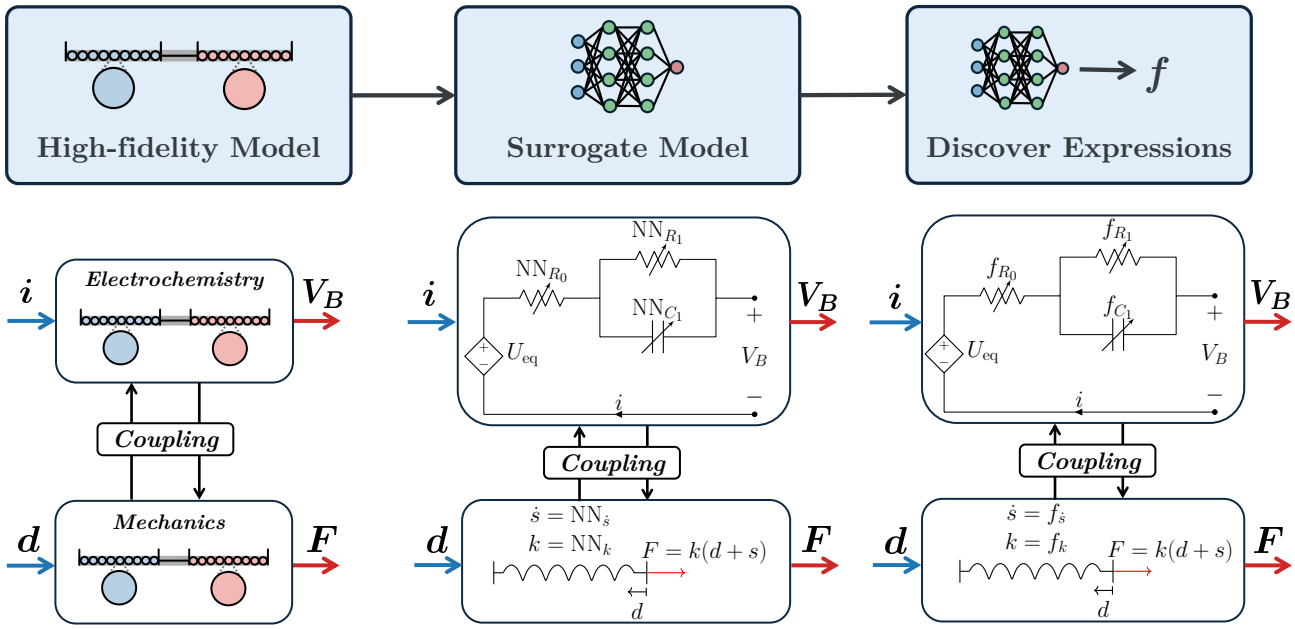
CHALMERS
UNIVERSITY OF TECHNOLOGY



Neural Network Enhanced Physics-based Surrogate Model Framework for Multiphysics Battery Simulations

Master's thesis in Physics

JESPER NOORD, JONATAN HARALDSSON

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's Thesis 2026

Neural Network Enhanced Physics-based Surrogate Model Framework for Multiphysics Battery Simulations

JESPER NOORD
JONATAN HARALDSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Computational Mechanics and Materials Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Neural network enhanced physics-based surrogate model framework for multiphysics battery simulations
JESPER NOORD & JONATAN HARALDSSON

Supervisors: David Carlstedt & Michele Godio, RISE Research Institutes of Sweden,
Fredrik Larsson, Chalmers University of Technology
Supervisor and examiner: Knut Andreas Meyer, Chalmers University of Technology

Master's Thesis 2026
Department of Industrial and Materials Science
Division of Computational Mechanics and Materials Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31-772 1000

Cover: Overview of the developed pipeline for the surrogate model framework. Illustrated with PowerPoint and TikZ.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Neural network enhanced physics-based surrogate model framework for multiphysics battery simulations

JESPER NOORD & JONATAN HARALDSSON
Department of Industrial and Materials Science
Division of Computational Mechanics and Materials Engineering
Chalmers University of Technology

Abstract

Physics-based battery models are important for understanding and predicting the behavior of batteries during operation. To fully describe their performance under different conditions high-fidelity multiphysics models are used. While these offer high accuracy, their computational cost is high. Simplified models, such as equivalent circuit models, provide a faster alternative. However, these models are less flexible and need to be recalibrated for different circumstances, often involving extensive testing.

This thesis presents a framework for using neural networks to enhance simplified physics-based models based on high-fidelity simulation data. Battery voltage and swelling force are simulated with a high-fidelity electrochemical-mechanical battery model for different applied currents and external pre-compression loads, while accounting for electrochemically induced swelling. The voltage response is then modeled with a first-order non-linear equivalent circuit model, and the mechanical response with a non-linear spring along with a swelling rate. Based on the high-fidelity data, neural networks are used to approximate the circuit elements, spring stiffness and swelling rate. Finally, symbolic regression is used to discover closed-form expressions for each neural network, revealing insights in dependencies.

The result suggests that highly non-linear circuit elements are required to capture the dynamics of battery voltage. Compared to a fully data-driven baseline, embedding neural networks within an equivalent circuit model preserved physical relations when extrapolating beyond the training domain. This work is limited to fully reversible swelling, and the voltage data considers only discharges. Although the thesis project demonstrates this framework on mechanically coupled battery simulations, we believe that the methodology and the overall workflow can be extended to include other processes, such as thermal effects or degradations in future work.

Keywords: *Multi-physics Surrogate Model, Batteries, Equivalent Circuit Models, Machine Learning, Symbolic Regression*

Acknowledgments

First of all, we want to dedicate a warm thanks to our supervisors at Rise, David and Michele. Your expertise and guidance provided throughout this work have been invaluable. We are grateful for your patience, openness, and for encouraging us to explore different ideas. Secondly, we also want to express our gratitude to our supervisor and examiner at Chalmers, Knut Andreas, for your involvement and dedication to the project. Your feedback on how to navigate the field of machine learning, equation discovery and modeling has been key for the project's outcome. We are also thankful to Fredrik for your support and interest in our work. In summary, we are truly thankful to have been surrounded by supervisors with such broad and complementary expertise. It has made this journey enriching, and given us great insight into life as an engineer and researcher. Thank you for this opportunity.

Moreover, both of us would like to extend an immense thanks to our family and friends. Not only throughout these last six months during this project, but throughout the years at Engineering Physics. Finally, we owe a huge thanks to our partners, Emma and Wilda, for your wholehearted support in everything we undertake, and for your acceptance when we had to stay late at the office. You give us the extra motivation and energy.

Jesper & Jonatan, Gothenburg 2026

Contents

1

Introduction

1.1 Background

As acknowledged by the Nobel Prize Committee in 2019, lithium-ion batteries (LiBs) have been at the heart of the revolution in modern portable electronic devices [1]. In fact, you might be reading this report on a device powered by a LiB. In addition to powering phones and laptops, LiBs are also viewed as a key technology in combating climate change by storing renewable energy in the electrical grid and in electric vehicles. When designing and developing products with LiBs, accurate models are essential to understand and predict battery performance under different conditions. Apart from the design stage, battery models are also needed for real-time applications, such as predicting the remaining battery capacity of an electric vehicle. While both use cases demand high accuracy, real-time analyses also benefit from a battery model with fast predictions.

Since LiBs chemically store electrical energy, coupled electrochemical models are often used to describe their behavior. However, in addition to electrochemical processes, mechanical and thermal effects are also be important to consider, adding another layer of complexity to high-fidelity battery models. Moreover, developing a high-fidelity model often requires detailed understanding of material parameters and how they develop in different scenarios, like increased temperature or mechanical loads. In contrast, there are simplified physics-based models that use circuits to approximate the response of a battery. Due to their simplicity, these equivalent circuit models (ECMs) are often used for real-time battery analyses [2]. Compared to high-fidelity models, ECMs can be calibrated with direct measurements of current and voltage, which means that no thorough investigation of a battery is needed. However, despite having low computational cost, ECMs sometimes fail to capture the strong non-linear nature of batteries. Unlike high-fidelity models, ECMs are also limited to electronic responses, as they do not inherently cover any thermal or mechanical effects. Common practice is, thus, to calibrate the ECMs separately at different temperatures and mechanical loads. To reach desired accuracy in predictions, an ECM in a real-world application is often accompanied by an extensive look-up table that covers possible scenarios (see [3] for an example). To obtain such a

look-up table, the battery responses have to be measured under many different conditions, which can be both expensive and time-consuming.

In contrast to physics-based models, advances in machine learning (ML) have opened up new possibilities to use data-driven models for modeling complex behavior at a lower computational cost. In this context, neural networks (NNs), which are central to the advancements in artificial intelligence, are especially prominent. That said, ML-based models (e.g. NNs) sometimes require large datasets to achieve satisfactory accuracy. Since ML models learn patterns by only looking at data, they have no inherent understanding of physics. In other words, there is no guarantee that a ML model will follow the laws of physics, which might lower the credibility of data-driven modeling approaches. However, to bridge the gap between physics-based models, on the one side, and ML models, on the other, scientific machine learning (SciML) have emerged as a middle-ground approach that combines physics and ML models with the ultimate goal of finding physically consistent models that use less computational resources [4].

In the context of batteries, surrogate models based on SciML have been developed and used to replicate the electrochemical responses at a lower computational cost (see Sec. 2.5 for a more thorough review). However, including multiphysics effects in surrogate modeling is less common, particularly mechanical effects induced during operation. We believe that the current thesis addresses this gap by presenting a framework that uses SciML to couple simple physics-based models in multiphysics scenarios.

1.2 Aim

This master thesis aims to demonstrate a framework for implementing a SciML surrogate model for coupled multiphysics battery simulations. The surrogate model is built on the structure of simpler physics-based models, with NNs calibrating their components. It aims to be an efficient and interpretable model capable of replicating the cell-level response of a high-fidelity battery model. A further aim is to discover how the components of the simplified models depend on different load cases, and find corresponding symbolic expressions. By discovering these relations, the framework could offer an alternative to the traditional ECM workflow. The framework is demonstrated for electrochemical-mechanical battery simulations considering different loads and internal swelling.

1.3 Scope and Limitations

To demonstrate the framework certain assumptions were made for both the high-fidelity model and the implemented surrogate model. The high-fidelity simulations were limited to

mechanically and electrochemically coupled models, neglecting thermal effects. The surrogate model predicts terminal voltage and reaction force under different applied currents and pre-compressions, accounting for electrochemically induced cyclic swelling. Moreover, single cycles were simulated, assuming fully reversible swelling, and neglecting long-term degradation processes. Although there are different materials and battery configurations, the framework is demonstrated for one type of LiB configuration. The framework was implemented in `Python` and `COMSOL` was used for the high-fidelity model.

1.4 Thesis Outline

This thesis report has the following outline:

Chapter 2, Theory:
The theory begins with giving an overview of batteries, emphasizing their working principle, high-fidelity battery models and equivalent circuit models as an alternative approach to modeling batteries. Next, we introduce mechanical models and how they couple to the electrochemical battery models. The subsequent part of the theory introduces ML, NNs and some approaches for SciML.

Chapter 3, Method:
The methodology is presented in the three main steps of the framework. Data generation from a high-fidelity coupled model, surrogate model training and uncovering closed-form expressions through symbolic regression.

Chapter 4, Results and Discussion:
The results and discussions mirrors the three main steps from the methodology. The chapter presents data from the high-fidelity model, followed by predictions of the surrogate model and lastly the discovered symbolic expressions.

Chapter 5, Conclusion and Future Outlook:
The conclusion presents a summary of the implemented framework and discusses its functionality. Lastly, suggestions for future work are proposed.

2

Theory

2.1 Overview of Batteries

First discovered on Utö in the Stockholm archipelago, Lithium (Li), the third lightest element and the least dense of all solids, is the main actor in LiBs [1]. Having only a single electron in the outer-most shell, Li has a low ionization energy, which together with its low mass makes it ideal as a charge carrier within the battery. In addition, a battery unit cell consists of a positive and a negative electrode (cathode and anode), separator and two current collectors. During discharge, Li-ions migrate from one electrode to the other, producing a current that can be used to drive an external load. However, this process is reversible, and applying an opposite current causes a reverse flow of Li-ions, effectively charging the battery.

In consumer products, such as phones, laptops and electric vehicles, a battery pack is made of several battery cells (often grouped into modules). Each cell consists of a multitude of repeatable unit cells. A full battery pack is typically on the cm to m scale and a battery cell on the mm to cm scale, whereas a unit cell is often on the µm scale, illustrated in Fig. 2.1. In other words, the scale difference between a unit cell and the entire battery pack can be several orders of magnitude.

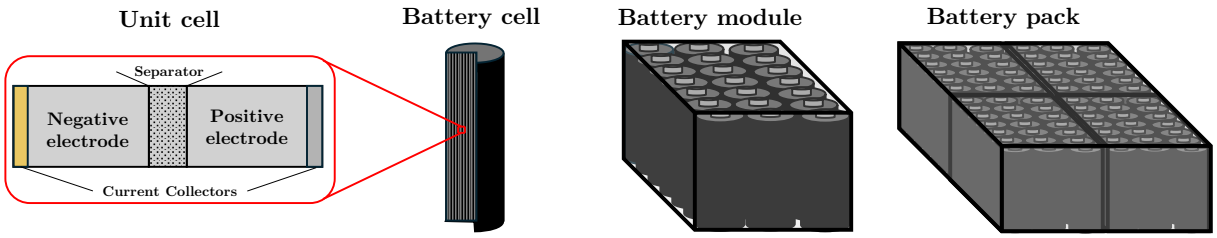


Fig 2.1: Battery hierarchy. A unit cell is the fundamental building block of a battery cell. The battery cell is built up of many sheets of unit cells consisting of two current collector layers, two electrode layers and one separator layer. Multiple battery cells can be arranged into a module and several modules form a battery pack.

2.1.1 Battery Components and Working Principle

The unit cell, shown schematically in Fig. 2.2, is the main electrochemical component of a LiB. In general, a unit cell is composed of a positive and negative electrode, divided by a porous separator. Both electrodes have an intricate porous structure that contains a connected network of solid particles. In-between the network, filling the pores, in both the electrodes and the separator is a Li salt solution, known as an electrolyte. When the two current collectors are connected, Li-ions are transported via the electrolyte from one electrode to the other. Since electrons are blocked by the separator, they flow in the external circuit, producing the current i; by convention, opposite to the electron flow.

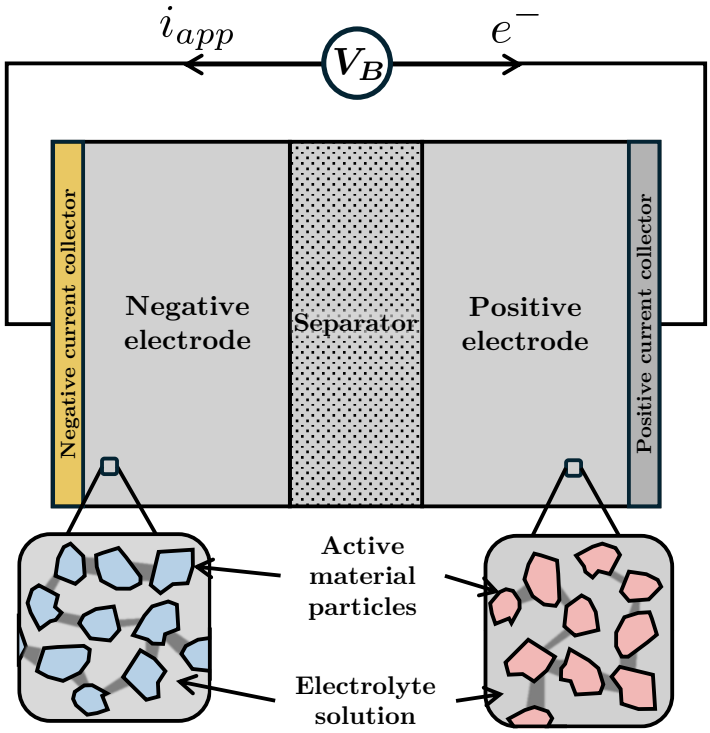


Fig 2.2: Schematic sketch of a unit cell, consisting of a positive and negative electrode, a separator and two current collectors. At the microscopic level, each electrode consists of active material particles, connected with binder that are submerged in an electrolyte solution. V_B denotes the cell voltage, under the applied current i_{app}. Note that electrons e^- and applied current flow in opposite directions.

In a LiB, each electrode has a solid phase and a liquid, electrolyte, phase*. The solid phase has a porous structure and consists of active material particles that are held together and connected by a binder (see Fig. 2.2). In the negative electrode, the active material is typically graphite, while transition metal oxides (e.g. Lithium Nickel Manganese Cobalt Oxide (NMC)) are common for the positive electrode [5]. Common materials for the

Solid-state batteries utilize a solid electrolyte instead of a liquid.

current collectors are Copper and Aluminum [6]. During operation, the active material stores and releases Li-ions through a process called (de)intercalation. Specifically, during the intercalation process Li-ions are inserted in-between the layers of the active material lattice [5]. Moreover, the binder, connecting the active material, has an electronically conductive coating, which enables the transport of electrons from the active material particles to the current collector [7].

Surrounding the electrode solid phase is the liquid electrolyte solution, whose main purpose is to facilitate the transport of Li-ions between the two electrodes. Importantly, Li-ions are soluble in the electrolyte, however, since the electrolyte is not electronically conductive the electrons remain in the solid phase, where they are transported via the binder. Furthermore, the porous separator, usually made from a polymers, also prevents electron transport between the electrodes [5].

In a fully charged battery, the active particles in the negative electrode have a high concentration of Li-ions, while the opposite is true for the positive electrode. By closing the circuit, electrons in the negative electrode are attracted to the positive electrode. This causes Li-ions to de-intercalate from the negative active material particle into the electrolyte. Next, the Li-ion, solved in the electrolyte, diffuses through the separator towards an active material particle in the positive electrode, where the Li-ion is intercalated. At the same time, the electron released during de-intercalation is transported from the negative- to the positive electrode through the external circuit, producing a current. Conversely, if an external current is applied in the opposite direction, electrons and Li-ions will flow from the positive- to the negative electrode, charging the battery. To summarize, the following reaction highlights the charging-discharging reactions

$$\mathrm{LiM} + \mathrm{C} \underset{\text{discharge}}{\overset{\text{charge}}{\rightleftharpoons}} \mathrm{Li}_{1-x}\mathrm{M} + \mathrm{Li}_x\mathrm{C}$$

where M denotes a metal oxide (positive electrode material), C is graphite (negative active material) and Li_x is a fraction $x \in (0,1)$ of Lithium [5].

2.1.2 Electrochemical Battery Models

High-fidelity electrochemical models that describe a battery during operation, can vary in how detailed the material matrices are resolved. A common simplification is the so-called Doyle-Fuller-Newman model (DFN), which simplifies the problem by assuming spherically symmetric active-material particles [8]. In effect, the dynamics within each particle is only resolved along the spherical radius $r \in [0, R]$ for some particle radius R. Furthermore, DFN can be employed in a three-dimensional geometry, producing the pseudo-four-dimensional (P4D) model (see e.g., [9]), or in a one-dimensional geometry,

giving the pseudo-two-dimensional (P2D) model. In both cases, the pseudo dimension refers to the radial dimension of the particle. For reference, a typical P2D setup is provided in Fig. 2.3, where each point along the negative electrode ($x \in [0, L_n]$) and positive electrode ($x \in [L_{\text{cell}} - L_p, L_{\text{cell}}]$) has an associated spherical active-material particle with characteristics that are resolved along r.

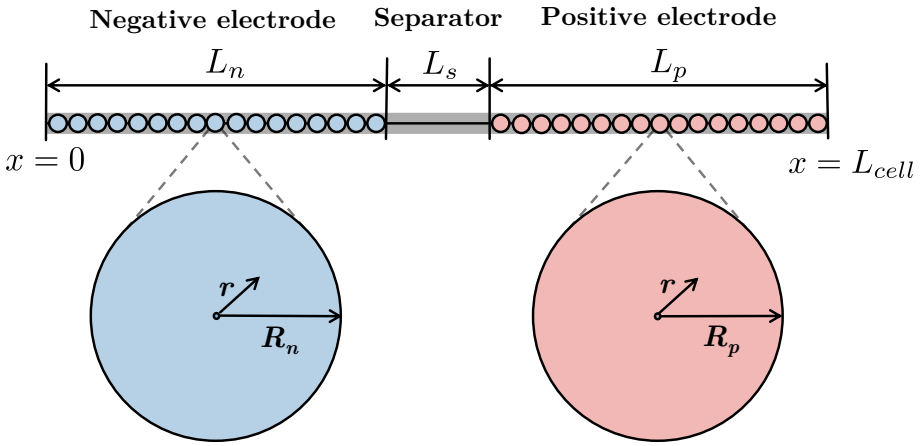


Fig 2.3: Schematic sketch of the P2D model. For each $x \in [0, L_n]$ at the negative electrode there is an active-material particle with radius R_n. Similarly, for the positive electrode ($x \in [L_{cell} - L_p, L_{cell}]$) there is an active-material particle with radius R_p.

To describe a battery using the P2D model, the overarching balance equations that govern the electrochemical response are charge and mass balance in both the electrolyte and the solid active material. The mass balance equations are governed by Fickian diffusion, accompanied by a constitutive relation for the ionic flux density. Similarly, the charge balance equations link the current density to the potential, using Ohm's law as the constitutive relation. The solid-electrolyte interface reactions are governed by the Butler-Volmer relation, which provides a connection between the balance equations in the electrolyte and solid phase. Note that the subsequent sections give a brief introduction to the governing equations for P2D. We refer to Brosa *et al.* [7], Torabi and Ahmadi [5], and Matty [6] for a more comprehensive review. For clarity, the index $l \in \{n, p\}$ denotes the negative and positive electrode, while the indices s and e denotes the solid and electrolyte phase respectively.

2.1.2.1 Electrochemical Mass Balance

Considering spherically symmetric active material particles, the conservation of Li in the solid phase of the electrodes is given by

$$\frac{\partial c_{s,l}}{\partial t} = -\frac{1}{r^2}\frac{\partial}{\partial r}\Big[r^2 N_{s,l}\Big], \tag{2.1}$$

where $c_{s,l} = c_{s,l}(r,x,t)$ is the radial concentration of Li-ions in the solid phase along $x \in [0,L_n]$ in the negative electrode and $x \in [L_p - L_{\text{cell}},L_{\text{cell}}]$ in the positive electrode. $N_{s,l} = N_{s,l}(r,x,t)$ denotes the radial flux of Li-ions within the active material particles. Next, the radial flux obeys the following constitutive relation

$$N_{s,l}(r,x,t) = -D_{s,l}(c_{s,l})\frac{\partial c_{s,l}}{\partial r},$$

where $D_{s,l}$ is the solid phase diffusion coefficient. To solve the mass balance, the following boundary/initial conditions apply,

$$N_{s,l}(r = 0,x,t)) = 0 \quad , \quad N_{s,l}(r = R_l,x,t)) = \frac{j_{BV}}{F} \quad \text{and} \quad c_{s,l}(r,x,t = 0) = c_{s,l,0}. \tag{2.2}$$

Here, j_{BV}, the interfacial current density given by the Butler-Volmer relation (see Eq. 2.11), and $F = eN_A$ is Faraday's constant.

Similar to the mass balance in the solid phase, conservation of Li-ions in the electrolyte is also governed by Fick's diffusion law, which reads

$$\frac{\partial(\nu_e c_e)}{\partial t} = -\frac{\partial N_e}{\partial x} + \frac{a_s j_{BV}}{F}. \tag{2.3}$$

In Eq. 2.3, $c_e = c_e(x,t)$ is the concentration of Li-ions in the electrolyte along $x \in [0,L_{\text{cell}}]$, and $N_e(x,t)$ is the Li flux in the electrolyte. Moreover, ν_e is the electrolyte volume fraction (or porosity), and $a_{s,l} = 3\nu_{s,l}/R_l$ is the active surface area for spherical particles with $\nu_{s,l}$ being the solid-phase volume fraction. Note that $j_{BV} = 0$ in the separator region, or $x \in [L_n,L_n + L_s]$.

Next, the ionic flux in the electrolyte is given by

$$N_e(x,t) = -D_e^{\text{eff}}\frac{\partial c_e}{\partial x} + \frac{t^+}{F}i_e, \tag{2.4}$$

where D_e^{eff} is the effective electrolyte diffusion coefficient, t^+ is the transference number (constant), and i_e is the electrolyte current density. D_e^{eff} is related to the porosity by the Bruggeman relation, giving $D_e^{\text{eff}} = D_e\nu_e^{\zeta}$ with ζ commonly set to $\zeta = 1.5$, however, should be determined from experiments [5]. When solving Eq. 2.3, the following boundary/initial conditions apply

$$N_e(x = 0,t) = N_e(x = L_{cell},t) = 0 \quad \text{and} \quad c_e(x,t = 0) = c_{e,0} = \text{const.}$$

2.1.2.2 Electrochemical Charge Balance

In addition to mass balance, the charge balance in the solid phase is given by

$$\frac{\partial i_{s,l}}{\partial x} = -a_{s,l} j_{BV}, \tag{2.5}$$

where $i_{s,l}$ is the solid current density in electrode l. A constitutive relation to the solid phase potential $\phi_s = \phi_s(x,t)$ is obtained through Ohm's law, which reads

$$i_{s,l} = -\kappa_{s,l}^{\text{eff}} \frac{\partial \phi_{s,l}}{\partial x}, \tag{2.6}$$

where $\kappa_{s,l}^{\text{eff}} = \kappa_{s,l} \nu_{s,l}^{\zeta}$ is the effective solid phase conductivity, corrected with the Bruggeman relation. Like the solid-phase charge balance, the electrolyte charge balance is given by

$$\frac{\partial i_e}{\partial x} = a_{s,l} j_{BV}. \tag{2.7}$$

Again, $j_{BV} = 0$ in the separator, $x \in [L_n, L_n + L_s]$. In this case, the corresponding constitutive relation, connecting i_e and the electrolyte potential $\phi_e(x,t)$ is

$$i_e = -\kappa_e^{\text{eff}} \frac{\partial \phi_e}{\partial x} + \kappa_D^{\text{eff}} \frac{\partial [\ln(c_e)]}{\partial x}, \tag{2.8}$$

where κ_e^{eff} and κ_D^{eff} are the effective electrolyte conductivity and the effective diffusion conductivity of the electrolyte respectively. These effective quantities are also corrected with the Bruggeman relation, yielding $\kappa_e^{\text{eff}} = \kappa_e \nu_e^{\zeta}$ and $\kappa_D^{\text{eff}} = \kappa_D \nu_e^{\zeta}$.

For charge balance equations, the following boundary conditions are assumed:

$$\begin{cases} \phi_{s,n}(x = 0,t) = 0 & i_{s,n}(x = L_n,t) = 0, \\ i_{s,p}(x = L_{\text{cell}} - L_p,t) = 0 & i_{s,p}(x = L_{\text{cell}},t) = i_{app}(t)/A_{\text{cell.}}, \end{cases} \tag{2.9}$$

where A_{cell} is the surface area of the cell, and $i_{app}(t)$ is the applied current. Henceforth, the applied current will be referred to as $i(t)$.

Finally, the difference in solid-phase potential measured at the boundary of the electrodes gives the cell voltage V_B,

$$V_B(t) = \phi_{s,p}(x = L_{cell},t) - \underbrace{\phi_{s,n}(x = 0,t)}_{=0,\ \text{Eq. 2.9}} = \phi_{s,p}(x = L_{cell},t). \tag{2.10}$$

2.1.2.3 Solid-electrolyte Interfacial Reaction

Connecting the solid-phase diffusion in Eq. 2.1 to the charge balance and the electrolyte mass balance in Eq. 2.5, 2.7 and 2.3, the Bulter-Volmer interfacial current density, j_{BV}, is given by

$$j_{BV} = j_0 \left[\exp\left(\frac{\alpha_n F \eta_l}{\mathcal{R}T}\right) - \exp\left(\frac{\alpha_p F \eta_l}{\mathcal{R}T}\right) \right], \tag{2.11}$$

where $j_0 = j_0(c_{s,l}, c_e)$ is the so-called exchange current density, $\mathcal{R}$ is the universal gas constant, T the temperature, and α_l is a charge transfer coefficient for electrode, typically set to 0.5 [6]. Lastly, η_l denotes the overpotential in electrode l, defined as

$$\eta_l = \phi_{s,l}(x,t) - \phi_e(x,t) - U_l, \tag{2.12}$$

where U_l refers to the intrinsic equilibrium, open-circuit potential for electrode l. Generally, U_l depends on $c_{s,l}(r = R_l, x, t)$, the Li concentration at the surface of the active material particles.

2.1.2.4 Other Important Battery Definitions

In addition to the governing equations, this section introduces the commonly used properties: open-circuit voltage (OCV), state of charge (SOC), and charge rate (C-rate). Firstly, from the open-circuit potential, U_l, the OCV is defined as $U_{eq} = U_p - U_n$. In practice, U_{eq} corresponds to the cell voltage measured at equilibrium, i.e., at $i = 0$. With OCV the cell voltage can be obtained by

$$V_B = U_{eq} - \eta_{\text{tot}}, \tag{2.13}$$

where η_{tot} denotes the total overpotential.

