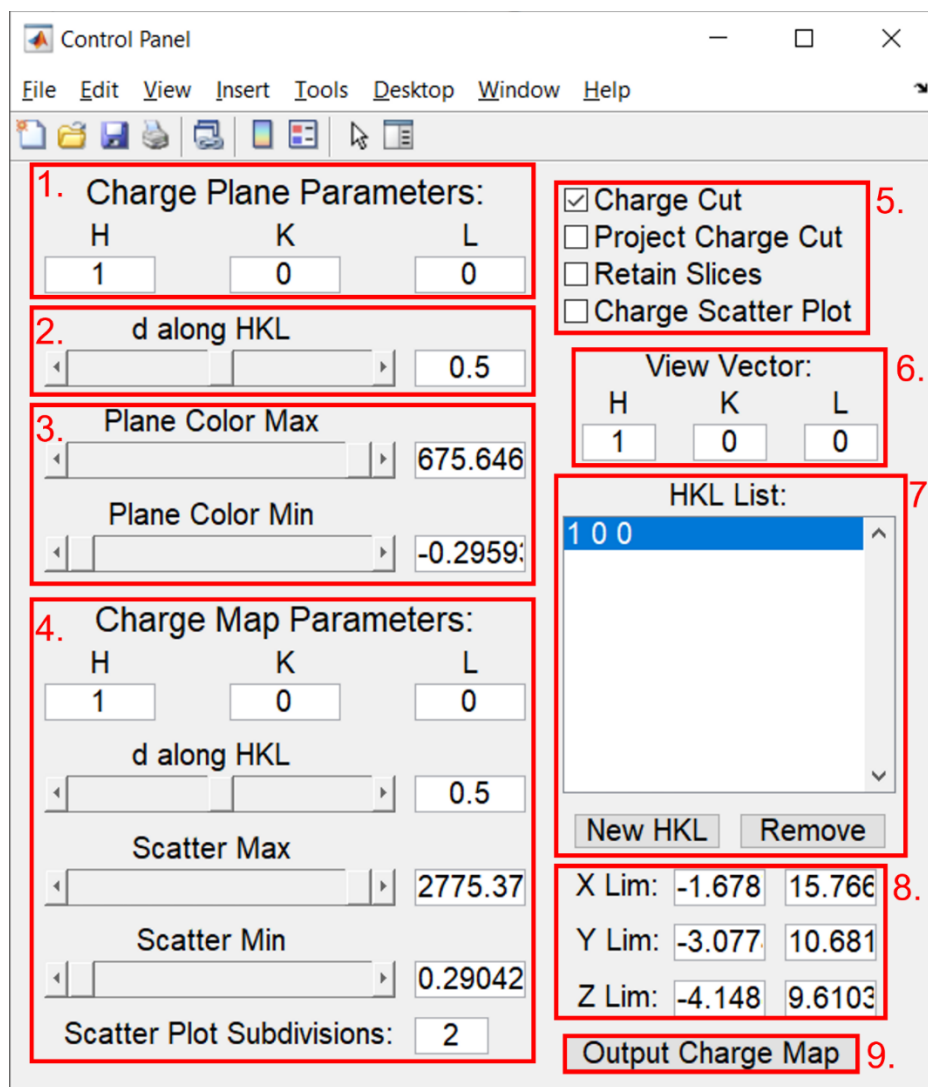


Figure G2. Control panel presented to the user upon loading a CHGCAR file into the program's memory. The sections of the panel are as follows:

1. HKL indices allowing the user to specify the plane along which the charge density is computed
2. Distance along the specified lattice vector where the charge density map is computed and displayed
3. Color sliders to change the minimum and maximum values of the color map to aid in visualization and thresholding
4. Parameters used to control a three-dimensional scatter plot of electron densities
5. Checkboxes determining specifically which options are used for visualization
6. HKL vector along which the structure is viewed
7. List of all HKL planes displayed in the current figure along with buttons to add or remove the selected HKL
8. Numeric limits on the x, y, and z ranges used for plotting
9. Button to output a charge density map at the current view vector with quantified electron densities based on the colormaps chosen by region 3

significant pixelation of the resulting charge density map. The structure displayed in the plot window will automatically update when the user defines a new HKL index as shown in Figure G3a, wherein the HKL plane is changed from the $[1\ 0\ 0]$ to the $[1\ 1\ 1]$. Additionally, by using the slider located in region 2 of the control panel the user can control the distance the plane is located along the specified HKL lattice vector. This is depicted in Figure G3b where the distance along the HKL vector d is changed from 0.3 in (a) to 0.6 in (b). Keep in mind that $d = 1$ places the lattice plane at the very edge of the unit cell and, as such, represents the % along a given lattice vector.

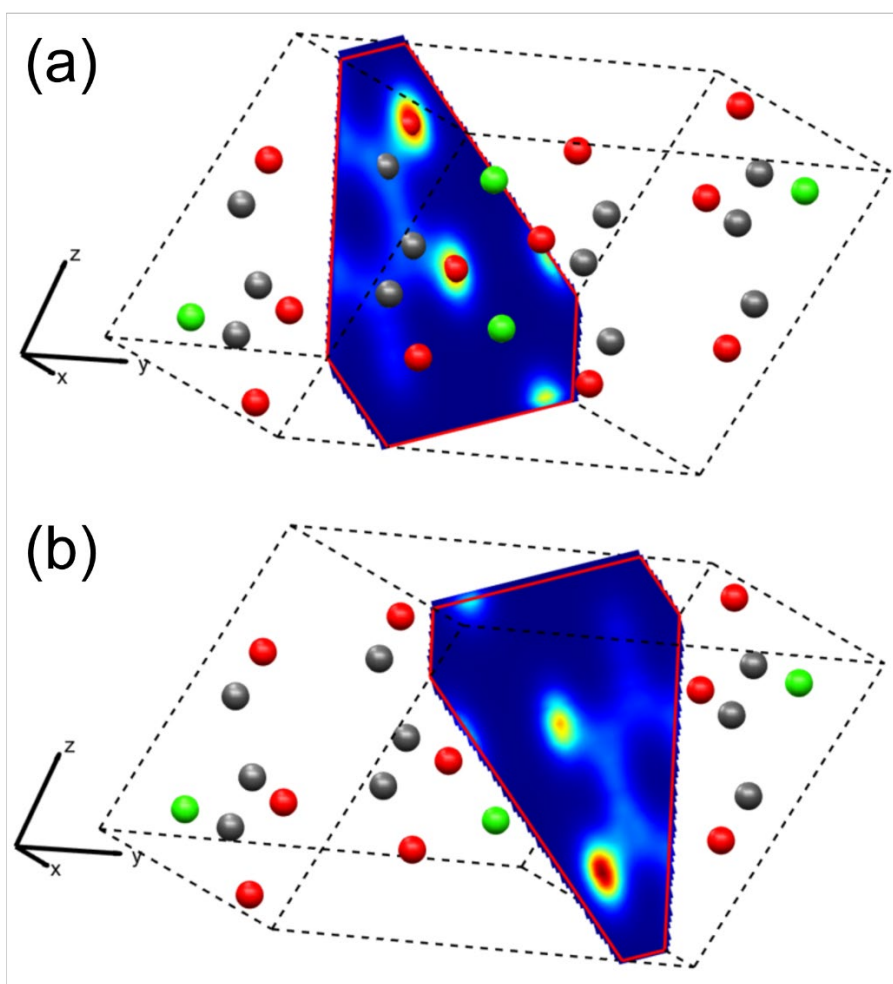


Figure G3. Visualizations of the charge density profiles along the $[1\ 1\ 1]$ lattice plane in $\text{Li}_2\text{C}_6\text{O}_6$ at (a) $d = 0.3$ and (b) $d = 0.6$.

Figure G3a displays another important function of this program, the ability to modify the color limits used in representing charge density information. The figure shows a very high local electron density around the single oxygen atom within the plane which obscures details about the electron density around the carbon framework. To resolve this the user need only lower the maximum allowable electron density using the slider in region 3 of the control panel. The results of doing so are presented in Figure G4b revealing additional detail in the charge distribution that would otherwise be obscured by the high electron density associated with oxygen. The same approach can be taken to help visualize density distribution in regions of high electron density by screening out the contributions from low density areas.

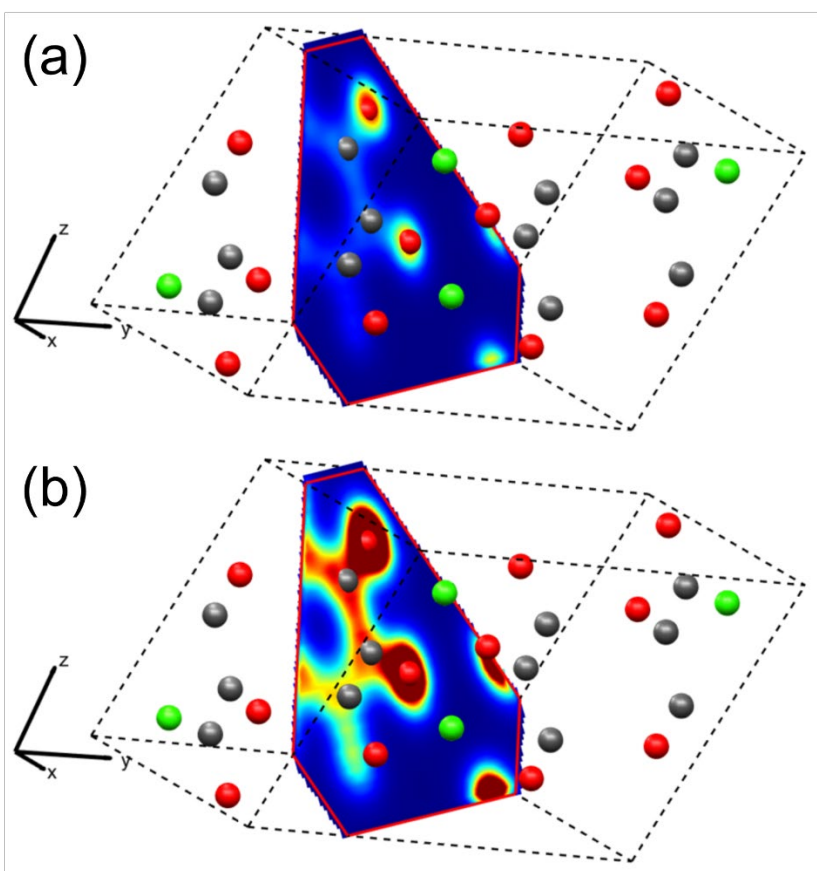


Figure G4. Two different color maps displaying the electron density within the carbon layer of $\text{Li}_2\text{C}_6\text{O}_6$ both (a) before any color scaling and (b) after reducing the maximum displayed electron density.

The examples displayed thus far display single slices of charge density through a given HKL plane, however, it is often useful to visualize the charge distribution across multiple lattice planes. This can be accomplished using the “New HKL” functionality found in region 7 of the control panel. On pressing this button a new HKL plane is defined whose indices are equivalent to those currently displayed in the HKL text boxes in region 1. Pressing the button will also add a new entry to the HKL list. This list allows the user to switch between different HKL planes to control each independently. In this implementation any changes to the d-spacing or HKL value are applied to the selected lattice plane. Figure G5 presents an example of this multi-plane display with one of the lattice planes aligned along $[1\ 0\ 0]$ and the other at $[0\ 1\ 0]$.

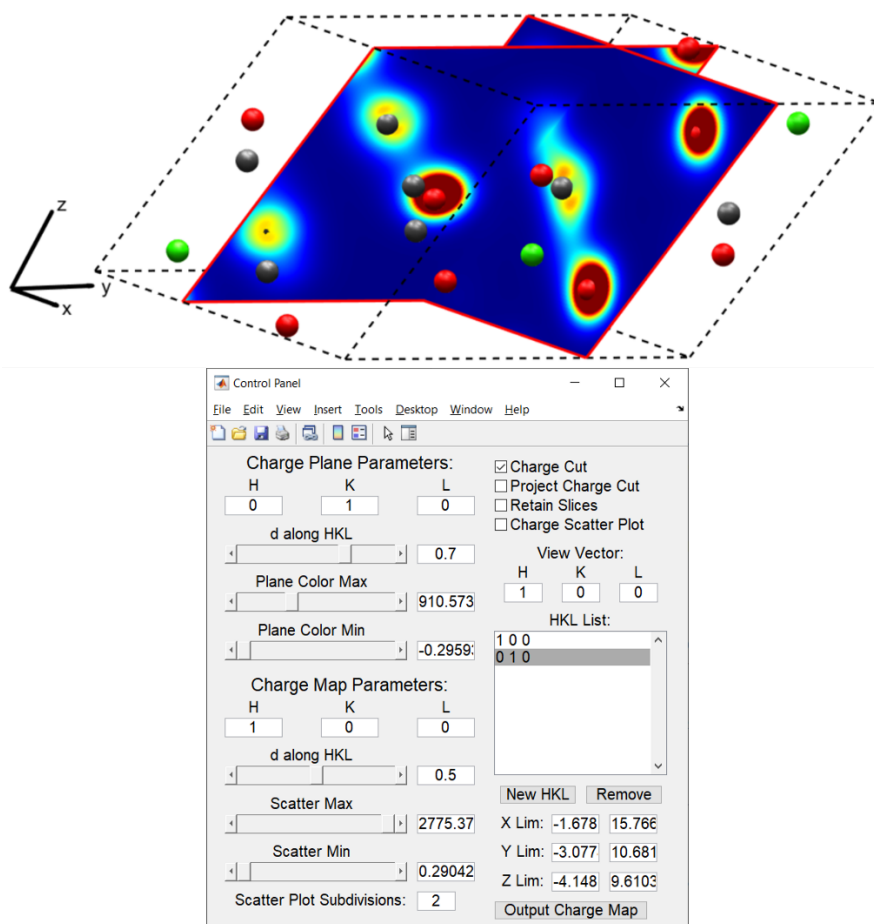


Figure G5. Output generated by visualizing two different HKL density maps simultaneously (top) along with the output presented on the control panel after adding the second plane.

While these transverse cuts of electron distribution are useful in the visualization of charge distribution within a single plane it is often to visualize the charge distribution as it exists in three-dimensional space. This functionality is present in the form of a three-dimensional scatter plot. To enable this option the user need only select the checkbox labeled “Charge Scatter Plot” in region 5 of the control panel. Likewise, the appearance of the generated scatter plot can be controlled from region 4 of the control panel. By default the scatter plot does not screen the charge density, thus the generated density map tends to fill the entirety of the cell volume (Figure G6a). This behavior can be changed using the charge density limit sliders in section 5 of the control panel. Screening out regions of low charge density clarifies the distribution of charge within the unit cell as demonstrated by Figure G6b.

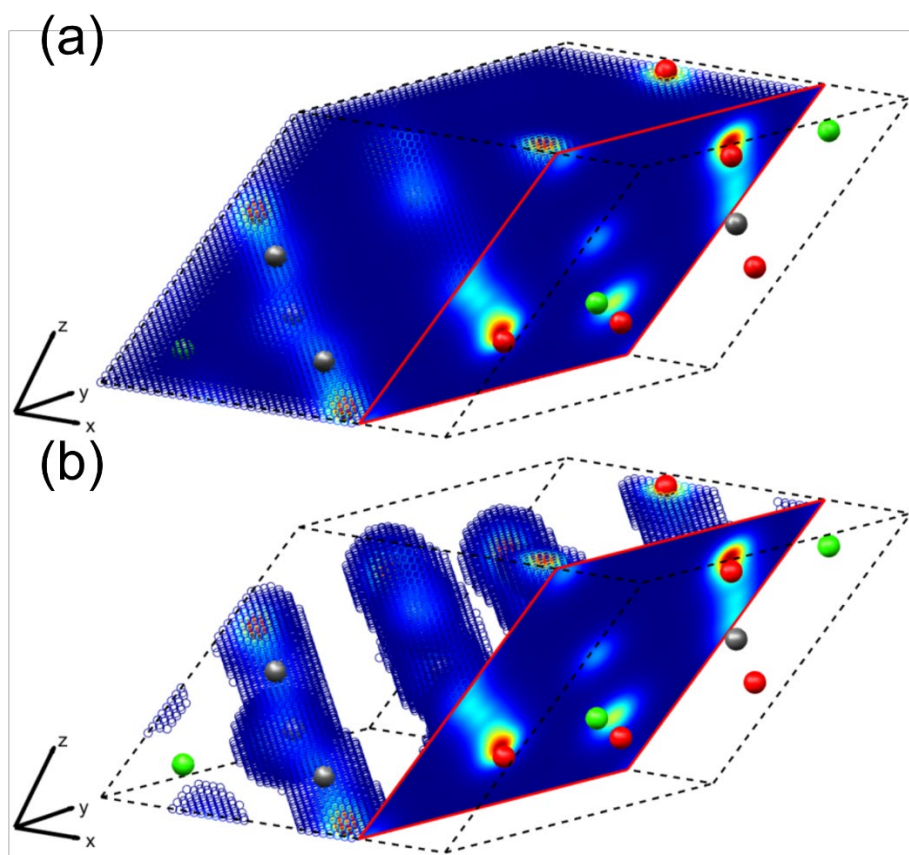


Figure G6. Charge distributions plotted as three-dimensional scatter plots both before (a) and after (b) screening out regions of low electron density for clarity.

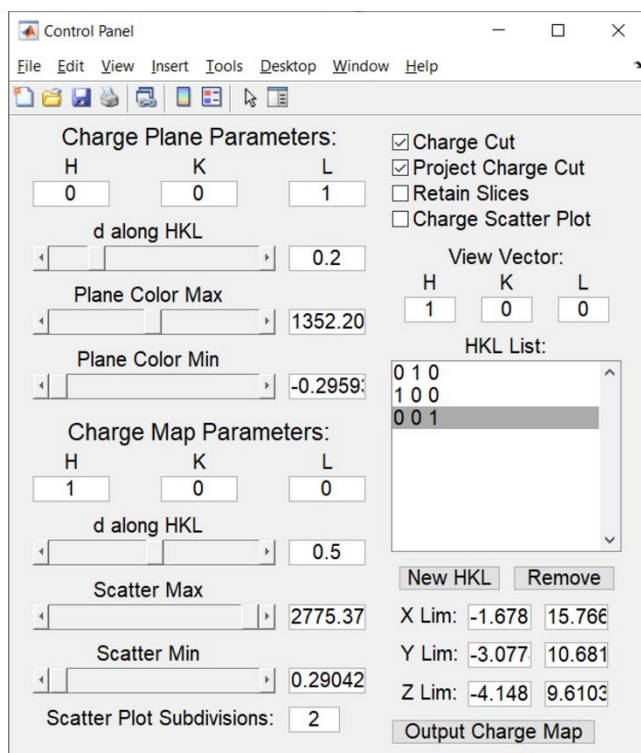
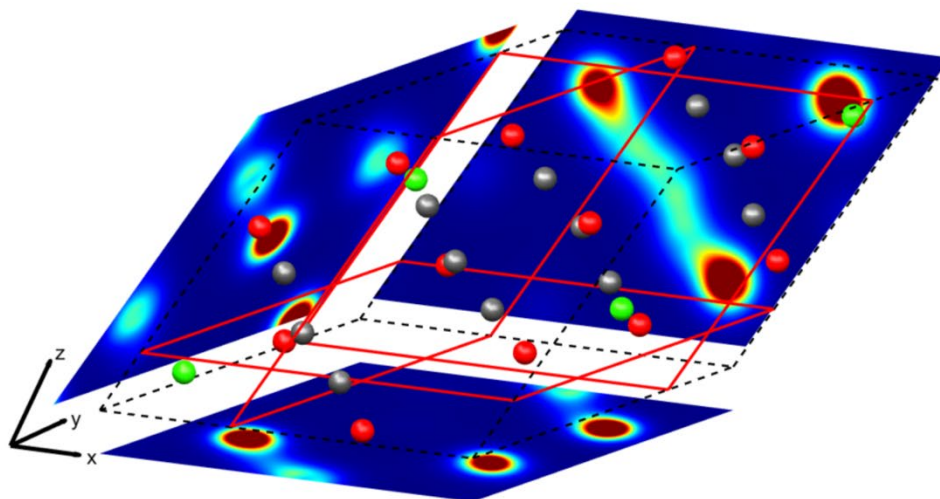


Figure G7. Charge densities plotted as projections to the outside of the unit cell.

In addition to these examples of charge density visualization within the unit cell the program can also project charge density maps to the outside of the cell to aid in visualization of desired. This option can be enabled by selecting the checkbox marked “Project Charge Cut” on the control panel. Finally, data can be presented in a more quantitative way by outputting a

charge density map to a separate figure window. Keep in mind that at time of publication this utility is only capable of outputting a single charge plane. Pressing the “Output Charge Map,” button generates a new figure window that plots the currently selected charge density map along with a color bar detailing the magnitudes of electron density presented in the figure, an example of which is presented in Figure G8.

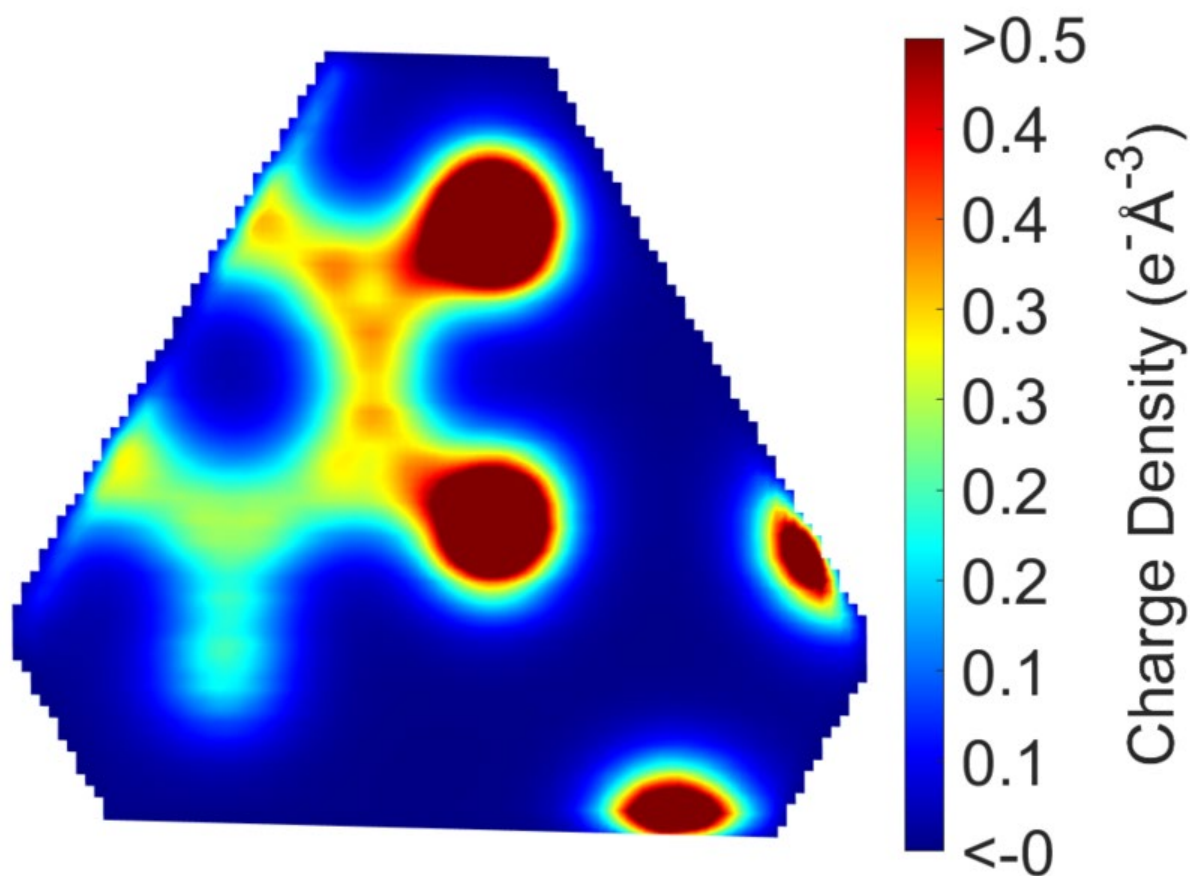


Figure G8. Example data output for a charge map generated along the $[1\ 1\ 1]$ lattice plane viewed along the $\langle 1\ 1\ 1 \rangle$ lattice vector.