Secondly, the SOC describes the remaining capacity of a battery in relation to the total nominal capacity. Thus, $\text{SOC} \in [0\,\%, 100\,\%]$, and it is typically computed with Coulomb counting

$$\frac{\mathrm{dSOC}}{\mathrm{d}t} = -\frac{i(t)}{Q_0} \Rightarrow \text{SOC}(t) = \text{SOC}(t_0) - \int_{t_0}^{t} \frac{i(t')}{Q_0}\,\mathrm{d}t', \tag{2.14}$$

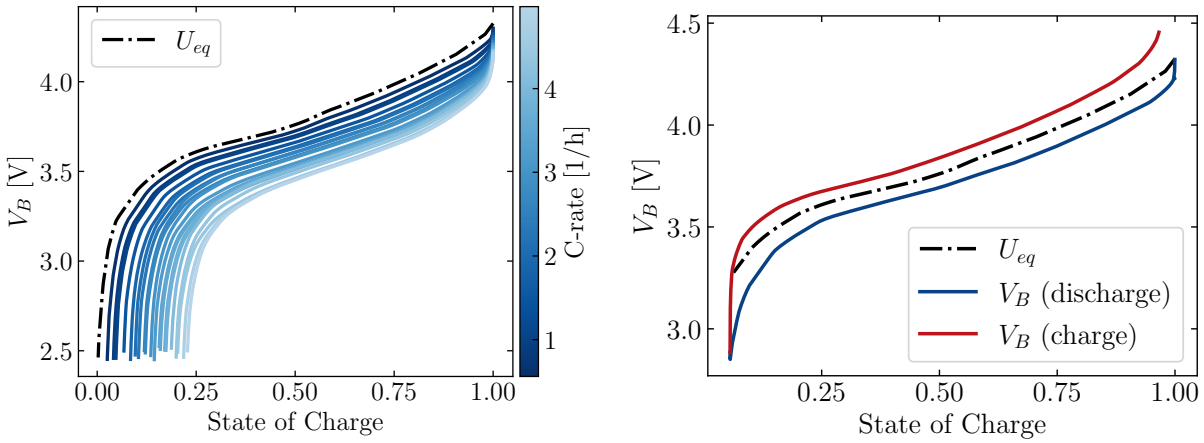
where Q_0 is the nominal cell capacity in A s and t_0 is some starting time [10]. Lastly, by normalizing current with the total discharge time and capacity, the so-called C-rate is obtained by

$$\text{C-rate} = \frac{i(t)}{Q_0}, \tag{2.15}$$

for Q_0 in A h. Put simply, the battery discharges in 1 h for a C-rate $= 1\,\text{h}^{-1}$.

2.1.2.5 Voltage Curves and Polarization

The discharge/charge curve, or the terminal voltage as a function of SOC or time, depends on the characteristics of the battery, and external factors such as the C-rate, stresses, and temperature. As observed in FIG. 2.4a, which shows discharge curves at constant C-rate $\in [0.5, 5]$ along with U_{eq}, increased applied currents cause the battery voltage to deviate further from the equilibrium voltage U_{eq}. This deviation in voltage is referred to as polarization, and its magnitude as the overpotential [11]. In addition, a full cycle with both discharge and charge is shown in FIG. 2.4b. Importantly, the battery voltage is greater than the equilibrium voltage during charge.



(a) (b)

FIG 2.4: (a) Discharge curve as a function of SOC for constant C-rates between 0.5 and 5 along with U_{eq}. (b) Discharge with constant C-rate $= 1\,\mathrm{h}^{-1}$ followed by a charge at constant C-rate $= -1\,\mathrm{h}^{-1}$ along with U_{eq}.

However, standard use of a battery might not always involve constant C-rates, instead the applied current may be alternating or switched off. In these cases, suddenly switching of the current causes the cell voltage to gradually relax towards U_{eq}, which corresponds to η_{tot} slowly decaying to zero (see EQ. 2.13). To illustrate, FIG. 2.5 shows the voltage response from an applied square wave current profile with a period of 300 s, simulated in `COMSOL`. More formally, the total overpotential can be split into three different terms, describing different physical phenomena, occurring at different time scales and spatial locations within the cell, giving

$$V_B = U_{eq} - [\eta_{\text{ohm}} + \eta_{\text{act}} + \eta_{\text{conc}}]\,, \tag{2.16}$$

where η_{ohm} denotes ohmic polarization, η_{act} activation polarization and η_{conc} concentration polarization [5].

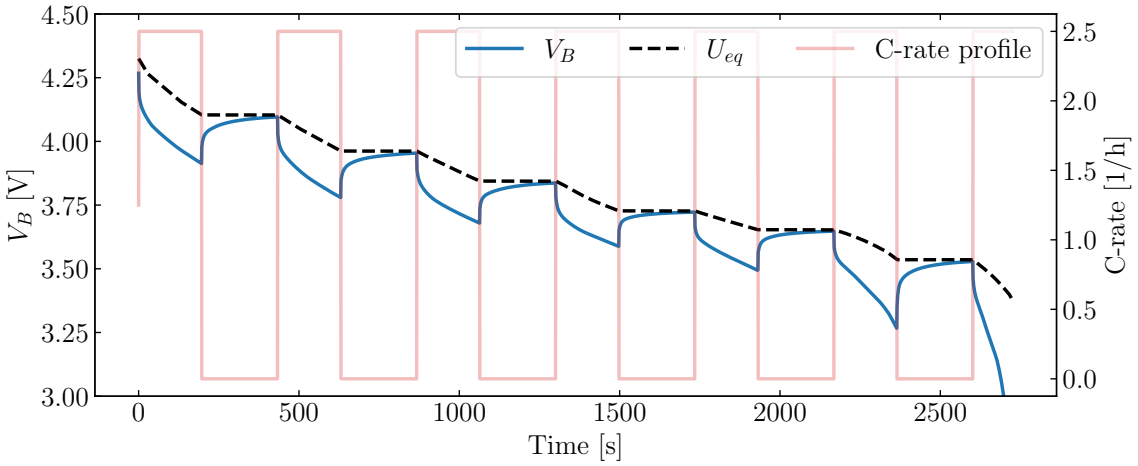


FIG 2.5: Discharge curve with a square wave C-rate profile along with U_{eq}.

Firstly, the ohmic polarization responds instantaneously to a change in applied current and is caused by an internal resistance in the battery. It is proportional to the applied current and is therefore conveniently modeled as a voltage drop over a resistor, with losses influenced by the electrolyte's ionic conductivity and the solid phase's electronic conductivity [12]. Secondly, the activation polarization responds relatively quickly and arises from the kinetic energy barrier for electrochemical redox (charge transfer) reactions at the electrode-electrolyte interface, modeled by the Butler-Volmer kinetics [13]. Thirdly, the slowest contribution is the concentration polarization [13]. This polarization is a result of concentration gradients that gradually builds up between the solid phase and the electrolyte, which is a process governed by the slow diffusion of lithium ions [14]. Activation and concentration combined produce the overpotential that develops slowly when the battery is put under load and relaxes towards U_{eq} when the load is removed, even at zero current and constant SOC, as shown in FIG. 2.5 [2]. The decay in overpotential under zero load can be seen as the diffusion process relaxing towards equilibrium following Fick's diffusion law EQ. 2.3.

Moreover, studying the polarizations as a function of time or SOC gives an insight into their individual contribution to the total overpotential, which can be used to motivate the properties and behavior of certain parameters when modeling battery voltage. This is especially important when designing simpler models based on equivalent circuits.

2.1.3 Equivalent Circuit Models

In contrast to electrochemical battery models, ECMs are alternative physics-based models, where the current-to-voltage response is modeled with a circuit. Since V_B is found by deviations from U_{eq}, U_{eq} is commonly modeled as a controlled voltage source in the circuits, whereas the overpotential is modeled with other circuit elements. A simple form of ECM

adds a linear resistor R_0, which accounts for instantaneous ohmic losses as iR_0 during discharge, according to Ohm's law (see Fig. 2.6). Note that the sign convention used here is $i > 0$ during discharge and $i < 0$ during charge. With $R_0 > 0$, this model will by design always give a $V_B < U_{eq}$ during discharge and $V_B > U_{eq}$ for charge. However, as argued, the voltage deviation from U_{eq} develops over time. Therefore, this ECM is often extended.

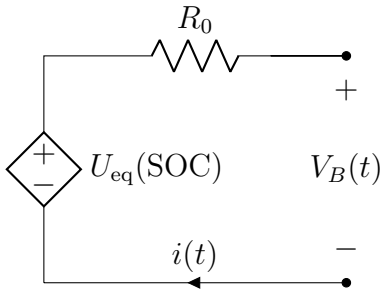


Fig 2.6: Circuit diagram of simple ECM, where U_{eq} is modeled with a SOC-dependent, controlled voltage source and R_0 represent instantaneous ohmic losses.

Building on the circuit in Fig. 2.6, a first-order ECM with parameters R_0, R_1 and C_1 is shown in Fig. 2.7. By introducing the RC-branch, the goal is to capture the relaxation of the overpotential more accurately, see Fig. 2.5. V_B can be obtained as

$$V_B = U_{eq} - iR_0 - V_1, \quad \text{where} \quad \frac{\mathrm{d}V_1}{\mathrm{d}t} = -\frac{V_1(t)}{R_1C_1} + \frac{i(t)}{C_1}. \tag{2.17}$$

For convenience, a time-constant $\tau_1 = R_1C_1$ can be introduced. Solving Eq. 2.17 for a constant C-rate, $i(t) = I_0$ gives

$$V_B = U_{eq} - I_0(R_0 + R_1(1 - e^{-t/\tau_1})). \tag{2.18}$$

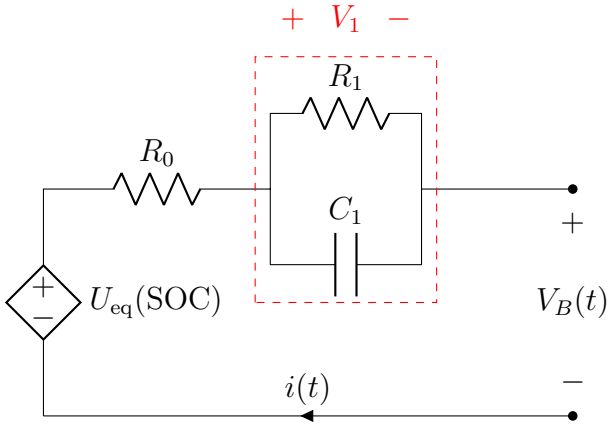


Fig 2.7: Circuit diagram of a first order ECM, where V_1 is the voltage over R_1 and C_1. U_{eq}, assumed to be a function of SOC, is modeled with a controlled voltage source, while R_0 represent instantaneous ohmic losses.

In addition, the ECM in FIG. 2.7 could be extended with further RC-branches, connected in series. For instance, introducing a second RC-branch, R_2 and C_2, gives the additional time-constant $R_2C_2 = \tau_2$. Note that this is one example of an ECM, and there are other possible ways of configuring equivalent circuits.

When the elements are treated as constants, capturing the end-of-discharge increase in overpotential might be not possible with this standard ECM. While the RC-branch covers the initial transient, it can fail to capture the polarization at the end of discharge. In practice, the resistor and capacitor values of an ECM vary with stress, temperature, C-rate and SOC, thus, a common approach is to estimate the ECM in each particular case.

2.1.4 Mechanical Material Models

Stress σ describes forces distributed over a surface area within a material body due to mechanical loads. External forces on a body can induce several types of internal force distributions (stress). Compression, naturally, induces compressive stress. The opposite external loads, which would be pulling forces, induces so called tensile stress. Moreover, the act of a "sliding" force distribution induces shear stress τ in the material.

A more formal definition of these stresses arises from dividing the force distribution in normal and tangential components. For simplicity, lets start with denoting x the normal direction and y, z the tangential directions. In this notation the compressive or tensile stress arises from the force per unit area in the normal direction σ_x. Whereas, the force distribution tangential to the surface results in shear stress τ_{xy}. Since stresses are point functions the following relations are revealed

$$\sigma_x = \frac{\mathrm{d}F_x}{\mathrm{d}A_x}, \quad \tau_{xy} = \frac{\mathrm{d}F_{xy}}{\mathrm{d}A_x} \quad \text{and} \quad \tau_{xz} = \frac{\mathrm{d}F_{xz}}{\mathrm{d}A_x}. \tag{2.19}$$

To describe the state of stress on any surface it is necessary to define the stress components in three dimensions. More precise, normal and tangential stress in three orthogonal directions (preferably the Cartesian coordinate basis (x, y, z)), see FIG. 2.8. This formulation gives rise to the stress tensor

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix}. \tag{2.20}$$

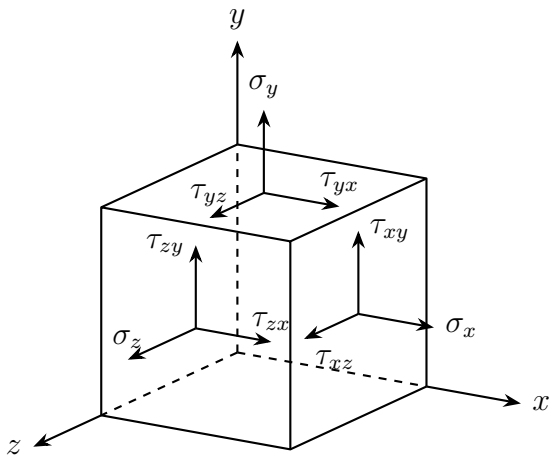


FIG 2.8: Stress on three orthogonal surfaces for a three dimensional unit cube, defined in Cartesian coordinates.

In a similar manner, normal strain ε and shear strain γ exists within any body under load. Normal strain describes the rate of change in length caused by normal stresses, and shear strain measures distortion or change in angle between orthogonal planes within the material caused by the tangential stresses. For small displacements, where $\boldsymbol{u}$ is the displacement field, the new length in x direction

$$\mathrm{d}x' \approx \mathrm{d}x + \frac{\partial u_x}{\partial x}\mathrm{d}x. \tag{2.21}$$

Thereby, normal strain and shear strain (for sufficiently small deformation angles) is defined as

$$\varepsilon_x = \frac{\partial u_x}{\partial x}, \quad \gamma_{xy} = \gamma_{yx} = \frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}, \quad \text{and} \quad \gamma_{xz} = \gamma_{zx} = \frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} \tag{2.22}$$

keeping the simplified notation of x as the normal direction [15]. Furthermore, this can be extended to a strain tensor analogous to $\boldsymbol{\sigma}$,

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{xz} \\ \frac{1}{2}\gamma_{yx} & \varepsilon_y & \frac{1}{2}\gamma_{yz} \\ \frac{1}{2}\gamma_{zx} & \frac{1}{2}\gamma_{zy} & \varepsilon_z \end{bmatrix} \tag{2.23}$$

In the case of linear elasticity, the constitutive relation between stress and strain is given by the tensor product

$$\boldsymbol{\sigma} = \mathbf{E} : \boldsymbol{\varepsilon}, \tag{2.24}$$

referred to as Hooke's law, where $\mathbf{E}$ is the elasticity tensor. If $\mathbf{E}$ is constant for all values

$\boldsymbol{\varepsilon}$, the material is linear elastic. On the other hand, $\boldsymbol{\sigma}$ could have a non-linear relation to $\boldsymbol{\varepsilon}$, for example a hyper-elastic behavior. Similar to the linear case, hyper-elasticity still assumes no dissipation, which means that energy is conserved. A common approach is to describe the hyper-elastic behavior in terms of strain energy density [16].

For a linear elastic material under one-dimensional constraints, the constitutive relation between stress and strain (Hooke's law) can be reduced to

$$F = k \cdot d, \tag{2.25}$$

where F is the reaction force, d the displacement and the stiffness k is represented as a single parameter which describes the continuum description above through the one-dimensional Young's modulus E, as $k = EA/L_{\text{cell}}$. Similar to the ECMs for the electrochemical equations, a corresponding equivalent mechanical model (EMM) is to use a spring as a material model, see Fig. 2.9. In the linear elastic case, the spring stiffness k is constant for all values of d. Conversely, in a hyper-elastic scenario, the displacement-to-force relation is non-linear, effectively giving k as a function of d.

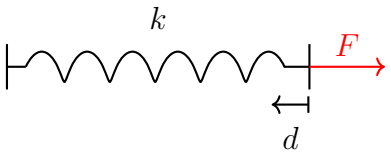


Fig 2.9: Illustration of a spring with stiffness k that is displaced a distance d, giving rise to a force F.

2.1.5 Electrochemical-Mechanical Battery Models

In addition to the electrochemistry, mechanically coupled models can give insight into structural degradation and its effect on performance. To understand how a battery deforms during operation, one important aspect is lithiation-induced swelling and shrinking of the electrode active material. Swelling/shrinking of the active material particles is directly linked to the intercalation process, and therefore to the concentration of Li [17]. Put simply, most active materials swell as more Li-ions are inserted in-between its layers. Thus, the increased concentration of Li leads to a volumetric expansion of active material particles, as illustrated in Fig. 2.10. From a modeling perspective, swelling in the active material of the electrode can be modeled as hygroscopic swelling, where electrode volumetric strain depends on average lithium concentration as

$$\varepsilon_{l,\text{vol.}} = \beta_l M_l (\bar{c}_{s,l} - \bar{c}_{s,l}^{\text{ref.}}) \quad l \in \{n, p\}. \tag{2.26}$$

Here, M_l is the molar mass of the active material, β_l is a swelling coefficient, and $\bar{c}_{s,l}$ denotes the average Li concentration within the active material of electrode l. Further, $\bar{c}_{s,l}^{\text{ref.}}$ is a reference concentration, for which the material is assumed to have zero volumetric strain. Even though swelling occurs in the active material particles, this modeling approach considers a volumetric strain in the entire electrode matrix; both the solid and electrolyte phases. With this, the electrolyte is considered incompressible, which further gives constant electrolyte and solid phase volume fractions. To summarize, Eq. 2.26 provides a one-way coupling between electrochemical and mechanical responses, where the average concentration gives a volumetric strain, $\varepsilon_{l,\text{vol.}}$, in the entire electrode.

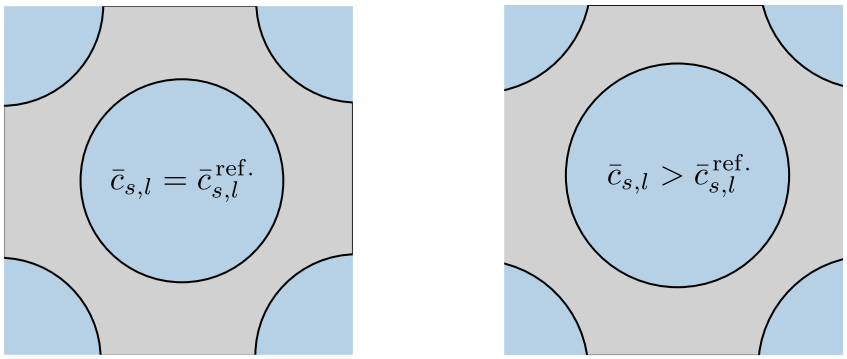


Fig 2.10: Sketch of the electrode matrix at different solid phase Li concentrations. To the right, increased $\bar{c}_{s,l}$ has caused a volumetric expansion of the active material particles and the surrounding matrix.

Next, the reverse coupling, from mechanics to electrochemistry, can be introduced by considering a strain-dependent porosity (electrolyte volume fraction) ν_e in the separator region. Since the separator connects the two electrodes, electrode swelling also causes a volumetric strain in the separator. In this case, a possible modeling approach is to consider the following strain-dependent porosity,

$$\nu_e = \nu_{e,0}(1 - \varepsilon_{\text{vol.}}), \tag{2.27}$$

where $\varepsilon_{\text{vol.}}$ denotes the volumetric strain for the separator, and $\nu_{e,0}$ is the zero-strain porosity. Introducing a dynamically changing porosity affects some electrochemical processes in the separator ($x \in [L_n, L_n + L_s]$), namely, the effective diffusion coefficient and ion conductivity in Eq. 2.4 and Eq. 2.8. Substituting Eq. 2.27 in to the expression for the effective diffusion coefficient and ion conductivities yields

$$\begin{aligned}
D_e^{\text{eff}} = D_e \nu_e^{\zeta} &\longrightarrow D_e^{\text{eff}} = D_e(\nu_{e,0}(1 - \varepsilon_{\text{vol.}}))^{\zeta}, \\
\kappa_e^{\text{eff}} = \kappa_e \nu_e^{\zeta} &\longrightarrow \kappa_e^{\text{eff}} = \kappa_e(\nu_{e,0}(1 - \varepsilon_{\text{vol.}}))^{\zeta}, \\
\kappa_D^{\text{eff}} = \kappa_D \nu_e^{\zeta} &\longrightarrow \kappa_D^{\text{eff}} = \kappa_D(\nu_{e,0}(1 - \varepsilon_{\text{vol.}}))^{\zeta}.
\end{aligned}$$

Moreover, a dynamic change in porosity also influences the electrolyte mass balance (Eq. 2.3). In summary, Eq. 2.26 and Eq. 2.27 introduce a two-way coupling between electrochemistry and mechanics.

2.2 Machine Learning

This section introduces ML, NNs, and approaches for combining prior knowledge about physics and NNs. The first two subsections are based on [18].

2.2.1 Data-driven Models and Finding Model Parameters

The main purpose of modeling is to find relationships between some dependent variables $\boldsymbol{y} = [y_0, y_1, ..., y_N]$ and independent variables $\boldsymbol{X} = [X_0, X_1, ..., X_N]$. An attempt to do this, is by constructing a model, M, such that it satisfies $\boldsymbol{y} = M(\boldsymbol{X})$. In general, M relates $\boldsymbol{y}$ to $\boldsymbol{X}$ with a set of adjustable or fixed model parameters $\boldsymbol{\theta}$. To illustrate, a simple model would be the following linear model

$$y_i = M(X_i; \boldsymbol{\theta}) = \theta_0 + \theta_1 X_i,$$

which relates the data point X_i to y_i via $\boldsymbol{\theta} = [\theta_0, \theta_1]$. However, in practice, a model never fully describes reality, thus, a more accurate description is $\boldsymbol{y} = M(\boldsymbol{X}; \boldsymbol{\theta}) + \delta M$, where $\delta M = \boldsymbol{y} - M(\boldsymbol{X}; \boldsymbol{\theta})$ is the model discrepancy or error.

For a given model, the goal of ML is to find a set of model parameters $\boldsymbol{\theta}^*$ that minimizes the error. However, comparing predictions $\hat{\boldsymbol{y}} = M(\boldsymbol{X}; \boldsymbol{\theta})$ with data $\boldsymbol{y}$ can be made more formal by employing so-called *loss functions*. Considering a set of N_D data points, some common examples include mean absolute error (MAE), mean squared error (MSE) and the root-mean-squared error (RMSE), which read

$$\text{MAE: } \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_D} \sum_{i=1}^{N_D} |y_i - \hat{y}_i(\boldsymbol{\theta})| \tag{2.28}$$

$$\text{MSE: } \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_D} \sum_{i=1}^{N_D} (y_i - \hat{y}_i(\boldsymbol{\theta}))^2 \tag{2.29}$$

$$\text{RMSE: } \mathcal{L}(\boldsymbol{\theta}) = \sqrt{\frac{1}{N_D} \sum_{i=1}^{N_D} (y_i - \hat{y}_i(\boldsymbol{\theta}))^2}. \tag{2.30}$$

By design, MAE and MSE loss functions treat errors slightly different, where the latter is more sensitive to outliers compared to the former. Utilizing loss functions, the optimal

set of model parameters is found by solving the optimization problem

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\mathcal{L}(\boldsymbol{\theta})\}. \tag{2.31}$$

Even though $\boldsymbol{\theta}$ is unknown, finding $\boldsymbol{\theta}^*$ with Eq. 2.31 still requires a known model structure that could be used to obtain $\hat{\boldsymbol{y}}$. Without prior knowledge about the underlying model structure, highly general and flexible models such as NNs provide an alternative approach that could be used within the same framework.

2.2.2 Artificial Neural Networks

In scenarios where the choice of model is difficult, artificial neural networks (ANNs), which has a highly flexible model structure, offer a way forward. Each neuron receives and manipulates inputs X to produce outputs $\hat{y}$. Like biological neurons, an artificial neuron fires only if the sum of all input, X, exceeds a threshold, usually referred to as an activation [18]. To illustrate, the output of an artificial neuron is computed as

$$\hat{y} = \mathcal{A}\left(\sum_{i=1}^{N} w_i X_i + b\right) = \mathcal{A}(z), \tag{2.32}$$

where w_i is a weight associated with input X_i and b is a bias term. Before computing the output, the weighted sum, z, is inserted into an activation function $\mathcal{A}$, which has the purpose of guiding when the neuron should fire and what values it takes [18]. Thus, the activation function provides a good basis for constraining a NN. When constructing NNs, the choice of activation function is often guided by the problem at hand. Fig. 2.11 show some common examples, i.e. sigmoid, tanh, rectified linear unit (ReLU), and softplus, which are given by

$$\text{sigmoid}(z) = \frac{1}{1+e^{-z}}, \qquad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}},$$
$$\text{ReLU}(z) = \max\{0,z\}, \qquad \text{softplus}(z) = \ln\left(1+e^{z}\right).$$

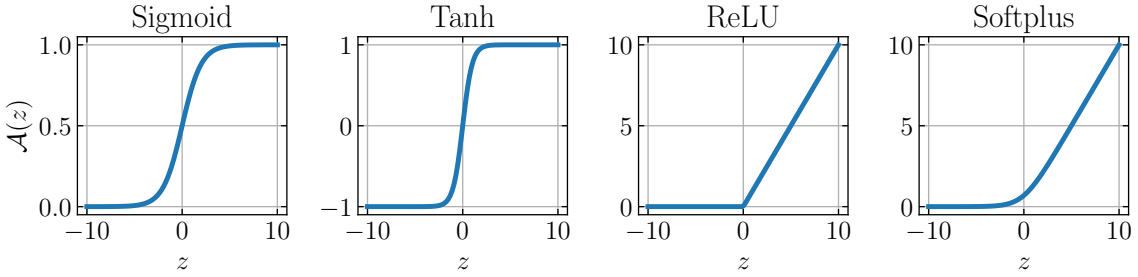


Fig 2.11: Sigmoid, tanh, ReLU, and softplus as a function of the weighted sum z.

In practice, ANNs are often arranged in a layered structure such that the output from prior neurons is fed into neurons in a subsequent layer, forming a so-called feed forward neural network (FFNN). ANNs with one or more layers in-between the input and output layer (hidden layers) are referred to as deep artificial neural network (DNN). As an example, a simple DNN with two inputs, two outputs and two hidden layers with four neurons each is shown in Fig. 2.12. Mathematically, the total outputs $\hat{\boldsymbol{y}}$ are computed by iterating Eq. 2.32 over all layers. Thus, a DNN is effectively a series of nested operations. Due to this nested structure, DNNs (henceforth both FFNNs and DNNs are referred to as just NNs) are fully differentiable with respect to both the parameters and the inputs, which is key when tuning the network based on data.

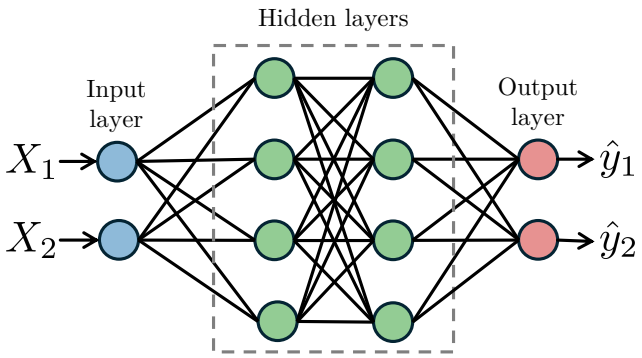


Fig 2.12: Schematic deep neural network where each circle represents a neuron. The inputs X_1 and X_2 enters the model at the input layer, proceeds through the hidden layers, and exits at the output layer. This process produces outputs $\hat{y}_1$ and $\hat{y}_2$.

For NNs, the overall goal is again an optimal set of parameters (following Eq. 2.31) based on some metric/loss $\mathcal{L}$. Employing the nomenclature used in the field of NNs, the model parameters $\boldsymbol{\theta}$ are usually referred to as weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$, and the tuning process is referred to as training. In short, NNs can be trained as follows,

0. initialize the parameters randomly,
1. make a prediction (forward pass) based on inputs $\boldsymbol{X}$ and $\boldsymbol{\theta}$

$$\hat{\boldsymbol{y}} = \mathrm{NN}(\boldsymbol{X}; \boldsymbol{\theta}),$$

2. compare $\hat{\boldsymbol{y}}$ to data $\boldsymbol{y}$ by computing the loss $\mathcal{L}(\boldsymbol{\theta})$,
3. compute $\vec{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ and adjust $\boldsymbol{\theta}$ accordingly (backward pass)

$$\boldsymbol{\theta}_{\mathrm{new}} = \boldsymbol{\theta}_{\mathrm{old}} - \varrho \vec{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\mathrm{old}}),$$

   with ϱ being the step size or *learning rate*,
4. return to step 1 and repeat the desired number of times or until $\mathcal{L}(\boldsymbol{\theta}) < \varepsilon_{tol}$ for some tolerance ε_{tol}.