Appendix H: General Peak Fitting Program for Multi-Spectrum Fitting (MATLAB):

Note: Requires Peak_Fit.fig to generate user interface.

```
function varargout = Peak_Fit(varargin)
%PEAK_FIT MATLAB code file for Peak_Fit.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @Peak_Fit_OpeningFcn, ...
                  'gui_OutputFcn',    @Peak_Fit_OutputFcn, ...
                  'gui_LayoutFcn',    [], ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before Peak_Fit is made visible.
function Peak_Fit_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    unrecognized PropertyName/PropertyValue pairs from the
%             command line (see VARARGIN)

% Choose default command line output for Peak_Fit
%% Initialize Variables
handles.output = hObject;
set(0,'DefaultFigureWindowStyle','normal');
hold(handles.axes1, 'on')
hold(handles.axes2, 'on')
handles.A = pwd;
handles.Spectra_index = {'Spectra:'};
handles.raw_data = {};
handles.working_data = {};
handles.xpnts = [];
handles.bkg = {};

handles.input_parameters = {};
handles.fitted_parameters = {};
handles.min_constraints = {};
handles.max_constraints = {};
handles.fit_profiles = {};
handles.RMS = {};
handles.Integrate = {};
handles.Risidual = {};
handles.Results = [];
```

```

handles.FileName = {};
handles.Filedir = {};
handles.xlsxname = {};
handles.FWHM = 5.0;
handles.peak_type = 'Pseudo Voigt';
handles.DOF = 4;

% Update handles structure
guidata(hObject, handles);
end

% --- Outputs from this function are returned to the command line.
function varargout = Peak_Fit_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
end

%% Buttons
% --- Executes on button press in Load_Data.
function Load_Data_Callback(hObject, eventdata, handles)
    [Filename, Filedir] = uigetfile('*..*', 'All Files (*.*)',
'MultiSelect','on');
    Fullname = fullfile(Filedir, Filename);
    handles.Filedir = Filedir;
    if iscell(Fullname)
        [filepath,name,ext] = fileparts(Fullname);
        handles.xlsxname = strcat(name{1},'_Fitting_Data', '.xlsx');
        for i = 1:size(Fullname, 2)
            if strcmp(ext{i}, '.xlsx')
                Data = xlsread(Fullname{i});
            elseif strcmp(ext{i}, '.asc')
                Data = importdata(Fullname{i}, ',');
            elseif strcmp(ext{i}, '.txt')
                Data = importdata(Fullname{i}, ',');
            elseif strcmp(ext{i}, '.CSV')
                Data = importdata(Fullname{i}, ',');
            else
                disp('Please Enter Valid File Format (.xlsx or .txt)');
            end
            Data = Data(:, 1:2);
            handles.raw_data{end+1} = Data;
            handles.Spectra_index{end+1} = name{i}; %=strcat('Spectrum_', i);
        end
    else
        [filepath,name,ext] = fileparts(Fullname);
        handles.xlsxname = strcat(name,'_Fitting_Data', '.xlsx');
        if strcmp(ext, '.xlsx')
            Data = readmatrix(Fullname);
        elseif strcmp(ext, '.asc')
            Data = importdata(Fullname, ',');
        elseif strcmp(ext, '.txt')
            Data = importdata(Fullname, ',');
        elseif strcmp(ext, '.CSV')
            Data = importdata(Fullname, ',');
        else
            disp('Please Enter Valid File Format (.xlsx or .txt)');
        end
    end
end

```



```

        for i = 2:size(Data, 2)
            handles.raw_data{end+1} = [Data(:, 1) Data(:, i)];
            if size(Data, 2) > 2
                handles.Spectra_index{end+1} = strcat('Spectrum_', int2str(i-
1));
            else
                handles.Spectra_index{end+1} = name;
            end
        end
    end

    handles.working_data = handles.raw_data;
    for i = 1 : size(handles.working_data, 2)
        handles.bkg{i} = [handles.working_data{i}(:, 1)
zeros(length(handles.working_data{i}(:, 1)), 1)];
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
        handles.Risidual{i} = [];
    end

    set(handles.Spectra, 'String', handles.Spectra_index);
    set(handles.Spectra, 'Value', 2);
    plot_current(hObject, handles);
    set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
    set(handles.X_Min, 'String', min(handles.working_data{1}(:, 1)));
    set(handles.X_Max, 'String', max(handles.working_data{1}(:, 1)));
    handles.FWHM = 0.1*(max(handles.working_data{1}(:, 1))-
min(handles.working_data{1}(:, 1)));
    set(handles.FWHM_DB, 'String', num2str(handles.FWHM));
    guidata(hObject, handles);
end

% --- Executes on button press in Background_Sub.
function Background_Sub_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    ord = 4;
    thresh = 0.01;
    if get(handles.Bkg_type, 'Value') == 1
        [bkg,a,it,ord,thresh, fct] =
backcor_multi(handles.working_data{value}(:, 1),
handles.working_data{value}(:, 2), ord, thresh);
        handles.bkg{value} = [handles.working_data{value}(:, 1) bkg];
        handles.working_data{value}(:, 2) = handles.working_data{value}(:,
2)-bkg;
        for i = 1:size(handles.working_data, 2)
            if i ~=value
                [bkg,a,it,ord,thresh, fct] =
backcor_multi(handles.working_data{i}(:, 1), handles.working_data{i}(:, 2),
ord, thresh, fct);
                handles.bkg{i} = [handles.working_data{i}(:, 1) bkg];
            end
        end
    end
end

```

```

        handles.working_data{i}(:, 2) = handles.working_data{i}(:,
2)-bkg;
    end
end

else
    [bkg,a,it,ord,thresh] = backcor_multi(handles.working_data{value}(:,
1), handles.working_data{value}(:, 2), ord, thresh);
    handles.bkg{value} = [handles.working_data{value}(:, 1) bkg];
    handles.working_data{value}(:, 2) = handles.working_data{value}(:,
2)-bkg;
end
plot_current(hObject, handles)
guidata(hObject, handles);
end

% --- Executes on button press in Spline_bkg.
function Spline_bkg_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    edit_window = figure('Position',[0 0 800
600],'NumberTitle','off','Name','Edit Constraints',...
    'MenuBar','none','ToolBar','figure','Resize','on','Color','w');
    ax = axes(edit_window);
    ax.Units = 'pixels';
    ax.Position = [75 75 500 500];
    ax.LineWidth = 3.0;
    ax.FontName = 'Arial';
    ax.FontSize = 18;
    box on
    hold on
    plot(ax, handles.working_data{value}(:, 1),
handles.working_data{value}(:, 2), 'LineWidth',2.0, 'Color', 'k');
    movegui(edit_window,'center');
    uicontrol('Style','text','String','Spline Order:', 'Position',
[600,500,100,25],'BackgroundColor','w')
    uicontrol('Style','text','String','Warning: Spline orders greater than
2 may yield unexpected results',...
    'Position', [600,350,150,100],'BackgroundColor','w')
    b_order = uicontrol('Style','edit','String','1', 'Position',
[700,500,50,25],'BackgroundColor','w');
    bok =
    uicontrol('Style','pushbutton','String','Submit','Position',[600,475,50,25],'
Callback',{@OKFcn},'BackgroundColor','w');
    bcompute = uicontrol('Style','pushbutton','String','Select
Knots','Position',[650,475,100,25],'Callback',{@Compute},'BackgroundColor','w
');
    bendpoint = uicontrol('Style','pushbutton','String','Endpoint
Subtraction','Position',[600,450,100,25],'Callback',{@Endpoint},'BackgroundCo
lor','w');
    bcancel =
    uicontrol('Style','pushbutton','String','Cancel','Position',[700,450,50,25],'
Callback',{@CancelFcn},'BackgroundColor','w');
    function CancelFcn(source,eventdata)
        % Just close the window
        uiresume(gcf);
        close(edit_window);
    end
end

```

```

function Compute(source, eventdata)
    cla(ax)
    plot(ax, handles.working_data{value}(:, 1),
handles.working_data{value}(:, 2), 'LineWidth',2.0, 'Color', 'k');
    order = str2num(get(b_order, 'String'))+1;
    Data = handles.working_data{value};
    [xpnts, ypnts] = getpts;
    [xpnts, x_order] = sort(xpnts);
    for i = 1:length(xpnts)
        [val, pos] = min(abs(Data(:, 1) - xpnts(i)));
        xpnts(i) = Data(pos, 1);
        ypnts(i) = Data(pos, 2);
    end
    %xpnts = [Data(1, 1); xpnts; Data(end, 1)];
    %ypnts = [Data(1, 2); ypnts; Data(end, 2)];
    handles.xpnts = xpnts;
    bkg = Build_Splines(order, xpnts, ypnts, Data);
    plot(ax, Data(:, 1), bkg, 'LineWidth', 2, 'Color', 'r');
    scatter(ax, xpnts, ypnts, 100, 'Marker', 'X',
'MarkerEdgeColor',[0.4941 0.1843 0.5569], 'LineWidth',3)
    handles.bkg{value} = [Data(:, 1), bkg];
end

function Endpoint(source, eventdata)
    cla(ax)
    plot(ax, handles.working_data{value}(:, 1),
handles.working_data{value}(:, 2), 'LineWidth',2.0, 'Color', 'k');
    order = str2num(get(b_order, 'String'))+1;
    Data = handles.working_data{value};
    xpnts = [Data(1, 1); Data(end, 1)];
    handles.xpnts = xpnts;
    ypnts = [Data(1, 2); Data(end, 2)];
    bkg = Build_Splines(order, xpnts, ypnts, Data);
    plot(ax, Data(:, 1), bkg, 'LineWidth', 2, 'Color', 'r');
    handles.bkg{value} = [Data(:, 1), bkg];
end

function OKFcn(source,eventdata)
    uiresume(gcf);
    if get(handles.Bkg_type, 'Value') == 1
        order = str2num(get(b_order, 'String'))+1;
        for i = 1 :size(handles.working_data, 2)
            Data = handles.working_data{i};
            xpnts = handles.xpnts;
            ypnts = xpnts;
            for j = 1:length(xpnts)
                [val, pos] = min(abs(Data(:, 1) - xpnts(j)));
                xpnts(j) = Data(pos, 1);
                ypnts(j) = Data(pos, 2);
            end
            handles.bkg{i} = [Data(:, 1) Build_Splines(order, xpnts,
ypnts, Data)];
            handles.working_data{i}(:, 2) = handles.working_data{i}(:,
2)-handles.bkg{i}(:, 2);
        end
    else

```

```

        handles.working_data{value}{:, 2) =
handles.working_data{value}{:, 2)-handles.bkg{value}{:, 2);
    end
    close(edit_window);
    plot_current(hObject, handles)
    guidata(hObject, handles);
end
function bkg = Build_Splines(order, xpnts, ypnts, Data)
    bkg = Data(:, 2);

    n = length(xpnts);
    mat = zeros((n-1)*order, (n-1)*order);
    temp = eye(order);
    temp(end-1, end-1:end) = [xpnts(1) 1];
    temp(end, end-1:end) = [xpnts(2) 1];
    mat(1:order, 1:order) = temp;
    Y = zeros(order, 1);
    Y(end-1) = ypnts(1);
    Y(end) = ypnts(2);
    for i = 2:order-1
        otemp = i+1;
        expon = (i:-1:0);
        temp = [zeros(order, order), eye(order, order)];
        temp(end, end)=0;
        xvals = xpnts(i:i+1).^expon;
        temp(order-i:order-i+1,end-size(xvals, 2)+1:end) = xvals;

        xvec = ones(i-1, size(xvals, 2))*xpnts(i);
        dmult = expon;
        expon = expon-1;

        for j = 3:otemp-1
            dmult = [dmult; dmult.*(dmult-1)];
            expon = [expon; expon-1];
        end
        xvals = dmult.*(xvec.^expon);
        temp(end-size(dmult, 1)+1:end, end-size(dmult, 2)+1:end) = xvals;
        xvals = -1*xvals(:, 2:end);
        temp(end-size(xvals, 1)+1:end, order-size(xvals, 2)+1:order) =
xvals;

        mat((i-1)*order+1:i*order, (i-2)*order+1:i*order) = temp;

        ytemp = zeros(order, 1);
        ytemp (order-i:order-i+1) = ypnts(i:i+1);
        Y = [Y; ytemp];

    end

    for i = order:n-1
        if order>2
            expon = (order-1:-1:0);
            dmult = expon;
            for j = 3:order-1
                dmult = [dmult; dmult.*(dmult-1)];
            end
            temp = zeros(order, 2*order);

```

```

temp(1:2, order+1:end) = xpnts(i:i+1).^expon;
expon = expon-1;

xvec = ones(order-2, order)*xpnts(i);
temp(3:order, 1:order) = -1*dmult.*(xvec.^expon);
temp(3:order, order+1:end) = dmult.*(xvec.^expon);
else
temp = zeros(order, 2*order);
expon = (order-1:-1:0);
xvals = xpnts(i:i+1).^expon;
temp(:, order+1:end) = xvals;
end

mat((i-1)*order+1:i*order, (i-2)*order+1:i*order) = temp;
ytemp = zeros(order, 1);
ytemp(1:2) = ypnts(i:i+1);
Y = [Y; ytemp];
end

coeffs = mat\Y;

for i = 1 : n-1
val = find((Data(:, 1)>xpnts(i))&(Data(:, 1)<xpnts(i+1)));
xtemp = Data(val,1);
expon = (order-1:-1:0);
xmat = xtemp.^expon;
bkg(val) = xmat*coeffs((i-1)*order+1 : i*order);
end
end
set( findall( edit_window, '-property', 'Units' ), 'Units', 'Normalized' )
end

% --- Executes on button press in Restore.
function Restore_Callback(hObject, eventdata, handles)
handles.working_data = handles.raw_data;
for i = 1 : size(handles.working_data, 2)
handles.bkg{i} = [handles.working_data{i}(:, 1)
zeros(length(handles.working_data{i}(:, 1)), 1)];
handles.input_parameters{i} = [];
handles.fitted_parameters{i} = [];
handles.min_constraints{i} = [];
handles.max_constraints{i} = [];
handles.fit_profiles{i} = [];
handles.RMS{i} = [];
handles.Integrate{i} = [];
handles.Risidual{i} = [];
end

set(handles.Spectra, 'String', handles.Spectra_index);
set(handles.Spectra, 'Value', 2);
axes(handles.axes1);
plot_current(hObject, handles)
handles.axes1.XTickLabels = [];
set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
set(handles.X_Min, 'String', min(handles.working_data{1}(:, 1)));

```

```

        set(handles.X_Max, 'String', max(handles.working_data{1}(:, 1)));
        handles.FWHM = 0.1*(max(handles.working_data{1}(:, 1))-
min(handles.working_data{1}(:, 1)));
        set(handles.FWHM_DB, 'String', num2str(handles.FWHM));
        guidata(hObject, handles);
    end

% --- Executes on button press in User_Define_Peaks.
function User_Define_Peaks_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    handles.input_parameters{value} = [];
    handles.fitted_parameters{value} = [];
    handles.min_constraints{value} = [];
    handles.max_constraints{value} = [];
    handles.fit_profiles{value} = [];
    handles.RMS{value} = [];
    handles.Residual{value} = [];
    plot_current(hObject, handles)

    [x,y]=getpts;
    [x, x_order] = sort(x);
    y = y(x_order,:);
    for i = 1:length(x)
        if strcmp(handles.peak_type, 'Voigt')
            handles.input_parameters{value} = [handles.input_parameters{value};
[x(i), y(i), handles.FWHM/2, handles.FWHM/2]];
        else
            handles.input_parameters{value} = [handles.input_parameters{value};
[x(i), y(i), handles.FWHM, 0.5]];
        end
    end
    set(handles.table, 'Data', handles.input_parameters{value})

    axes(handles.axes1)
    for i = 1:size(handles.input_parameters{value}, 1)
        if strcmp(handles.peak_type, 'Gaussian')
            plot(handles.working_data{value}(:, 1),
Create_Gaussian(handles.working_data{value}(:, 1),
handles.input_parameters{value}(i, :)), 'LineWidth', 2);
        elseif strcmp(handles.peak_type, 'Lorentzian')
            plot(handles.working_data{value}(:, 1),
Create_Lorentzian(handles.working_data{value}(:, 1),
handles.input_parameters{value}(i, :)), 'LineWidth', 2);
        elseif strcmp(handles.peak_type, 'Voigt')
            plot(handles.working_data{value}(:, 1),
Create_Voigt(handles.working_data{value}(:, 1),
handles.input_parameters{value}(i, :)), 'LineWidth', 2);
        elseif strcmp(handles.peak_type, 'Pseudo Voigt')
            plot(handles.working_data{value}(:, 1),
Create_Pseudo_Voigt(handles.working_data{value}(:, 1),
handles.input_parameters{value}(i, :)), 'LineWidth', 2);
        end
    end

    y = Build_Function(handles.working_data{value}(:, 1),
handles.input_parameters{value}, handles.peak_type);
    plot(handles.working_data{value}(:, 1), y, 'r', 'LineWidth', 2);

```

```

        guidata(hObject, handles);
end

% --- Executes on button press in Clear_Fit.
function Clear_Fit_Callback(hObject, eventdata, handles)
    for i = 1 : size(handles.working_data, 2)
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
        handles.Risidual{i} = [];
    end
    plot_current(hObject, handles)
    set(handles.table, 'Data', []);
    guidata(hObject, handles)
end

% --- Executes on button press in Set_Constraints.
function Set_Constraints_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    handles.min_constraints{value} = [];
    handles.max_constraints{value} = [];
    npeaks = size(handles.input_parameters{value}, 1);
    mheight = 25*npeaks;
    edit_window = figure('Position',[0 0 200*handles.DOF+200
mheight+100], 'NumberTitle','off', 'Name','Edit Constraints',...
    'MenuBar','none', 'ToolBar','figure', 'Resize','on', 'Color', 'w');
    movegui(edit_window, 'center');
    bok =
uicontrol('Style','pushbutton','String','Submit','Position',[510,10,50,25], 'C
allback',{@OKFcn}, 'BackgroundColor','w');
    bcancel =
uicontrol('Style','pushbutton','String','Cancel','Position',[600,10,50,25], 'C
allback',{@CancelFcn}, 'BackgroundColor','w');

    if handles.DOF == 3
        Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'fwhm min', 'fwhm
max'};
    elseif handles.DOF == 4
        if strcmp(handles.peak_type, 'Pseudo Voigt')
            Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'fwhm min',
'fwhm max', 'Eta min', 'Eta max'};
        else
            Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'Sigma min',
'Sigma max', 'Lambda min', 'Lambda max'};
        end
    end

    pos = handles.input_parameters{value}(:, 1);
    pht = max(handles.working_data{value}(:, 2));
    wid = max(handles.working_data{value}(:, 1))-
min(handles.working_data{value}(:, 1));

    if strcmp(handles.peak_type, 'Voigt')

```

```

        min_lims = [-0.2*wid, -pht, -wid, -wid];
        max_lims = [0.2*wid, pht, wid/2, wid/2];
    else
        min_lims = [-0.2*wid, -pht, -wid, -2.0];
        max_lims = [0.2*wid, pht, wid, 2.0];
    end

    peak_labels = get(handles.table, 'Data');
    peak_labels = peak_labels(:, 1);

    entries = cell(npeaks, 2*handles.DOF, 2);
    E_box = cell(npeaks, 2*handles.DOF, 2);
    for i = 1 : npeaks
        for j = 1: 2*handles.DOF
            if mod(j,2)
                switch j
                    case 1
                        entries{i, j} =
num2str(handles.input_parameters{value}(i, ceil(j/2))+min_lims(ceil(j/2)));
                    case {3, 5, 7}
                        entries{i, j} =
num2str(max(handles.input_parameters{value}(i,
ceil(j/2))+min_lims(ceil(j/2)), 0));
                    end
                else
                    switch j
                        case {2, 4, 6}
                            entries{i, j} = handles.input_parameters{value}(i,
ceil(j/2))+max_lims(ceil(j/2));
                        case 8
                            if strcmp(handles.peak_type, 'Voigt')
                                entries{i, j} = handles.input_parameters{value}(i,
ceil(j/2))+max_lims(ceil(j/2));
                            else
                                entries{i, j} =
min(handles.input_parameters{value}(i, ceil(j/2))+max_lims(ceil(j/2)), 1);
                            end
                        end
                    end
                end
            end
        end
    end

    for i = 1 : npeaks
        uicontrol('Style','text','FontName', 'Arial', 'FontSize', 12,
'String', num2str(peak_labels(i)), 'Position', [50, mheight-25*i+50, 100, 25],
'BackgroundColor','w');
        for j = 1 : 2*handles.DOF
            E_box{i, j} = uicontrol('Style','edit','String',entries{i,
j},'Position', [50+100*j, mheight-25*i+50, 50, 25], 'BackgroundColor','w');
        end
    end

    for i = 1:2*handles.DOF
        uicontrol('Style','text','FontName', 'Arial', 'FontSize', 12,
'String', Col_Names{i}, 'Position', [25+100*i, mheight+50, 100, 25],
'BackgroundColor','w');
    end
end

```



```

function CancelFcn(source,eventdata)
    % Just close the window
    uiresume(gcf);
    close(edit_window);
end

function OKFcn(source,eventdata)
    uiresume(gcf);
    for i = 1 : npeaks
        for j = 1: 2*handles.DOF
            if mod(j,2)
                handles.min_constraints{value}(i, ceil(j/2)) =
str2num(get(E_box{i, j}, 'String'));
            else
                handles.max_constraints{value}(i, ceil(j/2)) =
str2num(get(E_box{i, j}, 'String'));
            end
        end
    end
    close(edit_window);
    guidata(hObject, handles);
end

end

% --- Executes on button press in Minimize_Selected.
function Minimize_Selected_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    npeaks = size(handles.input_parameters, 1);
    pos = handles.input_parameters{value}(:, 1);
    pht = max(handles.working_data{value}(:, 2));
    wid = max(handles.working_data{value}(:, 1))-
min(handles.working_data{value}(:, 1));

    if strcmp(handles.peak_type, 'Voigt')
        min_lims = [-0.2*wid, -pht, -wid, -wid];
        max_lims = [0.2*wid, pht, wid/2, wid/2];
    else
        min_lims = [-0.2*wid, -pht, -wid, -0.5];
        max_lims = [0.2*wid, pht, wid, 0.5];
    end

    if size(handles.min_constraints{value}) ~=
size(handles.input_parameters{value}(:, 1:handles.DOF))
        for i = 1 : size(handles.input_parameters{value}, 1)
            handles.min_constraints{value} =
handles.input_parameters{value}(:, 1:handles.DOF) + min_lims(:,
1:handles.DOF);
        end
    end

    if size(handles.max_constraints{value}) ~=
size(handles.input_parameters{value}(:, 1:handles.DOF))
        for i = 1 : size(handles.input_parameters{value}, 1)

```

```

        handles.max_constraints{value} =
handles.input_parameters{value}(:, 1:handles.DOF) + max_lims(:,
1:handles.DOF);
    end
end

    opts = optimoptions('fmincon','Display','off');
    handles.fitted_parameters{value} = fmincon(@Cost_Function,
handles.input_parameters{value}(:, 1:handles.DOF),...
[], [], [], [], handles.min_constraints{value}(:,
1:handles.DOF),...
handles.max_constraints{value}(:, 1:handles.DOF), [],
opts);
    Simulated_Data = Build_Function(handles.working_data{value}(:, 1),
handles.fitted_parameters{value}, handles.peak_type);
    handles.Risidual{value} = (Simulated_Data-handles.working_data{value}(:,
2));
    handles.RMS{value} = sqrt(sum((handles.working_data{value}(:, 2) -
Simulated_Data).^2)/length(handles.working_data{value}(:,2)));

    peak_indv = [];
    tegrals = [];
    for i = 1:size(handles.input_parameters{value}, 1)
        if strcmp(handles.peak_type, 'Gaussian')
            peak_indv = [peak_indv
Create_Gaussian(handles.working_data{value}(:, 1),
handles.fitted_parameters{value}(i, :))];
        elseif strcmp(handles.peak_type, 'Lorentzian')
            peak_indv = [peak_indv
Create_Lorentzian(handles.working_data{value}(:, 1),
handles.fitted_parameters{value}(i, :))];
        elseif strcmp(handles.peak_type, 'Voigt')
            peak_indv = [peak_indv
Create_Voigt(handles.working_data{value}(:, 1),
handles.fitted_parameters{value}(i, :))];
        elseif strcmp(handles.peak_type, 'Pseudo Voigt')
            peak_indv = [peak_indv
Create_Pseudo_Voigt(handles.working_data{value}(:, 1),
handles.fitted_parameters{value}(i, :))];
        end
        tegrals = [tegrals; Int_Peak(handles.working_data{value}(:, 1),
peak_indv(:, i))];
    end
    handles.Integrate{value} = tegrals;

    handles.fit_profiles{value} = [handles.working_data{value}(:, 1)
handles.working_data{value}(:, 2) Simulated_Data peak_indv
handles.Risidual{value} handles.bkg{value}(:, 2)];
    assignin('base','fit_profiles',handles.fit_profiles);
    assignin('base','fit_parameters',handles.fitted_parameters);
    set(handles.table, 'Data', handles.fitted_parameters{value});
    set(handles.Error, 'String', num2str(handles.RMS{value}));
    plot_current(hObject, handles)
    handles.Results = update_results(hObject, handles);
    guidata(hObject, handles);

function [cost] = Cost_Function(params)

```

```

    Simulated_Data = Build_Function(handles.working_data{value}(:, 1),
params, handles.peak_type);
    cost = sqrt(sum((handles.working_data{value}(:, 2) -
Simulated_Data).^2)/length(handles.working_data{value}(:,2)));
end
end

% --- Executes on button press in Minimize_All.
function Minimize_All_Callback(hObject, eventdata, handles)
    opts = optimoptions('fmincon','Display','off');
    value = get(handles.Spectra, 'Value')-1;
    npeaks = size(handles.input_parameters{value}, 1);
    for i = 1:size(handles.working_data, 2)
        handles.input_parameters{i} = handles.fitted_parameters{value};
    end

    mheight = 25*npeaks;
    mwidth = 100*2*size(handles.input_parameters{value}, 2);
    edit_window = figure('Position',[0 0 200*handles.DOF+250
mheight+200],'NumberTitle','off','Name','Edit Constraints',...
    'MenuBar','none','ToolBar','figure','Resize','on','Color','w');
    movegui(edit_window,'center');
    bok =
uicontrol('Style','pushbutton','String','Submit','Position',[510,10,50,25],'C
allback',{@OKFcn},'BackgroundColor','w');
    bcancel =
uicontrol('Style','pushbutton','String','Cancel','Position',[600,10,50,25],'C
allback',{@CancelFcn},'BackgroundColor','w');

    if handles.DOF == 3
        Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'fwhm min', 'fwhm
max'};
    elseif handles.DOF == 4
        if strcmp(handles.peak_type, 'Pseudo Voigt')
            Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'fwhm min',
'fwhm max', 'Eta min', 'Eta max'};
        else
            Col_Names = {'X min', 'X max', 'Y min ', 'Y max', 'Sigma min',
'Sigma max', 'Lambda min', 'Lambda max'};
        end
    end

    pos = handles.input_parameters{value}(:, 1);
    pht = max(handles.working_data{value}(:, 2));
    wid = max(handles.working_data{value}(:, 1))-
min(handles.working_data{value}(:, 1));

    if strcmp(handles.peak_type, 'Voigt')
        min_lims = [-0.2*wid, -pht, -wid, -wid];
        max_lims = [0.2*wid, pht, wid/2, wid/2];
    else
        min_lims = [-0.2*wid, -pht, -wid, -1.0];
        max_lims = [0.2*wid, pht, wid, 1.0];
    end

    peak_labels = get(handles.table, 'Data');
    peak_labels = peak_labels(:, 1);

```

```

entries = cell(npeaks, 2*handles.DOF, 2);
E_box = cell(npeaks, 2*handles.DOF, 2);
for i = 1 : npeaks
    for j = 1: 2*handles.DOF
        if mod(j, 2)
            switch j
                case 1
                    entries{i, j} =
num2str(handles.input_parameters{value}(i, ceil(j/2))+min_lims(ceil(j/2)));
                case {3, 5, 7}
                    entries{i, j} =
num2str(max(handles.input_parameters{value}(i,
ceil(j/2))+min_lims(ceil(j/2)), 0));
            end
        else
            switch j
                case {2, 4, 6}
                    entries{i, j} = handles.input_parameters{value}(i,
ceil(j/2))+max_lims(ceil(j/2));
                case 8
                    if strcmp(handles.peak_type, 'Voigt')
                        entries{i, j} = handles.input_parameters{value}(i,
ceil(j/2))+max_lims(ceil(j/2));
                    else
                        entries{i, j} =
min(handles.input_parameters{value}(i, ceil(j/2))+max_lims(ceil(j/2)), 1);
                    end
            end
        end
    end
end

for i = 1 : npeaks
    uicontrol('Style','text','FontName', 'Arial', 'FontSize', 12,
'String', num2str(peak_labels(i)), 'Position', [50, mheight-25*i+50, 100, 25],
'BackgroundColor','w');
    for j = 1 : 2*handles.DOF
        E_box{i, j} = uicontrol('Style','edit','String',entries{i,
j},'Position', [50+100*j, mheight-25*i+50, 50, 25], 'BackgroundColor','w',
'Enable', 'off');
    end
end

for i = 1:2*handles.DOF
    uicontrol('Style','text','FontName', 'Arial', 'FontSize', 12,
'String', Col_Names{i}, 'Position', [25+100*i, mheight+50, 100, 25],
'BackgroundColor','w');
end

if handles.DOF == 3
    Text = {'X freedom', 'Y freedom', 'FWHM freedom'};
elseif handles.DOF == 4
    if strcmp(handles.peak_type, 'Pseudo Voigt')
        Text = {'X freedom', 'Y freedom', 'FWHM freedom', 'Eta freedom'};
    else

```

```

        Text = {'X freedom', 'Y freedom', 'Sigma Freedom', 'Lambda
freedom'};
    end
end

U_box = {};
for i = 1:length(Text)
    uicontrol('Style','text','FontName','Arial','FontSize',12,
'String',Text{i},'Position',[150*i, mheight+150, 150, 25],
'BackgroundColor','w');
    U_box{i} = uicontrol('Style','edit','String',max_lims(i),'Position',
[150*i+25, mheight+100, 100, 25], 'BackgroundColor','w');
end

cbox = uicontrol('Style','checkbox','FontName','Arial','FontSize',12,
'String','Apply to All','Position',[50, mheight+100, 100, 25],
'BackgroundColor','w','Value',1.0,'Callback',{@ChkBox});

function ChkBox(source, eventdata)
    if get(cbox, 'Value') == 0.0
        for i = 1:npeaks
            for j = 1:2*handles.DOF
                set(E_box{i, j}, 'Enable', 'on')
            end
        end
    else
        for i = 1:npeaks
            for j = 1:2*handles.DOF
                diff = str2num(get(U_box{ceil(j/2)}, 'String'));
                set(E_box{i, j}, 'String', Fill(i, j, diff, value,
hObject, handles))
                set(E_box{i, j}, 'Enable', 'off')
            end
        end
    end
    function [input] = Fill(i, j, diff, value, hObject, handles)
        if mod(j, 2)
            switch j
                case 1
                    input = handles.input_parameters{value}(i,
ceil(j/2))-diff;
                case {3, 5, 7}
                    input = max(handles.input_parameters{value}(i,
ceil(j/2))-diff, 0);
                end
            else
                switch j
                    case {2, 4, 6}
                        input = handles.input_parameters{value}(i,
ceil(j/2))+diff;
                    case 8
                        if strcmp(handles.peak_type, 'Voigt')
                            input = handles.input_parameters{value}(i,
ceil(j/2))+diff;
                        else
                            input = min(handles.input_parameters{value}(i,
ceil(j/2))+diff, 1);