The total number of iterations through step 1 to 3, referred to as *epochs*, is usually set in relation to the available computational resources and the amount of data. As shown above, the optimization scheme is based on gradient descent, taking advantage of the fact that NNs are differentiable with respect to the parameters. Naturally, with deeper and wider NNs, the number of parameters increases drastically, and while a larger parameter space generally improves the model flexibility, computing high-dimensional gradients is often expensive. In general, there is no silver bullet when choosing between high number of parameters and efficient optimization, it instead comes down to the particular problem.

Lastly, to further justify the use of NNs, NNs fulfill the so-called Universal Approximation Theorem, which states that any continuous function can be modeled to the desired degree of accuracy with a sufficiently large NN [19]. That said, even deeper NNs may still show limitations in extrapolating to data points far outside the training domain. This lack of generalizability could be conquered by constraining the NN. For instance, prior knowledge about the underlying physics or other hypotheses about the model outputs could be used to guide the model in areas not covered by the existing data.

2.3 SciML: Combining Physics and ML

In the field of SciML, Faroughi *et al.* presents a convenient notation to navigate through some of the different ways physics and ML are combined [20]. Based on NNs, these categories are: physics-guided NNs, physics-informed neural networks (PINNs), physics-encoded neural networks (PENNs), and neural operators (NOs).

Firstly, physics-guided NNs represent the simplest form of NN technique, where the introduction of physics consistency lies solely within the inputs and outputs, which are generated through experiments or simulations [20]. In other words, the network structure itself does not promote any physics. Next, the PINN architectures introduce the physical relationships with an additional *physics loss* to the loss function. With this setup, the network is penalized if it makes predictions that do not obey physical relationships in the physics loss. Despite their popularity, the performance of PINNs in terms of handling sparse data and generalization have been surpassed by PENN-based models, which have hard-coded the physical relations into the architecture itself [20]. While both PINNs and PENNs are reviewed below, readers particularly interested in NOs are referred to [21]. Finally, before exploring PINNs and PENNs, we would like to clarify what we mean by the term *physics*.

2.3.1 Explaining *Physics*

In the context of modeling, and ML in particular, the term physics is used to represent true underlying assumptions and constraints, however, it could also refer to sought-after characteristics usually found in laws of physics (idea from [22]). Firstly, physical laws are often highly *generalizable*, meaning that they can be applied in many scenarios. For instance, $F = ma$ may be used to derive planetary motion, but also the motion of a mass on a spring. Secondly, physical laws often tend to be simple, making them *easy to interpret*. Again, Newton's second law only consists of three quantities: acceleration, mass, and force, where mass is strictly positive. Even though ML models based on NNs can be complex, we believe the overall modeling approach should be inspired by the generalizable and easily interpretable models found in physics.

2.3.2 Physics-informed Neural Networks

In contrast to data-driven NNs, PINNs, introduced by Lagaris *et al.*, promotes physics through additional terms in the loss function [23]. Beneficially, NNs are differentiable with respect to their inputs, which means that the physics loss could include derivatives. Thus, a common use case for PINNs is to learn the solutions to partial differential equations (PDEs) [24]. The setup, schematically illustrated in Fig. 2.13, includes the PDE residual in the so-called physics loss. The physics and data loss are balanced with the scaling coefficient α_P. As an example, if $y(x,t)$ is a scalar field obeying the diffusion equation, the corresponding residual is given by

$$\text{Res}(y; x,t) = \frac{\partial y}{\partial t} - D\frac{\partial^2 y}{\partial x^2}.$$

During training, the PINN is penalized when Res $\neq 0$, which promotes the model to follow the underlying PDE. However, there are still no guarantees that model will produce solutions according to the PDE. As a final note, another interesting application of PINNs is that they can be trained without data, and only with the physics loss. This way, PINNs can be used to learn the spatio-temporal solution to a PDE.

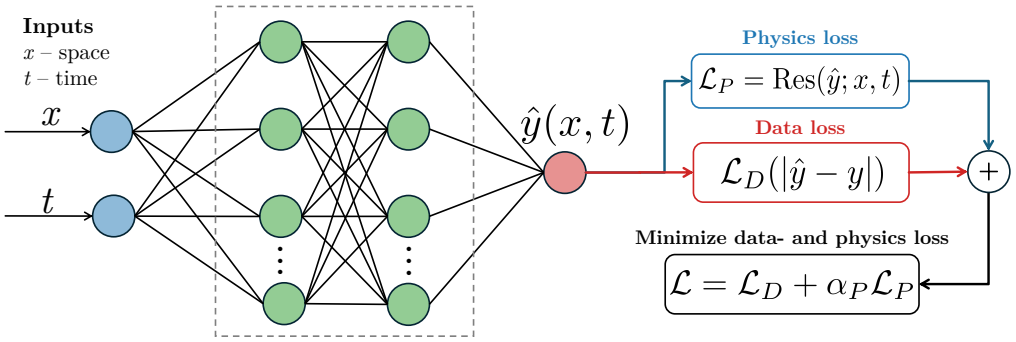


FIG 2.13: Schematic illustration of a PINN architecture with a spatial, x, and a temporal, t, input variable. The network output, $\hat{y}(x,t)$, is compared to data, y, in the data loss and to the PDE residual Res in the physics loss.

2.3.3 Physics-encoded Neural Networks

Instead of promoting physics with the loss function, PENNs hard-code physical relationships within the architecture itself, such that the physical laws are fulfilled per construction. Although implementing PENNs can lead to increased architectural complexity, they could offer improved robustness to data scarcity and a stronger ability to generalize compared to PINNs [20].

There are several proposed ways of encoding physical constraints, such as balance laws and constitutive laws, into NNs. The simplest, and perhaps elementary example, is "sensible" choice of activation functions on the last, output layer. With this approach, one can ensure that the output obtains the correct sign or lies within a specific range. As an example rooted in battery models, we can assume a NN that predicts the cell voltage as a function of current $V_B(t) = \mathrm{NN}(i(t))$. During discharge a positive current is applied and, according to EQ. 2.13, $V_B \leq U_{eq}$ and $V_B > 0$. This constraint could, for instance, be achieved through $V_B = U_{eq} \times \mathrm{sigmoid}(\mathrm{NN}\{i(t)\})$, where $\mathrm{sigmoid}(z) \in [0,1] \, \forall \, z$.

In contrast to the instance learning capabilities of PINNs, where PDE solutions can be inferred at an instance (tempo-spatial coordinate), PENNs can employ continuous learning [20]. When employing continuous learning Faroughi *et al.* mentions neural ordinary differential equations (NODEs), as one sub-category of PENNs [20]. NODEs were first introduced by Chen *et al.* as a way of parameterizing a system's dynamics, where a NN predicts the derivative of a system state variable [25].

To understand NODEs, consider a system with a time dependent state $\boldsymbol{\xi}(t)$. The dynamics of this state is then determined by its time derivative, which can be described by a known ordinary differential equation (ODE) $\mathcal{F}$ as

$$\dot{\boldsymbol{\xi}}(t) = \mathcal{F}(\boldsymbol{\xi}(t)). \tag{2.33}$$

Assume the initial and boundary conditions are well defined, then the state $\boldsymbol{\xi}$ can be determined at any time t using an integration technique of choice. However, if the dynamic behavior of the state $\boldsymbol{\xi}(t)$ is unknown, in other words the ODE $\mathcal{F}$ is unknown, then $\boldsymbol{\xi}$ at time t cannot be determined. Here, NODEs offers a way of determining $\boldsymbol{\xi}(t)$ by parameterizing the unknown ODE with a NN, as

$$\dot{\boldsymbol{\xi}}(t) = \mathrm{NN}(\boldsymbol{\xi}(t); \boldsymbol{\theta}),$$

with trainable parameters $\boldsymbol{\theta}$. Following the way of NNs, the true (albeit unknown) ODE, can be approximated arbitrarily well by training the NN's parameters $\boldsymbol{\theta}$. Since the true value of $\dot{\boldsymbol{\xi}}$ is unknown, the network output cannot be compared to $\boldsymbol{\xi}_{\text{true}}$ directly. Instead, the predicted trajectory $\hat{\boldsymbol{\xi}}(t)$ is obtained by integrating the NN forward in time

$$\hat{\boldsymbol{\xi}}(t_m) - \hat{\boldsymbol{\xi}}(t_0) = \int_{t_0}^{t_m} \dot{\boldsymbol{\xi}}(t')\, \mathrm{d}t',$$

assuming a known initial condition $\hat{\boldsymbol{\xi}}(t_0)$. The loss function then compares the predicted state $\hat{\boldsymbol{\xi}}(t_m)$ to state observations $\boldsymbol{\xi}(t_m)$ at discrete measurement times $t_m \in \{t_1, ..., t_N\}$. The training procedure is summarized as follows:

0. initialize the parameters randomly,
1. make a prediction (forward pass) $\dot{\boldsymbol{\xi}}(t) = \mathrm{NN}(\boldsymbol{\xi}(t); \boldsymbol{\theta})$,
2. integrate $\dot{\boldsymbol{\xi}}$ from t_0 with $\boldsymbol{\xi}(t_0)$ to obtain $\hat{\boldsymbol{\xi}}(t_m)$ at measurement times t_m,
3. compare $\hat{\boldsymbol{\xi}}(t_m)$ to data $\boldsymbol{\xi}(t_m)$ by computing the loss $\mathcal{L}(\boldsymbol{\theta})$,
4. adjust $\boldsymbol{\theta}$ in direction that minimizes $\mathcal{L}(\boldsymbol{\theta})$ (backward pass),
5. return to step 1 and repeat the desired number of times or until $\mathcal{L}(\boldsymbol{\theta}) < \varepsilon_{tol}$ for some tolerance ε_{tol}.

If the system's ODE is known but certain incorporated relations are not, a similar training procedure can still be a powerful tool [20], [26]. Here, the NN can learn unknown constitutive laws such that the governing balance law is satisfied. Instead of modeling the ODE as a network, it is assumed that $\mathcal{F}$ in Eq. 2.33 is known with an additional dependency on an unknown function $\boldsymbol{g}(t)$:

$$\dot{\boldsymbol{\xi}}(t) = \mathcal{F}(\boldsymbol{\xi}(t), \boldsymbol{g}(t)).$$

$\boldsymbol{g}(t)$ can then be parametrized with a NN as $\boldsymbol{g}(t) = \mathrm{NN}(t; \boldsymbol{\theta})$. Following the same procedure but with an addition to step 1:

```
...
 1. make a prediction g(t) = NN(θ; t) and compute ξ̇(t) = F(ξ(t), g(t)),
...
```

Since an integrator is embedded within the training, the choice of integration scheme must allow for backpropagation of gradients, and deliver adequate stability at reasonable computational cost. Explicit integration schemes are computationally cheap but imposes a step-size constraint. Whereas, implicit integrations schemes are unconditionally stable but at higher computational cost.

2.4 Symbolic Regression

Even though a NN efficiently can approximate unknown relations from data, it generally does not display these relations in an easily interpretable way. On the other hand, discovering simple expressions based on elementary functions might be difficult when dealing with large or high-dimensional datasets. To combat this, symbolic regression (SR) algorithms offer a different ML approach, where the optimization domain consists of analytic expressions, instead of a continuous space of parameter values. In a similar way as NNs are inspired by the structure of biological neural networks, SR algorithms usually employ a genetic optimization scheme, inspired by the evolution seen in nature [27]. While there are several SR algorithms, the following theory is based on the `PySR` module in `Python`, developed by Cranmer [27].

In essence, the evolutionary optimization for SR is based on comparing individuals through tournaments and then evolving them based on mutations and crossovers [27]. In each tournament, a subset of randomly drawn individuals, is compared with a loss (fitness) function. The fittest individual is copied and allowed to get offspring. The offspring either receives a random mutation, according to a predefined set of mutations, or characteristics from an individual from another tournament through a crossover. In addition to mutation and crossovers, other operations are also applied to the expression [27]. Next, the new individual replaces the oldest individual in the population. Since individuals are symbolic expressions, mutations could, for instance, be a swapped sign, whereas crossovers refer to interchanging parts of the expression with another individual (see Fig. 2.14).

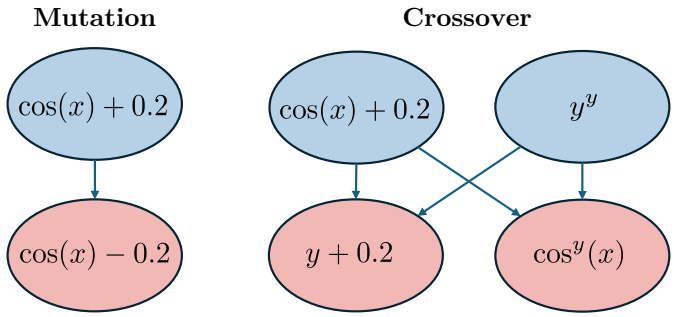


FIG 2.14: Mutation and crossover operation on two individuals. Inspired by [27].

In addition to comparing individuals with a pure data loss, such as the MSE or MAE, the loss function is extended with a complexity term that balances accuracy and sparsity. An expression's complexity is defined as the number of so-called "nodes", as shown in TAB. 2.1, however, the complexity of certain operators can also be dialed manually. During optimization, the default complexity regularization term is designed such that the algorithm keeps an equal number of expressions for each level of complexity [27]. This way, the algorithm gives a set of expressions at different complexities after optimization.

TAB 2.1: Expressions with default complexities

Expression	**Complexity**
10	1
$x + 10$	3
$1.2 \cdot \sin(x) + 10$	6

2.5 State of the Art: SciML and Batteries

In recent years, progress and accessibility of SciML frameworks have introduced new strategies for building effective surrogate models by combining prior knowledge about batteries with computationally efficient NNs. A common approach in literature is to use physics-informed architectures together with reduced-order models such as the single particle model (SPM): a simplification of the P2D, assuming a single active material particle per electrode and neglecting the electrolyte. For instance, Hassanaly *et al.* employed a PINN architecture to model the Li concentration and potential at each electrode along the particle radius r and time t [28]. In contrast to traditional PINNs, solving the same problem with NOs, such as the deep operator network (DeepONet), have been shown to improve generalization when predicting the solid concentration for more complex current profiles [29]. This was examined in Zhuang *et al.*'s work, where they compared PINNs to a physics-informed DeepONet (PI-DeepONet) in different scenarios and by varying the

amount of labeled data [29]. Moreover, Brendel *et al.* used a PI-DeepONet setup that replicated the SPM for different diffusion coefficients [30]. Similarly, Panahi *et al.* constructed a surrogate model based on parameter-embedded Fourier neural operator (FNO) that included both diffusion coefficients and the active material particle radii as parameters [31]. While the previously mentioned works model electrochemical behavior with basis in the SPM, Hassanaly *et al.* developed their PINN to replicate the P2D [32]. Moreover, Pang *et al.* extended the SPM to include thermal effects, and used a physics-informed ML approach to model heat generation [33]. For mechanics, Sun *et al.* suggest including a stress-dependent diffusion coefficient in a PINN architecture for the SPM [34].

For physics-based electrochemical-mechanical battery models, previous work has extended the P2D model to include mechanical coupling and multi-scale models [35], [36], [37]. Specifically, Bai *et al.* extended the P2D model to a two-level finite element method (FE^2) model, coupling electrochemical and mechanical responses [37]. Following Bai *et al.*, Asheri *et al.* modeled the interface between the solid state electrolyte and the active electrode material at the microscale level, particularly emphasizing interface damage [38]. Although the multi-scale model was based on FE^2, the micro-scale finite element (FE) calculations of concentration and damage derivatives were replaced with a NN. Some physical constraints were encoded with a strictly positive activation function.

In contrast to the aforementioned works, which are based on electrochemical battery models, an alternative approach for battery surrogate models is to use ML together with ECMs. In this context, Brucker *et al.* used NNs embedded within an ODE solver to predict circuit elements of a first-order ECM as a function of SOC and applied current [39], [40]. In their first work, values of R_0 and the time derivative of the RC-branch voltage ($\dot{V}_1$ in Fig. 2.7) were predicted with a NN [39]. Next, Brucker *et al.* developed their model, such that the NN predicted the R_1 resistor. The remaining circuit elements were modeled as trainable constant parameters R_0, and C_1 [40]. While Brucker *et al.*'s work focus on capturing the current-voltage response, Guo *et al.* utilized NNs to add a thermal coupling to an ECM by predicting deviations in R_0, R_1 and C_1 with respect to temperature [41]. As a final note, it is worth mentioning that ECMs and NNs are also commonly used in the opposite scenario, namely estimating SOC from measurements of the cell voltage, see [2], [42], [43] for reviews.

A summary of the referenced SciML studies, along with corresponding ML architectures, input and output variables are presented in App. A.

2.5.1 Research Novelty

To summarize, battery models often include multi-physical aspects, such as coupling between electrochemical, mechanical and even thermal processes. To reduce computational cost, surrogate models based on SciML have shown promising results for electrochemical simulations.

A common starting point for previous surrogate models is a known system of PDEs that provides a detailed description of the battery interior during operation. In this context, SciML architectures have been used to learn solutions to the system of electrochemical PDEs, providing a faster alternative than solving it numerically. In contrast, the surrogate model framework developed in this thesis focuses solely on cell-level responses, enabling a space and time independent surrogate model. While previous work have applied SciML to calibrate circuit elements of ECMs, the framework developed in this thesis includes a physical surrogate presentation of both the electrochemical and mechanical system as well as introducing symbolic regression as a method for discovering closed-form expressions of the circuit elements and their dependencies. Finally, the framework also provides a way of implementing surrogate models in a multiphysics scenario by coupling simpler physics-based models, describing different cell-level responses.

Since SciML surrogate models are still a relatively new research field, the mechanical effects of batteries are often not covered. On that note, we believe that the demonstration of the framework for electrochemical-mechanical battery simulations contributes to the body of work within the field.

3

Method

3.1 Pipeline

Before diving into the details of the methodology, we give a brief overview of the modeling pipeline by highlighting some key steps. For starters, a coupled high-fidelity battery model was constructed in the FE software `COMSOL`. Next, voltage curves and reaction forces were simulated for wide range of applied C-rates and compressions. Based on the generated data, a NN architecture was used to tune the values for resistors, capacitor, and spring stiffness according to ECM and EMM. The internal swelling, induced during a cycle, was modeled dynamically with a NODE. As a last step, the trained NN was fed into a symbolic regression scheme, eventually extracting symbolic expressions for the unknown equivalent circuit elements, the spring stiffness and the swelling rate. A graphical summary of the pipeline is given in Fig. 3.1.

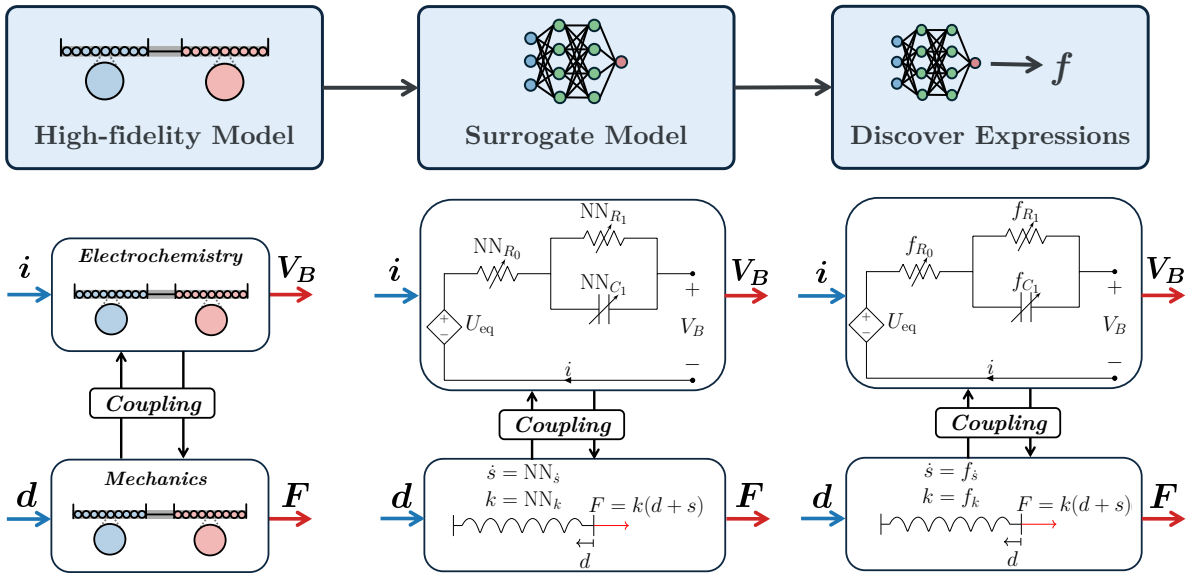


Fig 3.1: Overarching pipeline for the surrogate model framework. The steps involve: generating data from the high-fidelity model, training NNs to predict circuit elements, spring stiffness, and swelling rate, and discovering symbolic expressions from the NNs. Note that each step model the same inputs, i and d and the same outputs V_B and F.

To clarify, we use the following nomenclature to distinguish between the full surrogate model and the equivalent models within.

- *Observed inputs and observed outputs* – inputs to the surrogate model, applied C-rate and compression percentage ($\widetilde{d}$). Outputs from the surrogate model, cell voltage V_B and reaction force F.
- *Latent* (*internal*) *inputs and outputs* – inputs and outputs to the NNs or symbolic expressions. Inputs: Applied current i, compression $d = \widetilde{d}L_{\text{cell}}$ and SOC. Outputs: Circuit elements R_0, R_1, C_1, k, and swelling rate $\dot{s}$.
- *Trajectory* – time series of an observed input or observed output.

3.2 Generating Data from High Fidelity Model

To generate data under various operating conditions, a coupled electrochemical and mechanical battery model was implemented in COMSOL. The electrochemical model was based on a P2D battery model from the application library in COMSOL. This model was extended to include mechanical effects both through internal swelling and externally applied compressions. The following model assumptions were used to establish a bidirectional coupling between mechanical and electrochemical effects.

First, the electrochemical-mechanical coupling was introduced as swelling driven by Li concentration in the electrodes. In particular, hygroscopic swelling was considered with volumetric strain depending on concentration according to Eq. 2.26. With this assumption the swelling was entirely reversible, and considered no degradation effects. For initial conditions, the swelling stresses for each electrode were initialized to be zero at $t = 0$, by $\bar{c}_{s,l}(x,0) = \bar{c}_{s,l}^{\text{ref}}$ in Eq. 2.26. A simplified assumption was to set the hydroscopic swelling coefficients, β_l to $10^{-5}\,\text{m}^3\,\text{kg}^{-1}$ for both electrodes. These values for β_l gave a swelling-induced volumetric strain of $< 10\%$, which is in the same range as observed for real battery cells [7]. Secondly, a mechanical-electrochemical coupling was modeled through the evolution of porosity as a function of volumetric strain with Eq. 2.27 in the separator.

To avoid the *trivial* case of linear elasticity, the mechanical material response of the electrodes and separator was modeled with a neo-Hookean, hyper-elastic, material model. The volumetric response was modeled with the Simo-Pister strain energy density in COMSOL. For material parameters, the elasticity moduli were assumed to be 1.4 GPa for both electrodes and 0.7 GPa for the separator with a Poisson's ratio of 0.37 for all domains.

Moreover, the model considered a graphite negative electrode and an NMC positive elec-

trode with $LiPF_6$ liquid electrolyte with material data from the built-in material library. The total length of the cell was $L_{\text{cell}} = 144\,\mu\text{m}$. A schematic of the battery geometry is shown in Fig. 3.2. In terms of mechanic confinement, it was assumed that the negative current collector was fixed and that the positive was compressed with a prescribed compression d. The resulting reaction force was measured at the positive current collector as the force exerted by the battery on the surrounding wall, however, the total reaction force on the battery is the same but with the opposite sign. For the currents, a positive current density was applied at the positive current collector during discharge. Conversely, a negative current density during charge. The negative electrode potential was set to ground, thus, the overall cell voltage could be measured as the potential at the positive current collector.

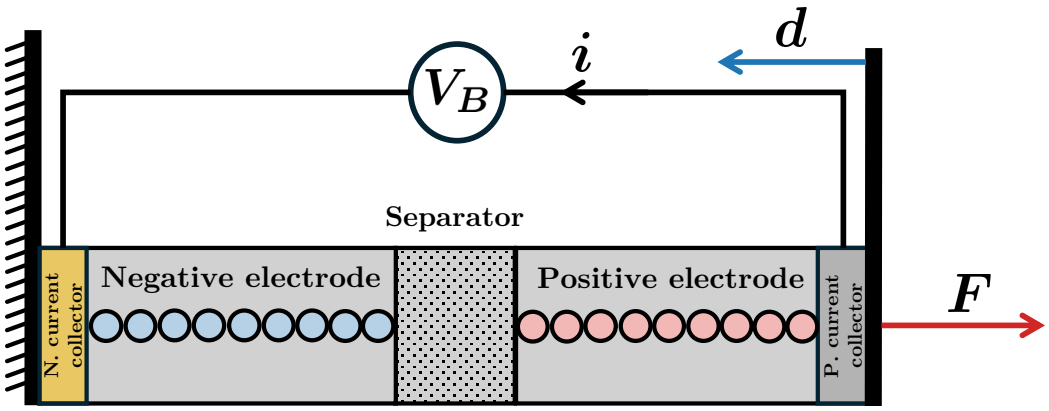


Fig 3.2: Schematic illustration of the high-fidelity model with a confined end at the negative current collector and an a compression applied at the positive. The compression results in a force F and the applied current i results in a cell voltage V_B.

In the discharging case, all simulations were run from an initial $\text{SOC}(t = 0) = 1$. Using a time-step of $\Delta t = 1\,\text{s}$, the simulation was run until $V_B < V_{\text{cutoff}}$, where V_{cutoff} was set to 2.3 V. The cutoff criterion was obtained as $U_p^{\min} - U_n^{\max}$, corresponding to the smallest value of the equilibrium potential, U_{eq}.

3.2.1 Alternative Battery Simulations

While this work used a battery model from `COMSOL`, there are other ways to simulate batteries such as the open-source `Python` extension `PyBaMM` [44]. The source code, based on both `COMSOL` and `PyBaMM`, can be downloaded from `github`.

3.2.2 Discharge with Varying Currents and Compressions

As an initial step, voltage curves and reaction forces for constant compressions and C-rates were recorded. The C-rates were converted to currents using Eq. 2.15. Similarly,

the percentage of compression was converted to displacement by $d = L_{\text{cell}}\widetilde{d}$. The overall boundaries were chosen such that C-rate $\in [0.5, 5]$ and $\widetilde{d} \in [0\,\%, 30\,\%]$. These ranges were constructed to maximize the observed inputs range in COMSOL, and still obtain a converged solution. In total, 125 unique pairs of C-rate and $\widetilde{d}$ values were randomly sampled, yielding 125 corresponding voltage and reaction force curves. In particular, some trajectories were generated without compression. For short, constant C-rate, or equivalently constant current, will be referred to as constant current (CC).

Next, a second data set, consisting of 75 trajectories, was generated with a varying current profile. For simplicity, the built-in waveform generator in COMSOL was used to generate the following current profile

$$i(t) = \frac{i_0}{2}\left[\text{square}\left(\frac{2\pi t}{\widetilde{T}}\right) + 1\right], \tag{3.1}$$

where i_0 represents a constant amplitude, $\widetilde{T}$ the period, and square$(\cdot)$ a square wave. To obtain a more general pulse data set, the period was randomly drawn from $\widetilde{T} \in [350\,\text{s}, 450\,\text{s}]$ and the duty cycle from [0.45,0.55]. Similarly, the amplitude i_0 and d were randomly sampled in the domain C-rate $\in [0.5\,,5] \times \widetilde{d} \in [0\,\%, 30\,\%]$. To avoid large scale differences the latent inputs were normalized to obtain values within [0,1] by dividing i and d by their respective maximum value.

3.3 NN-enhanced Surrogate Model

To replicate the coupled high fidelity model, the surrogate model was based on ECM and EMM structures. The goal was to simulate the cell voltage V_B and the reaction force F when the positive current collector is displaced by d and a current i is applied over the battery. To do this, the cell voltage was modeled with a first-order ECM as in Fig. 2.7, which mathematically corresponds to Eq. 2.17; repeated here as a reminder

$$V_B = U_{eq} - iR_0 - V_1, \quad \text{where} \quad \frac{\mathrm{d}V_1}{\mathrm{d}t} = -\frac{V_1(t)}{R_1 C_1} + \frac{i(t)}{C_1}.$$

To enhance this first-order ECM with mechanical coupling it was conceptually extended with a spring, to account for the reaction force, measured at the positive current collector. Despite the hyper-elastic stress-strain response, the following spring

$$F = k \cdot (d + s), \tag{3.2}$$

was assumed for the EMM where s is the internal swelling. To enhance the simpler models, the key concept was to use data for V_B and F to calibrate the unknown circuit

elements R_0, R_1, C_1 and k, along with the swelling rate $\dot{s}$ with NNs, as shown in FIG. 3.3 and FIG. 3.4.