```

```

        end
    end
end
end
end
function CancelFcn(source,eventdata)
    % Just close the window
    uiresume(gcf);
    close(edit_window);
end

function OKFcn(source,eventdata)
    uiresume(gcf);

    for i = 1 : npeaks
        for j = 1: 2*handles.DOF
            if mod(j,2)
                handles.min_constraints{value}(i, ceil(j/2)) =
str2num(get(E_box{i, j}, 'String'));
            else
                handles.max_constraints{value}(i, ceil(j/2)) =
str2num(get(E_box{i, j}, 'String'));
            end
        end
    end

    for i = 1 : size(handles.working_data, 2)
        handles.min_constraints{i} = handles.min_constraints{value};
        handles.max_constraints{i} = handles.max_constraints{value};
    end
    close(edit_window);

    for j = 1 : size(handles.working_data, 2)
        handles.fitted_parameters{j} = fmincon(@Cost_Function,
handles.input_parameters{j}(:, 1:handles.DOF),...
[], [], [], [], handles.min_constraints{j}(:,
1:handles.DOF),...
handles.max_constraints{j}(:, 1:handles.DOF),
[], opts);
        Simulated_Data = Build_Function(handles.working_data{j}(:,
1), handles.fitted_parameters{j}, handles.peak_type);
        handles.Residual{j} = (Simulated_Data-
handles.working_data{j}(:, 2));
        handles.RMS{j} = sqrt(sum((handles.working_data{j}(:, 2) -
Simulated_Data).^2)/length(handles.working_data{j}(:,2)));

        peak_indv = [];
        tegrals = [];
        for i = 1:size(handles.input_parameters{j}, 1)
            if strcmp(handles.peak_type, 'Gaussian')
                peak_indv = [peak_indv
Create_Gaussian(handles.working_data{j}(:, 1),
handles.fitted_parameters{j}(i, :))];
            elseif strcmp(handles.peak_type, 'Lorentzian')
                peak_indv = [peak_indv
Create_Lorentzian(handles.working_data{j}(:, 1),
handles.fitted_parameters{j}(i, :))];
            end
        end
    end
end

```

```

        elseif strcmp(handles.peak_type, 'Voigt')
            peak_indv = [peak_indv
Create_Voigt(handles.working_data{j}(:, 1), handles.fitted_parameters{j}(i,
:))];
        elseif strcmp(handles.peak_type, 'Pseudo Voigt')
            peak_indv = [peak_indv
Create_Pseudo_Voigt(handles.working_data{j}(:, 1),
handles.fitted_parameters{j}(i, :))];
        end
        tegrals = [tegrals; Int_Peak(handles.working_data{j}(:,
1), peak_indv(:, i))];
    end
    handles.Integrate{j} = tegrals;
    handles.fit_profiles{j} = [handles.working_data{j}(:, 1)
handles.working_data{j}(:, 2) Simulated_Data peak_indv handles.Risidual{j}
handles.bkg{j}(:, 2)];
    assignin('base','fit_profiles',handles.fit_profiles);
    assignin('base','fit_parameters',handles.fitted_parameters);
    guidata(hObject, handles);

end

function [cost] = Cost_Function(params)
    Simulated_Data = Build_Function(handles.working_data{j}(:, 1),
params, handles.peak_type);
    cost = sqrt(sum((handles.working_data{j}(:, 2) -
Simulated_Data).^2)/length(handles.working_data{j}(:,2)));
end

disp('Minimization Complete')
set(handles.table, 'Data', handles.fitted_parameters{value});
set(handles.Error, 'String', num2str(handles.RMS{value}));
plot_current(hObject, handles)
handles.Results = update_results(hObject, handles);
guidata(hObject, handles);
end
end

% --- Executes on button press in Export_Data.
function Export_Data_Callback(hObject, eventdata, handles)
    [file,path,indx] = uiputfile('*.xlsx','File Selection',handles.xlsxname);
    path = strcat(path, file);
    %disp(path)

    e = actxserver('Excel.Application');
    eWorkbook = e.Workbooks.Add;
    e.Visible = 1;
    eSheets = e.ActiveWorkbook.Sheets;
    eSheet1 = eSheets.get('Item', 1);
    eSheet1.Activate
    for i = 1 : size(handles.fit_profiles, 2)
        if ~isempty(handles.fit_profiles{i})
            eSheets.Item(eSheets.Count).Name = handles.Spectra_index{i+1};
            n = size(handles.fit_profiles{i}, 2);
            header_text = {'X', 'Y', 'Fit'};
            row_text = {};
            for j = 4 : n-2

```

```

        header_text{j} = strcat('Peak_', num2str(j-3));
        row_text{j-3, 1} = strcat('Peak_', num2str(j-3));
    end
    header_text{end+1} = 'Residual';
    header_text{end+1} = 'Background';

    %Input header
    hstart = strcat(xlscol(1), int2str(1));
    hfinish = strcat(xlscol(length(header_text)), int2str(1));
    hrnge = sprintf('%s:%s', hstart, hfinish);
    eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
    eActivsheetRange.Value = header_text;

    %Input data
    hstart = strcat(xlscol(1), int2str(2));
    hfinish = strcat(xlscol(n), int2str(size(handles.fit_profiles{i},
1)+1));

    hrnge = sprintf('%s:%s', hstart, hfinish);
    eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
    eActivsheetRange.Value = handles.fit_profiles{i};

    %Input Parameters
    num = handles.DOF;
    header_text = {'X Position', 'Intensity', 'FWHM'};
    switch num
        case 3
            if strcmp(handles.peak_type, 'Gaussian')
                header_text{end+1} = 'Integrated Intensities';
                header_text{end+1} = 'RMS';
                header_text{end+1} = 'Profile = Gaussian';
            else
                header_text{end+1} = 'Integrated Intensities';
                header_text{end+1} = 'RMS';
                header_text{end+1} = 'Profile = Lorentzian';
            end
        case 4
            if strcmp(handles.peak_type, 'Pseudo Voigt')
                header_text{end+1} = 'Eta';
                header_text{end+1} = 'Integrated Intensities';
                header_text{end+1} = 'RMS';
                header_text{end+1} = 'Profile = Pseudo-Voigt';
            else
                header_text{end} = 'Sigma';
                header_text{end+1} = 'Lambda';
                header_text{end+1} = 'Integrated Intensities';
                header_text{end+1} = 'RMS';
                header_text{end+1} = 'Profile = Voigt';
            end
    end

    %Input header
    hstart = strcat(xlscol(n+3), int2str(1));
    hfinish = strcat(xlscol(n+2+length(header_text)), int2str(1));
    hrnge = sprintf('%s:%s', hstart, hfinish);
    eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
    eActivsheetRange.Value = header_text;

```



```

%input Row Labels
hstart = strcat(xlscol(n+2), int2str(2));
hfinish = strcat(xlscol(n+2), int2str(1+size(row_text, 1)));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = row_text;

%input Fitting Parameters
hstart = strcat(xlscol(n+3), int2str(2));
hfinish = strcat(xlscol(n+2+num),
int2str(size(handles.fitted_parameters{i}, 1)+1));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = handles.fitted_parameters{i};

hstart = strcat(xlscol(n+3+handles.DOF), int2str(2));
hfinish = strcat(xlscol(n+3+handles.DOF),
int2str(size(handles.fitted_parameters{i}, 1)+1));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = handles.Integrate{i};

%input RMS
hstart = strcat(xlscol(n+4+handles.DOF), int2str(2));
hfinish = strcat(xlscol(n+4+handles.DOF), int2str(2));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = handles.RMS{i};

if i ~= size(handles.fit_profiles, 2)
    eSheets.Add([], eSheets.Item(eSheets.Count));
end
end
end
eSheets.Add([], eSheets.Item(eSheets.Count));
eSheets.Item(eSheets.Count).Name = 'Results';

num = handles.DOF;
header_text = {'Spectrum Index'};
plen = [];
for i = 1: size(handles.fitted_parameters, 2)
    plen = [plen; size(handles.fitted_parameters{i}, 1)];
end
npeaks = max(plen);
if strcmp(handles.peak_type, 'Voigt') | strcmp(handles.peak_type, 'Pseudo
Voigt')
    m = 5;
else
    m = 4;
end

for i = 1:npeaks
    header_text{end+1} = 'Position';
    header_text{end+1} = 'Intensity';

```

```

header_text{end+1} = 'FWHM';
switch num
case 3
    header_text{end+1} = 'Integrated Intensities';
case 4
    if strcmp(handles.peak_type, 'Pseudo Voigt')
        header_text{end+1} = 'Eta';
        header_text{end+1} = 'Integrated Intensities';
    else
        header_text{end} = 'Sigma';
        header_text{end+1} = 'Lambda';
        header_text{end+1} = 'Integrated Intensities';
    end
end

hstart = strcat(xlscol((i-1)*m+2), int2str(1));
hfinish = strcat(xlscol((i-1)*m+2), int2str(1));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = strcat('Peak_', num2str(i));
end
header_text{end+1} = 'RMS';

hstart = strcat(xlscol(1), int2str(2));
hfinish = strcat(xlscol(length(header_text)), int2str(2));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = header_text;

hstart = strcat(xlscol(1), int2str(3));
hfinish = strcat(xlscol(size(handles.Results, 2)),
int2str(size(handles.Results, 1)+2));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
eActivsheetRange.Value = handles.Results;

% Save Workbook
try
    SaveAs(eWorkbook, path);
    disp('Workbook Saved')
catch
    disp('Workbook not saved');
end
eWorkbook.Saved = 1;
e.Visible = 0;
Close(eWorkbook);
Quit(e);
delete(e);
end

% --- Executes on button press in Plot.
function Plot_Callback(hObject, eventdata, handles)
    value = get(handles.Spectra, 'Value')-1;
    handles.Results = update_results(hObject, handles);
    if ~isempty(handles.Results)
        if strcmp(handles.peak_type, 'Voigt') | strcmp(handles.peak_type, 'Pseudo
Voigt')
            m = 5;

```

```

else
    m = 4;
end
lim = size(handles.Results, 2)-1;
fig1 = figure('Name', 'Tiled Plot','NumberTitle','off', 'Position',[0 0
1400 750]);
set(gcf, 'Color', 'w');
tiledlayout(3, 5, 'TileSpacing','Compact');
ax1 = nexttile(1);
format(ax1);
for i = 2:m:lim-m+1
    plot(handles.Results(:, 1), handles.Results(:, i), 'LineWidth', 2,
'Marker','o')
end
ylabel('X Position');

ax2 = nexttile(2);
format(ax2);
for i = 3:m:lim-m+2
    plot(handles.Results(:, 1), handles.Results(:, i), 'LineWidth', 2,
'Marker','o')
end
ylabel('Intensity');

ax3 = nexttile(6);
format(ax3);
for i = 4:m:lim-m+3
    plot(handles.Results(:, 1), handles.Results(:, i), 'LineWidth', 2,
'Marker','o')
end
if strcmp(handles.peak_type, 'Voigt')
    ylabel('Sigma');
else
    ylabel('FWHM');
end

ax4 = nexttile(7);
format(ax4);
for i = 5:m:lim-m+4
    plot(handles.Results(:, 1), handles.Results(:, i), 'LineWidth', 2,
'Marker','o')
end
if strcmp(handles.peak_type, 'Voigt')
    ylabel('Lambda');
elseif strcmp(handles.peak_type, 'Pseudo Voigt')
    ylabel('Eta');
else
    ylabel('Integrated Intensities')
end

ax5 = nexttile(11);
format(ax5);
if strcmp(handles.peak_type, 'Voigt') | strcmp(handles.peak_type, 'Pseudo
Voigt')
    for i = 6:m:lim-m+5
        plot(handles.Results(:, 1), handles.Results(:, i), 'LineWidth',
2, 'Marker','o')
    end
end

```

```

        end
        ylabel('Integrated Intensities');
    else
        plot(handles.Results(:, 1), handles.Results(:, end), 'LineWidth', 2,
'Marker', 'o')
        ylabel('RMS')
    end

    if strcmp(handles.peak_type, 'Voigt') | strcmp(handles.peak_type, 'Pseudo
Voigt')
        ax6 = nexttile(12);
        format(ax6);
        plot(handles.Results(:, 1), handles.Results(:, end), 'LineWidth', 2,
'Marker', 'o')
        ylabel('RMS')
    end

    axbig = nexttile(3, [3, 3]);
    format(axbig)
    jetcustom = jet(size(handles.fit_profiles, 2));
    colormap(jet(size(handles.fit_profiles, 2)));
    for i = 1:size(handles.fit_profiles, 2)
        if ~isempty(handles.fit_profiles{i})
            plot3(i*ones(size(handles.fit_profiles{i}, 1), 1),
handles.fit_profiles{i}(:, 1), handles.fit_profiles{i}(:, 3), 'LineWidth', 3,
'Color', jetcustom(i, :))
            plot3(i*ones(size(handles.fit_profiles{i}, 1), 1),
handles.fit_profiles{i}(:, 1), handles.fit_profiles{i}(:, 2), 'LineWidth', 3,
'Color', [0,0,0,0.5])
        end
    end
    ylabel('X Value')
    zlabel('Intensity')
    axbig.View = [-30 30];
    else
        fig2 = figure('NumberTitle','off', 'Position',[0 0 800 600]);
        set(gcf, 'Color', 'w');
        ax10 = axes(fig2);
        format(ax10);
        jetcustom = jet(size(handles.fit_profiles, 2));
        colormap(jet(size(handles.fit_profiles, 2)));
        for i = 1:size(handles.fit_profiles, 2)
            plot3(i*ones(size(handles.working_data{i}, 1), 1),
handles.working_data{i}(:, 1), handles.working_data{i}(:, 2), 'LineWidth', 3,
'Color', jetcustom(i, :))
        end
        ylabel('X Value')
        zlabel('Intensity')
        ax10.View = [90 0];
        c = colorbar;
        c.Ticks = [0,1];
        c.TickLabels = {'1', num2str(size(handles.working_data, 2))};
        c.Label.String = 'Spectrum Number';
        c.Position(3) = 0.025;
        c.Position(1) = ax10.InnerPosition(1)+ax10.InnerPosition(3)-0.05;
    end
end

```

```

function format(ax)
    axes(ax)
    box on
    ax.FontSize = 16.0;
    ax.FontName = 'Arial';
    ax.LineWidth = 2.0;
    pbaspect([1.2 1 1]);
    xlabel('Spectrum Index');
    hold on
end
guidata(hObject, handles);
end

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
    children = get(handles.axes1, 'children');
    delete(children(:));
    children = get(handles.axes2, 'children');
    delete(children(:));
    set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
    set(handles.Pseudo_Voigt_CB, 'Value', 1.0);
    set(handles.Lorentzian_CB, 'Value', 0.0);
    set(handles.Voigt_CB, 'Value', 0.0);
    set(handles.Gaussian_CB, 'Value', 0.0);

    set(handles.Spectra, 'Value', 1);
    handles.Spectra_index = {'Spectra:'};
    set(handles.Spectra, 'String', handles.Spectra_index);
    set(handles.table, 'Data', []);
    set(0, 'DefaultFigureWindowStyle', 'normal');
    hold(handles.axes1, 'on')
    hold(handles.axes2, 'on')
    handles.A = pwd;
    handles.Spectra_index = {'Spectra:'};
    handles.raw_data = {};
    handles.working_data = {};
    handles.bkg = {};

    handles.input_parameters = {};
    handles.fitted_parameters = {};
    handles.min_constraints = {};
    handles.max_constraints = {};
    handles.fit_profiles = {};
    handles.RMS = {};
    handles.Integrate = {};
    handles.Residual = {};
    handles.Results = [];

    handles.FileName = {};
    handles.FileDir = {};
    handles.xlsxname = {};
    handles.FWHM = 5.0;
    handles.peak_type = 'Pseudo Voigt';
    handles.DOF = 4;
    guidata(hObject, handles);
end

```

```

%% List Boxes

% --- Executes on selection change in Spectra.
function Spectra_Callback(hObject, eventdata, handles)
    plot_current(hObject, handles);
    try
        set(handles.table, 'Data',
handles.fitted_parameters{get(handles.Spectra, 'Value')-1})

    catch
        set(handles.table, 'Data', [])
    end

end

% --- Executes during object creation, after setting all properties.
function Spectra_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

%% Entry Boxes
function X_Min_Callback(hObject, eventdata, handles)
value = get(handles.Spectra, 'Value')-1;
for i = 1 : size(handles.working_data, 2)
    val = str2num(get(handles.X_Min, 'String'));
    handles.bkg{i}(handles.bkg{i}(:, 1)<val, :) = [];
    handles.working_data{i}(handles.working_data{i}(:, 1)<val, :) = [];
end
handles.FWHM = 0.1*(max(handles.working_data{value}(:, 1))-
min(handles.working_data{value}(:, 1)));
set(handles.FWHM_DB, 'String', num2str(handles.FWHM));
for i = 1 : size(handles.working_data, 2)
    handles.input_parameters{i} = [];
    handles.fitted_parameters{i} = [];
    handles.min_constraints{i} = [];
    handles.max_constraints{i} = [];
    handles.fit_profiles{i} = [];
    handles.RMS{i} = [];
    handles.Integrate{i} = [];
    handles.Risidual{i} = [];
end
plot_current(hObject, handles)
guidata(hObject, handles);
end

% --- Executes during object creation, after setting all properties.
function X_Min_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

```

```

function X_Max_Callback(hObject, eventdata, handles)
value = get(handles.Spectra, 'Value')-1;
    for i = 1 : size(handles.working_data, 2)
        val = str2num(get(handles.X_Max, 'String'));
        handles.bkg{i}(handles.bkg{i}(:, 1)>val, :) = [];
        handles.working_data{i}(handles.working_data{i}(:, 1)>val, :) = [];

    end
    handles.FWHM = 0.1*(max(handles.working_data{value}(:, 1))-
min(handles.working_data{value}(:, 1)));
    set(handles.FWHM_DB, 'String', num2str(handles.FWHM));
    for i = 1 : size(handles.working_data, 2)
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
        handles.Risidual{i} = [];

    end
    plot_current(hObject, handles)
    guidata(hObject, handles);
end

% --- Executes during object creation, after setting all properties.
function X_Max_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function FWHM_DB_Callback(hObject, eventdata, handles)
    handles.FWHM = str2num(get(handles.FWHM_DB, 'String'));
    guidata(hObject, handles);
end

% --- Executes during object creation, after setting all properties.
function FWHM_DB_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

%% Check Boxes
% --- Executes on button press in Bkg_type.
function Bkg_type_Callback(hObject, eventdata, handles)
end

% --- Executes on button press in Lorentzian_CB.
function Lorentzian_CB_Callback(hObject, eventdata, handles)
    handles.peak_type = 'Lorentzian';
    set(handles.Gaussian_CB, 'Value', 0.0);
    set(handles.Voigt_CB, 'Value', 0.0);
    set(handles.Pseudo_Voigt_CB, 'Value', 0.0);

```

```

handles.DOF = 3;
for i = 1 : size(handles.working_data, 2)
    handles.input_parameters{i} = [];
    handles.fitted_parameters{i} = [];
    handles.min_constraints{i} = [];
    handles.max_constraints{i} = [];
    handles.fit_profiles{i} = [];
    handles.RMS{i} = [];
    handles.Integrate{i} = [];
    handles.Risidual{i} = [];
end
plot_current(hObject, handles)
set(handles.table, 'Data', []);
guidata(hObject, handles);
end

% --- Executes during object creation, after setting all properties.
function Lorentzian_CB_CreateFcn(hObject, eventdata, handles)
end

% --- Executes on button press in Gaussian_CB.
function Gaussian_CB_Callback(hObject, eventdata, handles)
    handles.peak_type = 'Gaussian';
    set(handles.Lorentzian_CB, 'Value', 0.0);
    set(handles.Voigt_CB, 'Value', 0.0);
    set(handles.Pseudo_Voigt_CB, 'Value', 0.0);
    handles.DOF = 3;
    for i = 1 : size(handles.working_data, 2)
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
        handles.Risidual{i} = [];
    end
    plot_current(hObject, handles)
    set(handles.table, 'Data', []);
    guidata(hObject, handles);
end

% --- Executes on button press in Voigt_CB.
function Voigt_CB_Callback(hObject, eventdata, handles)
    handles.peak_type = 'Voigt';
    set(handles.Lorentzian_CB, 'Value', 0.0);
    set(handles.Gaussian_CB, 'Value', 0.0);
    set(handles.Pseudo_Voigt_CB, 'Value', 0.0);
    handles.DOF = 4;
    for i = 1 : size(handles.working_data, 2)
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
    end
end

```



```

        handles.Risidual{i} = [];
    end
    plot_current(hObject, handles)
    set(handles.table, 'Data', []);
    guidata(hObject, handles);
end

% --- Executes on button press in Pseudo_Voigt_CB.
function Pseudo_Voigt_CB_Callback(hObject, eventdata, handles)
    handles.peak_type = 'Pseudo Voigt';
    set(handles.Lorentzian_CB, 'Value', 0.0);
    set(handles.Voigt_CB, 'Value', 0.0);
    set(handles.Gaussian_CB, 'Value', 0.0);
    handles.DOF = 4;
    for i = 1 : size(handles.working_data, 2)
        handles.input_parameters{i} = [];
        handles.fitted_parameters{i} = [];
        handles.min_constraints{i} = [];
        handles.max_constraints{i} = [];
        handles.fit_profiles{i} = [];
        handles.RMS{i} = [];
        handles.Integrate{i} = [];
        handles.Risidual{i} = [];
    end
    plot_current(hObject, handles)
    set(handles.table, 'Data', []);
    guidata(hObject, handles);
end

%% General Functions
%=====Build Lorentzian Peak =====%
function [y] = Create_Lorentzian(x, vars)
    pos = vars(1);
    I = vars(2);
    fwhm = vars(3);
    y = ((fwhm/2)^2) ./ ((x-pos).^2 + (fwhm/2)^2);
    if I >= 0
        y = y*I/max(y);
    else
        y = I*y/max(abs(y));
    end
end

%=====Build Gaussian Peak =====%
function [y] = Create_Gaussian(x, vars)
    pos = vars(1);
    I = vars(2);
    fwhm = vars(3);
    y = exp((-1)*(x-pos).^2/(2*(fwhm/2.35482)^2));
    if I >= 0
        y = y*I/max(y);
    else
        y = I*y/max(abs(y));
    end
end

%=====Build Pseudo Voigt =====%
function [y] = Create_Pseudo_Voigt(x, vars)
    eta = vars(4);

```

```

    I = vars(2);
    y = eta*Create_Lorentzian(x, vars(1:3))+(1-eta)*Create_Gaussian(x,
vars(1:3));
    if I >=0
        y = y*I/max(y);
    else
        y = I*abs(y)/max(abs(y));
    end
end
%=====Build Voigt=====
function [y] = Create_Voigt(x, vars)
    xshift = vars(1);
    I = vars(2);
    sigma = vars(3);
    lambda = vars(4);

    z = ((x-xshift)+1i*lambda)/(sigma*sqrt(2));
    den = (sigma*sqrt(2*pi));

    y = real(fadf(z))/den;
    if I >=0
        y = y*I/max(y);
    else
        y = I*y/max(abs(y));
    end
end

%=====General Build Function=====
function [y] = Build_Function(x, params, peak_type)
    if strcmp(peak_type, 'Gaussian')
        y = zeros(length(x), 1);
        for i =1:size(params, 1)
            y = y + Create_Gaussian(x, params(i, :));
        end
    elseif strcmp(peak_type, 'Lorentzian')
        y = zeros(length(x), 1);
        for i =1:size(params, 1)
            y = y + Create_Lorentzian(x, params(i, :));
        end
    elseif strcmp(peak_type, 'Voigt')
        y = zeros(length(x), 1);
        for i =1:size(params, 1)
            y = y + Create_Voigt(x, params(i, :));
        end
    elseif strcmp(peak_type, 'Pseudo Voigt')
        y = zeros(length(x), 1);
        for i =1:size(params, 1)
            y = y + Create_Pseudo_Voigt(x, params(i, :));
        end
    end
end

function plot_current(hObject, handles)
    axes(handles.axes1)
    value = get(handles.Spectra, 'Value')-1;

```

```

    if value ~= 0
        children = get(handles.axes1, 'children');
        delete(children(:));
        children = get(handles.axes2, 'children');
        delete(children(:));
        axes(handles.axes1);
        plot(handles.working_data{value}(:, 1), handles.working_data{value}(:,
2), 'LineWidth', 2, 'Color', 'k')
        handles.axes1.XTickLabels = [];
        set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
        set(handles.Error, 'String', []);

        if ~isempty(handles.fit_profiles{value})
            for i = 4 : (size(handles.fit_profiles{value}, 2)-2)
                plot(handles.fit_profiles{value}(:, 1),
handles.fit_profiles{value}(:, i), 'LineWidth', 2)
            end
            plot(handles.fit_profiles{value}(:, 1),
handles.fit_profiles{value}(:, 3), 'LineWidth', 2, 'Color', 'g')
            plot(handles.axes2, handles.fit_profiles{value}(:, 1),
handles.fit_profiles{value}(:, end-1), 'LineWidth', 2, 'Color', 'g');
            set(handles.Error, 'String', num2str(handles.RMS{value}));
        end
    end
    guidata(hObject, handles);
end

function [y] = update_results(hObject, handles)
    y = [];
    plen = [];
    for i = 1: size(handles.fitted_parameters, 2)
        plen = [plen; size(handles.fitted_parameters{i}, 1)];
    end
    npeaks = max(plen);

    for i = 1 : size(handles.fitted_parameters, 2)
        if ~isempty(handles.fitted_parameters{i})
            temp = [i];
            for j = 1 : npeaks
                try
                    temp = [temp handles.fitted_parameters{i}(j, :)]
handles.Integrate{i}(j)];
                catch
                    temp = [temp zeros(1, handles.DOF+1)];
                end
            end
            temp = [temp handles.RMS{i}];
            y = [y; temp];
        end
    end
    assignin('base', 'fit_results', y)
end

function [val] = Int_Peak(x, y)
    diff = x(2:end) - x(1:end-1);
    yav = (y(2:end) + y(1:end-1))./2;
    val = sum(yav.*diff);
end

```

end

```
%=====Fadeeva function =====%
function FF = fadf(z)
    ind_neg = imag(z)<0; % if some imag(z) values are negative, then ...
    z(ind_neg) = conj(z(ind_neg)); % ... bring them to the upper-half plane

    FF = zeros(size(z)); % define array

    ind_ext = abs(z)>8; % external indices
    ind_band = ~ind_ext & imag(z)<5*10^-3; % narrow band indices

    FF(~ind_ext & ~ind_band) = ...
        fexp(z(~ind_ext & ~ind_band)); % internal area. This area is ...
        % ... the most difficult for accurate computation
    FF(ind_ext) = contfr(z(ind_ext)); % external area
    FF(ind_band) = smallim(z(ind_band)); % narrow band

    function FE = fexp(z,tauM,maxN) % Fourier expansion approximation,
    ...
        % ... see [1, 2] for more information

        if nargin == 1 % assign default parameters tauM and maxN
            tauM = 12; % the margin value
            maxN = 23; % the upper limit summation integer
        end

        n = 1:maxN; % initiate an array
        aN = 2*sqrt(pi)/tauM*exp(-n.^2*pi^2/tauM^2); % Fourier
coefficients

        z1 = exp(1i*tauM*z); % define first repeating array
        z2 = tauM^2*z.^2; % define second repeating array

        FE = sqrt(pi)/tauM*(1 - z1)./z2; % initiate array FE
        for n = 1:maxN
            FE = FE + (aN(n)*((-1)^n*z1 - 1)./(n^2*pi^2 - z2));
        end
        FE = 1i*tauM^2*z/sqrt(pi).*FE;
    end

    function CF = contfr(z) % the Laplace continued fraction
approximation

        bN = 11; % initial integer
        bN = 1:bN;
        bN = bN/2;

        CF = bN(end)./z; % start computing from the last bN
        for k = 1:length(bN) - 1
            CF = bN(end-k)./(z - CF);
        end
        CF = 1i/sqrt(pi)./(z - CF);
    end

    function SIm = smallim(z) % approximation at small imag(z)
```

```

ind_0 = abs(real(z))<5*1e-3;

% If |Re[z]| < 5*1e-3, then:
SIm(ind_0) = small_z(z(ind_0));

x = real(z); % define the repeating array
ind_poles = false(size(x)); % initiate the array of indices

k = 1; % the counter
while k <= 23
    % These indices are to avoid the poles that can strongly ...
    % ... deteriorate the accuracy in computation
    ind_poles = ind_poles | abs(x - k*pi/12)<1e-4;
    k = k + 1; % just to increment the counter
end

% Else if |Re[z]| >= 5*1e-3, then:
SIm(~ind_0 & ~ind_poles) = narr_band(z(~ind_0 & ~ind_poles));
% -----
--
% Note that the margin value tauM in the line below is taken as
...
% 12.1 instead of the default value 12. This excludes all poles
...
% in computation even if Im[z] -> 0.
SIm(~ind_0 & ind_poles) = narr_band(z(~ind_0 &
ind_poles),12.1,23);
% -----
--

function SZ = small_z(z)

    % This equation improves accuracy near the origin. It is ...
    % obtained by the Maclaurin series expansion.

    % Define the repeating arrays
    zP2=z.^2;
    zP4=zP2.^2;
    zP6=zP2.*zP4;

    SZ = (((6 - 6*zP2 + 3*zP4 - zP6).*(15*sqrt(pi) + ...
        1i*z.*(30 + 10*zP2 + 3*zP4)))/(90*sqrt(pi)));
end

function NB = narr_band(z,tauM,maxN) % the narrow band

    % This is just an alternative representation of the ...
    % equation (14) from [1].

    if nargin == 1 % define default parameters
        tauM = 12; % the margin value
        maxN = 23; % the upper limit summation integer
    end

    n = 1:maxN; % initiate an array

```

```

aN = 2*sqrt(pi)/tauM*exp(-n.^2*pi^2/tauM^2); % The Fourier
...
% ... expansion coefficients

z1 = cos(tauM*z); % define first repeating array
z2 = tauM^2*z.^2; % define second repeating array

NB = 0; % initiate the array NB
for n = 1:maxN
    NB = NB + (aN(n)*((-1)^n*z1 - 1)./(n^2*pi^2 - z2));
end
NB = exp(-z.^2) - 1i*((z1 - 1)./(tauM*z) - ...
    tauM^2*z/sqrt(pi).*NB);
end
end

% Convert for negative imag(z) values
FF(ind_neg) = conj(2*exp(-z(ind_neg).^2) - FF(ind_neg));
end

function b = xlscol(a)
% Version 1.1 - Kevin Crosby

% DATE      VER  NAME      DESCRIPTION
% 07-30-10  1.0  K. Crosby  First Release
% 08-02-10  1.1  K. Crosby  Vectorized loop for numerics.

% Contact: Kevin.L.Crosby@gmail.com
base = 26;
if iscell(a)
    b = cellfun(@xlscol, a, 'UniformOutput', false); % handles mixed case
too
elseif ischar(a)
    if ~isempty(strfind(a, ':')) % i.e. if is a range
        b = cellfun(@xlscol, regexp(a, ':', 'split'));
    else % if isempty(strfind(a, ':')) % i.e. if not a range
        b = a(isletter(a)); % get rid of numbers and symbols
        if isempty(b)
            b = {};
        else % if ~isempty(a);
            b = double(upper(b)) - 64; % convert ASCII to number from 1 to 26
            n = length(b); % number of characters
            b = b * base.^((n-1):-1:0)';
        end % if isempty(a)
    end % if ~isempty(strfind(a, ':')) % i.e. if is a range
elseif isnumeric(a) && numel(a) ~= 1
    b = arrayfun(@xlscol, a, 'UniformOutput', false);
else % if isnumeric(a) && numel(a) == 1
    n = ceil(log(a)/log(base)); % estimate number of digits
    d = cumsum(base.^(0:n+1)); % offset
    n = find(a >= d, 1, 'last'); % actual number of digits
    d = d(n:-1:1); % reverse and shorten
    r = mod(floor((a-d)./base.^(n-1:-1:0)), base) + 1; % modulus
    b = char(r+64); % convert number to ASCII
end % if iscell(a)

% attempt to "flatten" cell, by removing nesting

```

```

    if iscell(b) && (iscell([b{:}]) || isnumeric([b{:}]))
        b = [b{:}];
    end % if iscell(b) && (iscell([b{:}]) || isnumeric([ba{:}]))
end

%% =====Backcor Function =====
function [z,a,it,ord,s,fct] = backcor_multi(n,y,ord,s,fct)

sord = ord;
t_old = s;
% Check arguments
if nargin < 2, error('backcor:NotEnoughInputArguments','Not enough input
arguments'); end
if nargin < 5, [z,a,it,ord,s,fct] = backcorgui(n,y,sord,t_old); return; end %
delete this line if you do not need GUI
if ~isequal(fct,'sh') && ~isequal(fct,'ah') && ~isequal(fct,'stq') &&
~isequal(fct,'atq'),
    error('backcor:UnknownFunction','Unknown function.');
```

end

```

% Rescaling
N = length(n);
[n,i] = sort(n);
y = y(i);
maxy = max(y);
dely = (maxy-min(y))/2;
n = 2 * (n(:)-n(N)) / (n(N)-n(1)) + 1;
y = (y(:)-maxy)/dely + 1;

% Vandermonde matrix
p = 0:ord;
T = repmat(n,1,ord+1) .^ repmat(p,N,1);
Tinv = pinv(T'*T) * T';

% Initialisation (least-squares estimation)
a = Tinv*y;
z = T*a;

% Other variables
alpha = 0.99 * 1/2;      % Scale parameter alpha
it = 0;                  % Iteration number
zp = ones(N,1);          % Previous estimation

% LEGEND
while sum((z-zp).^2)/sum(zp.^2) > 1e-9,

    it = it + 1;          % Iteration number
    zp = z;               % Previous estimation
    res = y - z;          % Residual

    % Estimate d
    if isequal(fct,'sh'),
        d = (res*(2*alpha-1)) .* (abs(res)<s) + (-alpha*2*s-res) .* (res<=-s)
+ (alpha*2*s-res) .* (res>=s);
    elseif isequal(fct,'ah'),
        d = (res*(2*alpha-1)) .* (res<s) + (alpha*2*s-res) .* (res>=s);
    elseif isequal(fct,'stq'),
```

```

        d = (res*(2*alpha-1)) .* (abs(res)<s) - res .* (abs(res)>=s);
    elseif isequal(fct,'atq'),
        d = (res*(2*alpha-1)) .* (res<s) - res .* (res>=s);
    end;

    % Estimate z
    a = Tinv * (y+d);    % Polynomial coefficients a
    z = T*a;             % Polynomial

end;

[~,j] = sort(i);
z = (z(j)-1)*dely + maxy;

a(1) = a(1)-1;
a = a*dely;% + maxy;

end
%=====Backcor GUI =====%
function [z,a,it,ord,s,fct] = backcorgui(n,y,sord,t_old)
z = [];
a = [];
it = [];
%ord = [];
%s = [];
fct = [];

order = sord;
threshold = t_old;
costfunction = 'atq';

% Main window
hwin = figure('Visible','off','Position',[0 0 750
400],'NumberTitle','off','Name','Background Correction',...

'MenuBar','none','ToolBar','figure','Resize','on','ResizeFcn',{@WinResizeFcn}
);
bgclr = get(hwin,'Color');

% Axes
haxes = axes('Units','pixels');

% Buttons OK & Cancel
hok =
uicontrol('Style','pushbutton','String','OK','Position',[600,40,80,25],'Callb
ack',{@OKFcn},'BackgroundColor',bgclr);
hcancel =
uicontrol('Style','pushbutton','String','Cancel','Position',[510,40,80,25],'C
allback',{@CancelFcn},'BackgroundColor',bgclr);

% Cost functions menu
hfctlbl = uicontrol('Style','text','String','Cost
function:','HorizontalAlignment','left','BackgroundColor',bgclr);
hfct =
uicontrol('Style','popupmenu','Value',4,'BackgroundColor','white','Callback',
{@CostFunctionFcn},...
```



```

    'String',{'Symmetric Huber function','Asymmetric Huber
function','Symmetric truncated quadratic','Asymmetric truncated quadratic'}});

% Threshold text
hthresholdlbl =
uicontrol('Style','text','String','Threshold:','HorizontalAlignment','left','
BackgroundColor',bgclr);
hthreshold =
uicontrol('Style','edit','String',num2str(threshold),'BackgroundColor','white
','Callback',{@ThresholdFcn});

% Order slider
horderlbl = uicontrol('Style','text','String','Polynomial
order:','HorizontalAlignment','left','BackgroundColor',bgclr);
horder = uicontrol('Style','slider','SliderStep',[0.05
0.05],'Min',0,'Max',20,'Value',order,'Callback',{@OrderFcn});
horderval =
uicontrol('Style','text','String',num2str(order),'BackgroundColor',bgclr);

% Move the GUI to the center of the screen
movegui(hwin,'center');

% Plot a first estimation
[ztmp,atmp,ittmp,order,threshold,costfunction] =
compute(n,y,order,threshold,costfunction);

% Make the GUI visible
set(hwin,'Visible','on');

% Callback functions

function CancelFcn(source,eventdata)
    % Just close the window
    uiresume(gcf);
    close(hwin);
end

function OKFcn(source,eventdata)
    % Return the current estimation and close the window
    z = ztmp;
    a = atmp;
    it = ittmp;
    ord = order;
    s = threshold;
    fct = costfunction;
    uiresume(gcf);
    close(hwin);
end

function CostFunctionFcn(source,eventdata)
    % Change cost function
    cf = get(hfct,'Value');
    if cf == 1,
        costfunction = 'sh';
    elseif cf == 2,
        costfunction = 'ah';
    elseif cf == 3,

```

```

        costfunction = 'stq';
    elseif cf == 4,
        costfunction = 'atq';
    end
    [ztmp,atmp,ittmp,ord,s,fct] =
compute(n,y,order,threshold,costfunction);
end

function OrderFcn(source,eventdata)
    % Change order
    order = get(horder,'Value');
    set(horderval,'String',num2str(order));
    [ztmp,atmp,ittmp,ord,s,fct] =
compute(n,y,order,threshold,costfunction);
end

function ThresholdFcn(source,eventdata)
    % Change threshold
    threshold = get(hthreshold,'String');
    threshold = str2double(threshold);
    [ztmp,atmp,ittmp,ord,s,fct] =
compute(n,y,order,threshold,costfunction);
end

function [ztmp,atmp,ittmp,order,threshold,costfunction] =
compute(n,y,order,threshold,costfunction)
    % Compute and plot an estimation (need to sort the data)
    [ztmp,atmp,ittmp,order,threshold,costfunction] =
backcor_multi(n,y,order,threshold,costfunction);
    [~,i] = sort(n);
    plot(haxes, n(i),y(i),'b-',n(i),ztmp(i),'r-');
end

function WinResizeFcn(source,eventdata)
    % Resize the window
    pos = get(hwin,'Position');
    w = pos(3);
    h = pos(4);
    if w>400 && h>100,
        set(haxes,'Position',[40,40,w-320,h-70]);
    end;
    set(hok,'Position',[w-90,30,80,25]);
    set(hcancel,'Position',[w-180,30,80,25]);
    set(hfctlbl,'Position',[w-240,h-30,220,20]);
    set(hfct,'Position',[w-240,h-50,220,25]);
    set(hthresholdlbl,'Position',[w-240,h-80,220,20]);
    set(hthreshold,'Position',[w-240,h-100,220,20]);
    set(horderlbl,'Position',[w-240,h-130,220,20]);
    set(horder,'Position',[w-210,h-150,190,20]);
    set(horderval,'Position',[w-240,h-150,20,20]);
end
uiwait(gcf);
end

```

Users Guide

The above program is a general spectral fitting utility written with the intent of deconvoluting positive and negative going peaks of an arbitrary dataset using a set of standard peak shapes. The program is also designed to allow the user to apply one exemplary fit to a series of spectra to monitor changes to spectral features across a collection of multiple data sets. The program was developed and implemented in MATLAB and can be run from the command line by calling `Peak_Fit`. Importantly the user must have both `Peak_Fit.m` and `Peak_Fit.fig` in the active directory when calling the function. Likewise, this program was developed for MATLAB 2021 b (it should run on newer and slightly older versions although this is not guaranteed) and requires both the “Optimization,” and “Image Analysis” toolboxes. On startup the user is presented with an interface displayed in Figure H1. The interface contains a series of buttons and checkboxes to load new data into active memory as well as to visualize, process, and export spectral data and accompanying spectrum fits.

To load data into the programs working memory the user can select the “Load Data,” button (object 2. In Figure H1) which will provide them with a windows prompt to select one, or multiple individual datasets. Accepted datasets are as follows:

1. Excel workbooks containing a single x column followed by an arbitrary number of y columns. Importantly all data must be located in sheet 1 of the excel workbook. Data on subsequent sheets will not be read.
2. Comma separated lists consisting of a single x column and single y column (.txt).
3. ASCII files of the same form mentioned in point 2.

To load multiple datasets at once the user must select each dataset they wish to load.

Additionally, multiple different types of datasets can be loaded into memory (for instance a multi-column excel workbook and several CSV files) however this must be done sequentially.

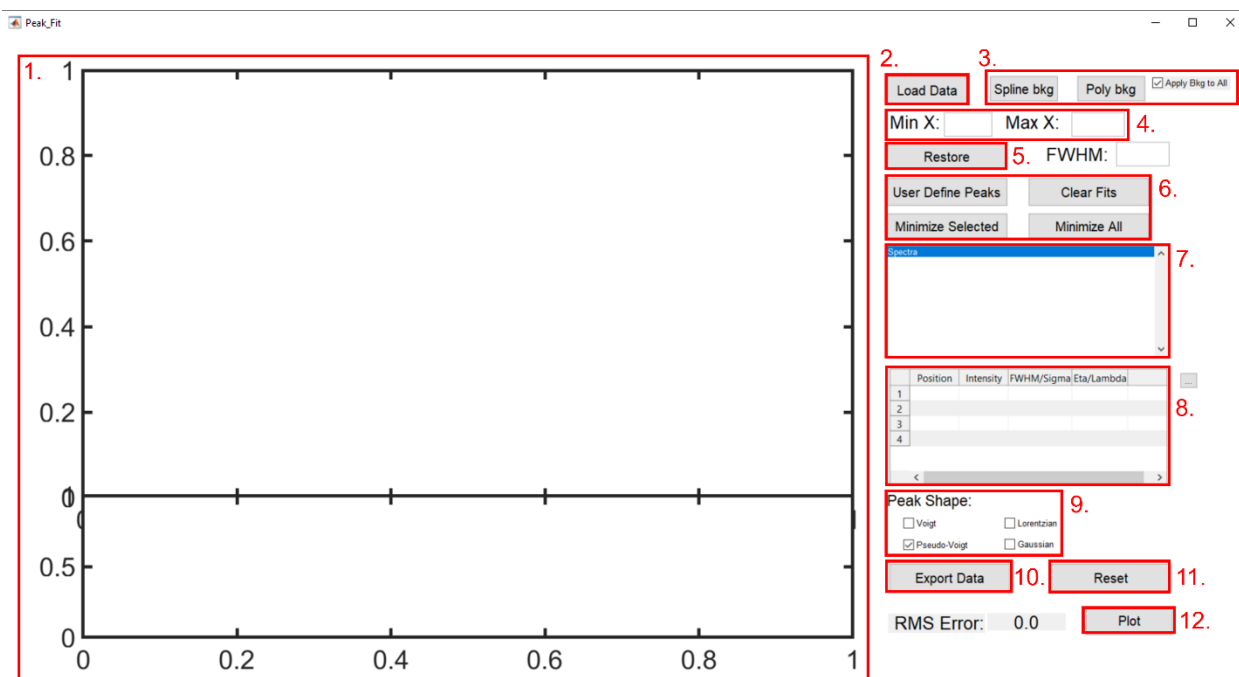


Figure H1. On startup the user is presented with the following interface. Components of the interface are as follows:

1. Main Plotting window
2. Button to load data from an external source
3. Background subtraction utilities including a spline-based background subtraction and a protocol based on fitting data minima to an n^{th} order polynomial.
4. Abscissa limits for plotting and analysis.
5. Restore button to undo any background subtraction, spectrum fitting, or changes to the Abscissa without removing data from active memory.
6. Main fitting functions allowing the user to define the location of spectral features, fit a single spectrum, fit all spectra currently loaded into memory, and remove fits from all spectra in the collection.
7. Spectra list
8. Peak list which includes the position, intensity, full width at half maximum, and eta parameter for individual peaks used to fit spectral data.
9. Check boxes allowing the user to define the peak profiles used to fit spectral data
10. Button to export a consolidated excel workbook with all the information associated with the data fits.

11. Reset button to clear all raw data and data fits from working memory.
12. Default plotting utility to help visualize both the quality of spectral fits, and to visualize changes across different spectra.

After data has been loaded into the programs working memory the user is provided with the interface presented in Figure H2 wherein the spectrum list has been populated and the first dataset of the list displayed in the plotting window. The user can select a spectrum to view either by clicking on the appropriate entry in the list or by using the up and down arrow keys to display each spectrum sequentially. It is often useful to view the raw data in a consolidated form to help locate both regions of interest and features that change from spectrum to spectrum. This is accomplished by pressing the “Plot,” button (object 12. In figure H1) which will overlay each individual spectrum color coded by their index in the list as shown in Figure H3. This plot window is accessible for plotting raw and background subtracted data as long as there have not been any data fits imposed on the raw data. If any data fits are present in memory the plot function operates slightly differently as will be detailed later.

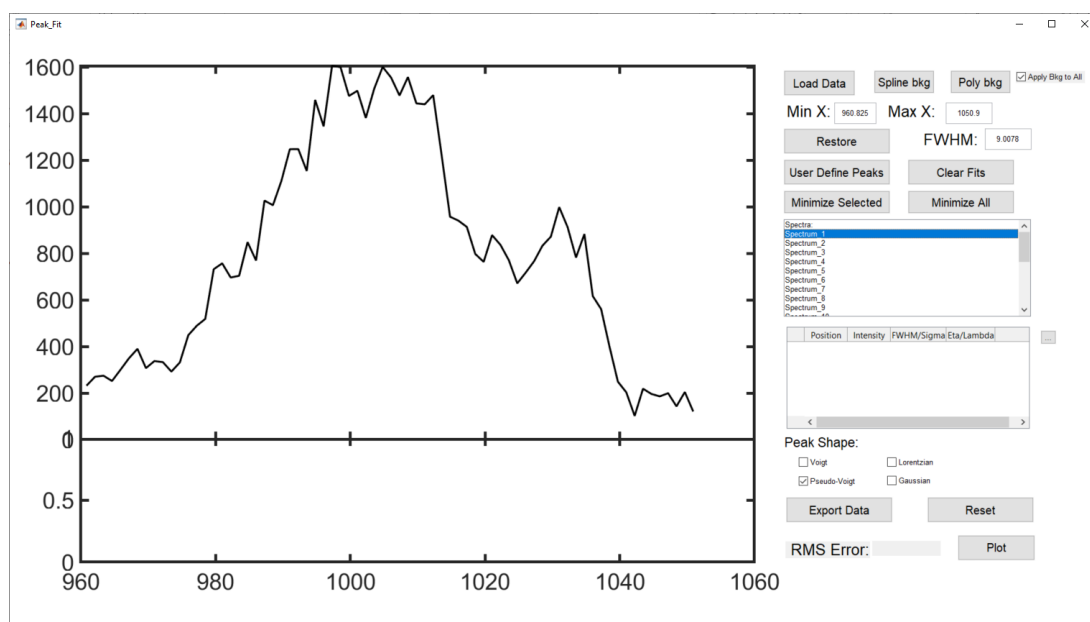


Figure H2. Program window after multiple datasets have been loaded into active memory. Note that the spectrum list is now populated, and the first dataset is displayed in the plotting window.

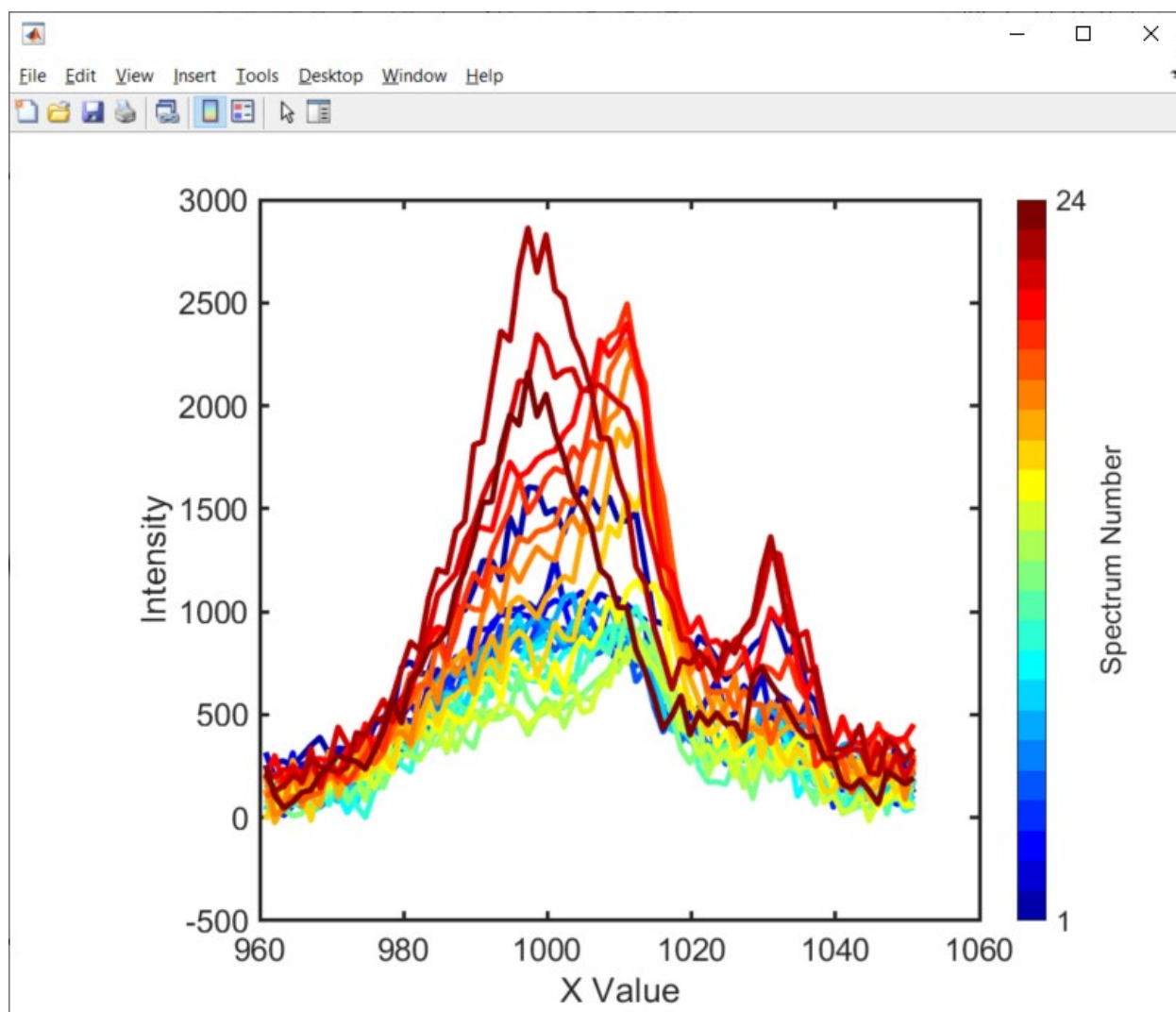


Figure H3. Consolidated overlay of raw spectral data. Note: the plot generated by pressing the “Plot,” button is formatted as a 3-d plot by default and can, therefore be rotated in MATLAB to aid in visualizing raw data.

The plot functionality likewise accounts for any changes to the raw data including changes to the x-range (caused by editing the values in the text boxes in 4. In figure H1) or any background subtractions that have been applied to the raw data. At present, there are two different background subtraction protocols implemented in the program: a spline-based interpolation method, and a polynomial background subtraction. Similarly, a radial checkbox has been implemented next to the background subtraction call buttons labeled “Apply Bkg to All.” When selected this will apply the same background subtraction protocol to each spectrum in the

spectral list. Importantly, it does not apply precisely the same numeric background subtraction. Rather it applies the given subtraction protocol; to use a concrete example if a 4th order polynomial subtraction is selected the “apply to all,” check box will perform a unique optimization of the polynomial to each spectrum in the list and subtract the background automatically. In instances where the background is complex, or great care is desired, it is recommended that the user deselect the “apply to all,” function to retain control of the specific background subtraction applied to each dataset.

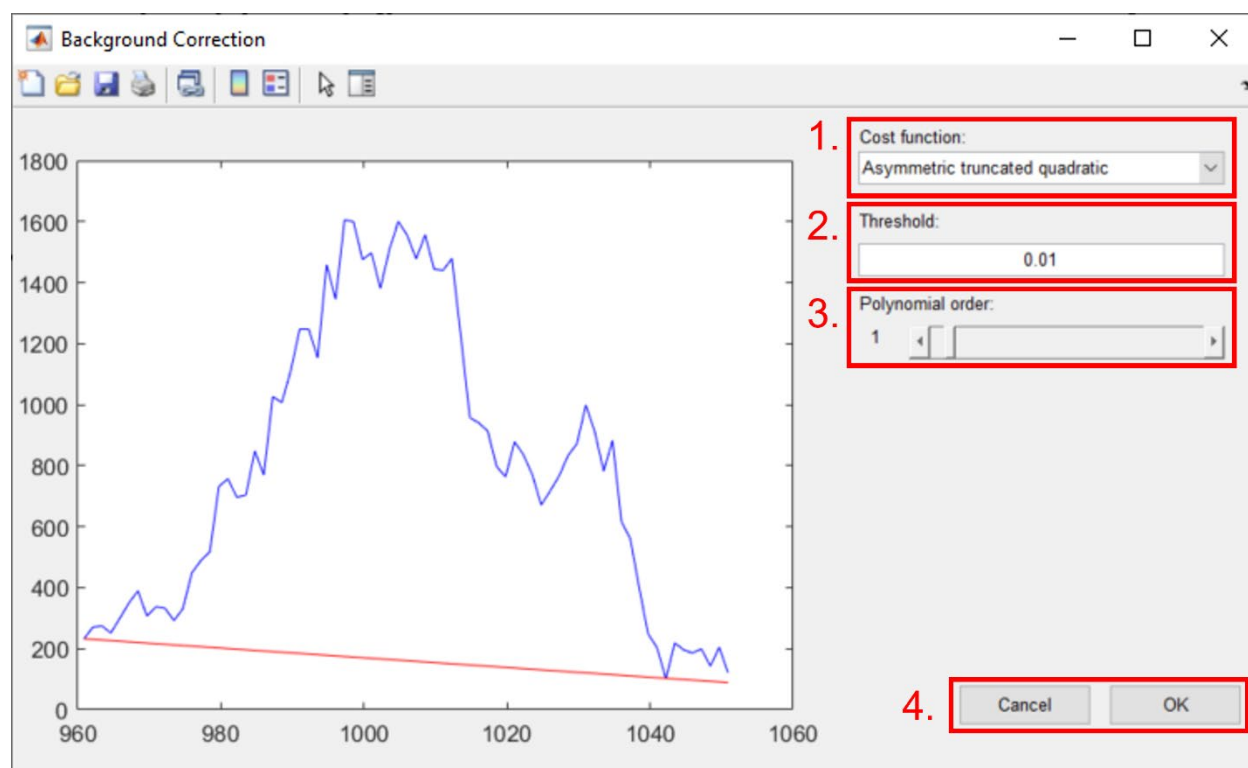


Figure H4. Polynomial background subtraction window. Components of this window include:

1. A drop-down menu to select the desired cost function
2. A text box to adjust the thresholding, and the for the stringency, of the cost function
3. The polynomial order used for subtraction (Warning polynomials greater than order 18 often encounter stability issues)
4. Cancel and accept buttons to either close the window or apply the selected background subtraction to the dataset.

The polynomial background subtraction is a slightly modified version of the “Backcor” background subtraction developed by Vincent Mazet. It applies an n^{th} order polynomial background which is fit by an appropriate cost function to the minima of a given dataset. The window presented to the user on selecting the polynomial subtraction is displayed in Figure H4. The main parts of the window include the desired cost function, the threshold dictating how stringent the cost function is, the polynomial order, and a cancel or accept function to apply the background subtraction to the given dataset.