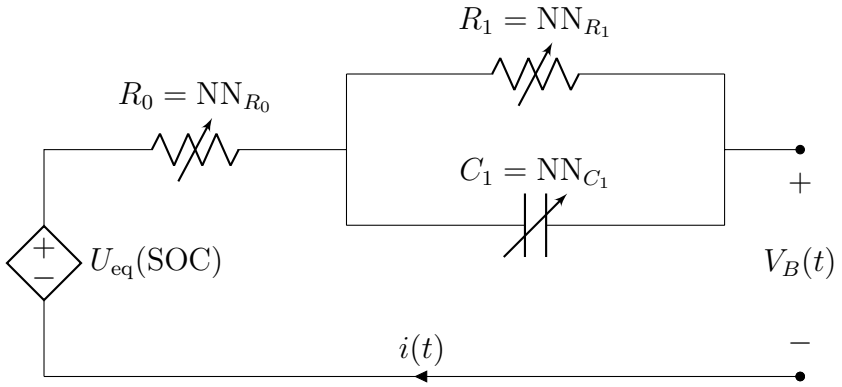


FIG 3.3: Circuit diagram of the ECM surrogate model structure. Note that the circuit has variable circuit elements that were determined by NNs.

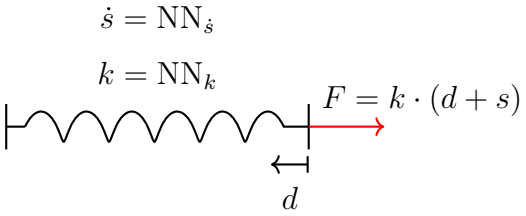


FIG 3.4: Schematic of the EMM, where both the spring stiffness and the swelling rate are determined by NNs.

Solving for the voltage and force with the assumed models involved solving initial value problems (IVPs) for V_1, s and SOC. The IVP for the RC voltage, V_1, was states as

$$\begin{cases} \dfrac{\mathrm{d}V_1}{\mathrm{d}t} = -\dfrac{V_1(t)}{R_1 C_1} + \dfrac{i(t)}{C_1} = -\dfrac{V_1(t)}{\mathrm{NN}_{R_1}\mathrm{NN}_{C_1}} + \dfrac{i(t)}{\mathrm{NN}_{C_1}}, \\ V_1(t_0) = 0. \end{cases} \tag{3.3}$$

For clarity, it was assumed that the capacitor had no charge at $t = t_0$, and consequently $V_1(t_0) = 0$. Furthermore, the swelling s was modeled to have a path dependence with an unknown ODE, $\mathrm{NN}_{\dot{s}}$, giving the IVP

$$\begin{cases} \dfrac{\mathrm{d}s}{\mathrm{d}t} = \mathrm{NN}_{\dot{s}}, \\ s(t_0) = 0, \end{cases} \tag{3.4}$$

where the battery was assumed to have no internal swelling at the start of discharge. The

SOC was modeled with the IVP:

$$\begin{cases} \dfrac{\mathrm{dSOC}}{\mathrm{d}t} = -\dfrac{i(t)}{Q_0}, \\ \mathrm{SOC}(t_0) = 1. \end{cases} \tag{3.5}$$

In this case, the overall model consisted of a system of three ODEs. Thus, training the NNs required solving the IVPs stated in Eq. 3.3, Eq. 3.4 and Eq. 3.5 to obtain the states V_1, s and SOC.

3.3.1 Training Procedure

A general assumption for the surrogate model would be to assume that the circuit elements, spring stiffness, and swelling rate depend on all latent inputs as well as the current states SOC, V_1 and s. In other words, the latent outputs are modeled accordingly:

$$\begin{aligned} \hat{R}_{0,j} &= \mathrm{NN}_{R_0}(i_j, d_j, \mathrm{SOC}_j, V_{1,j}, s_j), \\ \hat{R}_{1,j} &= \mathrm{NN}_{R_1}(i_j, d_j, \mathrm{SOC}_j, V_{1,j}, s_j), \\ \hat{C}_{1,j} &= \mathrm{NN}_{C_1}(i_j, d_j, \mathrm{SOC}_j, V_{1,j}, s_j), \\ \hat{k}_j &= \mathrm{NN}_{k}(i_j, d_j, \mathrm{SOC}_j, V_{1,j}, s_j), \\ \hat{\dot{s}}_j &= \mathrm{NN}_{\dot{s}}(i_j, d_j, \mathrm{SOC}_j, V_{1,j}, s_j), \end{aligned}$$

for each discrete time step, t_j, with index $j = 0, 1, ..., j_{\max}$. To clarify, each NN takes 5 scalar values as input and returns a single scalar.

From the IVPs, an initial prediction for V_B and F can be made by

$$\begin{aligned} \hat{V}_{B,0} &= U_{eq}(\mathrm{SOC}_0) - i_0 \hat{R}_{0,0} - V_{1,0}, \\ \hat{F}_0 &= \hat{k}_0 (d_0 + s_0). \end{aligned}$$

To obtain the next step, the states SOC, V_1 and s are advanced to $j = 1$ using

$$\begin{aligned} \mathrm{SOC}_1 &= \mathrm{SOC}_0 + \int_{t_0}^{t_1} \dot{\mathrm{SOC}}(t)\,\mathrm{d}t \approx \mathrm{SOC}_0 + \frac{-i_0}{Q_0}\Delta t \\ V_{1,1} &= V_{1,0} + \int_{t_0}^{t_1} \dot{V}_1(t)\,\mathrm{d}t \approx \frac{V_{1,0} + \Delta t\, i_0/\hat{C}_{1,0}}{1 + \Delta t/(\hat{R}_{1,0}\hat{C}_{1,0})}, \\ s_1 &= s_0 + \int_{t_0}^{t_1} \dot{s}(t)\,\mathrm{d}t \approx s_0 + \hat{\dot{s}}_0 \Delta t. \end{aligned}$$

With SOC_1, $V_{1,1}$ and s_1, the predictions $\hat{V}_{B,1}$ and $\hat{F}_1$ are obtained by computing

$$\begin{aligned}\hat{V}_{B,1} &= U_{eq}(\mathrm{SOC}_1) - i_1\hat{R}_{0,1} - V_{1,1},\\ \hat{F}_1 &= \hat{k}_1(d_1 + s_1),\end{aligned}$$

where

$$\begin{aligned}\hat{R}_{0,1} &= \mathrm{NN}_{R_0}(i_1, d_1, \mathrm{SOC}_1, V_{1,1}, s_1),\\ \hat{k}_1 &= \mathrm{NN}_k(i_1, d_1, \mathrm{SOC}_1, V_{1,1}, s_1).\end{aligned}$$

Computing $\hat{V}_{B,j}$ and $\hat{F}_j$ for all subsequent j's follow that exact same procedure, where SOC_j, $V_{1,j}$ and s_j are advanced from the previous step $j-1$.

To update NN parameters, the loss is computed between observed outputs $V_{B,j}$ and F_j and predicted $\hat{V}_{B,j}$ and $\hat{F}_j$ for all steps j in one trajectory. Lastly, the network parameters are updated with gradient descent, and the procedure is repeated for the next trajectory. Cycling through all trajectories concludes one epoch.

However, in the particular case implemented in this thesis the circuit elements and spring stiffness were assumed to only depend on the instantaneous current, compression and SOC. Assuming that all elements were independent of RC-voltage V_1 and swelling s (apart from the NODE $\dot{s} = \mathrm{NN}_{\dot{s}}$, which was assumed dependent on s), allowed for the mechanical and electrochemical branches to be implemented in parallel instead of simultaneously and produced the following relations.

$$\begin{aligned}\hat{R}_{0,j} &= \mathrm{NN}_{R_0}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{R}_{1,j} &= \mathrm{NN}_{R_1}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{C}_{1,j} &= \mathrm{NN}_{C_1}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{k}_j &= \mathrm{NN}_k(i_j, d_j, \mathrm{SOC}_j),\\ \hat{\dot{s}}_j &= \mathrm{NN}_{\dot{s}}(i_j, d_j, \mathrm{SOC}_j, s_j),\end{aligned}$$

for each discrete time step, t_j, with index $j = 0, 1, ..., j_{\max}$. In this case, each NN takes 3 or 4 scalar values as input and returns a single scalar.

The training procedure was implemented as follows. For each discrete step $j \in \{0,1,...,j_{\max}\}$

the networks were evaluated on the state SOC_j and latent input i_j and d_j

$$\begin{aligned}\hat{R}_{0,j} &= \mathrm{NN}_{R_0}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{R}_{1,j} &= \mathrm{NN}_{R_1}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{C}_{1,j} &= \mathrm{NN}_{C_1}(i_j, d_j, \mathrm{SOC}_j),\\ \hat{k}_j &= \mathrm{NN}_{k}(i_j, d_j, \mathrm{SOC}_j),\end{aligned}$$

where $\mathrm{SOC}_0 = 1$ when $j = 0$. To obtain SOC for all steps, it was advanced forward as

$$\mathrm{SOC}_{j+1} \approx \mathrm{SOC}_j - \frac{i_j}{Q_0}\Delta t.$$

SOC_{j+1} allowed $\hat{R}_{0,j+1}$, $\hat{R}_{1,j+1}$, $\hat{C}_{1,j+1}$ and $\hat{k}_{j+1}$ to be computed. Iterating SOC and making predictions for all j's, lead to an enhanced data set of $\{\hat{R}_0,\hat{R}_1,\hat{C}_1,\hat{k}\}$ for all j's.

This enhanced data set was then used to compute the observed output $\hat{V}_B$ and $\hat{F}$ by stepping through all j's. Starting with V_B at $j = 0$

$$V_{B,0} = U_{eq}(\mathrm{SOC}_0) - i_0\hat{R}_{0,0} - V_{1,0},$$

then advancing through all $j \in \{0, 1, \ldots j_{\max}\}$ with

$$\begin{aligned}V_{1,j+1} &\approx \frac{V_{1,j} + \Delta t\, i_j/\hat{C}_{1,j}}{1 + \Delta t/(\hat{R}_{1,j}\hat{C}_{1,j})},\\ \hat{V}_{B,j+1} &= U_{eq}(\mathrm{SOC}_{j+1}) - i_{j+1}\hat{R}_{0,j+1} - V_{1,j+1}.\end{aligned}$$

Continuing with F at $j = 0$:

$$\hat{F}_0 = \hat{k}_0 \cdot (d_0 + s_0),$$

then advancing through all $j \in \{0, 1, \ldots j_{\max}\}$ with

$$\begin{aligned}\hat{\dot{s}}_j &= \mathrm{NN}_{\dot{s}}(i_j, d_j, \mathrm{SOC}_j, s_j),\\ s_{j+1} &\approx s_j + \hat{\dot{s}}_j\Delta t,\\ \hat{F}_{j+1} &= \hat{k}_{j+1}(d_{j+1} + s_{j+1}).\end{aligned}$$

To update NN parameters, the MSE loss was computed between observed outputs $V_{B,j}$ and F_j and predicted $\hat{V}_{B,j}$ and $\hat{F}_j$ for all steps j in one trajectory. Lastly, the network

parameters were updated through gradient descent, and the procedure was repeated for the next trajectory. Cycling through all trajectories concludes one epoch.

This method allowed the latent outputs for electrochemistry along with k to be evaluated for all j at once, which provided the ability to use vectorization and obtain faster training.

The RC-voltage V_1 was integrated using semi-implicit Euler, while SOC and swelling used explicit Euler. Semi-implicit Euler was used since it provides additional stability, in contrast to explicit Euler. The training procedure was run for 2 500 epochs, and batched such that four trajectories were evaluated simultaneously. In total, the training was run for 85 trajectories with both CC and pulsed currents. The loss was computed using MSE in Eq. 2.30 and gradient descent was performed with the adaptive moment estimation (ADAM) optimizer using a learning rate of 0.001. The NN construction was implemented using `PyTorch` in `Python` [45]. Note that no further investigation was done to compare different loss functions and learning rates.

3.3.2 Network Structures and Constraints

The NNs were implemented with two hidden layers with 16 nodes each. The tanh activation function was used between layers to allow for internal values between -1 and $+1$. The latent output from each network was constrained with a softplus activation function, since all elements were assumed positive. To avoid large and small values for the NN predictions, the softplus activation function was scaled with a value, α, for each individual element. By letting y denote $R_0, R_1, C_1, k, \dot{s}$, a prediction $\hat{y}$ was computed by

$$\hat{y} = \text{softplus}\left(\text{NN}_y\right) \cdot \alpha_y.$$

The size of α_y was a rough estimate of the magnitude of each element and thus iteratively obtained (and estimated from fixed ECM calibrations). Since NNs are often initialized close to 1, introducing α_y also guided the network closer to the desired range of values, potentially improving convergence. The scaling values are presented in Tab. 3.1.

Tab 3.1: α_y values used in the softplus output layer constraint for each latent output y.

Latent output	α_y
R_0	$0.01\,\Omega$
R_1	$0.01\,\Omega$
C_1	$5000\,\text{F}$
k	$3\,\text{MN}\,\mu\text{m}^{-1}$
$\dot{s}$	$10^{-3}\,\mu\text{m}\,\text{s}^{-1}$

In addition to the positive constraint obtained by the softplus activation function, a min-max constraint was also implemented as

$$\hat{y} = y_{\min} + \text{sigmoid}(\text{NN}_y)(y_{\max} - y_{\min}). \tag{3.6}$$

However, this constraint was only used to train a model on CC currents and then predict on pulse currents, explained in Sec. 3.3.5.

3.3.3 Modeling Equilibrium Voltage from SOC

Inherent to ECMs is the controlled voltage source $U_{eq}(\text{SOC})$, which is assumed to be a function of SOC. Although data for U_{eq} could be obtained from the high-fidelity model at every time step, data for U_{eq} is not available when evaluating the surrogate model for current profiles not found in data. In other words, a fit was needed from SOC to U_{eq}. For this, data for U_{eq} and SOC from a slow discharge curve at C-rate = $0.1\,\text{h}^{-1}$ were fed through a Gaussian process regressor (see App. C.6 for details). The Gaussian regressor offered a continuous representation of $U_{eq}(\text{SOC})$, from which samples could be drawn. As a final step, SR was applied to the set of 20 000 samples, extracting a symbolic expression for $U_{eq}(\text{SOC})$. Details for SR are given in Sec. 3.4.

3.3.4 Reference Models for Electrochemistry

In addition to the complete surrogate model, some alternative models were developed for comparison.

3.3.4.1 ECM with Fixed Elements

The simplest form of baseline reference model was to use the first-order ECM with fixed values of the ECM circuit elements, determined from a basic calibration. For reference, the ciruit diagram is shown in Fig. 3.5. Although there are more sophisticated ways to estimate the element values, such as Kalman Filter or Bayesian analysis, the following loop provided the least-squares estimate. Since U_{eq} is viewed as a controlled voltage source in the ECM, the calibration optimized

$$\min\{\eta_{\text{tot}} - iR_0 - V_1\}, \quad \text{with} \quad \frac{\mathrm{d}V_1}{\mathrm{d}t} = -\frac{V_1(t)}{R_1 C_1} + \frac{i(t)}{C_1},$$

where $\eta_{\text{tot}} = U_{eq} - V_B$. For calibration, V_1 was obtained by integration with a default setting of the `scipy.odeint` solver, again, assuming the initial condition $V_1(0) = 0$. The least-square estimate was done with `scipy.curve_fit` with the bounds

$$R_0 \in [1\,\text{m}\Omega, 100\,\text{m}\Omega], \quad R_1 \in [5\,\text{m}\Omega, 500\,\text{m}\Omega], \quad C_1 \in [0.1\,\text{kF}, 25\,\text{kF}]$$

and the initial guess of $R_0 = 10\,\mathrm{m\Omega}$, $R_1 = 50\,\mathrm{m\Omega}$ and $C_1 = 5\,\mathrm{kF}$. The obtained values for these elements were kept fixed when simulating an entire discharge.

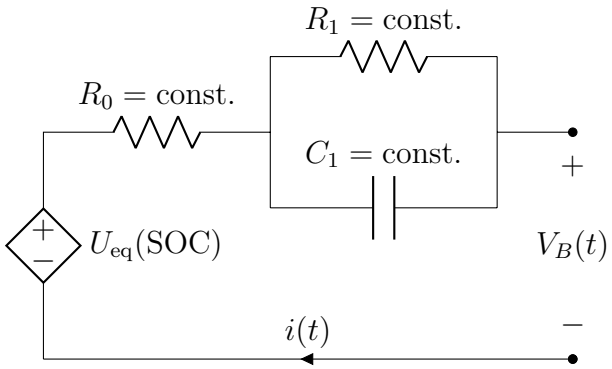


Fig 3.5: Circuit diagram of the ECM surrogate model structure with one set of fixed values for the circuit elements for each trajectory.

3.3.4.2 NN Reference Models

Instead of basing the surrogate on a first-order ECM, a simpler choice was to base a model on the ECM given in Fig. 3.6. With this setup, there was no need for an integration scheme, since the voltage over the resistor has no associated evolution law. Instead, the deviation from U_{eq} was found with $\hat{V}_B(t_j) = U_{eq}(\mathrm{SOC}_j) - i_j R_{\mathrm{NN}}(i_j, d_j, \mathrm{SOC}_j)$ for each time step t_j. This model's NN was constructed with two hidden layer of 16 nodes each and constrained with a softplus output layer.

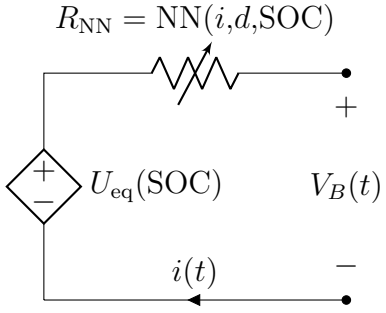


Fig 3.6: Circuit diagram of an ECM, where U_{eq} is modeled with a controlled voltage source governed by SOC and R_{NN} a variable resistor modeled by a NN with inputs i, d, and SOC for each time step.

In addition to the previous models that in some way incorporate an ECM, another approach for surrogate models is to use a fully data-driven "black-box" NN. To allow for a comparison between the different approaches, an identical NN with 16 nodes in two hidden layers was trained on the same data with same number of epochs as the models in Sec. 3.3 and Sec. 3.3.4.2. However, instead of calculating V_B from a circuit, V_B was given by a direct mapping $\hat{V}_B(t_j) = \mathrm{NN}(i_j, d_j, \mathrm{SOC}_j)$.

Apart from a different training procedure for the voltage response, the force response was also modified. The spring stiffness was left the same, however, to acquire an integration free training, the swelling was predicted instantaneously without any assumed path-dependence as $s_j = \mathrm{NN}_s(i_j, d_j, \mathrm{SOC}_j)$.

3.3.5 Evaluating Models

To show limitations and strengths of the surrogate model, it was evaluated for some different scenarios and compared to the reference models. The models were evaluated on V_B and F predictions, training time and extrapolation capabilities.

The elapsed time for the two training procedures in Sec. 3.3.1 and the two NN reference models in Sec. 3.3.4 were evaluated and compared. Each model was trained on the same data set for 2 500 epochs to obtain a qualitative measurement of the time. The models were trained on the same computer and without any speed optimizations, apart from the vectorization explained in the final training procedure in Sec. 3.3.1.

To evaluate the benefits of embedding NNs within the ECM structure, both the NN-enhanced ECM's and the black box NN's ability to extrapolate from discharge to charge were tested. The models were trained on 100 CC discharge trajectories and used for predicting a full cycle discharge and charge.

The surrogate model and the static reference NN model were trained on CC and then evaluated on pulse currents. Both models were trained on 100 trajectories with varying C-rates and compressions. In this case, the surrogate model was constructed with the sigmoid constraint, which allowed a lower bound to be set on R_1 and C_1, effectively constraining the lower bound on the time constant τ. Without the lower bound, the surrogate model would predict τ lower than a fixed value ECM would acquire for pulse predictions. The lower bound for the time constant was set to $\tau_{\min} = R_1 C_1 = 25\,\mathrm{s}$, which is similar to [3] of around 30 to 40 seconds. The upper bound ($R_1 = 250\,\Omega$, $C_1 = 14\,000\,\mathrm{F}$), on the other hand, were kept generous enough to not limit the predicted element.

3.4 Symbolic Regression

The last part of the methodology aimed to discover symbolic expressions that could be used alongside the NNs for predicting element values for the surrogate model. Compared to NNs, symbolic expressions also provide a more interpretable alternative representation of underlying relations. In essence, this step involved: evaluating the NNs and recording the latent outputs and the corresponding latent inputs; set up and run SR; and evaluating and comparing the discovered expressions. The main goal is summarized as

$$
\begin{aligned}
R_0 = \mathrm{NN}_{R_0}(i,d,\mathrm{SOC}) &\longrightarrow R_0 = f_{R_0}(i,d,\mathrm{SOC}),\\
R_1 = \mathrm{NN}_{R_1}(i,d,\mathrm{SOC}) &\longrightarrow R_1 = f_{R_1}(i,d,\mathrm{SOC}),\\
C_1 = \mathrm{NN}_{C_1}(i,d,\mathrm{SOC}) &\longrightarrow C_1 = f_{C_1}(i,d,\mathrm{SOC}),\\
k = \mathrm{NN}_{k}(i,d,\mathrm{SOC}) &\longrightarrow k = f_{k}(i,d,\mathrm{SOC}),\\
\dot{s} = \mathrm{NN}_{\dot{s}}(i,d,\mathrm{SOC},s) &\longrightarrow \dot{s} = f_{\dot{s}}(i,d,\mathrm{SOC},s).
\end{aligned}
$$

3.4.1 Generating Data for Symbolic Regression

After training, the NNs learned underlying relations for how the circuit elements, spring stiffness and swelling rate contribute to V_B and F based on the inputs. These hidden relations could be better understood by applying SR directly on samples from the NNs. In practice, the NNs were sampled according to latin hypercube sampling scheme. In contrast to the samples for the high-fidelity model, the NNs were sampled for C-rates down to zero to avoid potential singularities. In total, 10 000 samples were sampled for latent inputs within the domain

$$
\begin{cases}
\text{C-rate} \in [0, 5]\\
\widetilde{d} \in [0, 30\%]\\
\mathrm{SOC} \in [0, 1].
\end{cases}
$$

To ensure a somewhat similar range for the expressions, each latent input was normalized with the corresponding maximum value, which gave inputs between 0 and 1. The elements were also scaled to avoid large constants, see reference values in Tab. 3.2. As an example to show scaling and normalization, f_{R_0} was fitted to the NN by $\mathrm{NN}_{R_0} \approx R_0^{\mathrm{ref}} \cdot f_{R_0}(i/i_{max}, d/d_{max},\mathrm{SOC})$.

The swelling, $\dot{s}$, went through an almost identical procedure with the exception of how the data was generated. Since $\dot{s}$ depends on s, which is only accessible through integration, the values for s were obtained via explicit Euler integration.

Tab 3.2: List of the reference values for each latent output during SR.

Latent Output	**Reference value**
R_0	$10\,\mathrm{m\Omega}$
R_1	$10\,\mathrm{m\Omega}$
C_1	$1\,\mathrm{kF}$
k	$1\,\mathrm{MN\,\mu m^{-1}}$
$\dot{s}$	$0.001\,\mathrm{\mu m\,s^{-1}}$

3.4.2 Setting Up and Running Symbolic Regression

Before finding expressions, some hands-on adjustments were made to constrain the optimization space of the allowed expressions. Along with a list of unary operators (functions), given in Tab. 3.3, the binary operators, i.e., `+`, `-`, `*`, ÷ and `^`, determined the allowed space of expressions. To speed up the optimization, the power operator (`^`) was constrained such that it allowed for an arbitrary base but only exponents with a complexity of 1. As a result, expressions like $x^{\sqrt{x}}$ were not allowed. Note that this constraint did not apply to e^x, as it is considered a separate operator. The function space was further constrained by limiting some nested expressions (e.g., `exp(exp(exp(x)))` or `log(log(log(x)))`). For a complete view of the SR model, see App. B.

Tab 3.3: List of allowed operators together with their associated complexity.

Operators	**Expression**	**Complexity**
`square(x)`	x^2	1
`sqrt(x)`	$\sqrt{x}$	1
`cube(x)`	x^3	1
`exp(x)`	e^x	1
`log(x)`	$\ln(x)$	2
`+`, `-`, *, ÷		1
`^`		2

Before running SR, the optimization could be set up in different modes with different population sizes, and iterations. In general, the model could either optimize for `accuracy`, essentially minimized error, `score`, promoting more scarce models, or `best`, mix between the two former [46]. In this case, the `best` optimizing mode was used with 30 populations and 15 000 iterations. For the data loss, the default MSE was used.

After running the SR optimization, the candidate functions were compared with focus on overall dependencies between the latent inputs and the elements. Although SR automatically selects a preferred expression after optimization, a complexity-to-loss curve was used to find alternative expressions with lower complexities.

To simulate the observed outputs, the ECM and the spring equations were solved with the element values from SR expressions. Mathematically, this gave the following setup

$$
\begin{aligned}
V_B(t) &= U_{eq}(\mathrm{SOC}) - i(t)R_0(i,d,\mathrm{SOC}) - V_1, \\
F(t) &= k(i,d,\mathrm{SOC}) \cdot (d+s), \quad \text{with} \\
\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} V_1 \\ \mathrm{SOC} \\ s \end{bmatrix} &= \begin{bmatrix} \dfrac{-V_1}{R_1(i,d,\mathrm{SOC})C_1(i,d,\mathrm{SOC})} + \dfrac{i(t)}{C_1(i,d,\mathrm{SOC})} \\ -i(t)/Q_0 \\ \dot{s}(i,d,\mathrm{SOC},s) \end{bmatrix}.
\end{aligned}
$$

A solution to the ODE was found with `scipy.odeint` in `Python`.

4

Results and Discussion

The results and discussion consist of three parts, starting with visualizing data from the high-fidelity model then voltage and force predictions of the NN-enhanced ECM and comparisons between it and the reference models for different scenarios. A closer look at the NN calibrated circuit elements and the swelling rate, as well as their corresponding symbolic expression, is also presented and discussed.

4.1 High-fidelity Data Exploration

Simulating the battery with the high-fidelity model resolves the multi-physics throughout the full geometry, yielding spatially and temporally resolved fields, such as the average lithium concentration and the volumetric strain profiles in the electrodes, shown in Fig. 4.1. As mentioned in Sec. 2.1.1, Li-ions migrate from the negative to the positive electrode during discharge, effectively reducing the concentration as a function of SOC in the negative electrode and increasing for the positive electrode. Moreover, since the volumetric strain, modeled with Eq. 2.26, is proportional to the average concentration, the negative electrode shrinks while the positive swells. Despite identical swelling coefficients, β, the difference in molar masses contributes to larger strain for the positive electrode within a cycle. Due to the larger swelling in the positive electrode, the separator region will show a negative strain. Compared to the electrodes, the separator also has lower elasticity modulus, causing an increased negative strain.

While this shows that the high-fidelity model can be used to give a detailed picture of the battery unit cell, the surrogate model focuses on observed variables. The high-fidelity simulations of these are shown next.

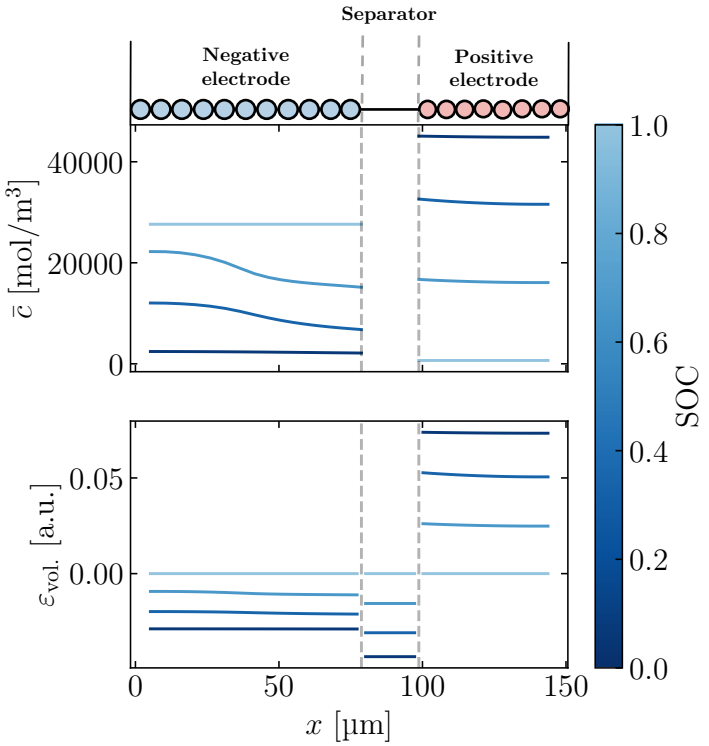
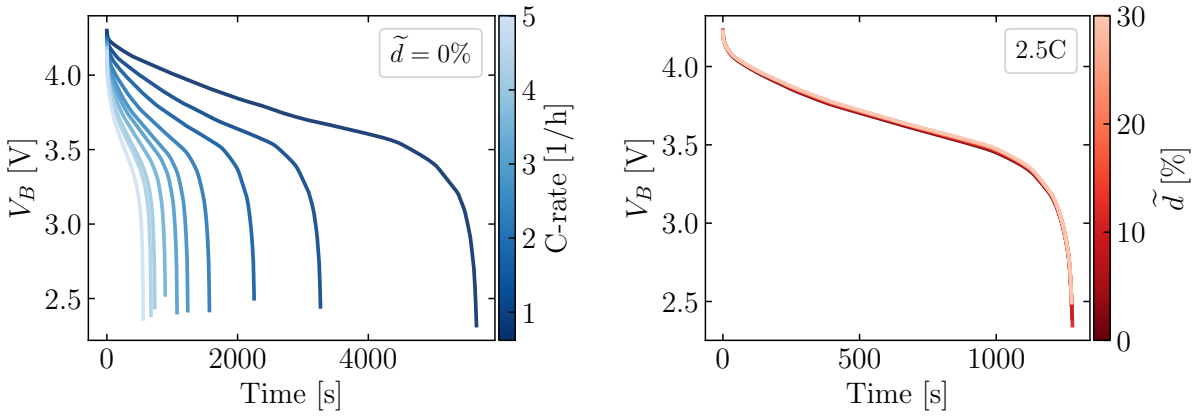


FIG 4.1: Average lithium concentration $\bar{c}$ and volumetric strain $\varepsilon_{\text{vol.}}$ along the x-dimension of the unit cell for varying SOC, computed by the `COMSOL` model.