The other background subtraction protocol that has been implemented is based on n^{th} order spline interpolation. Selecting the “Spline Bkg,” will open the window displayed in Figure H5 allowing the user to specify the knots of the spline in question along with the order of the spline interpolant. Importantly, during testing it was noticed that splines of order 3 or higher tend to display significant stability issues, thus splines of first or second order are recommended. Background fitting of this type can be accomplished either by selecting the desired background points or by selecting the “Endpoint Subtraction” function which will simply create a linear interpolation from the first to the last datapoint defined by the given dataset. If “Select Knots,” is pressed instead, the user will be asked to define points along the dataset that are associated with the background. Pressing the “Enter,” key will lock these points into the interpolation program. Following the computation of the desired spline the program will display the spline as a red trace, with the user selected knots presented as purple x’s. Figure H6 presents two examples of backgrounds generated from the spline background tool. E6a presents a first order spline interpolant across a series of arbitrary points while E6b presents a similar interpolant using a second order spline.

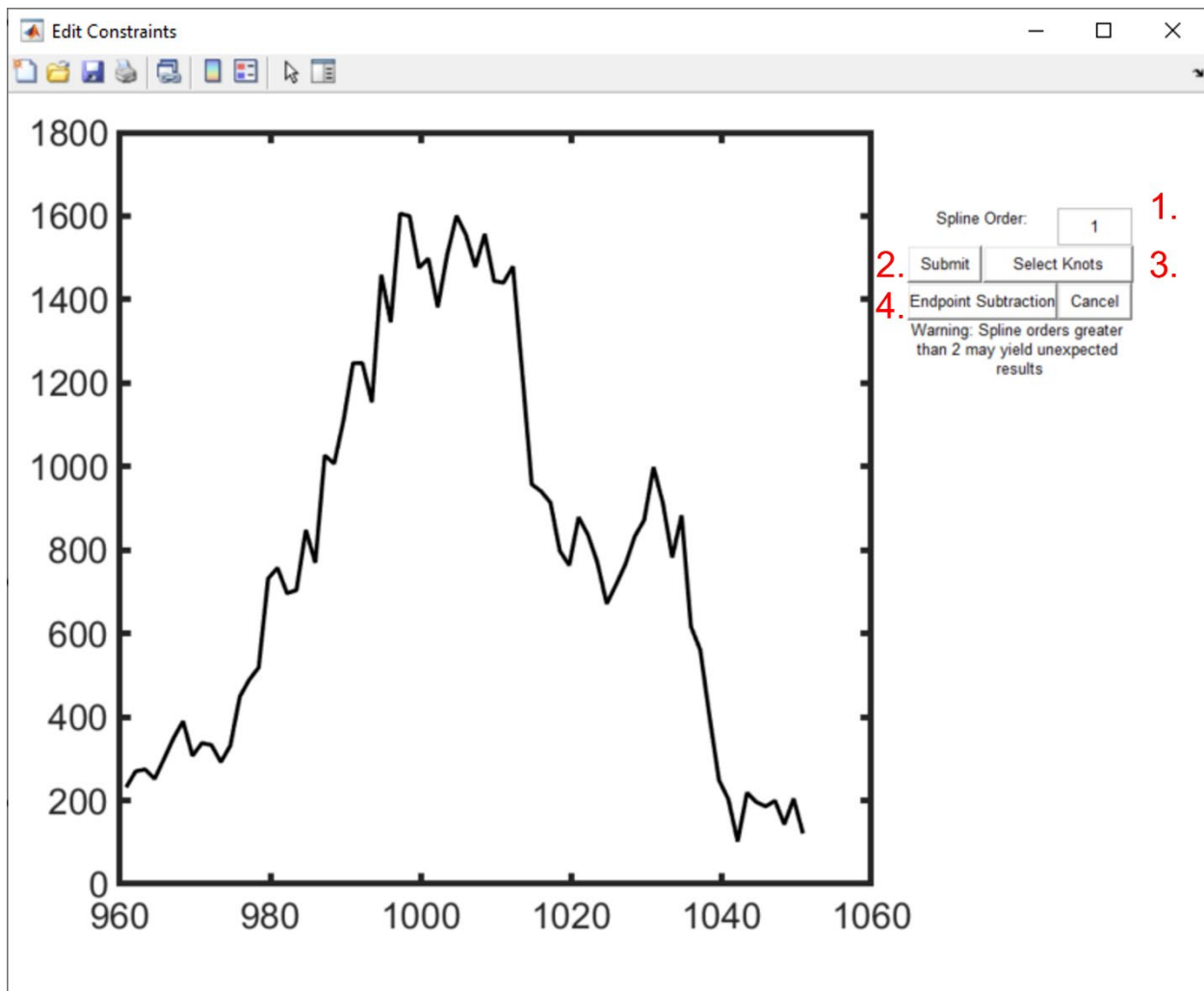


Figure H5. Window to allow the user to apply and submit a spline-based background subtraction to the selected dataset. Components of this window include:

1. The spline order describing the polynomial applied between individual knots. Note: for a given spline order n , the resulting spline interpolant will be $(n-1)^{\text{th}}$ order differentiable.
2. Submit button to apply the selected background subtraction to the dataset in question
3. Button to allow the user to select the knots which will define the spline interpolation
4. Endpoint subtraction function which simply applies a first order spline interpolant between the first and last datapoints to flatten these points to zero.

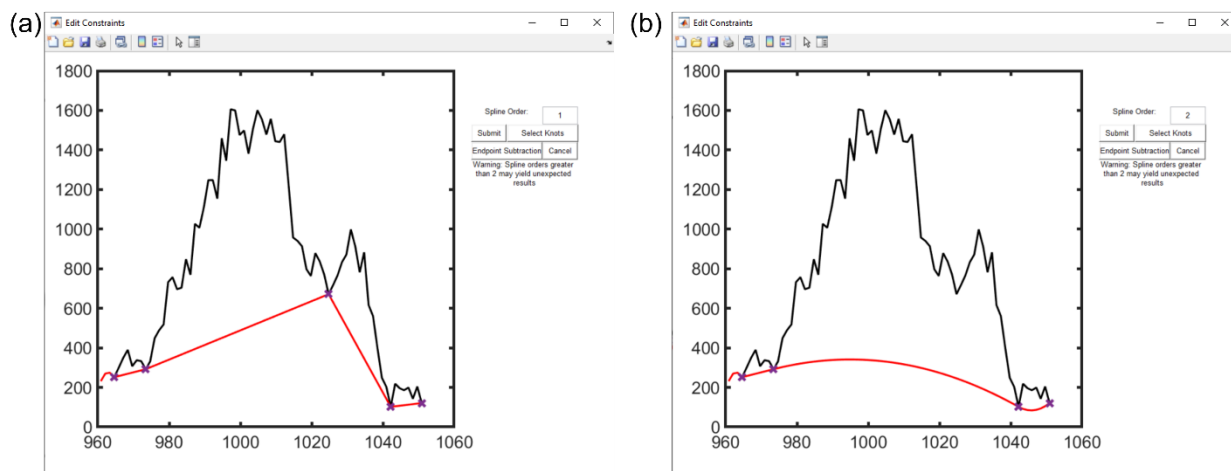


Figure H6. (a) demonstrates a first order spline interpolant (spline is presented in red with purple x's representing the knots used to generate the background) while (b) presents a similar background generated from a second order spline interpolant.

It is recommended that an appropriate background subtraction be applied to the datasets prior to data fitting as, by default, the fitting protocol presented here does not apply a vertical shift to fitting data. Thus, data whose background is not, to a reasonable extent, set to 0 will pose difficulties for fitting. This limitation is addressable by supplying an additional offset variable to the Nelder Mead minimization protocol, however at time of publication such an offset has not been implemented. After a reasonable background subtraction has been applied to all datasets under investigation the user may begin to define the locations of spectral features they wish to fit. This process, which is detailed in Figure H7, is accomplished by selecting the “User Define Peaks,” button which, very much like the spline background function, allows the user to select the peak positions that will be used to fit the data. After selecting the locations of all spectral features of interest the user can press the “Enter” key to lock these positions into the program’s memory (note that peak intensities can be positive or negative). After pressing “Enter,” preliminary peaks will be added to the plot window to give the user a chance to verify the location of the peaks they have selected. Importantly these peaks have not been fit, and their heights and widths are subject to change during the fitting protocol.

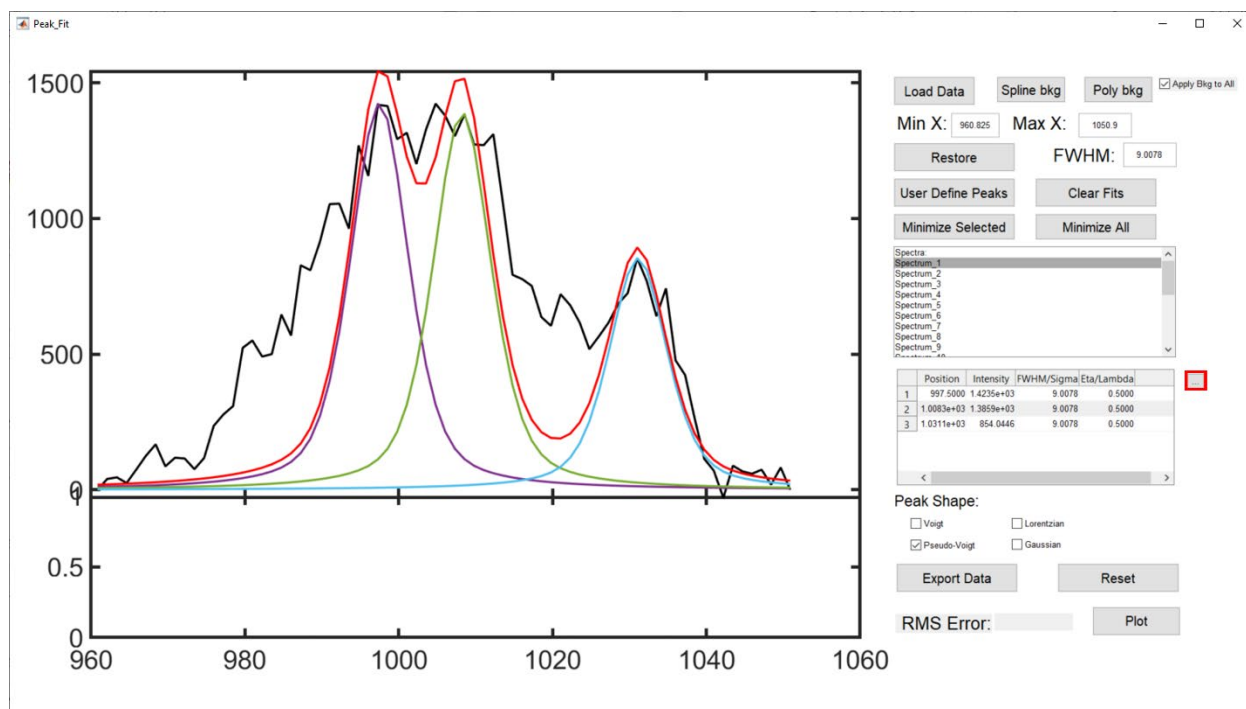


Figure H7. Plot window showing the preliminary locations of spectral features defined by the user prior to the fitting protocol. The selection box highlighted in red can be used prior to fitting the individual dataset to constrain the minimization depending on the specific application desired by the user. The window presented to the user upon selecting this function is presented in Figure H8.

Once the user has defined the locations of suspected spectral features, they may further constrain the subsequent optimization depending on the desired application. In particular, it is often necessary to limit the position, intensity or full width at half maximum of features to fall within the constraints of physical reality. When fitting an individual spectrum this can be accomplished by selecting the grey box next to the peak location table (which should automatically populate after peak locations have been defined). Selecting this box opens a separate window, an example of which is presented in Figure H8. Within this window the user can define the constraints applied to the given minimization prior to fitting the data.

	X min	X max	Y min	Y max	fwhm min	fwhm max	Eta min	Eta max
995.2816	977.266	1013.3	0	1689.9	0	99.0854	0	1
1008.1655	990.15	1026.18	0	1752.67	0	99.0854	0	1
1029.4966	1011.4811	1047.51	0	1313.28	0	99.0854	0	1

Submit Cancel

Figure H8. Window allowing the user to constrain the parameters used to fit the selected spectrum. Pressing the submit button applies these constraints. Pressing cancel closes the window without applying the constraints.

Once the user has defined the peak positions and any constraints to their satisfaction, they can fit the selected peaks to the experimental data by pressing the “Minimize Selected,” button. This button submits the data and the constraints to a Nelder Mead simplex minimization algorithm which aims to minimize the discrepancy between the fitted spectrum (constructed from a linear combination of the peaks defined by the user) to the experimental data. Upon completion of the minimization protocol the plot window will automatically update to something similar to that presented in Figure H9. Specifically, the fitted spectrum will be overlayed on top of the experimental data along with a deconvolution of the fundamental peaks used to define the spectral fit. Keep in mind that this program is constrained by the number of peaks defined by the user. It will never add or remove peaks on its own, however it may set the intensity of certain peaks to zero if it benefits the quality of the fit. After fitting, the deviation between the fit and the experiment is provided in the lower plot window and the total root mean square deviation is presented in the text box on the right-hand side of the window. Note: this is presented as an absolute error rather than a percent error. The data fit generated by the program is assigned to the spectrum to which it was applied. This allows the user to easily select a different spectrum for fitting and lets them return to earlier fits if desired. Likewise, if the fit is not satisfactory the user can simply redefine the peaks or change the constraints and retry the fitting protocol until a

satisfactory fit to the experimental data is generated. Finally, it is important to note that while there is some element of stochastic minima searching inherent to the Nelder Mead algorithm, the practical application of this fitting methodology was found to be effectively deterministic insofar as data fits using the same starting points and constraints invariably reach the same fit.

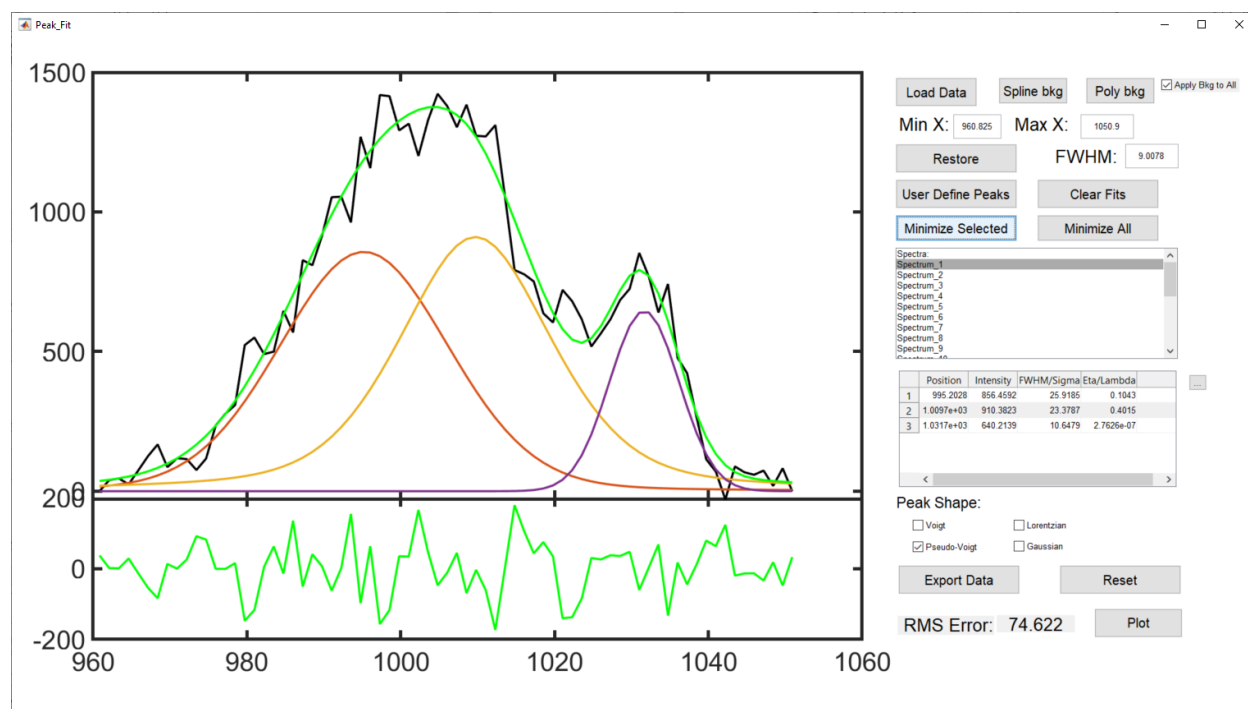


Figure H9. Fitting window following the fitting protocol as applied to a single dataset. Upon completion of the minimization algorithm the program will update the plot window as well as the information contained within the peak position table on the right-hand side.

As mentioned in the introductory comments about this program, it was written with the goal of expediting multi-spectrum fitting to monitor time dependent phenomena. This is accomplished using the same minimization protocol on each dataset within the spectra list. In practice this protocol uses the peaks defined by a single exemplary data fit as the starting point to fit each spectrum in the list. This function requires that at least one spectrum from the list be fit. Likewise, this spectrum must be actively selected to alert the fitting protocol to which fit it should use as its template. In practice it is recommended that the user picks a spectrum with as

many peaks present as possible to use as the template. This recommendation stems from the fact that the fitting protocol cannot add new peaks to the fit automatically; however, it can decrease the intensity of certain peaks to zero when appropriate. Once all of these conditions have been met, however, and the user is satisfied with their choice of template spectra, they may select the “Minimize all,” button, allowing the program to attempt to fit all spectra in the list to the template defined by the active spectrum window.

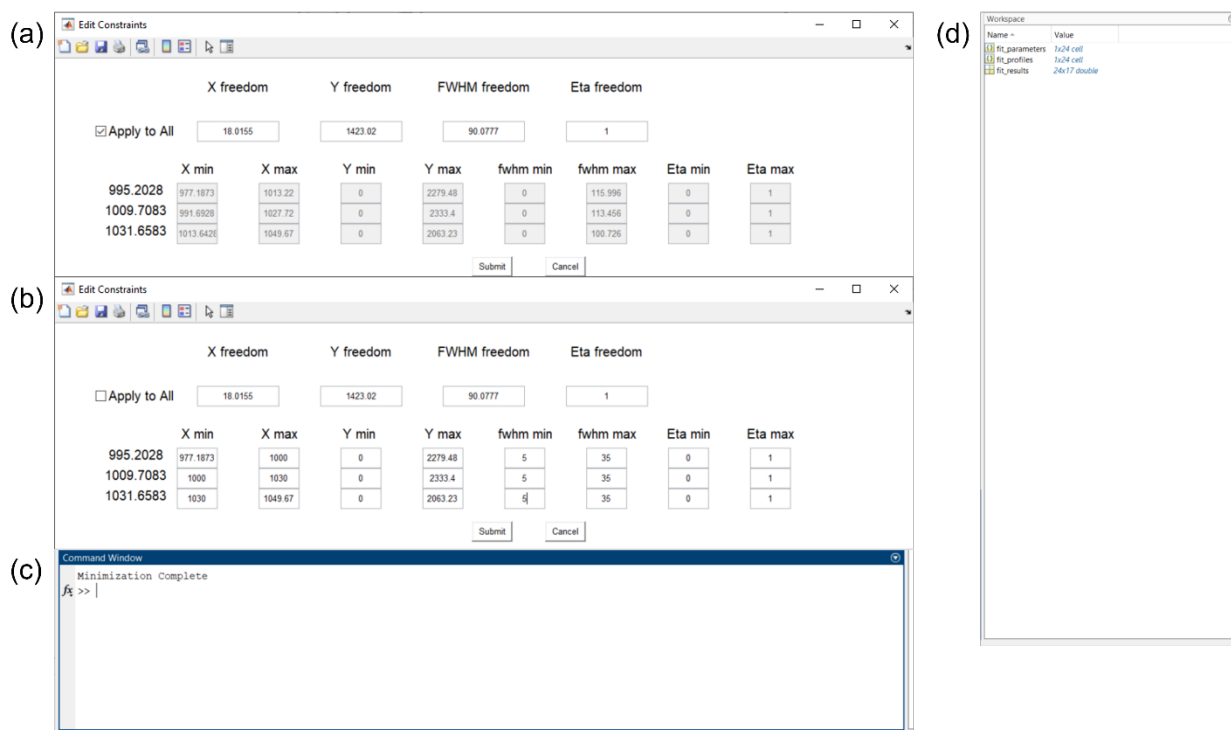


Figure H10. (a) Automatic constraints window opened prior to the application of multi-spectrum fitting. (b) the same window if the user chooses to de-select “apply to all” and edits the constraints manually. (c) Completion message exported to the command line after the successful fitting of all spectra in the spectrum list. (d) MATLAB variable explorer wherein all of the data used to fit each spectrum are presented.

Prior to fitting any of the spectra in the list the program will prompt the user with a constraint window to define the constraints within which the minimization must operate (Figure H10a,b). By default, the constraints are chosen automatically based on the template data fit. The user can easily change these constraints by deselecting the “apply to all” checkbox and editing

the text boxes individually. As discussed earlier this allows the user full freedom to constrain spectral features to regions that make sense within physical reality. Upon completion of all minimizations the software will print a completion message to the MATLAB command line and will populate the workspace with the variables used to generate fits as well as the data fits themselves. These variables are helpful if additional data processing in the MATLAB environment is desired. To verify the quality of the data fits in question the user is encouraged to look through each of the spectra contained within the spectrum list, each of which has now been edited to include the fit and the peak deconvolution applied to it.

Should the automatic data fits prove unsatisfactory the user can simply select the desired spectrum they wish to use as a fitting template and try again by selecting “Minimize All,” possibly using different constraints to arrive at data fits that align with the quality desired. Once the user is satisfied with the data fits achieved by the program, they can visualize changes to these fits by selecting the “Plot,” button. The plot function will now present all of the variables used to fit experimental data plotted against the spectrum index as presented in Figure H11. Finally, if the user wishes to export the data fits and parameters used to define them, they can select the “Export Data,” button which will output an excel document to the same directory from which the data was obtained.

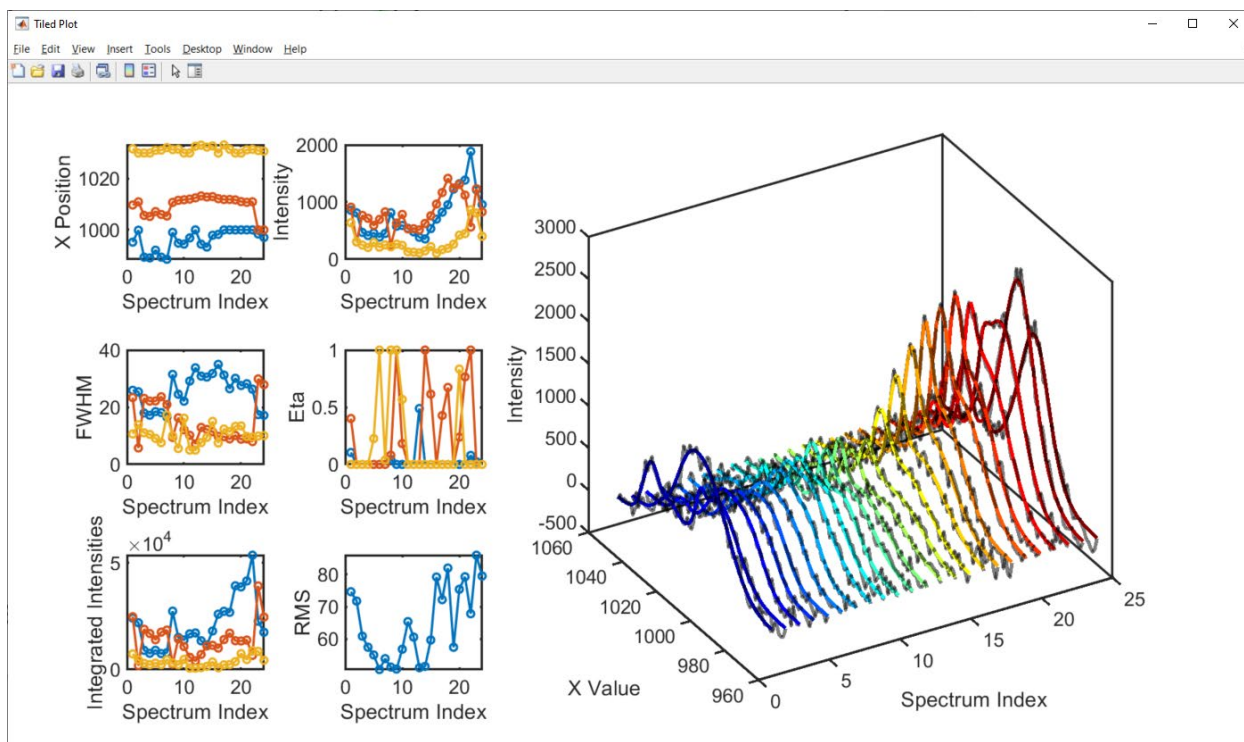


Figure H11. Example of the plot function used to visualize changes to fitting variables as a function of the spectrum index. This function provides the position, intensity, width, and integrated intensity of each peak as well as the total RMS deviation associated with each fit. This function also presents all of the fits generated on the right-hand side of the figure as colored traces. The raw data is also included as semitransparent gray traces overlaid with the data fits.

References:

(1) Vincent Mazet (2022). Background correction

(<https://www.mathworks.com/matlabcentral/fileexchange/27429-background-correction>),

MATLAB Central File Exchange. Retrieved February 27, 2022.

Appendix I: Program to Fit Heterogeneous Diffraction Data to a Sum of Lorentzians (MATLAB):

```
function varargout = Fit_Diffraction(varargin)
% FIT_DIFFRACTION MATLAB code for Fit_Diffraction.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @Fit_Diffraction_OpeningFcn, ...
                  'gui_OutputFcn',  @Fit_Diffraction_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before Fit_Diffraction is made visible.
function Fit_Diffraction_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to Fit_Diffraction (see VARARGIN)

% Choose default command line output for Fit_Diffraction
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fit_Diffraction wait for user response (see UIRESUME)
% uiwait(handles.figure1);
set(0,'DefaultFigureWindowStyle','normal');
handles.A = pwd;
handles.Filenames = {'Filenames:'};
handles.Standard_Data = {};
handles.Experimental_Data = [];
handles.Prepped_Experimental_Data = [];
handles.Fitting_Profiles = {};
handles.FWHM = 0.3;
parent_axes = handles.axes1;
handles.parent_figure = gcf;
handles.parent_axes = parent_axes;
hold(handles.parent_axes, 'on')
hold(handles.axes2, 'on')
handles.vertical_offset = [];
handles.scaling_factor = [];
handles.computed_background = [];
handles.start_parameters = [];
```

```

handles.simulated_spectrum = [];
handles.full_simulated_spectrum = [];
handles.filedir = [];
handles.xlsxname = [];
handles.T_Data = [];
handles.Fitting_Error = [];

set(handles.axes1, 'XTickLabel', []);
ylabel(handles.axes1, 'Normalized Intensity (a.u.)');
xlabel(handles.axes2, '2\theta (degrees)');
set(handles.axes2, 'YAxisLocation', 'right');
ylabel(handles.axes2, 'Deviation');
%handles.Min_FWHM = 0.1;
%handles.Max_FWHM = 1.0;

guidata(hObject, handles);
end

% --- Outputs from this function are returned to the command line.
function varargout = Fit_Diffraction_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
end

% --- Executes on button press in Add_Species.
function Add_Species_Callback(hObject, eventdata, handles)
%Obtain Filename
[Filename, filedir] = uigetfile('*..*', 'All Files (*.*)', 'MultiSelect',
'on');
if isempty(Filename)
    return
end

if ~iscell(Filename)
    z = 1;
else
    z = length(Filename);
end

for i = 1 : z
    if z == 1
        Fullname = fullfile(filedir, Filename);
        handles.Filenames{size(handles.Filenames, 2)+1} = Filename;
    else
        Fullname = fullfile(filedir, Filename{i});
        handles.Filenames{size(handles.Filenames, 2)+1} = Filename{i};
    end
    set(handles.Species, 'String', handles.Filenames);

    %Read Jade Datafile
    try
        text = fileread(Fullname);
        TextAsCells = regexp(text, '\n', 'split');
        mask = ~cellfun(@isempty, strfind(TextAsCells, '2-Theta'));
        startline = find(mask);
        TextAsCells = TextAsCells';
    end
end

```

```

        TextAsCells = {TextAsCells{startline+1:end}}';
        Dtemp = regexp(TextAsCells, ' ', 'split');
        Data = [];
        for i = 1:size(Dtemp, 1)
            Data = [Data;
str2double(Dtemp{i}(~cellfun('isempty',Dtemp{i}))))];
        end
        Data = [Data(:, 1) Data(:, 3)];
        Data = Data((min(handles.Experimental_Data(:, 1))<Data(:,
1))&(Data(:, 1)<max(handles.Experimental_Data(:, 1))), :);
        Data(:, 2) = Data(:, 2)/max(Data(:, 2));
    catch
        Data = importdata(Fullname);
        Data = [Data.data(:, 8) Data.data(:, 9)];
        Data(Data(:, 2)==0, :) = [];
        Data = Data((min(handles.Experimental_Data(:, 1))<Data(:,
1))&(Data(:, 1)<max(handles.Experimental_Data(:, 1))), :);
        Data(:, 2) = Data(:, 2)/max(Data(:, 2));
    end

    handles.Standard_Data{size(handles.Standard_Data, 2)+1} = Data;

    if get(handles.Lorentzian_CB, 'value') == 1.0
        y = Create_Lorentzian(handles.Prepped_Experimental_Data(:, 1), Data,
handles.FWHM);
    else
        y = Create_Gaussian(handles.Prepped_Experimental_Data(:, 1), Data,
handles.FWHM);
    end

    axes(handles.parent_axes);
    plot(handles.Prepped_Experimental_Data(:, 1), y, 'LineWidth', 2);
    handles.start_parameters = [handles.start_parameters; handles.FWHM; 0.3];
end

children = get(handles.axes2, 'children');
delete(children(:));

guidata(hObject, handles);
end

% --- Executes on button press in Minimize.
function Minimize_Callback(hObject, eventdata, handles)
%Define Cost Function to Optimize
handles.Fitting_Profiles = [];
children = get(handles.axes2, 'children');
delete(children(:));
function [cost] = Cost_Function(Params)
nspecies = uint8(length(Params)/2);
Simulated_Data = zeros(length(handles.Prepped_Experimental_Data), 1);
if get(handles.Lorentzian_CB, 'value') == 1
    for i = 1:nspecies
        temp = Create_Lorentzian(handles.Prepped_Experimental_Data(:, 1),
handles.Standard_Data{i}, Params(i*2-1));
        Simulated_Data = Simulated_Data + Params(2*i)*temp;
    end
else

```

```

        for i = 1:nspecies
            temp = Create_Gaussian(handles.Prepped_Experimental_Data(:, 1),
handles.Standard_Data{i}, Params(i*2-1));
            Simulated_Data = Simulated_Data + Params(2*i)*temp;
        end
    end
    cost = sqrt(sum((handles.Prepped_Experimental_Data(:, 2) -
Simulated_Data).^2)/length(handles.Prepped_Experimental_Data(:,2)));
end

%cost = Cost_Function(handles.start_parameters)
min_constraints = [];
max_constraints = [];
for j = 1 : length(handles.start_parameters)
    if mod(j,2)==0
        min_constraints = [min_constraints; 0];
        max_constraints = [max_constraints; 1.0];
    else
        min_constraints = [min_constraints; str2double(get(handles.MinFWHM,
'String'))];
        max_constraints = [max_constraints; str2double(get(handles.MaxFWHM,
'String'))];
    end
end
end

Test = fmincon(@Cost_Function, handles.start_parameters, [], [], [], [],
min_constraints, max_constraints);

function [y] = Build_Function(Params)
nspecies = uint8(length(Params)/2);
y = zeros(length(handles.Prepped_Experimental_Data), 1);
if get(handles.Lorentzian_CB, 'value') == 1
    for i = 1:nspecies
        temp = Create_Lorentzian(handles.Prepped_Experimental_Data(:, 1),
handles.Standard_Data{i}, Params(i*2-1));
        y = y + Params(2*i)*temp;
    end
    handles.simulated_spectrum = y;
else
    for i = 1:nspecies
        temp = Create_Gaussian(handles.Prepped_Experimental_Data(:, 1),
handles.Standard_Data{i}, Params(i*2-1));
        y = y + Params(2*i)*temp;
    end
    handles.simulated_spectrum = y;
end
end
end

simulated_data = Build_Function(Test);
handles.full_simulated_spectrum =
simulated_data*handles.scaling_factor+handles.computed_background+handles.ver
tical_offset;
children = get(handles.axes1, 'children');
delete(children(1:end-1));
%plot(handles.axes1, handles.Prepped_Experimental_Data(:, 1),
handles.Prepped_Experimental_Data(:, 2), 'LineWidth', 2, 'Color', 'k');

```

```

plot(handles.axes1, handles.Prepped_Experimental_Data(:, 1), simulated_data,
'LineWidth', 3, 'Color', 'r');
peak_areas = [];
for j = 1:size(handles.Standard_Data, 2)
    if get(handles.Lorentzian_CB, 'value') == 1
        temp =
Test(2*j)*Create_Lorentzian(handles.Prepped_Experimental_Data(:, 1),
handles.Standard_Data{j}, Test(j*2-1));
    else
        temp = Test(2*j)*Create_Gaussian(handles.Prepped_Experimental_Data(:,
1), handles.Standard_Data{j}, Test(j*2-1));
    end
    plot(handles.axes1, handles.Prepped_Experimental_Data(:, 1), temp,
'LineWidth', 1);
    handles.Fitting_Profiles{end+1} = [handles.Prepped_Experimental_Data(:,
1) temp];
    for k = 2 : size(handles.Fitting_Profiles{j}, 1)
        integral = ((handles.Fitting_Profiles{j}(k, 2) +
handles.Fitting_Profiles{j}(k-1, 2))/2)/(handles.Fitting_Profiles{j}(k, 1) -
handles.Fitting_Profiles{j}(k-1, 1));
    end
    peak_areas = [peak_areas; integral];
end
Ltext = handles.FileNames;
Ltext{1} = {'Data'};
Ltext = [Ltext{1:1}, {'Fit'}, Ltext{2:end}];
legend(handles.axes1, Ltext, 'Box', 'off');
figure
plot(handles.Experimental_Data(:, 1), handles.Experimental_Data(:, 2),
'LineWidth', 2, 'Color', 'k');
hold on
plot(handles.Prepped_Experimental_Data(:, 1),
handles.full_simulated_spectrum, 'LineWidth', 2, 'Color', 'r');

Col_label = {'Species', 'Peak Intensity', 'FWHM', 'Area'};
T_Data = Col_label;
for j = 1:size(handles.Standard_Data, 2)
    T_Data = [T_Data; {handles.FileNames{j+1}, Test(j*2), Test(j*2-1),
peak_areas(j)}];
end

%handles.Fitting_Error = sqrt((handles.Prepped_Experimental_Data(:, 2)-
simulated_data).^2);
handles.Fitting_Error = handles.Prepped_Experimental_Data(:, 2)-
simulated_data;
plot(handles.axes2, handles.Prepped_Experimental_Data(:, 1),
handles.Fitting_Error, 'LineWidth', 1, 'Color', 'g');
line(handles.axes2, get(handles.axes2, 'XLim'), [0 0], 'Color', 'Black',
'LineWidth', 3, 'LineStyle', '--')
handles.T_Data = T_Data;

cost = sqrt(sum((handles.Prepped_Experimental_Data(:, 2) -
simulated_data).^2)/length(handles.Prepped_Experimental_Data(:, 2)));
set(handles.Error, 'String', num2str(cost));
assignin('base', 'T_Data', T_Data);
assignin('base', 'Fitted_Profiles', handles.Fitting_Profiles);
assignin('base', 'Simulated_Spectrum', handles.simulated_spectrum);

```

```

guidata(hObject, handles);
end

% --- Executes on selection change in Species.
function Species_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function Species_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in Lorentzian_CB.
function Lorentzian_CB_Callback(hObject, eventdata, handles)
set(handles.Gaussian_CB, 'value', 0.0);
end

% --- Executes on button press in Gaussian_CB.
function Gaussian_CB_Callback(hObject, eventdata, handles)
set(handles.Lorentzian_CB, 'value', 0.0);
end

function MinFWHM_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function MinFWHM_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function MaxFWHM_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function MaxFWHM_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes during object creation, after setting all properties.
function Lorentzian_CB_CreateFcn(hObject, eventdata, handles)
end

% --- Executes on button press in Load_Data.
function Load_Data_Callback(hObject, eventdata, handles)
[Filename, filedir] = uigetfile('*..*', 'All Files (*.*)');
Fullname = fullfile(filedir, Filename);
handles.filedir = filedir;

```

```

handles.xlsxname = strcat(Filename,'Fitting_Data', '.xlsx');
Exp_Data = table2array(readtable(Fullname));
handles.Experimental_Data = Exp_Data;
bkg = backcor(Exp_Data(:, 1), Exp_Data(:, 2));
handles.computed_background = bkg;
Exp_Data = [Exp_Data(:, 1), Exp_Data(:, 2) - bkg];
handles.vertical_offset = min(Exp_Data(:, 2));
Exp_Data(:, 2) = Exp_Data(:, 2) - min(Exp_Data(:, 2));
handles.scaling_factor = max(Exp_Data(:, 2));
Exp_Data(:, 2) = Exp_Data(:, 2)/max(Exp_Data(:, 2));
handles.Prepped_Experimental_Data = Exp_Data;
axes(handles.parent_axes);
%plot(Exp_Data(:, 1), Exp_Data(:, 2), 'LineWidth',2, 'Color', 'r');
scatter(Exp_Data(:, 1), Exp_Data(:, 2), 100, [0.4940 0.1840 0.5560], 'x');
set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
%set(handles.axes2, 'XTickLabel', get(handles.axes1, 'XTickLabel'));
guidata(hObject, handles);
end

function [y] = Create_Lorentzian(x, pnts, fwhm)
pnts = pnts((min(x)<pnts(:, 1)) & (pnts(:, 1)<max(x)), :);
y = zeros(length(x), 1);
for i = 1 : size(pnts, 1)
    y = y + (pnts(i, 2)*fwhm/2)*((fwhm/2)./(x-pnts(i, 1)).^2+(fwhm/2)^2));
end
y = y/max(y);
end

function [y] = Create_Gaussian(x, pnts, fwhm)
pnts = pnts((min(x)<pnts(:, 1)) & (pnts(:, 1)<max(x)), :);
y = zeros(length(x), 1);
for i = 1 : size(pnts, 1)
    y = y + (pnts(i, 2))*exp((-1)*(x-pnts(i, 1)).^2/(2*(fwhm/2.35482)^2));
end
y = y/max(y);
end

% --- Executes on button press in Remove_Species.
function Remove_Species_Callback(hObject, eventdata, handles)
selection = get(handles.Species, 'Value');
set(handles.Species, 'Value', 1);
handles.Filenames(selection) = [];
set(handles.Species, 'String', handles.Filenames);
handles.Standard_Data(selection - 1) = [];
children = get(handles.parent_axes, 'children');
if isempty(handles.full_simulated_spectrum)
    delete(children(length(children)-selection+1));
else
    handles.Fitting_Profiles(selection - 1) = [];
    delete(children(end-1));
    delete(children(length(children)-selection));
    %delete(children(length(children)-selection+1));
end
handles.start_parameters((selection-1)*2) = [];
handles.start_parameters((selection-1)*2-1) = [];
children = get(handles.axes2, 'children');

```

```

delete(children(:));
handles.full_simulated_spectrum = [];
handles.simulated_spectrum = [];
guidata(hObject, handles);
end

% --- Executes on button press in Export_Data.
function Export_Data_Callback(hObject, eventdata, handles)
warning('off', 'all');
chdir(handles.filedir)
if isfile(handles.xlsxname)
    delete(handles.xlsxname);
end
writematrix([handles.Experimental_Data handles.computed_background
handles.Prepped_Experimental_Data], handles.xlsxname, 'Sheet', 1);
writematrix([handles.Experimental_Data handles.full_simulated_spectrum
handles.Prepped_Experimental_Data(:, 2) handles.simulated_spectrum],
handles.xlsxname, 'Sheet', 2);
outp = [handles.Experimental_Data(:, 1) handles.simulated_spectrum];
for i = 1:size(handles.Fitting_Profiles, 2)
    outp = [outp handles.Fitting_Profiles{i}(:, 2)];
end
writematrix(outp, handles.xlsxname, 'Sheet', 3);
writecell(handles.T_Data, handles.xlsxname, 'Sheet', 4);
chdir(handles.A);
end

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
children = get(handles.parent_axes, 'children');
delete(children(:));

children = get(handles.axes2, 'children');
delete(children(:));
set(handles.axes2, 'XLim', get(handles.axes1, 'XLim'));
handles.A = pwd;
handles.Filenames = {'Filenames:'};
set(handles.Species, 'Value', 1);
set(handles.Species, 'String', handles.Filenames);
handles.Standard_Data = {};
handles.Experimental_Data = [];
handles.Prepped_Experimental_Data = [];
handles.Fitting_Profiles = {};
handles.FWHM = 0.3;
parent_axes = gca;
handles.parent_figure = gcf;
handles.parent_axes = parent_axes;
hold(handles.parent_axes, 'on')
handles.vertical_offset = [];
handles.scaling_factor = [];
handles.computed_background = [];
handles.start_parameters = [];
handles.simulated_spectrum = [];
handles.full_simulated_spectrum = [];
handles.filedir = [];
handles.xlsxname = [];
handles.T_Data = [];
handles.Fitting_Error = [];

```



```

guidata(hObject, handles);
end

% --- Executes on button press in Aprox.
function Aprox_Callback(hObject, eventdata, handles)
prompt = handles.Filenames;
prompt(1) = [];
dlgtitle = 'Scattering Coefficients';
answer = inputdlg(prompt,dlgtitle);
end

```

User Guide:

The above program was developed to assist in rapid processing of X-Ray diffraction data collected from composite materials without the need for an external database. The program was developed and tested in MATLAB 2021b and its performance cannot be verified on older or newer versions. Similarly this program makes use of Nelder Mead simplex minimization which is implemented in MATLAB's optimization toolbox. The program provides the user with an interface for plotting experimental diffraction data to compare easily against files obtained from the ICDD crystallographic database or generated using the VESTA software produced by JP Mineral. From this framework any material can be used as reference as long as a .cif file has been generated for the given structure. The program can be executed from the MATLAB command line by simply running `Fit_Diffraction`. Importantly the program requires both `backcor.m` and `Fit_Diffraction.fig` in its home directory to function properly. On initialization the program presents the user with the interface presented in Figure I1.

On selecting the "Load Data," button the program will prompt the user to select a raw data file from the provide file explorer window. The program can accept two column data in several formats including an excel spreadsheet, .csv file, or as a .txt document with comma separated value formatting. Upon loading the raw experimental data into the program's memory, the program will prompt the user with the background subtraction window presented in Figure I2. The background subtraction makes use of the `backcor` function described in appendix G to conduct a polynomial background subtraction using a top bounded cost function. The user can choose the threshold tolerance, fitting function, and polynomial order to obtain a satisfactory background subtraction. On completion of the background subtraction the background subtracted data will be presented in the main function plot window as shown in Figure I3. Note that

Fit_Diffraction by default normalizes the intensity of experimental data by setting the absolute maximum to 1. This aids in subsequent data fitting.

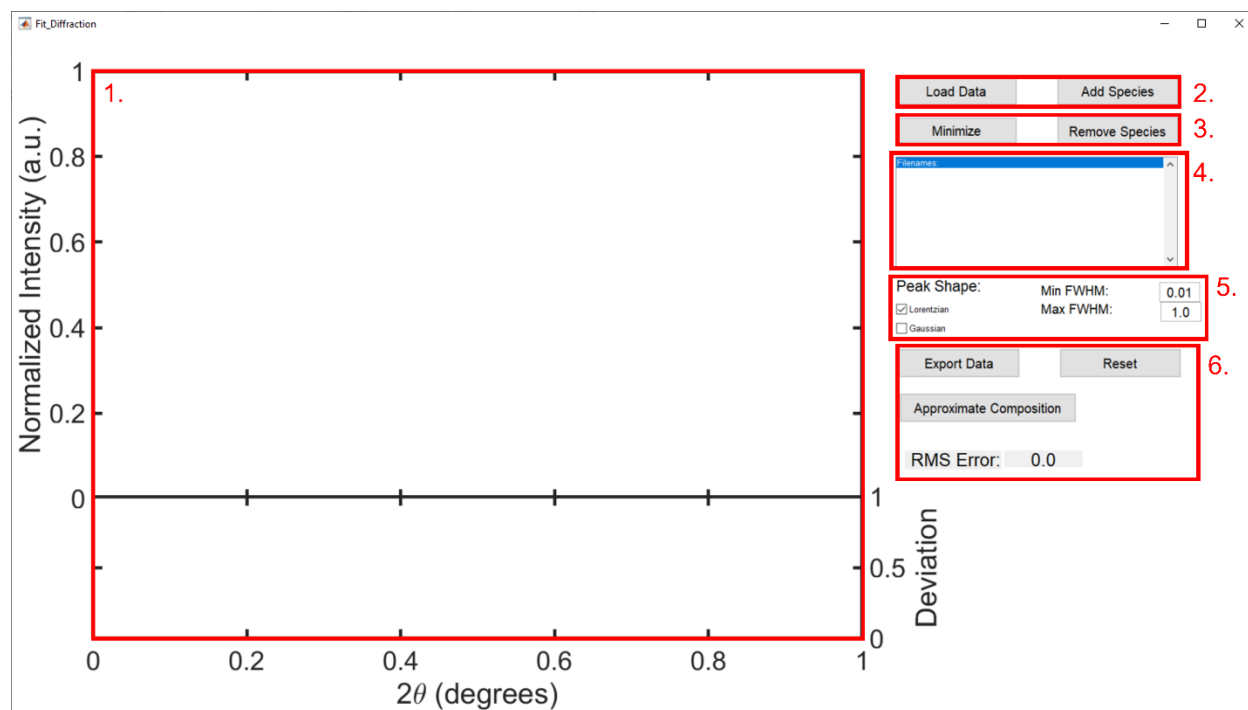


Figure II. User interface generated on initialization of Fit_Diffraction.m. The user interface contains the following components:

1. Main plot window to display experimental data and the deviation between the computed data fit and the raw data.
2. Buttons to load raw experimental data “Load Data” and to add reference crystallographic data “Add Species”. Data can be added in the form of ICDD data files or reflection table data generated by VESTA
3. Buttons to minimize the data fit as a linear combination of simulated spectra and to remove reference spectra from the data list
4. List of datasets held within the program’s memory
5. Parameters used to define the peak shape of diffraction features and to constrain the data fit
6. Buttons to export the data fit, reset the program and a report on the total error incurred during data fitting.

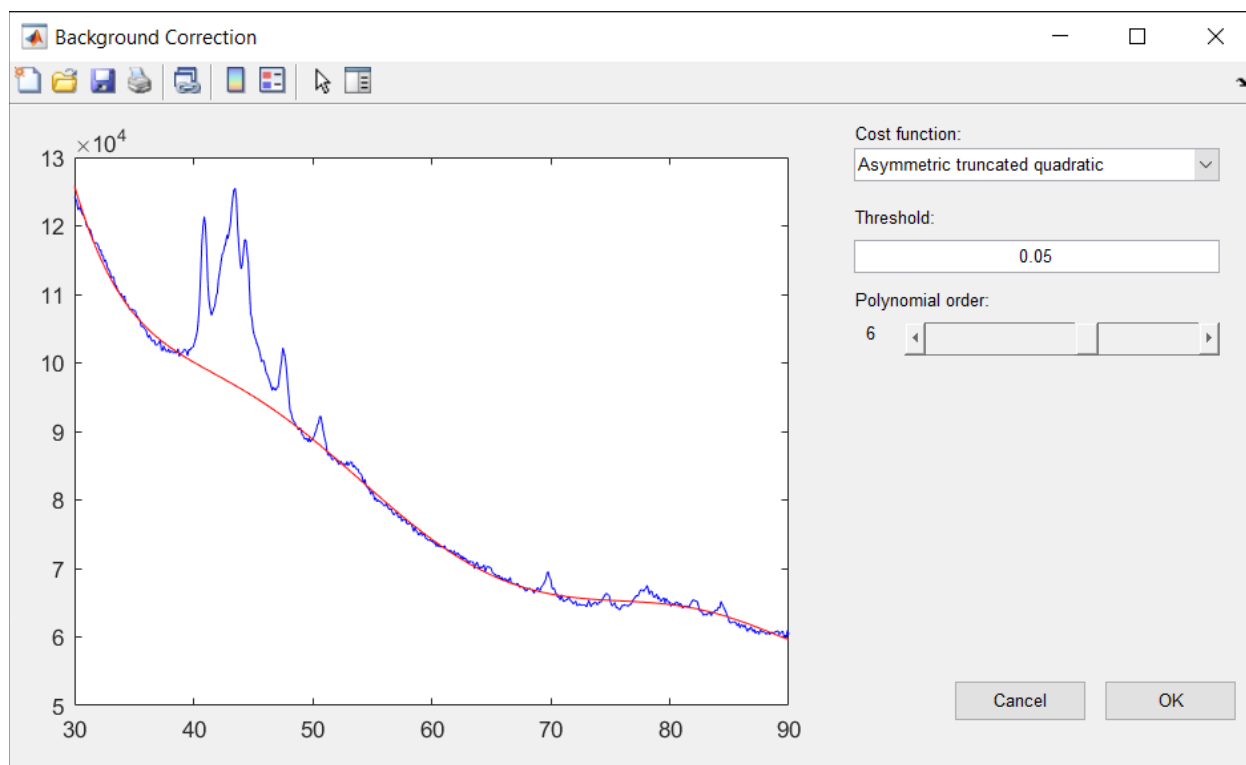


Figure I2. Background subtraction function to process raw data.

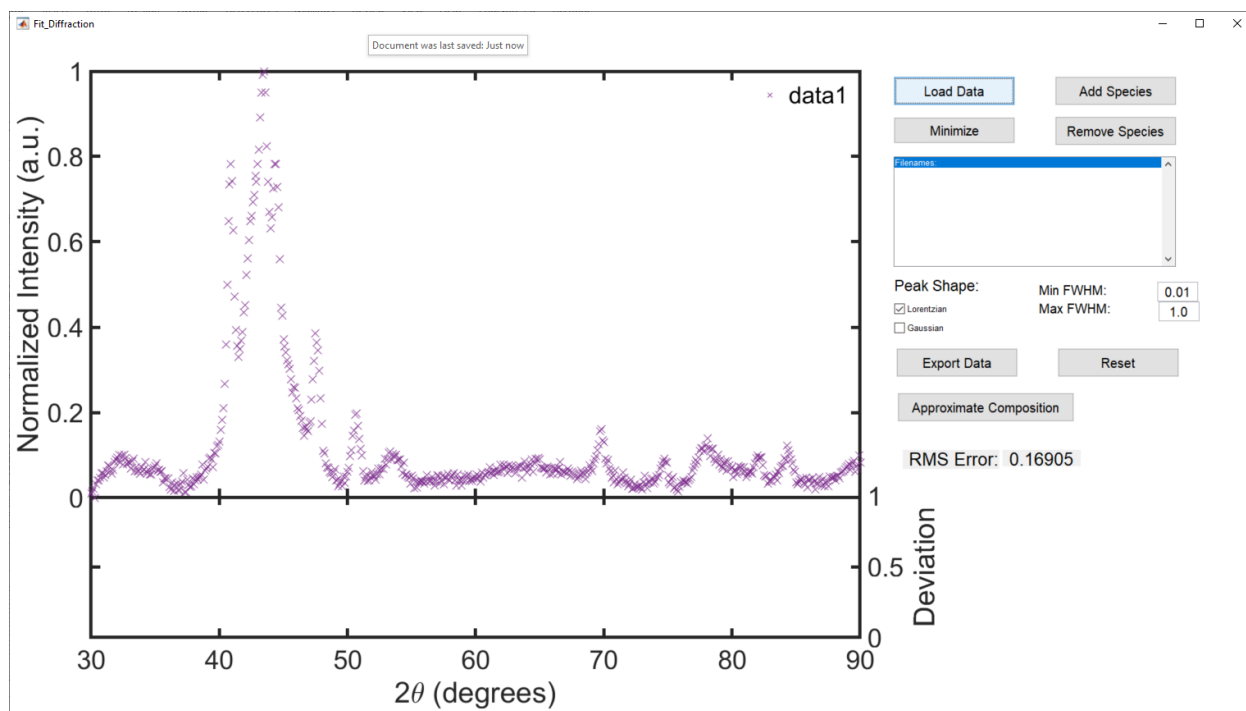


Figure I3. Raw data presented in the main function plot window following background subtraction and normalization.

Once raw data has been processed and displayed the user is free to add any number of reference structures to compare to experimental data. By default, Fit_Diffraction accepts ICDD data sets as well as .int files exported from VESTA. These files are formatted as a set of powder diffraction peak positions and associated intensities. The benefits of this approach allow the user to generate the reference data set as long as a .cif file exists for a given crystal structure, or as long as one can be generated in VESTA. This can be accomplished by simulating the powder diffraction pattern for a given structure in VESTA followed by exporting the generated reflection table as a text file. Each reference pattern selected by the user will appear in the plotting window and will populate the filename list as shown in Figure I4. The program offers either a Lorentzian or gaussian peak shape which is applied to each peak in the reference spectrum list to generate a simulated diffraction pattern.

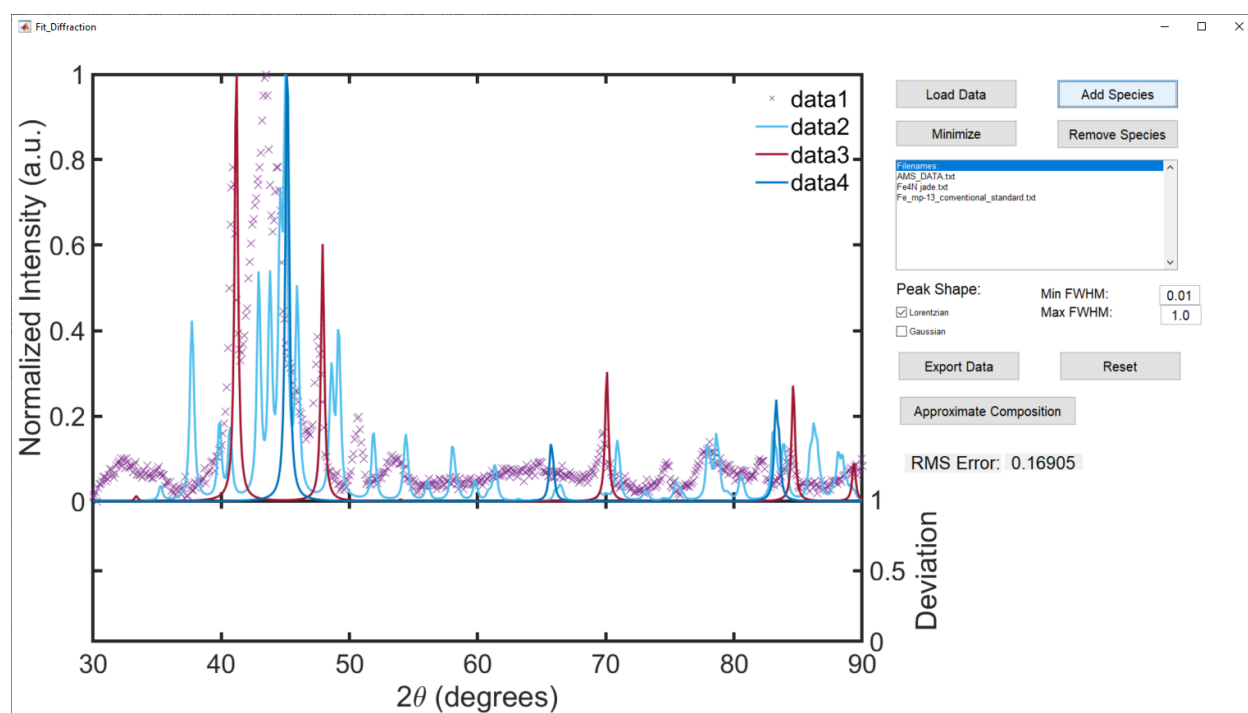


Figure I4. Program window after three different sets of reference data have been added.