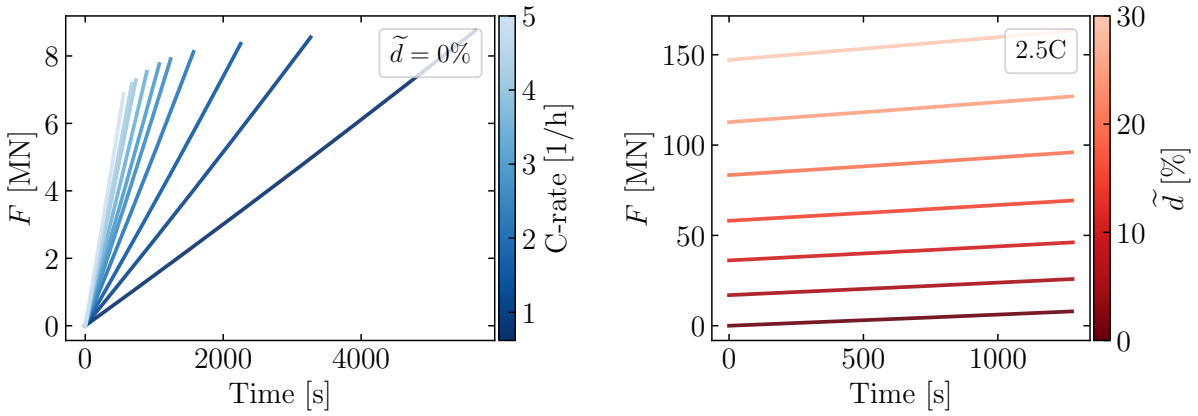
To visualize the observed output's (V_B's and F's) dependency on the observed input, discharge trajectories from the high-fidelity `COMSOL` model are shown. The figures FIG. 4.2 and FIG. 4.3 show the observed output during CC discharge for constant compression $\widetilde{d} = 0$ under varying C-rate, as well as for a constant C-rate 2.5C under linearly increasing compression. FIG. 4.2 shows the voltage dependence on C-rate as introduced before, where discharge time is inversely proportional to C-rate. The coupling between V_B and d is generally weaker, however, some slight increase in voltage can be seen for high compression. This likely stems from the mechanical coupling between volumetric strain and porosity, given in EQ. 2.27. During discharge the solid volume fraction in the separator $\varepsilon_{\text{vol.}}$ is negative and decreasing, thus increasing the electrolyte porosity ν_e. In turn, a growing ν_e increases the diffusion coefficient and ion conductivity in the separator and therefore it might contribute to the increased voltage output, effectively decreasing the battery's internal resistance. This observed dependence therefore reflects the particular mechanical-electrochemical coupling assumed in the high-fidelity model. Stronger dependencies could be expected with a more refined coupling scheme or different materials and cell design.



(a) (b)

FIG 4.2: Battery voltage during discharge for (a) fix compression and various C-rates, and for (b) fix C-rate under increasing compression.

Nevertheless, the coupling between F, C-rate and $\widetilde{d}$ is noticeable. FIG. 4.3 shows that for $\widetilde{d} = 0$, the force starts at zero and then increases proportional to C-rate and time. This implies that the battery is swelling during discharge and thus exerting a positive force on the enclosing structure. Note that the simplified model assumptions give an opposite response compared to conventional LiBs, which typically swell during charge [47]. The figure also shows a non-linear increase of F with respect to $\widetilde{d}$, as further demonstrated in FIG. 4.4, which is consistent with the response of the hyper-elastic material model (per construction of the battery model in `COMSOL`).



(a) (b)

FIG 4.3: Battery reaction force during discharge for (a) fix compression and various C-rates, and for (b) fix C-rate under increasing compression.

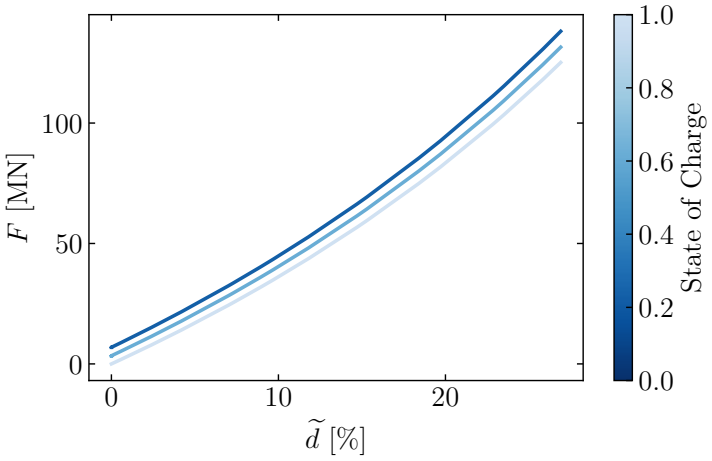
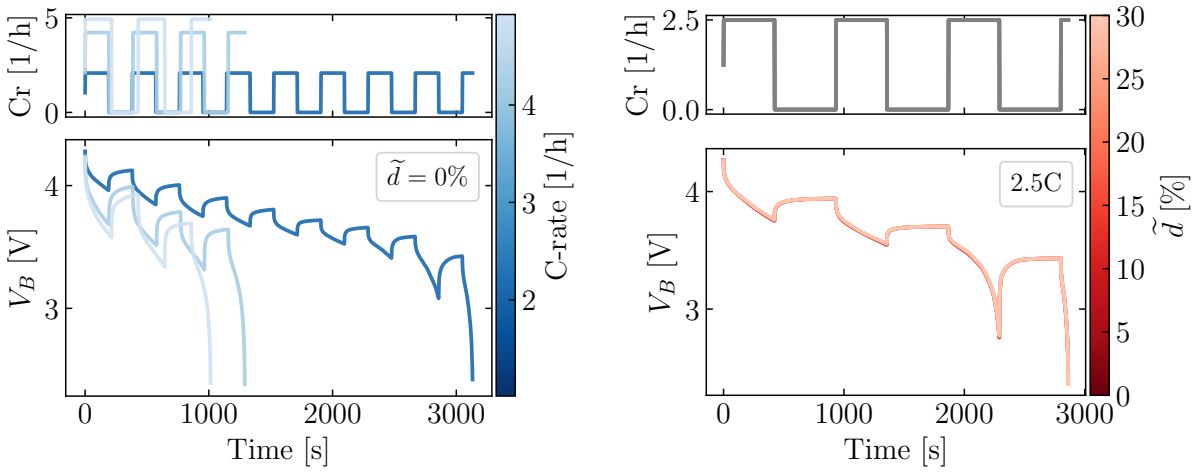


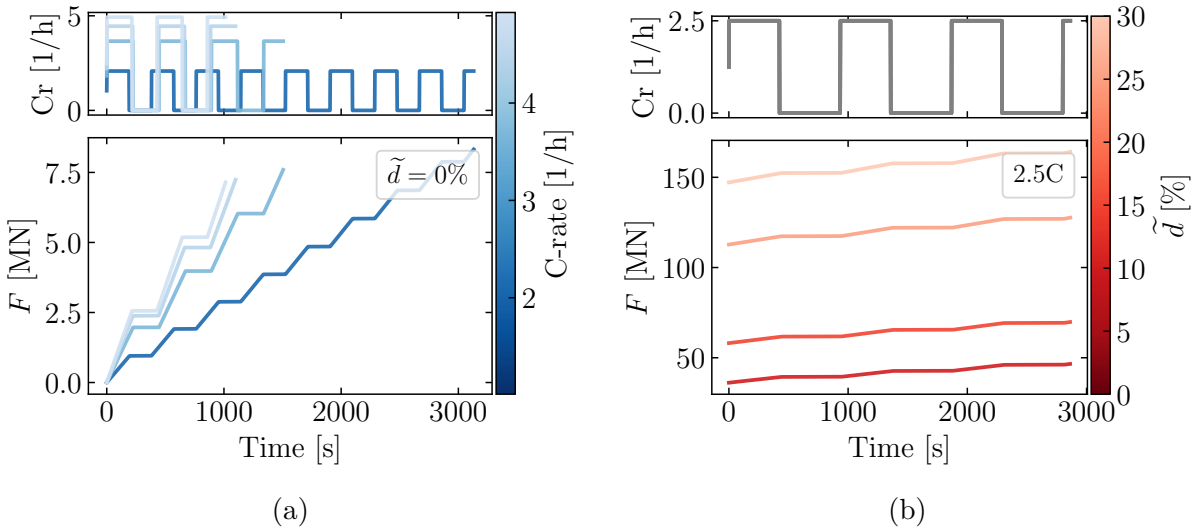
FIG 4.4: Battery reaction force as a function of compression for different SOC.

Analogous to previous figures, FIG. 4.5 and FIG. 4.6 show the observed output during pulsed discharge for constant compression along with constant C-rate. Noticeably, the voltage shows the characteristic transient relaxation following a transition from a finite current to zero current. While the voltage shows a gradual relaxation after a sudden change to zero current, the force response is instant, see FIG. 4.6. This validates the use of the ECM with a RC-branch and an instantly responsive spring for modeling the mechanical response.



(a) (b)

FIG 4.5: Battery voltage during pulsed current discharge for (a) fixed compression and varying C-rates, and for (b) fixed C-rate under varying compression. Current profiles are shown in the upper panels where Cr is C-rate.



(a) (b)

FIG 4.6: Battery reaction force during pulsed current discharge for (a) fixed compression and varying C-rates, and for (b) fixed C-rate under varying compression. Current profiles are shown in the upper panels where Cr is C-rate.

The randomly sampled training domain of observed inputs for both CC and pulses are shown in FIG. 4.7. For pulses, C-rate refers to the pulse wave amplitude, i_0 in EQ. 3.1. Worth noting is the increased sampling of C-rates during no compression.

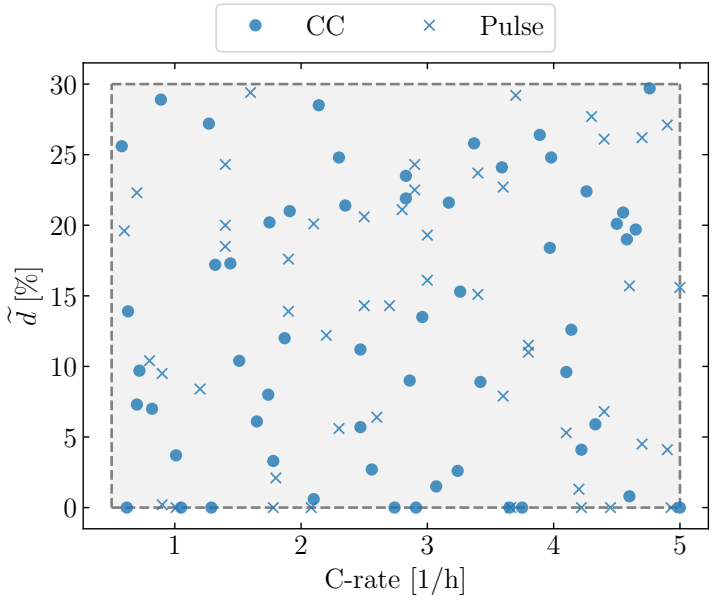


FIG 4.7: Training data samples for compression and C-rate for both constant current and pulse trajectories. The dashed box indicates the sampled parameter domain.

4.2 The Surrogate Model

The following section presents predictions from surrogate model in comparison to the high-fidelity model. The section is divided in voltage and force predictions respectively. Prior to that, a brief reflection on the training procedure is provided.

The surrogate model was trained on the CC and pulse data for 2 500 epochs, and the training and test loss in terms of RMSE for V_B and F are shown in Fig. 4.8. Although the loss curves in Fig. 4.8 suggest that training might not have fully converged. The training was limited to 2 500 epochs, due to long training times. For reference, it took approximately 700 minutes on a standard laptop CPU. With more advanced hardware and optimized programming, both training time and convergence could possibly be improved.

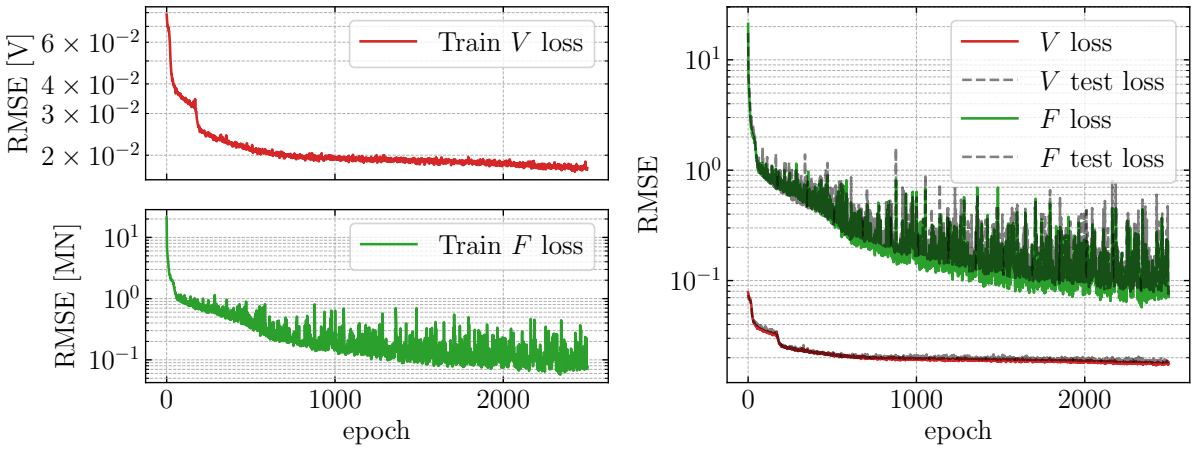


Fig 4.8: RMSE loss at each epoch during training of the surrogate model. (a) Training loss for voltage and force show separately. (b) Training and test loss for voltage and force shown on a shared axis.

Integration schemes and vectorization had a significant impact on training times. Training without integration for both the voltage and force branches, as done for the static NN and black box NN reference models, was around 100 times faster than the surrogate model. When assuming a full dependence on internal swelling and voltage, the surrogate model was trained without vectorization, leading to an increased training time by a factor of two. This highlights the computational cost of the fully general formulation, placing it outside of the scope of this thesis.

4.2.1 Voltage Predictions

To compare the surrogate model to the high-fidelity model and the fixed reference ECM introduced in Sec. 3.3.4.1, discharge curves are shown in Fig. 4.9. In this case, the

values for the fixed ECM circuit elements were obtained from a fit to a trajectory with C-rate = $3.22\,\mathrm{h}^{-1}$. Visually comparing the curves, it is clear that the surrogate model captures the end-of-discharge voltage drop more efficiently than the ECM with fixed element values. In other words, the result suggests that non-constant circuit elements are required. As a matter of fact, ECM element calibrations typically also include a SOC dependence (see e.g. [3]). Including a dependence on SOC further complicates calibrating ECMs.

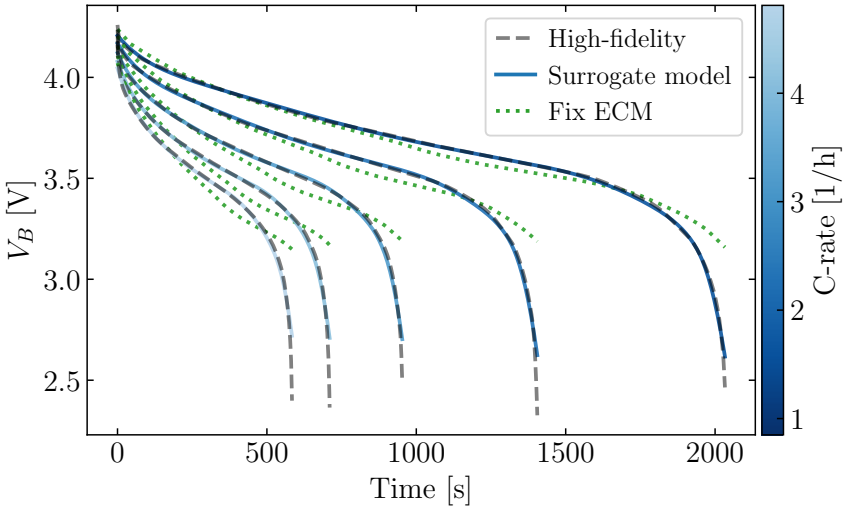


Fig 4.9: Battery voltage during constant current discharge at different C-rates for the high-fidelity model, along with predictions from the surrogate model and fix ECM. Note that the fix ECM has been calibrated for C-rate = $3.22\,\mathrm{h}^{-1}$.

Fig. 4.10 shows predictions on pulses for the surrogate model, the fix ECM and the high-fidelity model. Since both the surrogate model and the fixed ECM reference model are physics-based they do capture the transient behavior during pulsed discharge in a similar manner. The circuit element values for the fix ECM were obtained from a fit to C-rate = $2.8\,\mathrm{h}^{-1}$. Although the fix ECM captures the relaxing voltage, it does not capture the end-of-discharge drop as well as the non-linear elements.

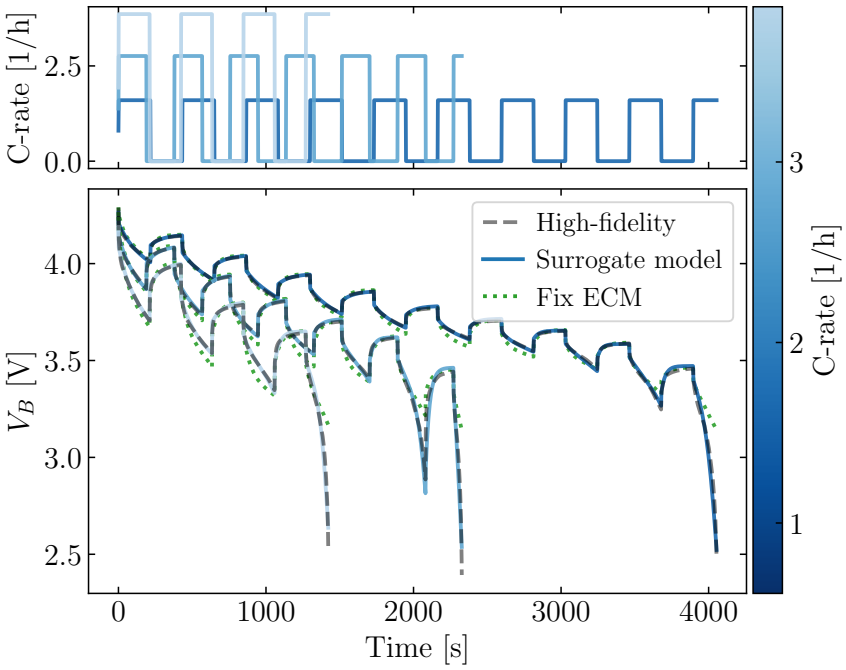


FIG 4.10: The upper panel shows pulsed current profiles corresponding to the voltage curves in the lower panel. Battery voltage is shown for the high-fidelity model along with the predictions from the surrogate model and the fix ECM. Note that the fix ECM has been calibrated for C-rate $= 2.8\,\mathrm{h}^{-1}$.

The circuit elements of the surrogate model are shown in FIG. 4.11 for different SOC, C-rates, and compressions. Here, the elements were constrained to always be larger than zero with a softplus acitvation function. Despite that single constraint, the predicted range for R_0 and R_1 is vastly different, and it seems like the end-of-discharge drop is not captured by R_0. Instead, R_1 and C_1 show significant changes as SOC approaches zero, where R_1 increases and C_1 decreases. Moreover, the dependence on SOC for R_1 and C_1 suggests that higher C-rates cause the change to occur at a higher SOC, which is in accordance with arguments in SEC. 2.1.2.5. Interestingly, even though $R_1 > 0$ was the only constraint, R_1 seems to flatten out at around $300\,\mathrm{m\Omega}$ for lower SOC and higher C-rates. This might be due to the cut-off voltage in high-fidelity model, as trajectories with high C-rate reaches the cut-off voltage for a higher SOC, see FIG. 2.4a for reference.

In addition to C-rate, the elements also show some relation for different compressions. In relation to the introduced coupling between mechanics and electrochemistry, a compressed cell gives a negative volumetric strain, which in turn causes an increase in electrolyte volume fraction as $\nu_e = \nu_{0,e}(1 - \varepsilon_{\mathrm{vol.}})$ in the separator. Further, ν_e is proportional to the electrolyte diffusivity and ion conductivities. Since conductivity is inversely proportional to resistance, an increased ionic conductivity could lead to a decreased internal resistance in the separator. Interpreting the elements for different SOC and compressions, as in FIG. 4.11, R_0 seems to be consistently lower with increased d. Moreover, the peak for C_1

also seems to be slightly lower as d increases, while R_1 is mildly affected. Although this is a qualitative analysis, the subsequent symbolic regression part of the pipeline aims to describe the relations.

To summarize, it is obvious that using an ECM with constant elements for all SOC, C-rates and applied compressions does not fully describe the voltage response of a battery, where the end-of-discharge drop is particularly apparent. Instead, by studying the NN-predicted elements for different SOC, C-rates and compressions, the introduction of a stronger non-linearity in R_1 and C_1 for lower SOC seems to replicate the high-fidelity model more accurately. While the overall influence of compression is less than the influence of C-rates, R_0 seems to pick up some variations.

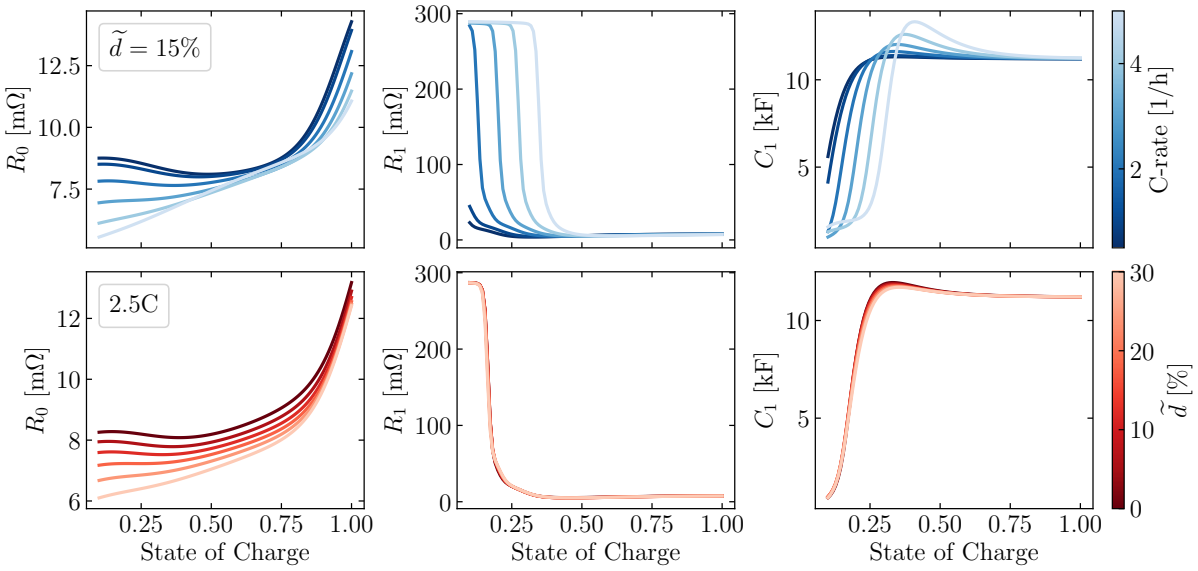


FIG 4.11: NN predicted circuit elements as functions of SOC. The upper panel shows different C-rate at a fixed compression of $\widetilde{d} = 15\,\%$, while the lower panel shows different compressions $\widetilde{d}$ for fixed C-rate $= 2.5\,\mathrm{h}^{-1}$.

4.2.1.1 Extrapolation

Training both the surrogate model and the black box model on discharge response only, and then extrapolating to charge (negative C-rate), yields the predictions shown in FIG. 4.12. The surrogate manages to predict the fundamental physical relationship between V_B and U_{eq}, where $V_B < U_{eq}$ during discharge and $V_B > U_{eq}$ during charge, as modeled by EQ. 2.17. The purely data-driven black box NN also manages to extrapolate to the charging conditions, however, it fails to satisfy the physical constraint of $V_B > U_{eq}$ during charge. The physically inconsistent predictions of the black box model promote the use of physical constraints. Although the surrogate model conserves this relation, the accuracy of the surrogate model is limited outside the training domain, as seen for the NN-predicted circuit elements.

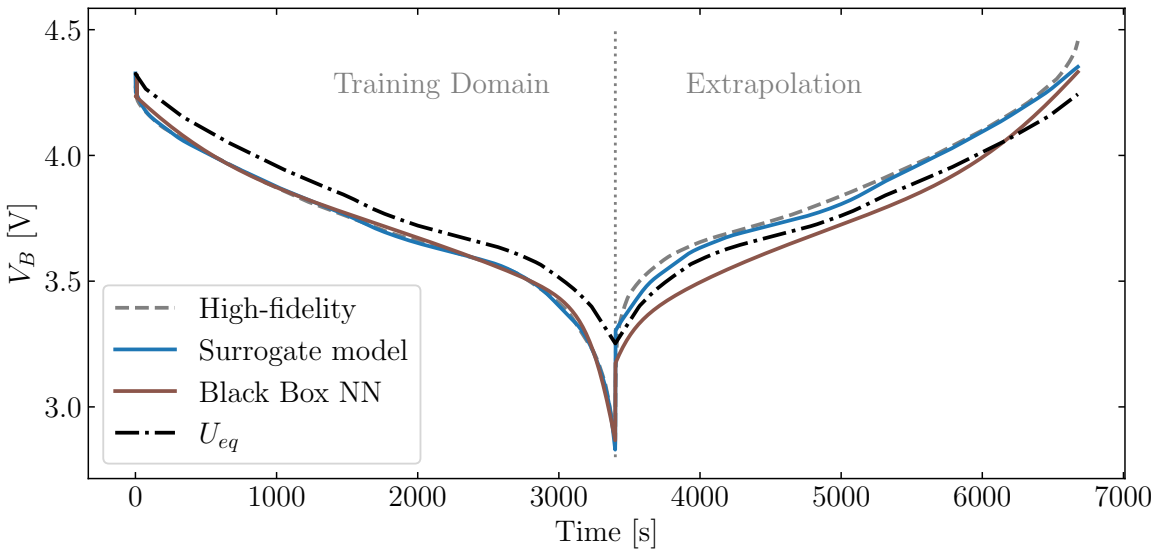


FIG 4.12: Extrapolation to full cycle for the surrogate model and the black box NN. The high-fidelity simulation and the equivalent voltage, U_{eq} are shown for reference. Both the surrogate and the black box were trained on the same data, which consisted of only discharge voltage curves.

To model the voltage curve during charging, a negative current was used as input for the surrogate model. Circuit element predictions for negative C-rates are shown in FIG. 4.13 for different SOC and displacements. Since the surrogate model was solely trained on positive C-rates, predicting for negative currents yields a different behavior compared to what was observed in FIG. 4.11. In particular, R_0 retains its increasing behavior with respect to SOC, however, the NN generally predicts higher values for negative currents. Considering the trend for positive currents, namely that decreased C-rates caused increased R_0, the extrapolation to negative currents continue this trend. A more significant difference between positive and negative currents is seen for R_1. Here, R_1 approaches zero for increasing negative C-rates. Thus, the predictions for R_1 continue the trend observed in FIG. 4.11, i.e., decreased C-rate leads to decreased R_1. The dependence on d, still remained the same for both R_0 and R_1, but not for C_1. Instead, C_1 showed a reverse dependence on d for negative currents.

This extrapolation behavior is somewhat expected for NNs only trained on positive currents. In general, the NNs extrapolated the element predictions in a sensible direction based on training data. Nevertheless, it showcases the obvious need for training data that captures all relevant operating conditions.