Once the raw data and reference patterns have been supplied by the user the program is ready to attempt to fit the experimental pattern to a linear sum of the reference data. Importantly the cost function is allowed degrees of freedom in both intensity and full width at half maximum (FWHM) of the peaks used to construct the simulated diffractograms. The FWHM can be constrained depending on what is expected experimentally. The data fitting is initiated by selecting the minimize button. On completion the program will add the simulated diffractogram, along with its constituent parts to the main plot window. It will also add the deviation from experiment in the lower plot window as shown in Figure I5. Bear in mind that this example is merely used to demonstrate the capabilities of this program. Data fits significantly better than those shown can be achieved if adequate care is taken in selecting reference structures.

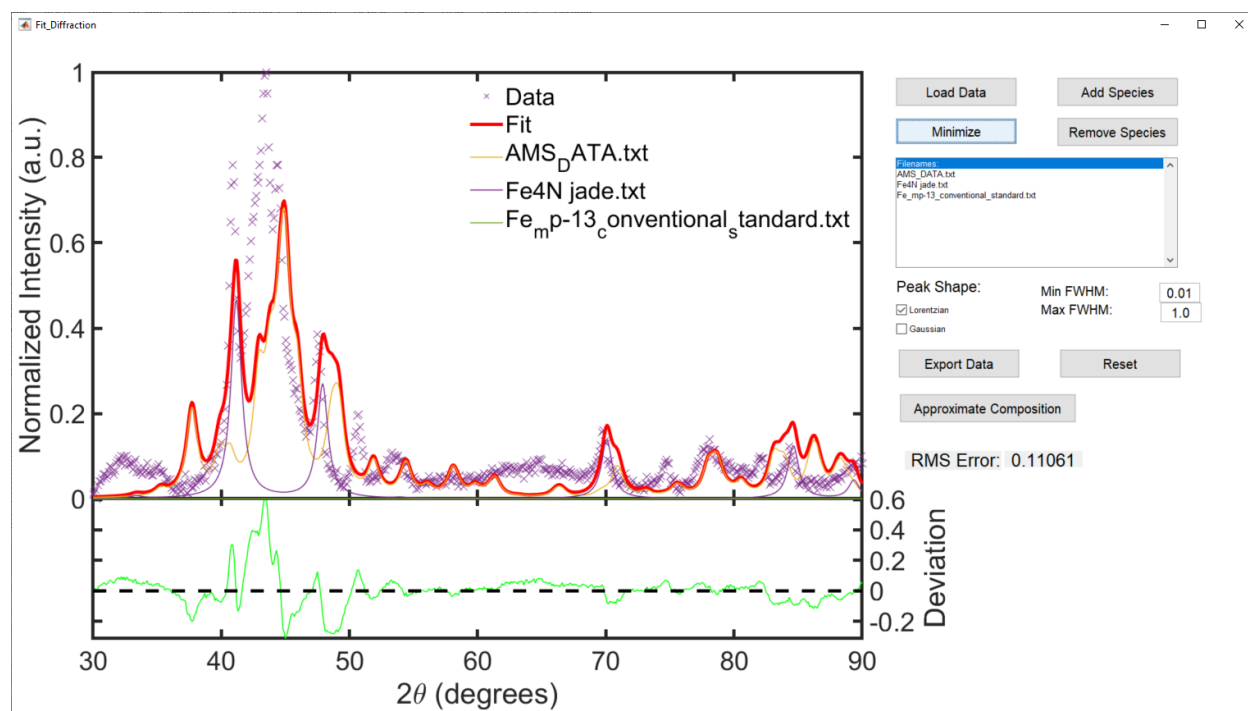


Figure I5. Program window following Nelder Mead Simplex minimization.

In addition to the plots shown in Figure I5 the program will also produce an external plot window which shows the original, non-background-subtracted dataset in black along with the simulated diffraction pattern, including simulated background, in red as shown in Figure I6.

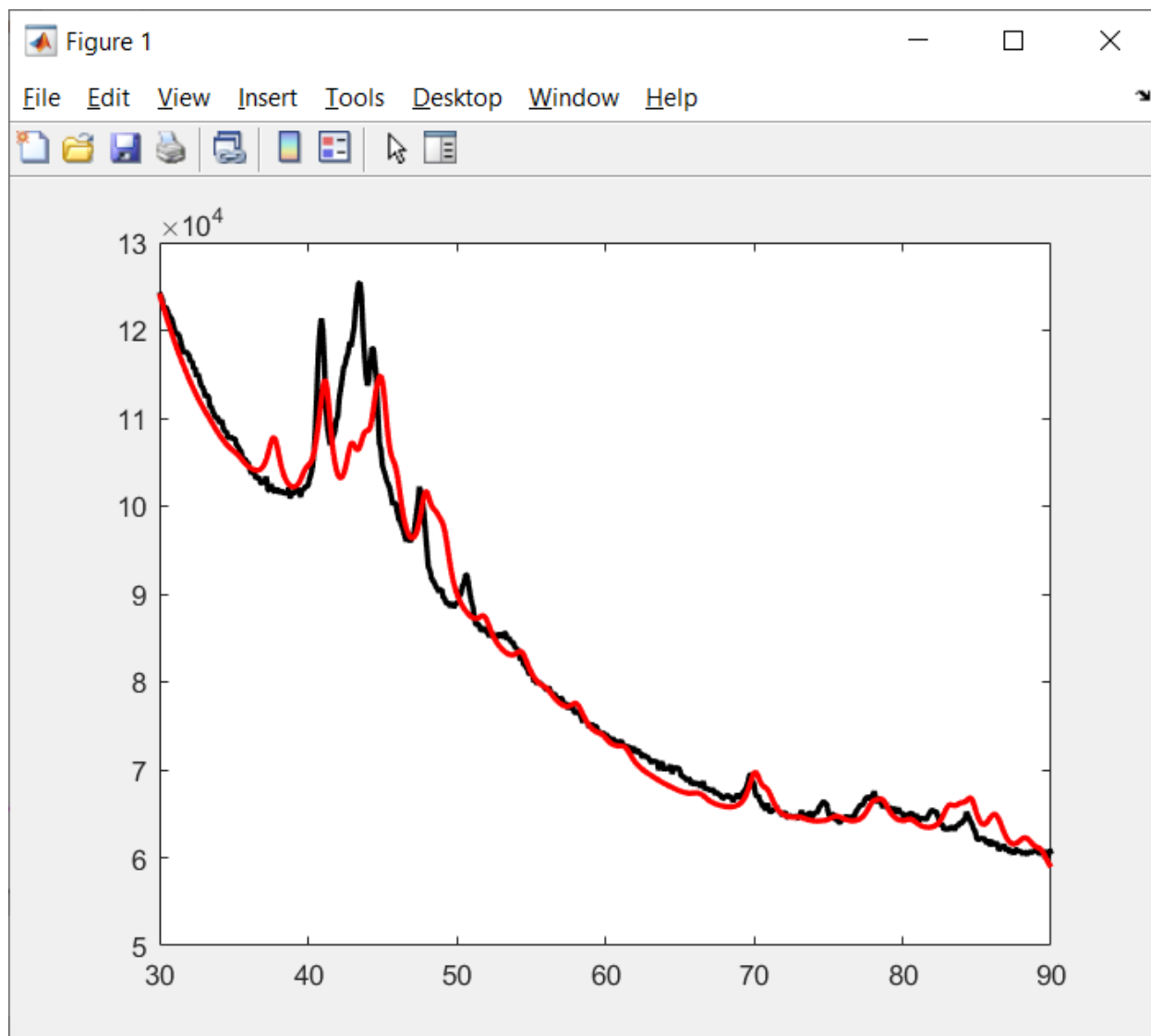


Figure I6. Generated simulated diffraction pattern compared against raw experimental data.

Finally, the user can export the parameters used for data fitting to a consolidated excel workbook for further quantification including the integrated intensity ratios of each reference pattern used in the fit.

Appendix J: Program to Process Galvanostatic Battery Cycling Data (MATLAB):

Note: At time of publication this program can process data exported from Arbin, CH Instruments, and NewWare battery cyclers. Program only accurately processes galvanostatic charge/discharge data at constant currents.

```
%===== Initialization =====%
clear all
close all
set(0,'DefaultFigureWindowStyle','normal')

%===== Main Script =====%
[Filename, filedir] = uigetfile('*..*', 'All Files (*.*)');
if Filename == 0; return; end
[filepath, name, ext] = fileparts(Filename);
Filename = fullfile(filedir, Filename);
List = {'Arbin', 'New Ware', 'CHI', 'Processed Data'};
[indx,tf] = listdlg('ListString',List);

switch indx
    case 1
        Data = Read_Arbin_Data(Filename);
    case 2
        if exist('m', 'var'); delete(m); end
        Data = Read_Neware_Data(Filename);
        try
            str=Find_Description(Filename); m = msgbox(str,'Remarks');
            m.Position(1) = 300; m.Position(2) = 300; catch
        end
    case 3
        Data = importdata(Filename, ','); Data = Data.data;
    case 4
        [Charge, Discharge, DCharge, DDischarge, CCapacity, DCapacity, CE] =
Read_Processed_Data(Filename);
end

if indx ~= 4
    inpt = inputdlg({'Rate (mA g-1)', 'dQ/dV Point Spread', 'Plot dQ/dV as
log (y/n)'});
    rate = str2num(inpt{1});
    span = str2num(inpt{2}); if isempty(span); span = 1; end
    dQlog = inpt{3};
    [Charge, Discharge, CCapacity, DCapacity, CE] = Segment(Data, rate);
    [DCharge, DDischarge] = Differentiate(Charge, Discharge, span, dQlog);
    Export_Data(name, filedir, Charge, Discharge, DCharge, DDischarge,
DCapacity, CCapacity, CE, dQlog);
else
    [~, ynlog] = xlsread(Filename, 3, 'B2:B2');
    if strcmp(ynlog, 'log(dQ/dV)'); dQlog = 'y'; else dQlog = []; end
end

Plot_Capacities(CCapacity, DCapacity, CE);
Plot_Traces(Charge, Discharge, DCharge, DDischarge, dQlog)
Plot_Tiled(Charge, Discharge, DCharge, DDischarge, dQlog)
clearvars -except Charge Discharge DCharge DDischarge CCapacity DCapacity CE
Data Filename m
```



```

%===== Function Library =====%
%===== Read Data from Arbin Cyclor =====%
function Data = Read_Arbin_Data(Filename)
    [status, sheets, xlFormat] = xlsfinfo(Filename);
    num = length(sheets);
    hExcel = actxserver('Excel.Application');
    hExcel.Visible = 1;
    hWorkbook = hExcel.Workbooks.Open(Filename);
    T = [];
    I = [];
    V = [];
    for i = 2 : num-1
        hWorksheet = hWorkbook.Sheets.Item(i);
        s1 = get(hExcel.Sheets, 'Item', i);
        s1.Select
        nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
        nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
        Range = ['A', num2str(1), ':', xlscol(nCol), num2str(1)];
        RangeObj = hWorksheet.Range(Range);
        Headers = RangeObj.value;
        Cols = find(~cellfun(@isempty, strfind(Headers, 'Test_Time(s)')));
        Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Current(A)')))];
        Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Voltage(V)')))];
        Range = [xlscol(Cols(1)), num2str(2), ':', xlscol(Cols(1)),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        T = [T; cell2mat(RangeObj.value)];
        Range = [xlscol(Cols(2)), num2str(2), ':', xlscol(Cols(2)),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        I = [I; cell2mat(RangeObj.value)];
        Range = [xlscol(Cols(3)), num2str(2), ':', xlscol(Cols(3)),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        V = [V; cell2mat(RangeObj.value)];
    end
    Data = [T I V];
    hExcel.Visible = 0;
    release(hWorksheet);
    hWorkbook.Close
    hExcel.Quit
    delete(hExcel)
end

%===== Read Data from Neware Cyclor =====%
function Data = Read_Neware_Data(Filename)
    [status, sheets, xlFormat] = xlsfinfo(Filename);
    num = length(sheets);
    hExcel = actxserver('Excel.Application');
    hExcel.Visible = 1;
    hWorkbook = hExcel.Workbooks.Open(Filename);
    T = [];
    I = [];
    V = [];
    try
        for i = 4:num

```

```

hWorksheet = hWorkbook.Sheets.Item(i);
s1 = get(hExcel.Sheets, 'Item', i);
s1.Select
nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
Range = ['A', num2str(1), ':', xlscol(nCol), num2str(1)];
RangeObj = hWorksheet.Range(Range);
Headers = RangeObj.value;

Cols = find(~cellfun(@isempty, strfind(Headers, 'Absolute
Time')));
Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Cur (mA)')))];
Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Voltage (V)')))];

Range = [xlscol(Cols(1)), num2str(2), ':', xlscol(Cols(1)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);
T = [T; Parse_Absolute_Time(RangeObj.value)];
Range = [xlscol(Cols(2)), num2str(2), ':', xlscol(Cols(2)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);
I = [I; cell2mat(RangeObj.value)];
Range = [xlscol(Cols(3)), num2str(2), ':', xlscol(Cols(3)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);
V = [V; cell2mat(RangeObj.value)];
end
catch
for i = 4:num
hWorksheet = hWorkbook.Sheets.Item(i);
s1 = get(hExcel.Sheets, 'Item', i);
s1.Select
nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
Range = ['A', num2str(1), ':', xlscol(nCol), num2str(1)];
RangeObj = hWorksheet.Range(Range);
Headers = RangeObj.value;

Cols = find(~cellfun(@isempty, strfind(Headers, 'Realtime')));
Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Current (mA)')))];
Cols = [Cols; find(~cellfun(@isempty, strfind(Headers,
'Voltage (V)')))];

Range = [xlscol(Cols(1)), num2str(2), ':', xlscol(Cols(1)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);
T = [T; Parse_Absolute_Time(RangeObj.value)];
Range = [xlscol(Cols(2)), num2str(2), ':', xlscol(Cols(2)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);
I = [I; cell2mat(RangeObj.value)];
Range = [xlscol(Cols(3)), num2str(2), ':', xlscol(Cols(3)),
num2str(nRows)];
RangeObj = hWorksheet.Range(Range);

```

```

        V = [V; cell2mat(RangeObj.value)];
    end
end
T = T-T(1);
Data = [T I V];
hExcel.Visible = 0;
release(hWorksheet);
hWorkbook.Close
hExcel.Quit
delete(hExcel)
end
%===== Read Data from Output File =====%
function [Charge, Discharge, DCharge, DDischarge, CCapacity, DCapacity, CE] =
Read_Processed_Data(Filename)
    Charge = {};
    Discharge = {};
    DCharge = {};
    DDischarge = {};
    CCapacity = [];
    CE = [];
    hExcel = actxserver('Excel.Application');
    hExcel.Visible = 1;
    hWorkbook = hExcel.Workbooks.Open(Filename);

    hWorksheet = hWorkbook.Sheets.Item(1);
    s1 = get(hExcel.Sheets, 'Item', 1);
    s1.Select
    nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
    nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
    for i = 1:floor(nCol/2)
        Range = [xlscol(i*2-1), num2str(3), ':', xlscol(i*2),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        Charge{i} = cell2mat(RangeObj.value);
    end

    hWorksheet = hWorkbook.Sheets.Item(2);
    s1 = get(hExcel.Sheets, 'Item', 2);
    s1.Select
    nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
    nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
    for i = 1:floor(nCol/2)
        Range = [xlscol(i*2-1), num2str(3), ':', xlscol(i*2),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        Discharge{i} = cell2mat(RangeObj.value);
    end

    hWorksheet = hWorkbook.Sheets.Item(3);
    s1 = get(hExcel.Sheets, 'Item', 3);
    s1.Select
    nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
    nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
    for i = 1:floor(nCol/2)
        Range = [xlscol(i*2-1), num2str(3), ':', xlscol(i*2),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);

```

```

        DCharge{i} = cell2mat(RangeObj.value);
    end

    hWorksheet = hWorkbook.Sheets.Item(4);
    s1 = get(hExcel.Sheets, 'Item', 4);
    s1.Select
    nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
    nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
    for i = 1:floor(nCol/2)
        Range = [xlscol(i*2-1), num2str(3), ':', xlscol(i*2),
num2str(nRows)];
        RangeObj = hWorksheet.Range(Range);
        DDischarge{i} = cell2mat(RangeObj.value);
    end

    hWorksheet = hWorkbook.Sheets.Item(5);
    s1 = get(hExcel.Sheets, 'Item', 5);
    s1.Select
    nRows = hExcel.ActiveSheet.UsedRange.Rows.Count;
    nCol = hExcel.ActiveSheet.UsedRange.Columns.Count;
    Range = [xlscol(1), num2str(2), ':', xlscol(2), num2str(nRows)];
    RangeObj = hWorksheet.Range(Range);
    CCapacity = cell2mat(RangeObj.value);

    Range = [xlscol(3), num2str(2), ':', xlscol(4), num2str(nRows)];
    RangeObj = hWorksheet.Range(Range);
    DCapacity = cell2mat(RangeObj.value);

    [L, I] = min([length(CCapacity), length(DCapacity)]);
    if I == 1
        CE = [CCapacity(:, 1) 100*DCapacity(1:L, 2)./CCapacity(1:L, 2)];
    else
        CE = [DCapacity(:, 1) 100*DCapacity(1:L, 2)./CCapacity(1:L, 2)];
    end

    hExcel.Visible = 0;
    release(hWorksheet);
    hWorkbook.Close
    hExcel.Quit
    delete(hExcel)
end

%===== Split Raw Data by Cycle Number =====%
function [Charge, Discharge, CCapacity, DCapacity, CE] = Segment(Data, rate)
Charge = {};
Discharge = {};
Data(Data(:, 2) == 0, :) = [];
if size(Data, 2) > 2
    I = Data(:, 2);
    if I(1) > 0
        Start_Polarization = 'Charge';
    else
        Start_Polarization = 'Discharge';
    end
    I_Shift = [(-1*I(1)); I(1:end-1)];
    Mult = I.*I_Shift;
    Mult = (Mult<0);
    [Rn, Cn] = find(Mult ==1);

```

```

for i = 1 : length(Rn)-1
    if I(Rn(i)) > 0
        col = (Data(Rn(i):Rn(i+1)-1, 1)-Data(Rn(i), 1))*rate/3600;
        Charge{size(Charge, 2)+1} = [col Data(Rn(i):Rn(i+1)-1, 3)];
    else
        col = (Data(Rn(i):Rn(i+1)-1, 1)-Data(Rn(i), 1))*rate/3600;
        Discharge{size(Discharge, 2)+1} = [col Data(Rn(i):Rn(i+1)-1,
3)];
    end
end
if I(Rn(end)) > 0
    col = (Data(Rn(end):end, 1)-Data(Rn(end), 1))*rate/3600;
    Charge{size(Charge, 2)+1} = [col Data(Rn(end):end, 3)];
else
    col = (Data(Rn(end):end, 1)-Data(Rn(end), 1))*rate/3600;
    Discharge{size(Discharge, 2)+1} = [col Data(Rn(end):end, 3)];
end
else
    T = Data(:, 1);
    V = Data(:, 2);

    %Remove data points occurring at the same time points
    T_Shift = [-1; T(1:end-1)];
    T_Diff = T_Shift-T;
    [Rdel, Cdel] = find(T_Diff == 0);
    T(Rdel)=[];
    V(Rdel)=[];
    V_max = max(V(floor(length(V)/3):end));
    V_min = min(V(floor(length(V)/3):end));
    V_Buffer = (V_max-V_min)/10;

    V_Shift = [(V(1)); V(1:end-1)];
    Diff = [linspace(1, length(V), length(V))' V sign(V-V_Shift)];
    Diff(Diff(:, 3)==0, :) = [];
    [Rdel, Cdel] = find((V_min+V_Buffer < Diff(:, 2)) & (Diff(:, 2) < V_max-
V_Buffer));
    Diff(Rdel, :)=[];
    Mult = Diff(:, 3).*[Diff(1, 3); Diff(1:end-1, 3)];
    Mult = (Mult<0);
    [Rn, Cn] = find(Mult ==1);
    Rn = [1; Diff(Rn-1, 1); length(V)];
    if V(Rn(2)) > V_max-V_Buffer
        Start_Polarization = 'Charge';
    else
        Start_Polarization = 'Discharge';
    end
end
for i = 1 : length(Rn)-2
    if V(Rn(i+1)-1) > V_max-V_Buffer
        col = (T(Rn(i):Rn(i+1)-1)-T(Rn(i)))*rate/3600;
        Charge{size(Charge, 2)+1} = [col V(Rn(i):Rn(i+1)-1)];
    else
        col = (T(Rn(i):Rn(i+1)-1)-T(Rn(i)))*rate/3600;
        Discharge{size(Discharge, 2)+1} = [col V(Rn(i):Rn(i+1)-1)];
    end
end
end

```

```

    if V(Rn(end)) > V(Rn(end-1))
        col = (T(Rn(end-1):Rn(end))-T(Rn(end-1)))*rate/3600;
        Charge[size(Charge, 2)+1] = [col V(Rn(end-1):Rn(end))];
    else
        col = (T(Rn(end-1):Rn(end))-T(Rn(end-1)))*rate/3600;
        Discharge[size(Discharge, 2)+1] = [col V(Rn(end-1):Rn(end))];
    end
end

CCapacity = [];
DCapacity = [];
CE = [];
for i = 1:size(Charge, 2)
    CCapacity = [CCapacity; [i Charge{i}(end, 1)]];
end
for i = 1:size(Discharge, 2)
    DCapacity = [DCapacity; [i Discharge{i}(end, 1)]];
end

if strcmp(Start_Polarization, 'Charge')
    for i = 1:min([length(CCapacity); length(DCapacity)])
        CE = [CE; [i DCapacity(i, 2)*100/CCapacity(i, 2)]];
    end
else
    for i = 2:min([length(CCapacity); length(DCapacity)])
        CE = [CE; [i DCapacity(i, 2)*100/CCapacity(i-1, 2)]];
    end
end
end

%===== Compute Differential Capacity =====%
function [DCharge, DDischarge] = Differentiate(Charge, Discharge, span,
dQlog)
    L = ceil(span/2);
    DCharge = {};
    DDischarge = {};
    if dQlog == 'y'
        for i = 1 : size(Charge, 2)
            dC = (Charge{i}(1+L:end, 1)-Charge{i}(1:end-L,
1))./(Charge{i}(1+L:end, 2)-Charge{i}(1:end-L, 2));
            dC = sign(dC).*log10(dC.*sign(dC));
            DCharge{i} = [(Charge{i}(1+L:end, 2)+Charge{i}(1:end-L, 2))/2 dC];
            DCharge{i}(isnan(DCharge{i}(:, 2)), :) = [];
        end
        for i = 1 : size(Discharge, 2)
            dD = (Discharge{i}(1+L:end, 1)-Discharge{i}(1:end-L,
1))./(Discharge{i}(1+L:end, 2)-Discharge{i}(1:end-L, 2));
            dD = sign(dD).*log10(dD.*sign(dD));
            DDischarge{i} = [(Discharge{i}(1+L:end, 2)+Discharge{i}(1:end-L,
2))/2 dD];
            DDischarge{i}(isnan(DDischarge{i}(:, 2)), :) = [];
        end
    else
        for i = 1 : size(Charge, 2)
            DCharge{i} = [(Charge{i}(1+L:end, 2)+Charge{i}(1:end-L, 2))/2
(Charge{i}(1+L:end, 1)-Charge{i}(1:end-L, 1))./(Charge{i}(1+L:end, 2)-
Charge{i}(1:end-L, 2))];
        end
    end
end

```

```

        for i = 1 : size(Discharge, 2)
            DDischarge{i} = [(Discharge{i}(1+L:end, 2)+Discharge{i}(1:end-L,
2))/2 (Discharge{i}(1+L:end, 1)-Discharge{i}(1:end-L,
1))./(Discharge{i}(1+L:end, 2)-Discharge{i}(1:end-L, 2))];
        end
    end
end
%===== Write Output File =====%
function Export_Data(name, directory, Charge, Discharge, DCharge, DDischarge,
DCapacity, CCapacity, CE, dQlog)
    rngc = {};
    rngd = {};
    for m = 1 : size(Charge, 2)
        start = strcat(xlscol(2*m-1), int2str(3));
        finish = strcat(xlscol(2*m), int2str(size(Charge{m}, 1)+ 2));
        rngc{m} = sprintf('%s:%s', start, finish);
    end

    for m = 1 : size(Discharge, 2)
        start = strcat(xlscol(2*m-1), int2str(3));
        finish = strcat(xlscol(2*m), int2str(size(Discharge{m}, 1)+ 2));
        rngd{m} = sprintf('%s:%s', start, finish);
    end

    e = actxserver('Excel.Application');
    eWorkbook = e.Workbooks.Add;
    e.Visible = 1;
    eSheets = e.ActiveWorkbook.Sheets;
    eSheet1 = eSheets.get('Item', 1);
    eSheet1.Activate

    % Input Charge and Discharge Data
    for i = 1 : size(Charge, 2)
        %Cycle header
        eActivsheetRange = get(e.Activesheet, 'Range', strcat(xlscol(2*i-1),
int2str(1)));
        eActivsheetRange.Value = strcat('Cycle_', int2str(i));

        %Column headers
        hstart = strcat(xlscol(2*i-1), int2str(2));
        hfinish = strcat(xlscol(2*i), int2str(2));
        hrngc = sprintf('%s:%s', hstart, hfinish);
        eActivsheetRange = get(e.Activesheet, 'Range', hrngc);
        eActivsheetRange.Value = {'Capacity (mA h /g)', 'Potential (V)'};

        %Input data
        eActivsheetRange = get(e.Activesheet, 'Range', rngc{i});
        eActivsheetRange.Value = Charge{i};
    end
    eSheets.Item(1).Name = 'Charge Curves';
    eSheets.Add([], eSheets.Item(eSheets.Count));
    for i = 1 : size(Discharge, 2)
        %Cycle header
        eActivsheetRange = get(e.Activesheet, 'Range', strcat(xlscol(2*i-1),
int2str(1)));
        eActivsheetRange.Value = strcat('Cycle_', int2str(i));
    end
end

```

```

    %Column header
    hstart = strcat(xlscol(2*i-1), int2str(2));
    hfinish = strcat(xlscol(2*i), int2str(2));
    hrnge = sprintf('%s:%s', hstart, hfinish);
    eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
    eActivsheetRange.Value = {'Capacity (mA h /g)', 'Potential (A)'};

    %Input Data
    eActivsheetRange = get(e.Activesheet, 'Range', rng2{i});
    eActivsheetRange.Value = Discharge{i};
end

rng2 = {};
rng2 = {};
for m = 1 : size(DCharge, 2)
    start = strcat(xlscol(2*m-1), int2str(3));
    finish = strcat(xlscol(2*m), int2str(size(DCharge{m}, 1)+ 2));
    rng2{m} = sprintf('%s:%s', start, finish);
end

for m = 1 : size(DDischarge, 2)
    start = strcat(xlscol(2*m-1), int2str(3));
    finish = strcat(xlscol(2*m), int2str(size(DDischarge{m}, 1) + 2));
    rng2{m} = sprintf('%s:%s', start, finish);
end
eSheets.Item(2).Name = 'Discharge Curves';
eSheets.Add([], eSheets.Item(eSheets.Count));
for i = 1 : size(DCharge, 2)
    %Cycle number header
    eActivsheetRange = get(e.Activesheet, 'Range', strcat(xlscol(2*i-1),
int2str(1)));
    eActivsheetRange.Value = strcat('Cycle_', int2str(i));

    %Column headers
    hstart = strcat(xlscol(2*i-1), int2str(2));
    hfinish = strcat(xlscol(2*i), int2str(2));
    hrnge = sprintf('%s:%s', hstart, hfinish);
    eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
    if dQlog == 'y'
        eActivsheetRange.Value = {'Potential (V)', 'log(dQ/dV)'};
    else
        eActivsheetRange.Value = {'Potential (V)', 'dQ/dV'};
    end

    %Input Data
    eActivsheetRange = get(e.Activesheet, 'Range', rng2{i});
    eActivsheetRange.Value = DCharge{i};
end
eSheets.Item(3).Name = 'Charge Differential';
eSheets.Add([], eSheets.Item(eSheets.Count));
for i = 1 : size(DDischarge, 2)
    %Cycle number header
    eActivsheetRange = get(e.Activesheet, 'Range', strcat(xlscol(2*i-1),
int2str(1)));
    eActivsheetRange.Value = strcat('Cycle_', int2str(i));