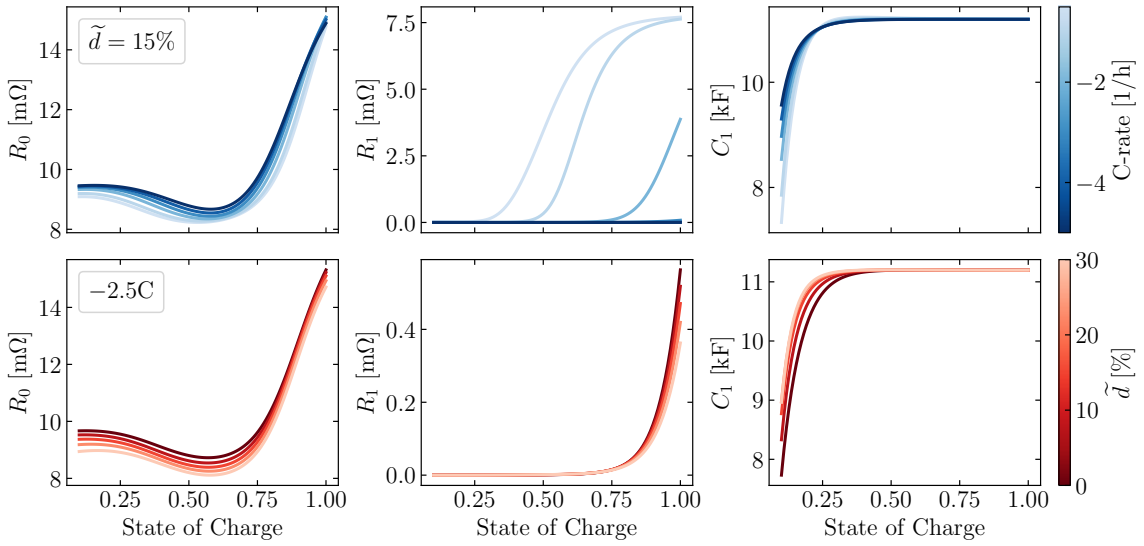


FIG 4.13: NN predicted circuit elements for negative current as functions of SOC. The upper panel shows different C-rate at a fixed compression of $\widetilde{d} = 15\,\%$, while the lower panel shows different compressions $\widetilde{d}$ for fixed C-rate $= -2.5\,\mathrm{h}^{-1}$.

While the surrogate model was used to show that physics-based modeling does not break physics when extrapolating to charging, we could step down in complexity and show that even the static ECM reference model respects the physical constraint $V_B > U_{eq}$ during charge. This can be seen in the dependency on i in the model equations and the positive resistor value R_{NN} in SEC. 3.3.4.2. Arguably, if following this physical concept is the main concern, the reference model could suffice. Moreover, the static model is around 100 times faster to train. However, extrapolating from a training domain of CC discharge to pulsed current discharge shows a potential downside of the static reference model.

The static ECM reference model, described in SEC. 3.3.4.2, is compared with the surrogate model for pulsed currents in FIG. 4.14. Note that both models were trained on CC discharges. As expected, the surrogate model captures the voltage relaxation dynamics. This is a direct consequence of the embedded ODE governing this transient behavior. However, since the static model lacks a dependence on time, it produces an instantaneous response to the changes in current.

The τ_{min} constraint explained in SEC. 3.3.5 demonstrates that incorporating prior knowledge of element values can improve extrapolation capabilities. Without the constraint, the NN-enhanced ECM struggles to identify all three elements from CC data alone, to a degree that makes pulse predictions satisfactory. Particularly difficult to capture is the capacitance C_1, whose role in capturing the relaxation behavior is minimal under constant current. Prior testing showed that an unbounded $C_1 > 0$ tends to be modeled smaller when solely trained on CC data, than when trained on pulses.

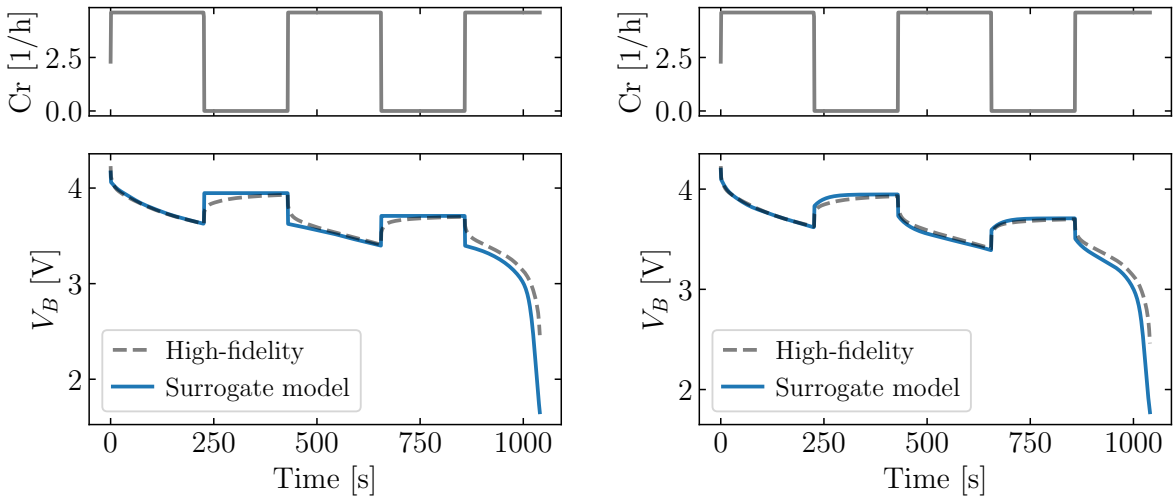
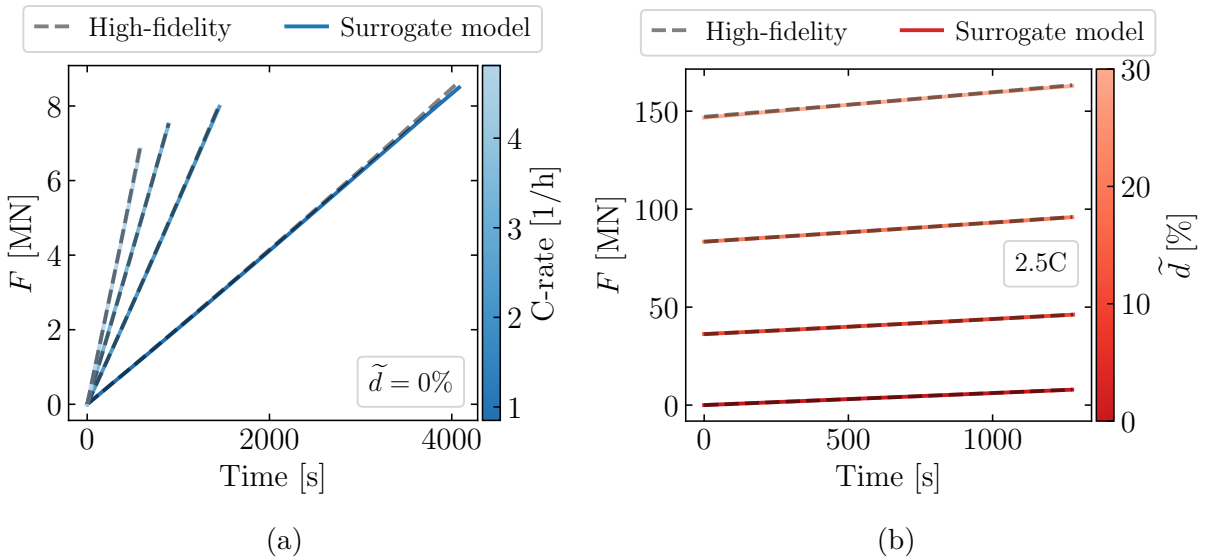


FIG 4.14: (a) Predictions with static ECM reference model and (b) surrogate model predictions on the pulsed currents shown in upper panels. Note that both the static NN and the surrogate model were only trained on CC.

4.2.2 Force Predictions

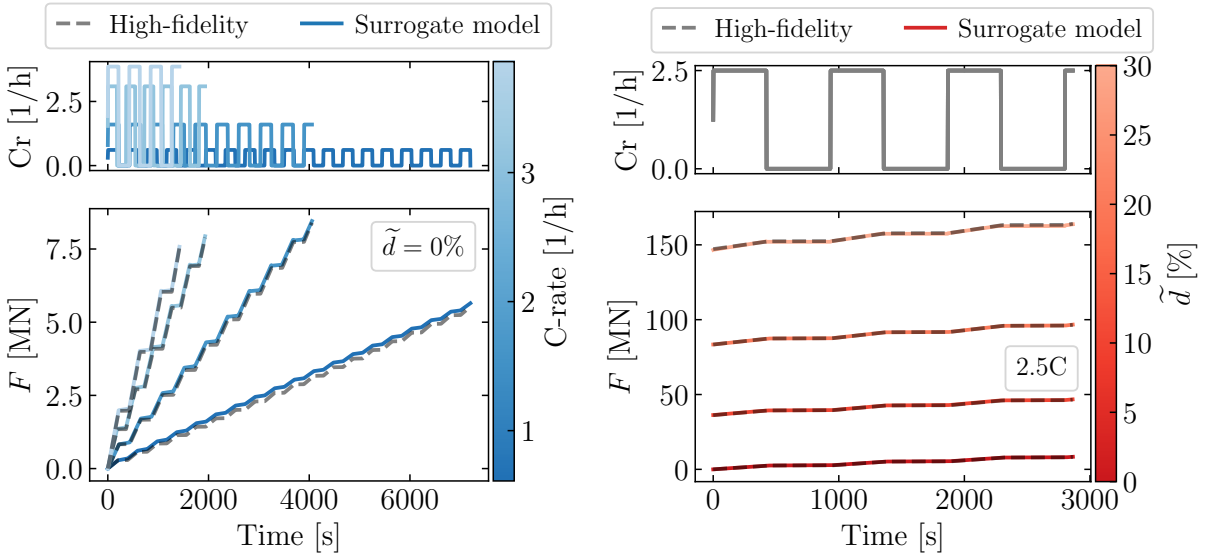
To visualize the surrogate model's ability to predict the force response for the high-fidelity model, a number of force trajectories during discharge are shown in FIG. 4.15 for CC and in FIG. 4.16 for pulses. Apart from voltage, the force depends strongly on compression and is therefore shown for fixed compression with varying C-rate and fixed C-rate with varying compression. For CC the force predictions in FIG. 4.15 appear to capture the increasing trend of the force data under (a) $\widetilde{d} = 0\,\%$ and (b) C-rate $= 2.5\,\mathrm{h}^{-1}$. However, for lower C-rate the prediction eventually drifts slightly away from the data. This could possibly be attributed to the integration of $\mathrm{NN}_{\dot{s}}$ deviating over many time steps.

The same behavior carries over to pulses, shown in FIG. 4.16, where a slight deviation becomes apparent for lower C-rates. Interestingly, the deviation is larger for the lowest C-rate at 2000 s than for a higher C-rate curve, ending at 2000 s. This suggests that the deviation might not only be attributed to integration drift, due to the amount of time steps.



(a) (b)

FIG 4.15: Battery reaction force data and predictions during CC discharge for (a) fix compression and various C-rates, and for (b) fix C-rate under increasing compression.



(a) (b)

FIG 4.16: Battery reaction force data and predictions during pulsed discharge for (a) fix compression and various C-rates, and for (b) fix C-rate under increasing compression. Current profiles are shown in the upper panels.

The force predictions are governed by k, $\dot{s}$ and s, shown in FIG. 4.17 to demonstrate their dependence on C-rate and $\widetilde{d}$. The stiffness k decreases approximately linearly with SOC, suggesting that the battery becomes stiffer throughout discharge. Moreover, k's dependence on C-rate is weak, the top left curves are nearly indistinguishable, whereas its dependence on $\widetilde{d}$ is significant and non-linear. This non-linear relation between k and $\widetilde{d}$ is consistent with the hyper-elastic material response, shown in FIG. 4.4.

The swelling s is assumed to increase with decreasing SOC as a simplified model assumption. In practice, battery swelling depends strongly on factors such as electrode materials. Conventional Li-ion batteries typically exhibit swelling during charging and shrinkage during discharge (see for instance [47]). This discrepancy does not limit the generality of the framework, as it allows for other dependencies of s on SOC. The predicted $\dot{s}$ decreases for increasing compression while it increases with increasing C-rate. The C-rate dependence of $\dot{s}$ is validated in the force predictions with fixed compression in Fig. 4.15 and Fig. 4.16, where the force increases faster with higher C-rate, which is consistent with faster swelling.

The apparent contrast between upper-middle and upper-right panels in Fig. 4.17 is a consequence of the change of variables. Since $\mathrm{dSOC}/\mathrm{d}t = -i/Q_0$, one obtains $\mathrm{d}s/\mathrm{dSOC} = -\dot{s}\,Q_0/i \propto \dot{s}/\text{C-rate}$. Since the predicted $\dot{s}$ scales approximately linear with C-rate at fix compression, dividing $\dot{s}$ with C-rate produces the observed collapse in the upper-right panel. In the lower panels, however, C-rate is fixed, which means that the $\widetilde{d}$ dependence propagates unchanged into $\mathrm{d}s/\mathrm{dSOC}$. The same $\widetilde{d}$-dependence can be seen in the time-domain force predictions in Fig. 4.15 and Fig. 4.16, where the baseline F spans orders of magnitude when $\widetilde{d}$ is swept. The near-linear scaling of $\dot{s}$ with C-rate implies a consistent result with swelling being driven by the rate of lithium intercalation, which is proportional to current. On the other hand, the relation between $\dot{s}$ and $\widetilde{d}$, is less straightforward to ground in physics, highlighting a limitation in the assumed EMM.

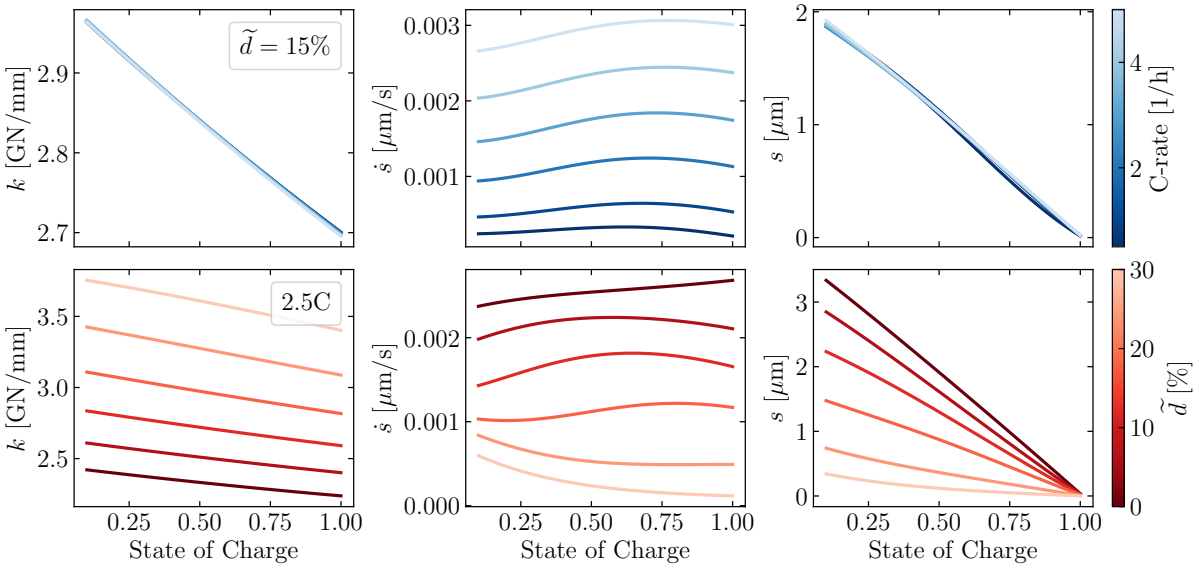


Fig 4.17: NN predicted stiffness, swelling rate and swelling as functions of SOC. The upper panel shows different C-rate at a fixed compression of $\widetilde{d} = 15\,\%$, while the lower panel shows different compressions $\widetilde{d}$ for fixed C-rate $= 2.5\,\mathrm{h}^{-1}$.

4.2.2.1 Mechanical Limitations

For the mechanical response to swelling and the applied pre-compression, the model setup together with the data does not give a clear distinction between the dynamically evolving swelling and stiffness. First of all, the modeling assumption is still based on the linear Hooke's law, which by default may not be sufficient to capture the non-linear mechanical response. Moreover, the evolution of the force during operation, given by $F = k \cdot (s - d) = ks - kd$, involves a multiplication of $k \cdot s$; both determined by NNs. Without prior knowledge about what values and ranges to expect, several possible combinations of k and s could be give the same evolution of F. Simply phrased, s could be strongly varying for SOC and k not, or vice versa.

To address this, prior knowledge, such as experimental data, could be used to constrain the networks. However, if prior knowledge about k and s is still lacking, another way to address the issue could be to generate more specific data from the high-fidelity model. For instance, one could interrupt the discharge early and then apply other compressions.

4.2.2.2 Extrapolation

As with the voltage response extrapolated to charge, the spring stiffness and swelling rate were predicted for negative currents, see Fig. 4.18. Note the model's inability to predict negative swelling rate, due to the softplus constraint. Strictly positive $\dot{s}$ for all inputs does not capture the shrinkage, which for these modeling assumptions occurs during charge. The predicted mechanical elements for negative currents in Fig. 4.19 this limitation.

However, by assuming that the predicted swelling rate follows the same sign as the current, the model reproduces the mirrored response from discharge to charge, as shown in Fig. 4.18. This assumes that the absolute current is used for the network prediction

$$\dot{s} = \text{sign}(i) \times \text{softplus}(\text{NN}(|i|,d,\text{SOC},s)).$$

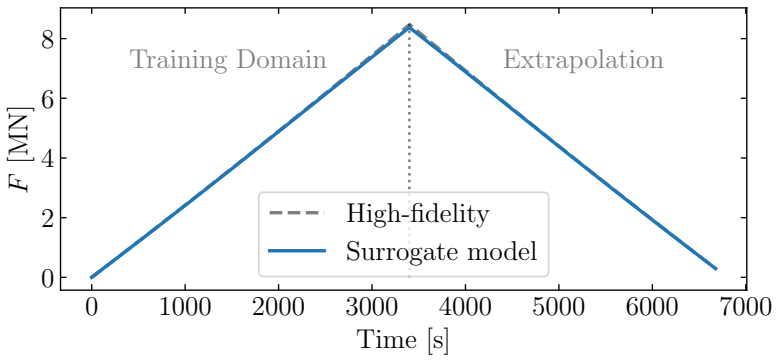


Fig 4.18: Force prediction within discharge training domain and extrapolated to charge.

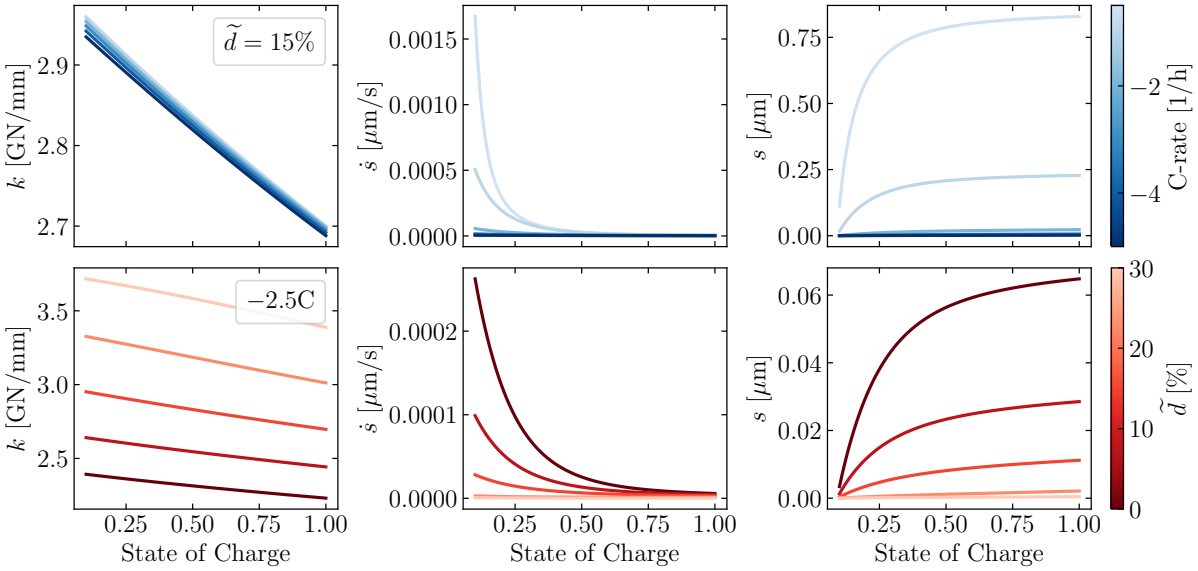


FIG 4.19: NN predicted spring stiffness and swelling rate for negative current as functions of SOC. The upper panel shows different C-rate at a fixed compression of $\widetilde{d} = 15\,\%$, while the lower panel shows different compressions $\widetilde{d}$ for fixed C-rate $= -2.5\,\mathrm{h}^{-1}$.

4.3 Symbolic Expressions

Before presenting some examples of symbolic expressions for the circuit elements, spring stiffness and swelling rate, TAB. 4.1 gives a summary of the overall dependencies. Specifically, TAB. 4.1 shows the frequency of latent inputs occurring in the expressions for each latent output. This simple study suggests that SOC and C-rate are the inputs that appear most frequently in the expressions for the circuit elements. Moreover, SR identified a dependence on $\widetilde{d}$ for R_0, however, not for R_1 and C_1. The spring stiffness depended on $\widetilde{d}$ and SOC but SR did not find a dependence on C-rate. Lastly, SR found $\dot{s}$ to be related to C-rate, $\widetilde{d}$ and s more often than on SOC.

TAB 4.1: Frequency of latent inputs occurring in the symbolic expressions for the latent outputs.

Latent Outputs	**Latent Inputs**			
	SOC	**C-rate**	$\widetilde{\boldsymbol{d}}$	$\boldsymbol{s}$
R_0	96%	80%	72%	–
R_1	92%	84%	0%	–
C_1	96%	78%	0%	–
k	78%	0%	96%	–
$\dot{s}$	25%	100%	89%	75%

To visualize and compare expressions from an optimization run with SR, the complexity

versus loss was plotted for each circuit element, the spring stiffness and $\dot{s}$. From the list of expressions, two expressions were selected, namely, the default expression chosen by the SR algorithm and a simpler expression chosen for contrast. The full list of all expressions, along with their loss and complexity plots can be found in App. C.

4.3.1 Symbolic Expressions for ECM Elements

Firstly, the extracted expressions for R_0 read

$$\text{Low complexity:} \quad R_0 = R_0^{\text{ref}} \cdot \sqrt{\text{SOC}^6 + 0.70\,(e^d)^{-C}}$$

$$\text{High complexity:} \quad R_0 = R_0^{\text{ref}} \cdot \sqrt{-0.24d + \left(\text{SOC}^2 - 0.31\right)^2 + 0.24\left(-\sqrt{C} + 2.1\text{SOC}^2 - 0.26\right)^3 + 0.77},$$

where C denotes C-rate and $R_0^{\text{ref}} = 10\,\text{m}\Omega$. Note that C, d and SOC are normalized, such that they lie between 0 and 1. The more complex expression was the default expression chosen by the SR algorithm. Both the simpler and the complex expression have a dependence on i, d and SOC. Moreover, the expressions show a decreased resistance for lower SOC, higher currents, increased compressions, which agrees with the observed NN predictions, in Fig. 4.20. Further, a limitation with the simpler expression may be that it suggests that R_0 does not depend on C-rate at $d = 0$. Moreover, it is obvious that these expressions were fitted in relation to how the NN was sampled, i.e., with positive values for C-rate. In particular, plugging in $C < 0$ for the expression with higher complexity yields a non-physical, complex-valued R_0.

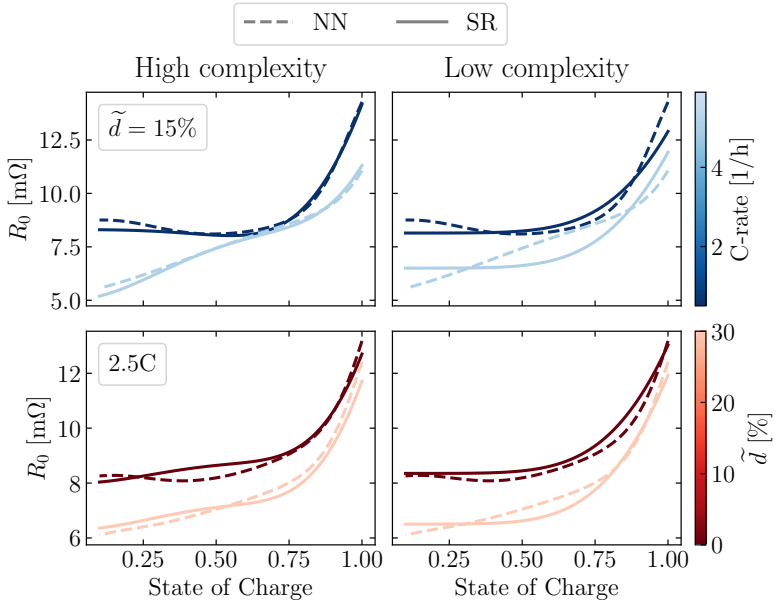


Fig 4.20: Visual comparison between SR and NN for two different C-rates and two different compressions for R_0. The complex expression is to the left and the simpler to the right.

Secondly, for R_1 the extracted expressions were

$$\text{Low complexity:} \quad R_1 = R_1^{\text{ref}} \cdot 29.1e^{-\left(7.81^{\text{SOC}} - C\right)^{12.3}}$$
$$\text{High complexity:} \quad R_1 = R_1^{\text{ref}} \cdot \left(\left(4.5e^{-\text{SOC}^{4.5}\left(51.3\text{SOC}^{2C} - 3.9\right)^4} - 1.6\right)^3 + 4.9\right),$$

with $R_1^{\text{ref}} = 10\,\text{m}\Omega$. As expected, the symbolic expressions for R_1 focuses entirely on capturing the strong non-linearity for lower SOC values. Specifically, both expressions have SOC as a negative value in the exponent, which means that a decrease in SOC corresponds to higher values of R_1. Arguably, the simpler expression will fail at higher SOC as it converges to 0. Moreover, comparing the expressions to the NN, it follows that the simpler expressions do not capture the location of the non-linearity as well; see Fig. 4.21. Another limitation from the data sampling, might be that the SR also does a fit to the plateau at around $300\,\text{m}\Omega$ for lower SOC. As discussed in Sec. 4.2.1, data points with low SOC and high C-rates are outside the training domain for the NN, due to the cut-off voltage, resulting in a plateau.

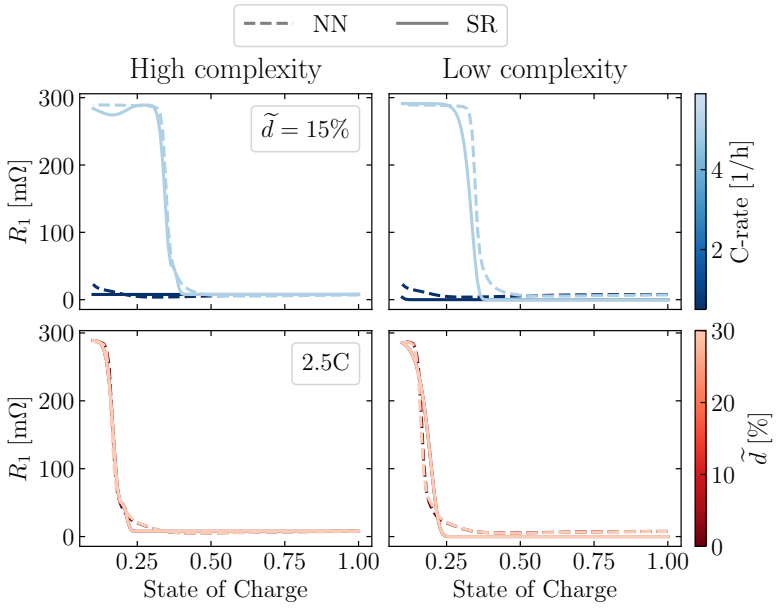


Fig 4.21: Visual comparison between SR and NN for two different C-rates and two different compressions for R_1. The complex expression is to the left and the simpler to the right.

Finally, for C_1 the two extracted expressions are given by

$$\text{Low complexity:} \quad C_1 = C_1^{\text{ref}} \cdot \left(2.25 - e^{(C - 7.53\text{SOC})^3}\right)^3$$
$$\text{High complexity:} \quad C_1 = C_1^{\text{ref}} \cdot \left(C^2\left(5.97\text{SOC} - 5.97\sqrt{\text{SOC}}\right)^2 + 11.1e^{-9.88e^{\left(C^{1.3} - 4.6\text{SOC} - 0.96\right)^3}}\right),$$

with $C_1^{\text{ref}} = 1\,\text{kF}$. While both expressions feature exponential C-rate and SOC, the expression with higher complexity has an additional term that contributes to an increase in C_1 for SOC $\in [0.4,1]$. Visually comparing the two expressions with NN predictions, as in FIG. 4.22, the simpler expression seems to miss the peak in C_1 at around SOC $= 0.5$ at higher C-rates. The first term in the expression with higher complexity accounts for this peak. Interestingly, none of the symbolic expressions for C_1 manage to capture variations in C_1 caused by different compressions, as observed in FIG. 4.11. However, comparing to C-rate and SOC, compressions have a mild overall effect on C_1.