```



```

%Column headers
hstart = strcat(xlscol(2*i-1), int2str(2));
hfinish = strcat(xlscol(2*i), int2str(2));
hrnge = sprintf('%s:%s', hstart, hfinish);
eActivsheetRange = get(e.Activesheet, 'Range', hrnge);
if dQlog == 'y'
    eActivsheetRange.Value = {'Potential (V)', 'log(dQ/dV)'};
else
    eActivsheetRange.Value = {'Potential (V)', 'dQ/dV'};
end

%Input Data
eActivsheetRange = get(e.Activesheet, 'Range', rng2{i});
eActivsheetRange.Value = DDischarge{i};
end
eSheets.Item(4).Name = 'Discharge Differential';

% Export Charge and Discharge Capacity to Spreadsheet
eSheets.Add([], eSheets.Item(eSheets.Count));
eActivsheetRange = get(e.Activesheet, 'Range', 'A1:B1');
eActivsheetRange.Value = {'Cycle Number', 'Charge Capacity'};
eActivsheetRange = get(e.Activesheet, 'Range', 'C1:D1');
eActivsheetRange.Value = {'Cycle Number', 'Discharge Capacity'};
eActivsheetRange = get(e.Activesheet, 'Range', 'E1:F1');
eActivsheetRange.Value = {'Cycle Number', 'Coulombic Efficiency'};

CRange = sprintf('%s:%s', strcat(xlscol(1), int2str(2)),
strcat(xlscol(2), int2str(length(CCapacity) + 1)));
DRange = sprintf('%s:%s', strcat(xlscol(3), int2str(2)),
strcat(xlscol(4), int2str(length(DCapacity) + 1)));
CERange = sprintf('%s:%s', strcat(xlscol(5), int2str(2)),
strcat(xlscol(6), int2str(length(CE) + 1)));

eActivsheetRange = get(e.Activesheet, 'Range', CRange);
eActivsheetRange.Value = CCapacity;
eActivsheetRange = get(e.Activesheet, 'Range', DRange);
eActivsheetRange.Value = DCapacity;
eActivsheetRange = get(e.Activesheet, 'Range', CERange);
eActivsheetRange.Value = CE;
numsheets = eSheets.Count;
eSheets.Item(numsheets).Name = 'Charge and Discharge Capacity';

% Save Workbook
newfile = [name, ' processed'];
name = sprintf('%s', newfile, '.xlsx');
try
    SaveAs(eWorkbook, fullfile(directory, name));
catch
    disp('Workbook not saved');
end
eWorkbook.Saved = 1;
e.Visible = 0;
Close(eWorkbook);
Quit(e);

```

```

delete(e);
end
%===== Plot Capacity and C.E. =====%
function Plot_Capacities(CCapacity, DCapacity, CE)
    if ~isempty(DCapacity)
        figure('Name', 'Capacities and Coulombic
Efficiencies', 'NumberTitle', 'off', 'Position', [50 50 1000 700]);
        set(gcf, 'Color', 'w');
        scatter(DCapacity(:, 1), DCapacity(:, 2), 200, 'MarkerEdgeColor',
'b', 'LineWidth', 3.0);
        ax1 = gca;
        ax1.FontSize = 28.0;
        ax1.FontName = 'Arial';
        ax1.LineWidth = 3.0;
        ax1.Color = 'None';
        ax1.YLim = [0 (1.5*max(DCapacity(:, 2)))];
        pbaspect([1.2 1 1]);
        xlabel('Cycle Number');
        ylabel('Discharge Capacity (mA h g-1)');
    end

    if ~isempty(CE)
        ax2 = axes;
        linkaxes([ax1 ax2], 'x');
        scatter(CE(:, 1), CE(:, 2), 200, 'MarkerEdgeColor', 'r', 'LineWidth',
3.0);
        ax2.YColor = 'r';
        ax2.Color = 'None';
        ax2.YAxisLocation = 'right';
        ax2.XAxisLocation = 'top';
        ax2.XTick = [];
        ax2.XTickLabel=[];
        ax2.Position = ax1.Position;
        ax2.PlotBoxAspectRatio = ax1.PlotBoxAspectRatio;
        ax2.LineWidth = 3.0;
        ax2.FontSize = 28.0;
        ax2.FontName = 'Arial';
        ylabel('C.E. (%)');
        ax2.YLim = [0 120];
        ax1.XLim = [-1 (max(DCapacity(:, 1))+1)];
        axes(ax1);
    end
end
%===== Plot Charge/Discharge Traces =====%
function Plot_Traces(Charge, Discharge, DCharge, DDischarge, dQlog)
    figure('Name', 'Charge/Discharge Traces', 'NumberTitle', 'off',
'Position', [50 50 1000 700]);
    set(gcf, 'Color', 'w');
    box on
    hold on
    jetcustom = jet(max([size(Charge, 2) size(Discharge, 2)]));
    for i = 1:size(Charge, 2)
        plot3(i*ones(length(Charge{i}), 1), Charge{i}(:, 1), Charge{i}(:, 2),
'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    for i = 1:size(Discharge, 2)

```

```

        plot3(i*ones(length(Discharge{i}), 1), Discharge{i}(:, 1),
Discharge{i}(:, 2), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    ax1 = gca;
    ax1.View = [90 0];
    ax1.FontSize = 28.0;
    ax1.FontName = 'Arial';
    ax1.LineWidth = 3.0;
    pbaspect([1.2 1 1]);
    xlabel('Cycle Number');
    ylabel('Capacity (mA h g^{-1})');
    zlabel('Potential (V vs Li/Li^{+})');
    colormap(jet(max([size(Charge, 2) size(Discharge, 2)])));
    c = colorbar;
    c.Ticks = [0, 0.5, 1];
    c.TickLabels = {'1', num2str(round(max([size(Charge, 2) size(Discharge,
2)])/2)), num2str(round(max([size(Charge, 2) size(Discharge, 2)])))};
    c.Label.String = 'Cycle Number';
    c.Position(3) = 0.025;
    c.Position(1) = ax1.InnerPosition(1)+ax1.InnerPosition(3)-0.1;

    figure('Name', 'Differential Charge/Discharge
Traces','NumberTitle','off', 'Position',[50 50 1000 700]);
    set(gcf, 'Color', 'w');
    box on
    hold on
    jetcustom = jet(max([size(DCharge, 2) size(DDischarge, 2)]));
    for i = 1:size(DCharge, 2)
        plot3(i*ones(length(DCharge{i}), 1), DCharge{i}(:, 1), DCharge{i}(:,
2), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    for i = 1:size(DDischarge, 2)
        plot3(i*ones(length(DDischarge{i}), 1), DDischarge{i}(:, 1),
DDischarge{i}(:, 2), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    ax1 = gca;
    ax1.View = [90 0];
    ax1.FontSize = 28.0;
    ax1.FontName = 'Arial';
    ax1.LineWidth = 3.0;
    pbaspect([1.2 1 1]);
    xlabel('Cycle Number');
    ylabel('Potential (V vs Li/Li^{+})');
    if dQlog == 'y'
        zlabel('Log_{10} dQ/dV');
    else
        zlabel('dQ/dV (mA h g^{-1} V^{-1})');
    end
    colormap(jet(max([size(DCharge, 2) size(DDischarge, 2)])));
    c = colorbar;
    c.Ticks = [0, 0.5, 1];
    c.TickLabels = {'1', num2str(round(max([size(DCharge, 2) size(DDischarge,
2)])/2)), num2str(round(max([size(DCharge, 2) size(DDischarge, 2)])))};
    c.Label.String = 'Cycle Number';
    c.Position(3) = 0.025;
    c.Position(1) = ax1.InnerPosition(1)+ax1.InnerPosition(3)-0.1;
end

```

```

%===== Create Tiled Plot =====%
function Plot_Tiled(Charge, Discharge, DCharge, DDischarge, dQlog)
    figure('Name', 'Tiled Plot','NumberTitle','off', 'Position',[50 50 1200
700]);
    set(gcf, 'Color', 'w');
    tiledlayout(1, 9, 'TileSpacing','Compact');
    ax1 = nexttile(1, [1 5]);
    box on
    hold on
    jetcustom = jet(max([size(Charge, 2) size(Discharge, 2)]));
    for i = 1:size(Charge, 2)
        plot3(i*ones(length(Charge{i})), 1, Charge{i}(:, 1), Charge{i}(:, 2),
'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    for i = 1:size(Discharge, 2)
        plot3(i*ones(length(Discharge{i})), 1, Discharge{i}(:, 1),
Discharge{i}(:, 2), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end

    view(90, 0);
    ax1.FontSize = 28.0;
    ax1.FontName = 'Arial';
    ax1.LineWidth = 3.0;
    ylabel('Capacity (mA h g^{-1})');
    zlabel('Potential (V vs Li/Li^{+})');

    ax2 = nexttile([1 3]);
    if dQlog == 'y'
        factor = 1;
    else
        factor = 1000;
    end
    hold on
    box on
    jetcustom = jet(max([size(DCharge, 2) size(DDischarge, 2)]));
    for i = 1:size(DCharge, 2)
        plot3(i*ones(length(DCharge{i})), 1, DCharge{i}(:, 2)/factor,
DCharge{i}(:, 1), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    for i = 1:size(DDischarge, 2)
        plot3(i*ones(length(DDischarge{i})), 1, DDischarge{i}(:, 2)/factor,
DDischarge{i}(:, 1), 'LineWidth', 3, 'Color', jetcustom(i, :))
    end
    ax2.View = [90 0];
    ax2.FontSize = 28.0;
    ax2.FontName = 'Arial';
    ax2.LineWidth = 3.0;
    ax2.ZTickLabel = [];
    zlabel([])
    if dQlog == 'y'
        ylabel('Log_{10} dQ/dV');
    else
        ylabel('dQ/dV (A h g^{-1} V^{-1})');
    end
    colormap(jet(max([size(DCharge, 2) size(DDischarge, 2)]));
    c = colorbar;
    c.Ticks = [0, 0.5, 1];

```

```

c.TickLabels = {'1', num2str(round(max([size(DCharge, 2) size(DDischarge, 2)]/2)), num2str(round(max([size(DCharge, 2) size(DDischarge, 2)]))));
c.Label.String = 'Cycle Number';
drawnow
ylims = max(abs(ax2.YLim));
ax2.YLim = [-1*ylims ylims];
ax2.ZLim = ax1.ZLim;
line([0 0], [0 0], ax2.ZLim, 'Color', 'Black', 'LineWidth', 3,
'LineStyle', '--')
c.Position(3) = 0.015;
c.Position(1) = ax2.InnerPosition(1)+ax2.InnerPosition(3) + 0.02;
end
%===== Utility for converting numbers to excel cells =====
function b = xlscol(a)
% Version 1.1 - Kevin Crosby

% DATE      VER  NAME      DESCRIPTION
% 07-30-10   1.0  K. Crosby   First Release
% 08-02-10   1.1  K. Crosby   Vectorized loop for numerics.

% Contact: Kevin.L.Crosby@gmail.com
base = 26;
if iscell(a)
    b = cellfun(@xlscol, a, 'UniformOutput', false); % handles mixed case
too
elseif ischar(a)
    if ~isempty(strfind(a, ':')) % i.e. if is a range
        b = cellfun(@xlscol, regexp(a, ':', 'split'));
    else % if isempty(strfind(a, ':')) % i.e. if not a range
        b = a(isletter(a)); % get rid of numbers and symbols
        if isempty(b)
            b = {[]};
        else % if ~isempty(a);
            b = double(upper(b)) - 64; % convert ASCII to number from 1 to 26
            n = length(b); % number of characters
            b = b * base.^((n-1):-1:0)';
        end % if isempty(a)
    end % if ~isempty(strfind(a, ':')) % i.e. if is a range
elseif isnumeric(a) && numel(a) ~= 1
    b = arrayfun(@xlscol, a, 'UniformOutput', false);
else % if isnumeric(a) && numel(a) == 1
    n = ceil(log(a)/log(base)); % estimate number of digits
    d = cumsum(base.^(0:n+1)); % offset
    n = find(a >= d, 1, 'last'); % actual number of digits
    d = d(n:-1:1); % reverse and shorten
    r = mod(floor((a-d)./base.^(n-1:-1:0)), base) + 1; % modulus
    b = char(r+64); % convert number to ASCII
end % if iscell(a)

% attempt to "flatten" cell, by removing nesting
if iscell(b) && (iscell([b{:}]) || isnumeric([b{:}]))
    b = [b{:}];
end % if iscell(b) && (iscell([b{:}]) || isnumeric([ba{:}]))
end
%===== Find Remarks in Neware Files =====
function str=Find_Description(Filename)
[~, str] = xlsread(Filename, 1, 'J8:J8');

```

```

end
%===== Convert Date/Time to seconds =====%
function TTime=Parse_Absolute_Time(times)
    TTime = [];
    for i = 1:size(times, 1)
        t_split = strsplit(times{i});
        dte = cell2mat(cellfun(@str2num,strsplit(t_split{1},
'/' ),'UniformOutput',false));
        try
            tme = cell2mat(cellfun(@str2num,strsplit(t_split{2},
':' ),'UniformOutput',false));
            m = dte(1); d = dte(2); y = dte(3); h = tme(1); min = tme(2); s =
tme(3);
            if strcmp(t_split{3}, 'PM')&&h<12; h = h+12;
            elseif strcmp(t_split{3}, 'AM')&&h==12; h=h-12; end
        catch
            s = s+1;
        end
        TTime = [TTime; datenum(y,m,d)*24*3600+3600*h+60*min+s];
    end
end
end

```

User Guide:

The program described above is a simple script to process the output files obtained from galvanostatic cycling on several different instruments common to electrochemical laboratories including CHInstruments potentiostats, Arbin electrochemical workstations, and NewWare battery systems. The program was developed and tested in MATLAB 2021b and, as such, its performance cannot be verified on older or newer versions. The user interface employed here was written with the goal of providing a simple and fast method to process battery cycling data. The program can be executed by running `Galvanostat_v5` in the MATLAB command line. On initialization the user is presented with the following window asking for information about the experimental parameters it needs to process the data. First the user will be asked to select the specific instrument used to collect the data as this will inform the specific data acquisition program used to read the raw data (note that one of the options is “Processed Data,” as this program has the capability to re-plot data that it has already processed).

Following this dialogue, the user will be prompted to enter the cycling rate used for the experiment in mA g^{-1} which will be used to compute the overall gravimetric capacity. It will also ask for the point spread to be used when calculating the differential capacity. This is used to help cope with noisy data which can prove troublesome upon differentiation. If nothing is defined the program will assume that the point spread is 1 and adjacent points will be used to compute the numeric derivative. Finally, the window asks if the user wants to plot the differential capacity on a logarithmic scale which is often useful if the charge and discharge plateaus are very flat. In addition to this a small window will appear that will report any information the program can find in the file describing the specific measurement.

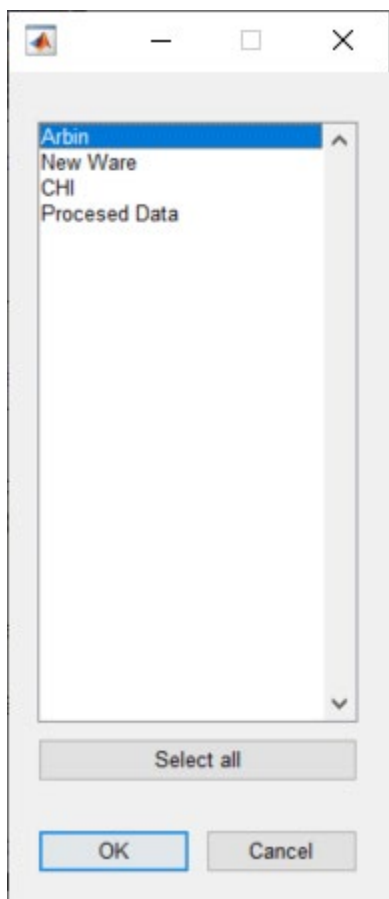


Figure J1. Dialogue to select the appropriate data format for import.

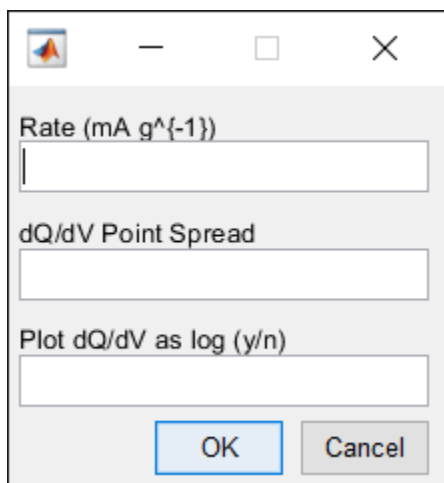


Figure J2. Dialogue to input cycling conditions.

Once the appropriate cycling parameters have been entered the program will process the raw data, (outputting an excel document appended with the word “processed” to the same directory as the original dataset with each charge/discharge cycle separated) and generate a series of plots to visualize it. The Program will export a capacity vs cycle number plot (Figure J3), a consolidated plot of all the charge/discharge traces (Figure J4), a consolidated plot of differential capacity traces (Figure J5), and a combined plot with both the charge/discharge traces and the differential capacity traces (Figure J6). All plots are color coded by cycle number apart from the first one.

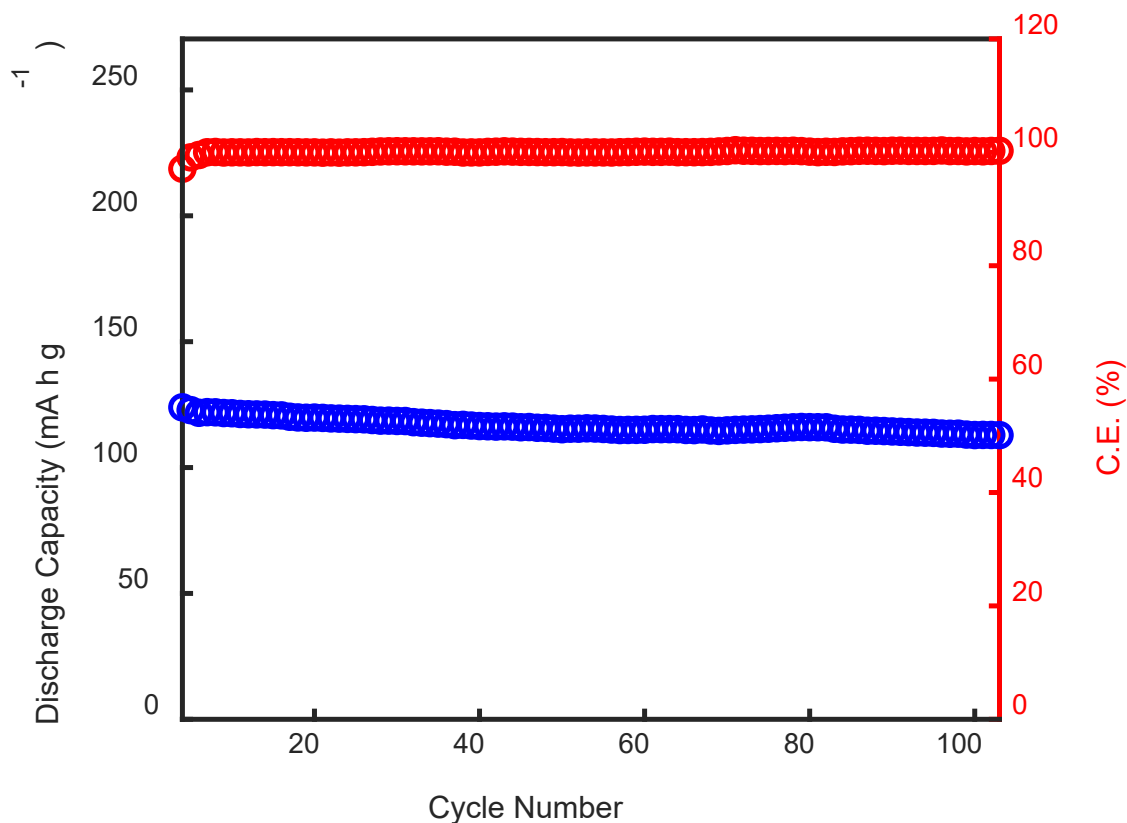


Figure J3. Example plot of discharge capacity against cycle number.

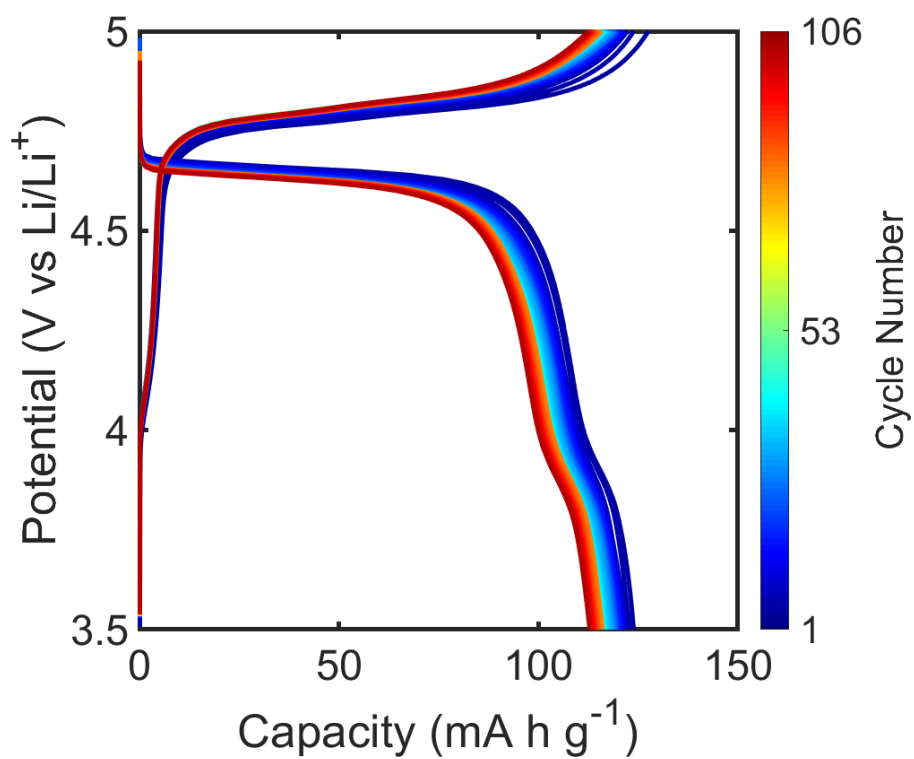


Figure J4. Example plot of consolidated charge and discharge traces.

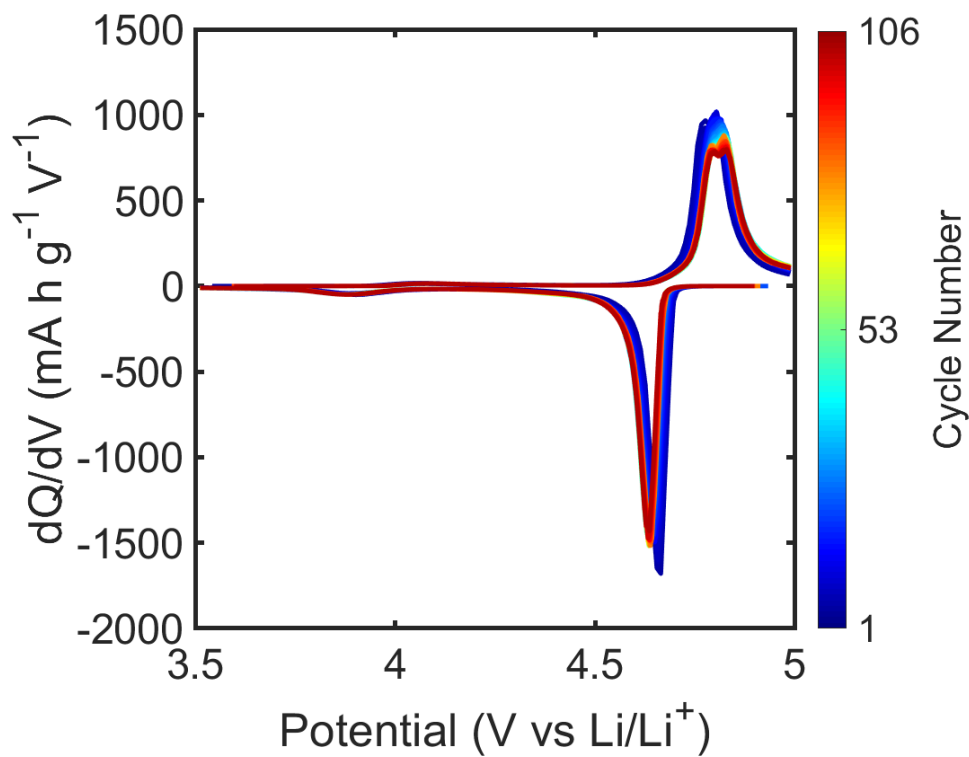


Figure J5. Example plot of consolidated differential capacity traces.

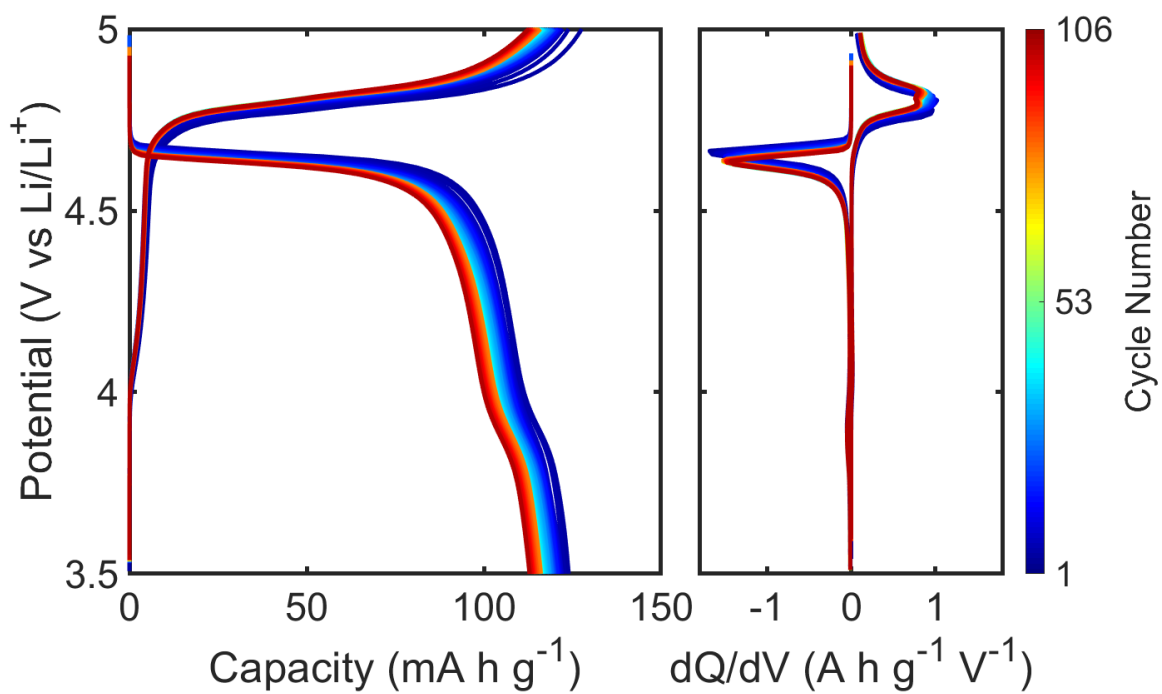


Figure J6. Combined charge/discharge traces and differential capacity traces to visualize the potential at which redox processes occur.