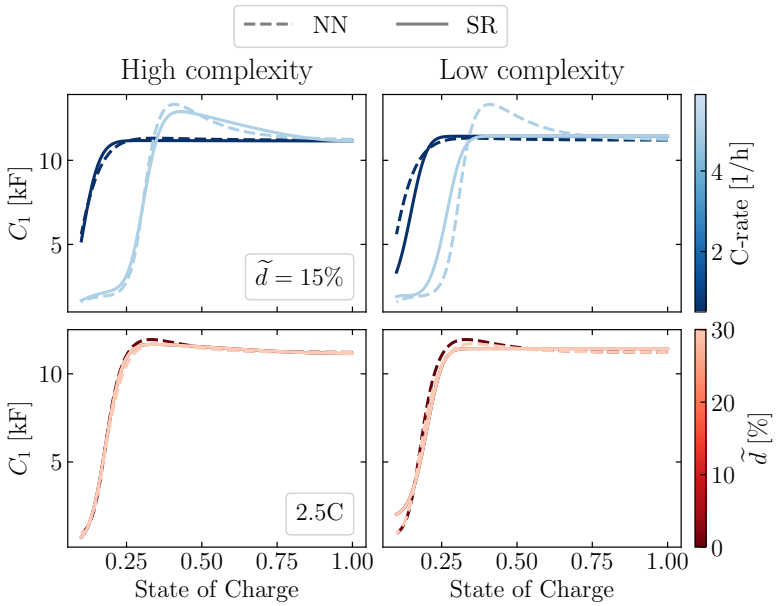


FIG 4.22: Comparison between SR and NN for two different C-rates and two different compressions for C_1. The complex expression is to the left and the simpler to the right.

4.3.2 Symbolic Expressions for Mechanics

The extracted expressions for the spring constant were

$$\text{Low complexity:} \quad k = k^{\text{ref}} \cdot \frac{6.58}{-d + 0.23\text{SOC} + 2.70}$$

$$\text{High complexity:} \quad k = k^{\text{ref}} \cdot \left(0.51d + 0.23\left(\text{SOC}^{0.94} - e^{d}\right)\left(0.19d^{6}\text{SOC}^{\text{SOC}} - d - 0.86\right) + 2.24\right),$$

where $k^{\text{ref}} = 1\,\text{MN}/\mu\text{m}$. Both expressions show an increasing k for higher compressions and lower SOC. Moreover, none of the extracted expressions, provided in TAB. C.4, had a dependence on C-rate, suggesting that k only depends on the present state and not the rate at which the battery is drained. Examining the simpler expression reveals a singularity at very high compressions and a non-symmetric behavior around $d = 0$. Compared to predictions from the NN, the simpler expression deviates slightly for lower SOC and higher compressions, see FIG. 4.23. Otherwise, both expressions seem to align with the NN predictions.

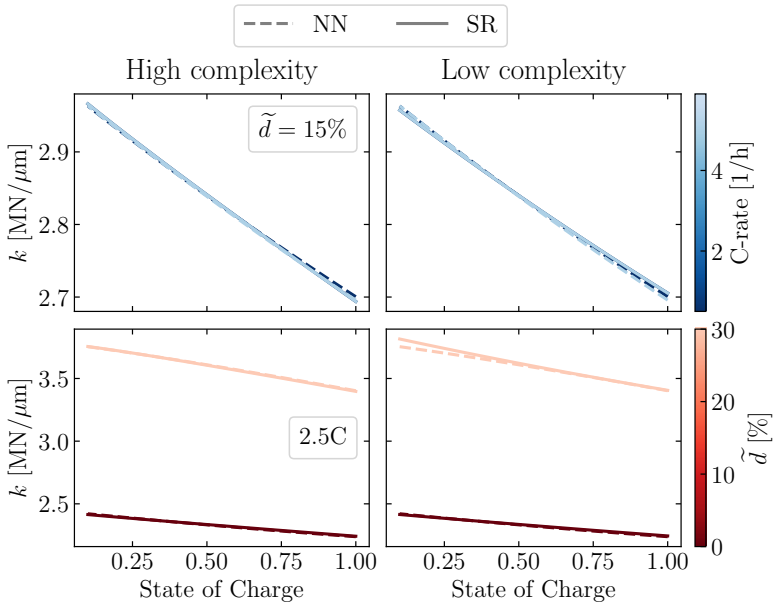


FIG 4.23: Visual comparison between SR and NN for two different C-rates and two different compressions for k. The complex expression is to the left and the simpler to the right.

In contrast to k, taking into account the expressions for $\dot{s}$, available in TAB. C.5, all depend on C and most of the expressions have $\dot{s} \propto C$. Since the time derivative of SOC is proportional to $-C$, there is a clear relation between the swelling rate and the rate of change in SOC. Examining FIG. 4.17, the relation between SOC and s is close to linear with slope that varies with d. This relation is clearly shown for the simpler of the two extracted expressions for $\dot{s}$ with the reference value $\dot{s}^{\text{ref}} = 0.001\,\mu\text{m}\,\text{s}^{-1}$,

$$\text{Low complexity:} \quad \dot{s} = \dot{s}^{\text{ref}} \cdot (4.901C\,(1.062 - d))$$

$$\text{High complexity:} \quad \dot{s} = \dot{s}^{\text{ref}} \cdot \left(5.881C^{1.056} + 0.105\right)\left(e^{(-d-0.323)^3 e^{s^2-(d-s-\text{SOC}+0.099)^3}} - 0.056\right),$$

which assumes that $\dot{s}$ is not affected by SOC. However, as seen in FIG. 4.24, the NN suggests a slight variation in $\dot{s}$ for different SOC. The complex expression suggests a more intricate dependence on C, d, SOC and s, and seems to capture the overall curve better. Interestingly, the expression with higher complexity suggests that a change in s can occur for $C = 0$.

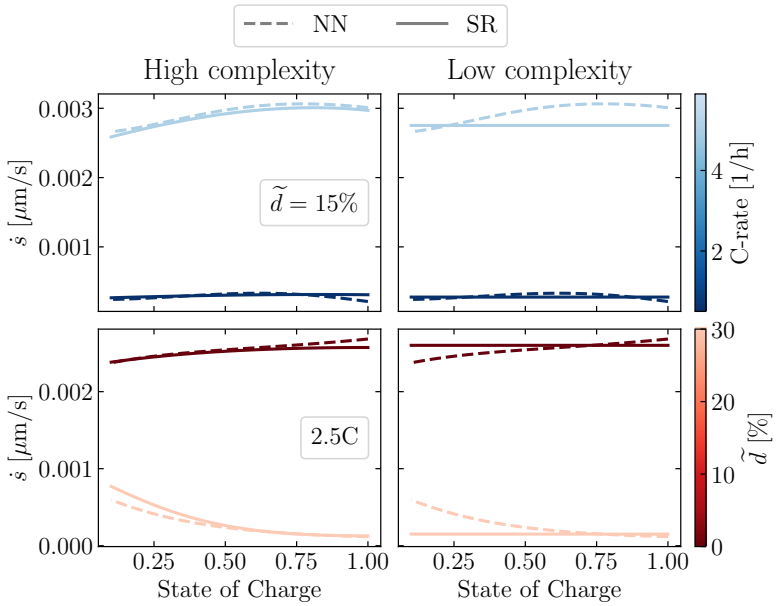


FIG 4.24: Visual comparison between SR and NN for two different C-rates and two different compressions for $\dot{s}$. The complex expression is to the left and the simpler to the right.

To summarize, SR provides an alternative and more interpretable way to describe the non-linear circuit elements, spring stiffness and swelling. However, as for any fitting method, adding complexity generally improves the fit. Thus, using a simpler expression to simulate the voltage and force response might not be ideal for all cases. For instance, the simple expression for R_1 converges to 0 for higher SOC. Since $\dot{V}_1 = V_1/(R_1 C_1) - i/C_1$, the limit $R_1 \to 0$ is problematic. On the other hand, if the main goal is to understand the hidden relations, a simple expression might still cover the overall trends, allowing for an easier interpretation. In fact, considering the loss-to-complexity curves in APP. C, the simpler expressions were chosen after a significant drop in loss. Adding more complexity does not drastically improve the loss, highlighting that the simpler expressions capture at least some fundamental relations.

We also claim that the implemented SR scheme should be seen as an iterative process, where knowledge from previous runs could be encoded as constraints in future runs. As an example, running another SR for R_1, we have learned that the expression needs a lower constraint to avoid $R_1 \to 0$, for SOC $\to 1$. This could be achieved by adjusting the optimization accordingly

$$R_1 = R_1^{\text{ref}} \cdot f_{R_1}(i,d,\text{SOC}) = R_1^{\text{ref}} \cdot \left[(g_{R_1}(i,d,\text{SOC}))^2 + R_{1,\text{low}}\right],$$

with $R_{1,\text{low}}$ being a constant and $g_{R_1}(i,d,\text{SOC})$ being the symbolic expression that we want to discover. Embedding g_{R_1} in an overarching expression, as shown, could be done

by specifying a template in `PySR`. Finally, this is somewhat analogous to encoding physical constraints by choosing activation functions and constraints for the last layer of a NN.

4.3.3 Symbolic Expression for Equivalent Voltage

In addition to the symbolic expressions for the elements, the SR was used to discover a symbolic expression for U_{eq}. The expression chosen by the SR algorithm was

$$U_{eq}(\mathrm{SOC}) = (0.313 - \mathrm{SOC})^2 - \left(1.45 - \left(2.17\mathrm{SOC} + \left(\mathrm{SOC}^2 + 0.436\right)^6\right)^{0.25}\right)^3 + 3.8.$$

The obtained expression's prediction of U_{eq} is shown in Fig. 4.25.

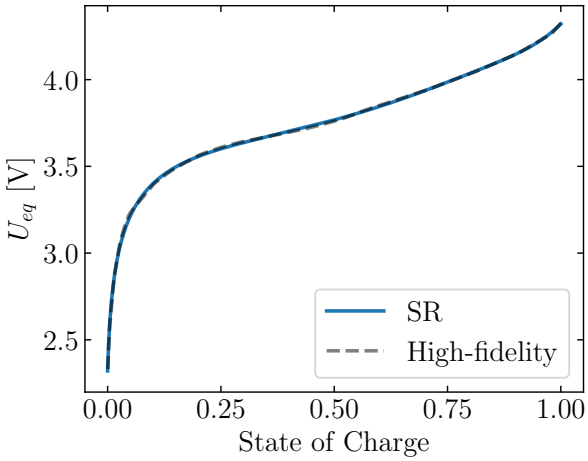


Fig 4.25: Equivalent voltage from SR expression and data from the high-fidelity model.

4.3.4 Voltage and Force Predictions with Symbolic Expressions

To show examples of the symbolic expressions' ability to model the voltage and force, Fig. 4.26 and Fig. 4.27 shows their predictions under pulse currents. The SR predictions were made using the expressions with high complexity together with the U_{eq} expression. The symbolic expression reproduces a similar result as the NNs. However, as discussed earlier the symbolic expression should be used with caution when extrapolating beyond the training data domain.

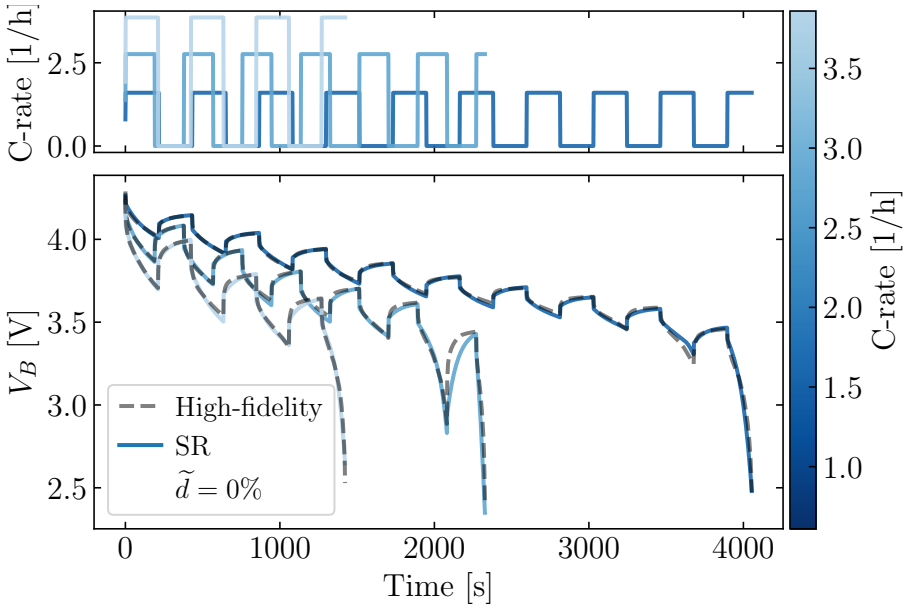


Fig 4.26: Voltage predictions for the symbolic expressions calculated with the ECM equations for zero compression, along with voltage data from the high-fidelity model. The upper panel shows the pulsed currents.

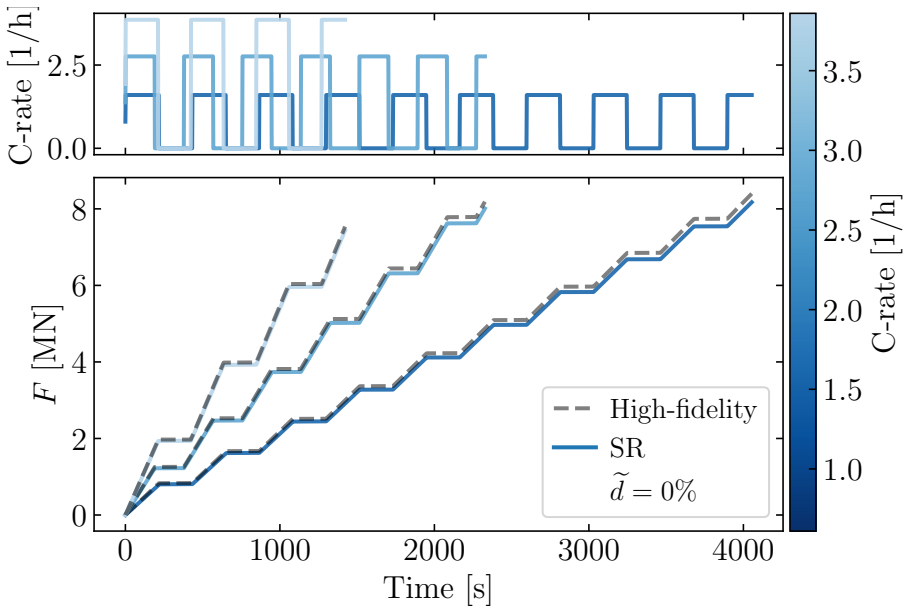


Fig 4.27: Force predictions for the symbolic expressions calculated with the spring equation for zero compression, along with force data from the high-fidelity model. The upper panel shows the pulsed currents

5

Conclusion and Future Outlook

5.1 Summary

In summary, this thesis developed and demonstrated a framework for enhancing simpler physics-based models using data from high-fidelity battery simulations. While the framework allows for different equivalent models, a first-order ECM was assumed for the current-to-voltage response, and a non-linear spring and a swelling rate were used to model the electrochemically induced swelling force for different compressions. Couplings between the simplified models were introduced by letting NNs predict the circuit elements, spring stiffness and swelling rate under different C-rates, compressions and SOC.

Compared with an otherwise identical ECM that uses constant circuit elements, the NN-enhanced surrogate model captures the current-to-voltage response more accurately, especially towards the end of discharge. It does so without the need for recalibration, or continuous adjustment of the circuit elements during the simulation. In contrast to a fully data-driven approach, embedding NNs in this particular ECM structure preserved two important aspects outside the training domain: the physical relation to the equilibrium voltage, U_{eq}, and a time-dependent voltage relaxation. Finally, the framework also included a SR scheme, which gave a symbolic representation of how the circuit elements, spring stiffness and swelling rate vary in different conditions.

Although this framework shows potential in replicating a high-fidelity battery model, implementing a physics-based surrogate model with NNs is more involved than a purely data-driven approach. For this demonstration, a central challenge was the need to integrate the model equations during training. Unlike a data-driven method, which directly predicts the targets, the integration increased computational cost and training time significantly. Moreover, applying adequate constraints and scaling factors for the NNs requires prior physical knowledge. This knowledge could either be obtained from experiments or from iterations, as done in this work. Due to the slow training, iteratively finding satis-

factory result might be tedious. Improving the training procedure and reducing training time is, therefore, a relevant consideration for future work.

Looking forward, this framework can be a useful tool to complement fixed-value ECM models, as a fast modeling alternative that can be adapted to changes in a high-fidelity model. We also believe that the framework can enable more targeted experimental designs, aimed at identifying circuit element relations rather than to cycle through different operating conditions such as various currents, temperatures and pre-compressions. With access to a high-fidelity model, the surrogate model could be trained on different scenarios, and then validated with a smaller set of experiments. If new variables, like temperature, are added, updating the high-fidelity model and re-training the surrogate model might be less cumbersome than expanding the look-up table with new experiments.

While this thesis demonstrated the entire pipeline from high-fidelity model to symbolic expressions, we believe that it is possible to "enter" the pipeline at any stage. For instance, if one already has experimental data for voltage-to-current response under different conditions, the surrogate model can still be trained and used to extrapolate. Moreover, if one already has a look-up table, SR can still be used to gain insights into how the circuit elements vary in different scenarios, potentially guiding future experiments. To end, the framework essentially provides a way to move from discrete to continuous look-up tables, while offering the capability of incorporating different physical considerations.

5.2 Future Work

Since the scope of this thesis has been to demonstrate a methodology for constructing a surrogate model over finding the optimal the NN implementation, we believe design choices for the NNs could be further investigated. For instance, factors like NN size, regularization schemes and learning rates all affect the training and performance of the NNs. A natural extension of this work is to conduct a more thorough study, where different designs are compared. In addition to improving the NNs, one could implement a more rigorous SR construction by iteratively applying constraints.

Another extension is to generalize the surrogate model to batteries not covered in the present study. With the current framework, this requires new data from a high-fidelity model that could be used to train new surrogate models for different battery types. Ideally, a future model could perhaps cover several batteries within a single surrogate model.

Following previous work, the surrogate model could also be evaluated on more complex current profiles, such as more randomized pulses with varying height or continuous profiles like Gaussian random fields [29], [30], [31]. More complex current profiles could be used

in a testing scenario to show limitations of the model. In addition, including them in the training could potentially yield more robust predictions on unseen current profiles.

5.2.1 Other Equivalent Models

For this work, the surrogate model assumed a first-order ECM and a non-linear spring as simplified equivalent models. However, in different scenarios, these assumed models might lack the ability to describe features in the high-fidelity model.

5.2.1.1 Other Equivalent Electrochemical Models

As argued in Sec. 2.1.2.5, several phenomena acting on different timescales affect current-to-voltage response. One of the assumptions for this model was to use a first-order ECM with a single RC-branch. However, an extension would be to introduce a second RC-branch with elements R_2 and C_2, producing a second time-constant $\tau_2 = R_2C_2$, which typically fulfills $\tau_1 < \tau_2$ [12]. Since the two branches have identical shape, it might require a more carefully designed training scheme, some additional constraints for the second branch, or that the second branch is introduced after the first was trained. Nonetheless, a second RC-branch might be needed in scenarios that require insights into relaxations at different timescales.

5.2.1.2 Other Equivalent Mechanical Model

The surrogate model in this work considers only an instantaneous mechanical force response from an applied compression and swelling, whereas a more general scenario could include a dynamic response. Importantly, this requires data or a high-fidelity model that models a dynamic force response. Taking inspiration from the electrochemical part of the surrogate model, a natural extension would be to include a viscous damper in the mechanical model in addition to the spring. The force-to-displacement relation for a damper is given by $F = c\dot{d}$, where c is a damping coefficient.

Analogous to the structure of the first-order ECM, two springs and a damper could be arranged as in Fig. 5.1. Similar to the capacitance C_1 in the ECM, the damper, c_1, determines a time-constant for the mechanical relaxation. With this extension, the force ODE has to be integrated inside the training loop.

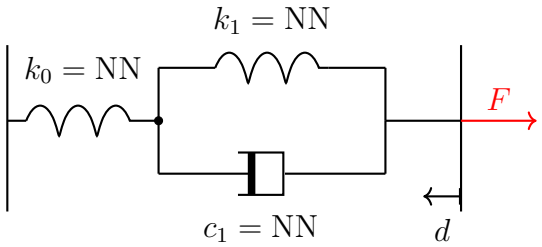


FIG 5.1: Extended EMM with an additional parallel damper-and-spring, k_2 and c, connected in series with k_1.

5.2.1.3 Equivalent Thermal Models

In addition to the ECM for electrochemical responses and the EMM for mechanical responses, equivalent thermal models could be used to describe thermal properties. An example, implemented by Kumar *et al.*, is shown in FIG. 5.2 [48]. Here, the generated heat $h(t)$ is modeled with a current source, and the ambient temperature as a voltage source $V_a = T_a$. Moreover, the voltage V_s resembles the battery's surface temperature T_s, and R_T and C_T determine the evolution of T_s. Solving for T_s gives the corresponding ODE

$$\frac{\mathrm{d}T_s}{\mathrm{d}t} = -\frac{T_s}{R_T C_T} + \frac{T_a}{R_T C_T} + \frac{h(t)}{C_T}.$$

Similar to how the ECM is extended to include mechanical effects, R_T and C_T could also be tuned using NNs. For this approach, a simple connection to the ECM could be achieved by assuming that all Ohmic losses are converted into heat, i.e., $h(t) = i^2 R_0$. An opposite connection would be to assume temperature dependent circuit elements in the ECM. However, this extension also requires a modified high-fidelity model that accounts for thermal effects. With a dynamically evolving temperature, the ODE needs to be integrated during training.

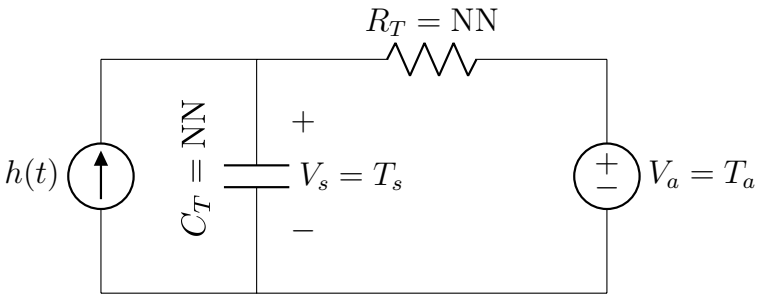


FIG 5.2: Equivalent thermal model, consisting of a current source $h(t)$ and voltage source V_a along with a resistor, R_T, and a capacitor C_T. $h(t)$ models the generated heat and V_a the ambient temperature, while C_T and R_T governs the evolution of surface temperature.

5.2.2 Irreversible Processes over Multiple Cycles

Another future consideration is to model other mechanical effects on the battery, such as degradation effects. While the present study only covered reversible swelling, a realistic behavior would include irreversible degradation over many cycles. Modeling this would require a scheme for simulating many cycles, rather than simply one discharge. In addition, the high-fidelity model would also have to be extended to capture longer term degradation effects. Considering irreversible swelling, it is crucial to let the circuit elements have a dependence on s. In terms of training, this means that the circuit elements are predicted inside the integrator, which aligns with the general assumptions in the training procedure Sec. 3.3.1.

The current training scheme integrates through the training loop, which can be costly. The integration itself is more time consuming than the network predictions. Furthermore, first-order integrators might introduce drift over long trajectories. Both of these effects scale with the trajectory length. Therefore, training on multiple cycles to probe degradation might require an alternative approach or more powerful hardware.

Bibliography

[1] Nobel Prize Outreach. "Popular information: The Nobel Prize in Chemistry 2019." Nobelprize.org. Accessed: Jan. 28, 2026. [Online]. Available: https://www.nobelprize.org/prizes/chemistry/2019/popular-information/.

[2] D. P. Laudani, D. Milillo, M. Quercio, F. R. Fulginei, and L. Sabino, "A comprehensive review of equivalent circuit models and neural network models for battery management systems," *Batteries*, vol. 12, no. 1, Jan. 2026, ISSN: 2313-0105. DOI: 10.3390/batteries12010037. [Online]. Available: https://www.mdpi.com/2313-0105/12/1/37.

[3] M.-K. Tran et al., "A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters," *Journal of Energy Storage*, vol. 43, p. 103 252, Nov. 2021, ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.103252. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X2100949X.

[4] F. Dietrich and W. Schilders, "Scientific machine learning," *Mathematische Semesterberichte*, vol. 72, no. 2, pp. 89–115, Sep. 2025. DOI: 10.1007/s00591-025-00399-4. [Online]. Available: https://doi.org/10.1007/s00591-025-00399-4.

[5] F. Torabi and P. Ahmadi, "Lithium-based batteries," in *Simulation of Battery Systems: Fundamentals and Applications*, Academic Press, 2020, ch. 8, pp. 263–309, ISBN: 978-0-12-816212-5. DOI: 10.1016/B978-0-12-816212-5.00012-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012816212500012X.

[6] A. R. Matty, "Modelling of electrode swelling in lithium- ion batteries," Master's thesis, Department of Industrial and Materials Science, Chalmers University of Technology, Gothenburg, Sweden, 2025. [Online]. Available: https://odr.chalmers.se/items/fac77df5-06d2-40d0-9e10-c17587b915a0.

[7] F. Brosa Planella et al., "A continuum of physics-based lithium-ion battery models reviewed," *Progress in Energy*, vol. 4, no. 4, p. 042 003, Oct. 2022, ISSN: 2516-1083. DOI: 10.1088/2516-1083/ac7d31. [Online]. Available: https://iopscience.iop.org/article/10.1088/2516-1083/ac7d31.

[8] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," *Journal of The Electrochemical Society*, vol. 140, no. 6, pp. 1526–1533, Jun. 1993, ISSN: 0013-4651. DOI: 10.1149/1.2221597. [Online]. Available: https://iopscience.iop.org/article/10.1149/1.2221597.

[9] S. Kulathu et al., "A three-dimensional thermal-electrochemical-mechanical-porous flow multiscale formulation for battery cells," *International Journal for Numerical Methods in Engineering*, vol. 125, no. 13, e7464, Jul. 2024, ISSN: 0029-5981. DOI: 10.1002/nme.7464. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/nme.7464.

[10] A. Stefanopoulou and Y. Kim, "System-level management of rechargeable lithium-ion batteries," in *Rechargeable Lithium Batteries*, A. A. Franco, Ed. Elsevier, 2015, pp. 281–302, ISBN: 978-1-78242-090-3. DOI: 10.1016/B978-1-78242-090-3.00010-9. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9781782420903000109.

[11] J. S. Newman and K. E. Thomas-Alyea, *Electrochemical Systems*, 3rd ed. Hoboken, New Jersey: John Wiley & Sons, 2004, ISBN: 0-471-47756-7.

[12] A. Nyman, T. G. Zavalis, R. Elger, M. Behm, and G. Lindbergh, "Analysis of the polarization in a li-ion battery cell by numerical simulations," *Journal of The Electrochemical Society*, vol. 157, no. 11, A1236, Sep. 2010, ISSN: 00134651. DOI: 10.1149/1.3486161. [Online]. Available: https://iopscience.iop.org/article/10.1149/1.3486161.

[13] M. Winter and R. J. Brodd, "What are batteries, fuel cells, and supercapacitors?" *Chemical Reviews*, vol. 104, no. 10, pp. 4245–4270, Oct. 2004, ISSN: 0009-2665. DOI: 10.1021/cr020730k. [Online]. Available: https://pubs.acs.org/doi/10.1021/cr020730k.

[14] B. Xia, B. Ye, and J. Cao, "Polarization voltage characterization of lithium-ion batteries based on a lumped diffusion model and joint parameter estimation algorithm," *Energies*, vol. 15, no. 3, p. 1150, Feb. 2022, ISSN: 1996-1073. DOI: 10.3390/en15031150. [Online]. Available: https://www.mdpi.com/1996-1073/15/3/1150.

[15] W. C. Young and R. G. Budynas, "Stress and strain: Important relationships," in *Roark's Formulas for Stress and Strain*, 7th ed. New York: McGraw-Hill, 2002, pp. 9–17, Originally published 1938, ISBN: 0-07-072542-X.

[16] A. Kossa, M. T. Valentine, and R. M. McMeeking, "Analysis of the compressible, isotropic, neo-hookean hyperelastic model," *Meccanica*, vol. 58, no. 1, pp. 217–232, Jan. 2023. DOI: 10.1007/s11012-022-01633-2. [Online]. Available: https://doi.org/10.1007/s11012-022-01633-2.

[17] F. Yang, "Lithiation-induced swelling of electrodes," *Journal of Energy Storage*, vol. 75, p. 109 634, Jan. 2024, ISSN: 2352152X. DOI: 10.1016/j.est.2023.109634. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352152X23030323.

[18] C. Forssén, "Learning from data TIF285: Lecture notes," Chalmers University of Technology, Gothenburg, Sweden, 2022. Accessed: Mar. 5, 2026. [Online]. Available: https://cforssen.gitlab.io/tif285-book.

[19] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989, ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0893608089900208.

[20] S. A. Faroughi et al., "Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing," 2023. arXiv: 2211.07377. [Online]. Available: https://arxiv.org/abs/2211.07377.

[21] N. Kovachki et al., "Neural operator: Learning maps between function spaces," *The Journal of Machine Learning Research*, vol. 24, no. 89, pp. 1–97, 2023. [Online]. Available: https://jmlr.org/papers/v24/21-1524.html.

[22] S. L. Brunton, University of Washington. "AI/ML+Physics Part 3: Designing an Architecture [Physics Informed Machine Learning]." (Aug. 3, 2024). Accessed: Feb. 5, 2026. [Online]. Available:

https://www.youtube.com/watch?v=fiX8c-4K0-Q&list=PLMrJAkhIeNNQ0BaKuBKY43k4xMo6NSbBa&index=4.

[23] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sep. 1998. DOI: 10.1109/72.712178. [Online]. Available: https://ieeexplore.ieee.org/document/712178.

[24] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," 2023. arXiv: 2308.08468. [Online]. Available: https://arxiv.org/abs/2308.08468.

[25] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, *Neural ordinary differential equations*, 2019. arXiv: 1806.07366. [Online]. Available: https://arxiv.org/abs/1806.07366.

[26] M. Innes et al., "A differentiable programming system to bridge machine learning and scientific computing," 2019. arXiv: 1907.07587. [Online]. Available: https://arxiv.org/abs/1907.07587.

[27] M. Cranmer, "Interpretable machine learning for science with pysr and symbolicregression.jl," 2023. arXiv: 2305.01582. [Online]. Available: https://arxiv.org/abs/2305.01582.

[28] M. Hassanaly et al., "Pinn surrogate of li-ion battery models for parameter inference, part i: Implementation and multi-fidelity hierarchies for the single-particle model," *Journal of Energy Storage*, vol. 98, p. 113 103, Sep. 2024, ISSN: 2352152X. DOI: 10.1016/j.est.2024.113103. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352152X24026896.

[29] Y. Zhuang, Y. Zheng, Y. Che, and R. Teodorescu, "Physics-informed neural network surrogate modeling of single particle model for lithium-ion batteries," 2026. arXiv: 2601.01466. [Online]. Available: https://arxiv.org/abs/2601.01466.

[30] P. Brendel, I. Mele, A. Rosskopf, T. Katrašnik, and V. Lorentz, "Parametrized physics-informed deep operator networks for design of experiments applied to lithium-ion-battery cells," *Journal of Energy Storage*, vol. 128, p. 117 055, Aug. 2025, ISSN: 2352152X. DOI: 10.1016/j.est.2025.117055. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X25017682.

[31] A. A. Panahi, D. Luder, B. Wu, G. Offer, D. U. Sauer, and W. Li, "Fast and generalisable parameter-embedded neural operators for lithium-ion battery simulation," *Energy and AI*, vol. 22, p. 100 647, Dec. 2025, ISSN: 26665468. DOI: 10.1016/j.egyai.2025.100647. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S266654682500179X.

[32] M. Hassanaly et al., "Pinn surrogate of li-ion battery models for parameter inference. part ii: Regularization and application of the pseudo-2d model," 2024. arXiv: 2312.17336. [Online]. Available: https://arxiv.org/abs/2312.17336.

[33] H. Pang, L. Wu, J. Liu, X. Liu, and K. Liu, "Physics-informed neural network approach for heat generation rate estimation of lithium-ion battery under various driving conditions," *Journal of Energy Chemistry*, vol. 78, pp. 1–12, Mar. 2023, ISSN: 20954956. DOI: 10.1016/j.jechem.2022.11.036. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2095495622006349.

[34] M. Sun et al., "Residual-channel attention physics-informed neural network for surrogate modeling of a sodium-ion battery single-particle model," *Journal of Power Sources*, vol. 682, p. 240 366,

Aug. 2026, ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2026.240366. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037877532601116X.

[35] J. Christensen, "Modeling diffusion-induced stress in li-ion cells with porous electrodes," *Journal of The Electrochemical Society*, vol. 157, no. 3, A366, Jun. 2010, ISSN: 00134651. DOI: 10.1149/1.3269995. [Online]. Available: https://iopscience.iop.org/article/10.1149/1.3269995.

[36] S. Golmon, K. Maute, and M. L. Dunn, "Numerical modeling of electrochemical–mechanical interactions in lithium polymer batteries," *Computers & Structures*, vol. 87, no. 23-24, pp. 1567–1579, Dec. 2009, ISSN: 00457949. DOI: 10.1016/j.compstruc.2009.08.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0045794909002053.

[37] Y. Bai, Y. Zhao, W. Liu, and B.-X. Xu, "Two-level modeling of lithium-ion batteries," *Journal of Power Sources*, vol. 422, pp. 92–103, May 2019, ISSN: 03787753. DOI: 10.1016/j.jpowsour.2019.03.026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378775319302654.

[38] A. Asheri, M. Fathidoost, V. Glavas, S. Rezaei, and B.-X. Xu, "Data-driven multiscale simulation of solid-state batteries via machine learning," *Computational Materials Science*, vol. 226, p. 112 186, Jun. 2023, ISSN: 09270256. DOI: 10.1016/j.commatsci.2023.112186. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025623001805.

[39] J. Brucker, W. G. Bessler, and R. Gasper, "Grey-box modelling of lithium-ion batteries using neural ordinary differential equations," *Energy Informatics*, vol. 4, no. S3, p. 15, Sep. 2021, ISSN: 2520-8942. DOI: 10.1186/s42162-021-00170-8. [Online]. Available: https://energyinformatics.springeropen.com/articles/10.1186/s42162-021-00170-8.

[40] J. Brucker, R. Behmann, W. G. Bessler, and R. Gasper, "Neural ordinary differential equations for grey-box modelling of lithium-ion batteries on the basis of an equivalent circuit model," *Energies*, vol. 15, no. 7, p. 2661, Apr. 2022, ISSN: 1996-1073. DOI: 10.3390/en15072661. [Online]. Available: https://www.mdpi.com/1996-1073/15/7/2661.

[41] Z. Guo, Y. Li, Z. Yan, and M.-Y. Chow, "A neural-network-embedded equivalent circuit model for lithium-ion battery state estimation," 2024. arXiv: 2407.20262. [Online]. Available: https://arxiv.org/abs/2407.20262.

[42] Z. Al-Hashimi, T. Khamis, M. A. Kouzbary, N. Arifin, H. Mokayed, and N. A. A. Osman, "A decade of machine learning in lithium-ion battery state estimation: A systematic review," *Ionics*, vol. 31, no. 3, pp. 2351–2377, Mar. 2025. DOI: 10.1007/s11581-024-06049-4. [Online]. Available: https://link.springer.com/article/10.1007/s11581-024-06049-4.

[43] C. Wang et al., "A review of battery soc estimation based on equivalent circuit models," *Journal of Energy Storage*, vol. 110, p. 115 346, Feb. 2025, ISSN: 2352152X. DOI: 10.1016/j.est.2025.115346. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352152X25000593.

[44] V. Sulzer, S. G. Marquis, R. Timms, M. Robinson, and S. J. Chapman, "Python battery mathematical modelling (pybamm)," *Journal of Open Research Software*, vol. 9, no. 1, p. 14, Jun. 2021, ISSN: 2049-9647. DOI: 10.5334/jors.309. [Online]. Available: https://openresearchsoftware.metajnl.com/article/10.5334/jors.309/.

[45] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," 2019. arXiv: 1912.01703. [Online]. Available: https://arxiv.org/abs/1912.01703.

[46] M. Cranmer. "PySR API Reference," Accessed: May 11, 2026. [Online]. Available: https://astroautomata.com/PySR/v1.5.9/api.html.

[47] Z. Guo, M. A. Kashem, C. Briggs, L. Oldham, A. Barai, and J. Marco, "Lithium-ion battery swelling characterisation using high-resolution optical fibre-based surface profiler and thermal imaging," *Journal of Power Sources*, vol. 681, p. 240 296, Jul. 2026, ISSN: 03787753. DOI: 10.1016/j.jpowsour.2026.240296. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378775326010463.

[48] P. Kumar, G. Rankin, K. R. Pattipati, and B. Balasingam, "Model-based approach to long term prediction of battery surface temperature," *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, vol. 4, no. 1, pp. 389–399, Jan. 2023, ISSN: 2687-9735. DOI: 10.1109/JESTIE.2022.3214060. [Online]. Available: https://ieeexplore.ieee.org/document/9917283/.

[49] K. Cui, T. Gao, and D. Shi, "Physics-informed bayesian neural network for li-ion battery continual learning," in *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)*, IEEE, May 2025, pp. 01–06, ISBN: 979-8-3315-4299-3. DOI: 10.1109/ICPS65515.2025.11087868. [Online]. Available: https://ieeexplore.ieee.org/document/11087868/.

Declaration of Generative AI Use

During the code development of this project GitHub Copilot, ChatGPT and Claude has been used to streamline the coding work and accelerate debugging. In particular, Copilot assisted with repetitive tasks and inline suggestions. ChatGPT and Claude were used for brainstorming, debugging and generating implementation suggestions, docstrings and comments.

During the preparation of the report Claude was used to improve grammar, readability and suggesting synonyms. Moreover, Claude and ChatGPT have been used to implement and solve errors in generating TikZ figures.

Code and Data Availability

The `Python` source code and the data from `COMSOL` can be downloaded from `github.com/JesperNoord/Msc_thesis_battery_surrogate`.

List of Acronyms

ADAM adaptive moment estimation.

ANN artificial neural network.

C-rate charge rate.

CC constant current.

DeepONet deep operator network.

DFN Doyle-Fuller-Newman model.

DNN deep artificial neural network.

ECM equivalent circuit model.

EMM equivalent mechanical model.

FE finite element.

FE^2 two-level finite element method.

FFNN feed forward neural network.

FNO Fourier neural operator.

HGR heat generation rate.

IVP initial value problem.

Li Lithium.

LiB lithium-ion battery.

MAE mean absolute error.

ML machine learning.

MSE mean squared error.

NMC Lithium Nickel Manganese Cobalt Oxide.

NN neural network.

NO neural operator.

NODE neural ordinary differential equation.

OCV open-circuit voltage.

ODE ordinary differential equation.

P2D pseudo-two-dimensional.

P4D pseudo-four-dimensional.

PDE partial differential equation.

PENN physics-encoded neural network.

PI-DeepONet physics-informed DeepONet.

PINN physics-informed neural network.

ReLU rectified linear unit.

RMSE root-mean-squared error.

SciML scientific machine learning.

SOC state of charge.

SPM single particle model.

SR symbolic regression.

A

Appendix: Literature Summary

Outlined in Tab. A.1 are previous work, where battery and ML models are combined. Note the abbreviations:

ECMech – Electrochemical-mechanical model.

SPMT – Single particle thermal model.

PE-FNO – Parameter-embedded Fourier neural operator.

PI-BNN – Physics-informed Bayesian neural network.

PI-BiLSTM – Physics-informed bidirectional long-short term memory.

PI-DeepONet – Physics-informed deep neural operator network.

FFNN – Feed forward neural network.

Tab A.1: Literature overview of SciML in battery research, highlighting battery model, ML architecture, and inputs and outputs.

Researcher	**Bat. Model**	**Mech**	**ML Arch.**	**Input**	**Output**	**Comment**
Hassanali *et al.*[28]	SPM	-	PINN	r, t	c_n,c_p, ϕ_e,ϕ_p	
Hassanali *et al.*[32]	P2D	-	PINN	r, t, x, $j_{0,n}$, D_p	c_n,c_p, ϕ_e,ϕ_p,ϕ_n, c_e	
Sun *et al.*[34]	SPM	$\sim$	PINN	r, t	c_n, ϕ_e, ϕ_p, j	Na-ion bat. $\sigma \rightarrow D$ coupling. ICs and BCs are encoded
Zhuang *et al.*[29]	SPM	-	PINN, DeepONet	r,t, $i(t)$	$c_{n,p}(r,t)$	Comparing architectures and extracts V from c
Panahi *et al.*[31]	SPM	-	DeepONet, FNO, PE-FNO	r,t, $i(t)$, D, R	$c_{n,p}(r,t)$	Extracts V from c
Brendel *et al.*[30]	SPM	-	PI-DeepONet	r,t, $i(t)$, $D_{p,n}$	$c_{n,p}(r,t)$	
Asheri *et al.*[38]	ECMech	✔	FFNN	J_s, $c^i_{s,n}$, $c^i_{b,n}$	$\langle \dot{d} \rangle$, $\dot{c}_{s,n}$, $\dot{c}_{b,n}$	Active material damage
Pang *et al.*[33]	SPMT	-	PI-BiLSTM	$i(t)$, $V(t)$, $T_{amb}(t)$, $c_{s,n}(t)$, $c_{s,p}(t)$	heat generation rate (HGR)	Models heat rates in LiBs
Cui *et al.*[49]	ECM	-	PI-BNN	i, SOC	R_0, R_1, C_1, R_2, C_2, U_{eq}	Probabilistic parameter estimation. Recurrent structure
Guo *et al.*[41]	ECM	-	FFNN	$i(t)$, $V(t)$, $T(t)$	R_0, R_1, C_1	Estimates R and C values based on current, voltage and temperature
Brucker *et al.*[40]	ECM	-	FFNN	i, SOC	R_1	R_0 and C_1 trainable constant parameters

- $r \in [0,R_k]$ – particle radius at electrode $k \in \{n,p\}$
- t – time
- j – current density
- x – spatial dimension for P2D
- c_k – concentration of solid particles at $k \in \{n,p,e\}$

- $c_{s,k}$ – surface concentration of at electrode $k \in \{n,p\}$
- $c_{b,k}$ – bulk concentration of at electrode $k \in \{n,p\}$
- ϕ_k – potential at $k \in \{n,p,e\}$
- $j_{0,k}$ – exchange current density at $k \in \{n,p\}$
- D_k – diffusion coefficient at electrode $k \in \{n,p\}$
- $i(t)$ – applied current
- J_s – Pore wall flux
- $\langle d \rangle$ – accumulated damage
- SOC – state of charge

B

Appendix: Symbolic Regression Setup

B.1 PySR Setup

```python
def get_default_settings():
    bin_ops = ["+", "*", '-','/','^']
    un_ops = ['sqrt','square', 'cube','exp','log', 'sin', 'cos', 'tan']
    nest_const = {'sin':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                          'cos':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                          'tan':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                          'log':  {'sin': 0, 'cos': 0, 'tan': 0},
                          'exp':  {'sin': 0, 'cos': 0, 'tan': 0},
                          'exp':  {'exp': 0}
                          }
    consts = {"^": (-1, 1)}
    op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1, 'cube':
    ↪  1,'exp': 1,'log': 1, 'sin': 3, 'cos': 3, 'tan': 3}
    var_names = ['i','d','soc']
    return bin_ops, un_ops, nest_const,consts, op_comps, var_names

def get_var_names(elem):
    if elem in ['R0', 'R1', 'C1','k'] or elem == None:
        return ['C','d','soc']
    elif elem in ['s']:
        return ['C','d','soc','s']
    elif elem in ['Ue']:
        return ['soc']
    else:
        raise ValueError(f"Unknown element: {elem}")

def get_settings(elem):
    bin_ops, un_ops, nest_const,consts, op_comps, var_names = get_default_settings()

    if elem == 'R0':
```

```
        un_ops = ['exp', 'log','sqrt','square', 'cube',]
        nest_const = {'exp':  {'exp': 1, 'log': 1},
                      'log':  {'exp': 1, 'log': 1}}
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 2}
    if elem == 'R1':
        un_ops = ['exp', 'log','sqrt','square', 'cube',]
        nest_const = {'exp':  {'exp': 1, 'log': 1},
                      'log':  {'exp': 1, 'log': 1}}
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 2}
    if elem == 'C1':
        un_ops = ['exp', 'log','sqrt','square', 'cube',]
        nest_const = {'exp':  {'exp': 1, 'log': 1},
                      'log':  {'exp': 1, 'log': 1}}
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 2}

    if elem == 'k':
        un_ops = ['exp', 'log','sqrt','square', 'cube',]
        nest_const = {'exp':  {'exp': 1, 'log': 1},
                      'log':  {'exp': 1, 'log': 1}}
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 2}
    if elem == 's':
        un_ops = ['exp', 'log','sqrt','square', 'cube',]
        nest_const = {'exp':  {'exp': 1, 'log': 1},
                      'log':  {'exp': 1, 'log': 1}}
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 2}

    if elem == 'Ue':
        bin_ops = ["+", "*", '-','/']
        un_ops = ['sqrt','square', 'cube','exp','log', 'sin', 'cos', 'tan']
        nest_const = {'sin':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                      'cos':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                      'tan':  {'sin': 0, 'cos': 0, 'tan': 0, 'log': 0, 'exp': 0},
                      'log':  {'sin': 0, 'cos': 0, 'tan': 0},
                      'exp':  {'sin': 0, 'cos': 0, 'tan': 0},
                     }
        op_comps = {"+": 1, "*": 1, '-': 1, '/': 1,'^':2, 'sqrt': 1, 'square':1,
        ↪  'cube': 1,'exp': 1,'log': 1, 'sin':3,'cos':3,'tan':3}

    var_names = get_var_names(elem)

    return bin_ops, un_ops, nest_const,consts, op_comps, var_names

```

```
71 def setup_model(its = int(1e3), pops = 30, selection = "accuracy",run_id = None, elem
   ↪  = None):
72     bin_ops, un_ops, nest_const, consts,op_comps, var_names = get_settings(elem)
73 
74     model = PySRRegressor(
75         model_selection=selection,
76         niterations=its,
77         binary_operators=bin_ops,
78         unary_operators=un_ops,
79         populations=pops,
80         nested_constraints = nest_const,
81         verbosity=0,
82         variable_names=var_names,
83         constraints = consts,
84         batching = True,
85         complexity_of_operators = op_comps,
86         complexity_of_constants = 1,
87         maxsize = 30,
88         batch_size = 512,
89         run_id= run_id )
90     return model
```

C

List of Symbolic Expressions

C.1 Expressions for R_0

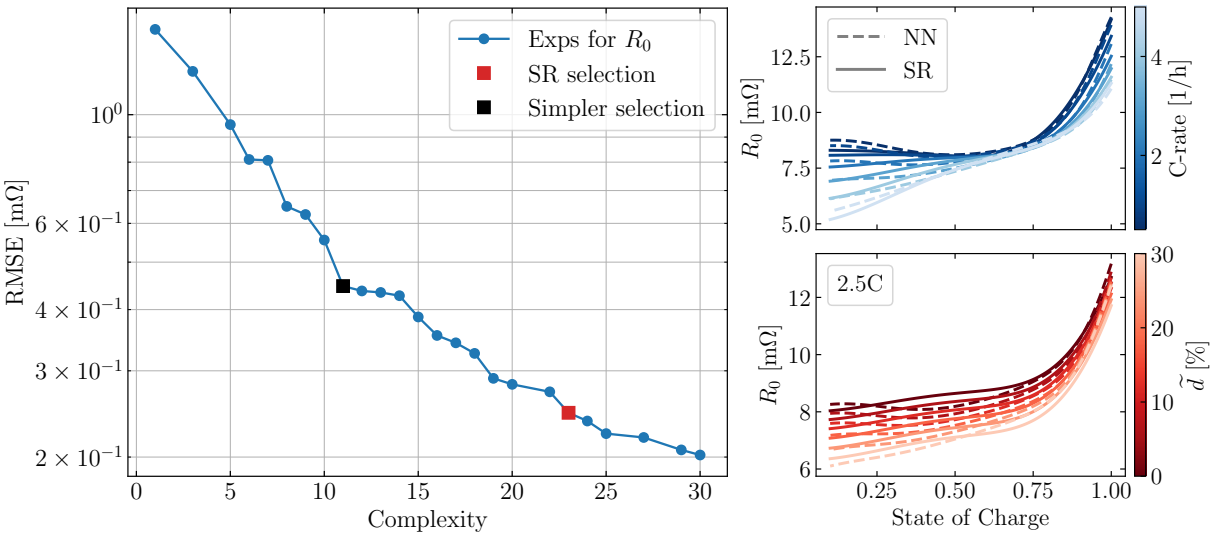


FIG C.1: To the left, RMSE loss versus complexity for R_0 symbolic expressions. The chosen expression by SR and an additional simpler expression are marked in red and green respectively. To the right, predictions of R_0 with the NN and the best expression from SR.

Tab C.1: List of all symbolic expressions for R_1

Comp.	**Loss** [mΩ]	**Expression**
1	1.493	0.824
3	1.226	soc^{soc}
5	0.954	$\left(soc^3 + 0.465\right)^{0.5}$
6	0.810	$\left(soc^6 + 0.552\right)^{0.5}$
7	0.807	$0.739 \left(e^{soc^4}\right)^{0.5}$
8	0.650	$\left(\frac{e^{soc^3}}{C+1.48}\right)^{0.5}$
9	0.625	$\left(\frac{e^{soc^4}}{C+1.38}\right)^{0.5}$
10	0.555	$\left(\frac{e^{soc^3}}{Cd+1.69}\right)^{0.5}$
11	0.447	$\left(soc^6 + 0.698 \left(e^d\right)^{-C}\right)^{0.5}$
12	0.437	$\left(-0.27C - 0.27d + soc^6 + 0.827\right)^{0.5}$
13	0.434	$\left(soc^6 + 0.728 \left(e^d + 0.148\right)^{-C}\right)^{0.5}$
14	0.427	$\left(0.4390.304^C + 0.4390.443^d + soc^6\right)^{0.5}$
15	0.386	$\left(\left(soc^9 + 0.724\right) \left(\left(0.274e^{soc}\right)^C\right)^d\right)^{0.5}$
16	0.354	$\left(\left(soc^9 + 0.729\right) \left(\left(0.597soc + 0.17\right)^C\right)^d\right)^{0.5}$
17	0.342	$\left(\left(soc^9 + 0.758\right) \left(\left(0.5250.057^d + 0.525soc\right)^C\right)^{0.5}\right)^{0.5}$
18	0.326	$0.834soc^9e^{-C} + 0.834 \left(\left(0.515soc + 0.465\right)^C\right)^d$
19	0.290	$0.85soc^9e^{-\sqrt{C}} + 0.85 \left(\left(0.552soc + 0.425\right)^C\right)^d$
20	0.282	$\left(\frac{soc^9}{C+0.633} + 0.717\right)^{0.5} \left(\left(0.536soc + 0.434\right)^C\right)^d$
22	0.272	$0.846soc^9e^{-C-0.35065746d} + 0.846 \left(\left(0.587soc + 0.428\right)^C\right)^d$
23	0.246	$\left(-0.241d + \left(soc^2 - 0.305\right)^2 + 0.241 \left(-C^{0.5} + 2.12soc^2 - 0.262\right)^3 + 0.765\right)^{0.5}$
24	0.237	$\left(-0.228d + \left(soc^{1.5} - 0.387\right)^2 + 0.228 \left(-C^{0.5} + 2.3soc^2 - 0.323\right)^3 + 0.756\right)^{0.5}$
25	0.223	$\left(-0.234d + \left(\left(soc + 0.0819\right)^2 - 0.419\right)^2 + 0.234 \left(-C^{0.5} + 2.15soc^2 - 0.329\right)^3 + 0.753\right)^{0.5}$
27	0.219	$\left(-0.24d + \left(\left(soc + 0.0858\right)^2 - 0.429\right)^2 + 0.24 \left(2.17soc^2 - \left(C + 0.037\right)^{0.5} - 0.302\right)^3 + 0.756\right)^{0.5}$
29	0.207	$\left(-0.306C^{0.25}d + 0.634soc^4 - 0.306soc + 0.306 \left(2.0soc^2 - \left(C + 0.234\right)^{0.5}\right)^3 + 0.868\right)^{0.5}$
30	0.202	$\left(-0.307C^{0.276}d - 0.307soc + 0.307 \left(1.97soc^2 - \left(C + 0.248\right)^{0.5}\right)^3 + 0.691 \left(soc^9\right)^{0.5} + 0.876\right)^{0.5}$

C.2 Expressions for R_1

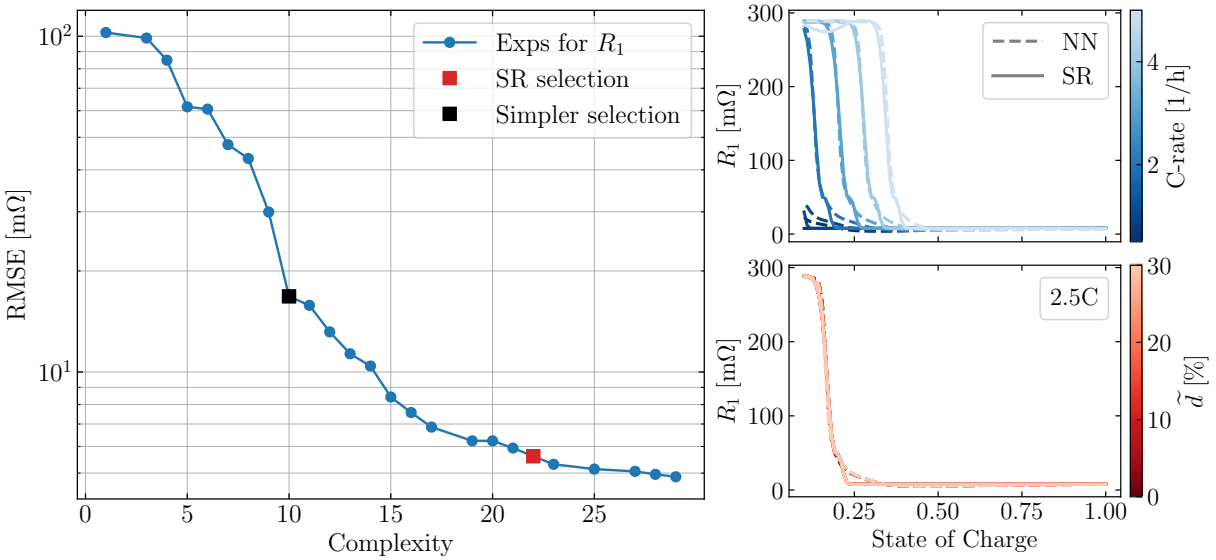


FIG C.2: RMSE loss versus complexity for R_1 symbolic expressions. The chosen expression by SR and an additional simpler expression are marked in red and green respectively.

Tab C.2: List of all symbolic expressions for R_1

Comp.	Loss [mΩ]	Expression
1	102.573	5.68
3	98.746	$10.9C$
4	84.843	$(2.36 - soc)^{3}$
5	61.627	$(1.55 - soc)^{7.98}$
6	60.667	$82.8e^{-28.00609^{soc}}$
7	47.546	$\left(C + (1.39 - soc)^{3}\right)^{3}$
8	43.240	$\left(1.49 - soc^{2.0C}\right)^{9}$
9	29.960	$34.5e^{-(8.005652^{soc} - C)^{3}}$
10	16.799	$29.1e^{-(7.810039^{soc} - C)^{12.346904}}$
11	15.795	$e^{3.36847661791267e^{-(7.0995107^{soc} - C)^{19.581877}}}$
12	13.167	$29.2e^{-295130.938570492soc^{3}soc^{9C}}$
13	11.333	$e^{3.37910675195931e^{-102724.364140792soc^{3}soc^{9C}}}$
14	10.418	$e^{3.36847661791267e^{-7836539.96824415soc^{4}soc^{12C}}}$
15	8.422	$e^{3.3673443264226e^{-soc^{3}\left(34.627403soc^{2C} - 2.3802423\right)^{3}}}$
16	7.572	$e^{3.36560890776492e^{-soc^{4}\left(38.309746soc^{2C} - 2.5478015\right)^{4}}}$
17	6.854	$e^{e^{\left(-soc^{4}\left(29.301643soc^{2C} - 1.9828179\right)^{4} + 1.0667994\right)^{3}}}$
19	6.237	$e^{e^{\left(-soc^{4.5130028}\left(41.422971300624soc^{2C} - 3.2068205\right)^{4} + 1.0667049\right)^{3}}}$
20	6.237	$e^{e^{\left(-soc^{4.5130028}\left(41.422971300624soc^{2C} - 3.2068205\right)^{4} + 1.0667049\right)^{3}}}$
21	5.936	$e^{e^{\left(-soc^{4.5138984}\left(41.44322soc^{2C} - 3.218325\right)^{4} + 1.0673679\right)^{3}}} - 0.212$
22	5.613	$\left(-1.61 + 4.49e^{-soc^{4.4675528}\left(51.3012soc^{2C} - 3.9479158\right)^{4}}\right)^{3} + 4.91$
23	5.311	$\left(e^{1.70039061806463e^{-soc^{4.477104}\left(43.531536soc^{2C} - 3.371759\right)^{4}}} - 2.58\right)^{3} + 4.7$
25	5.138	$\left(e^{1.70532310980796e^{-\left(soc^{2.2722908}\left(48.658314soc^{2C} - 3.854382\right)^{2} - 0.030778294\right)^{2}}} - 2.63\right)^{3} + 5.08$
27	5.059	$-soc + \left(e^{1.69608626521479e^{-\left(soc^{2.26109}\left(48.660423soc^{2C} - 3.843869\right)^{2} - 0.035951883\right)^{2}}} - 2.58\right)^{3} + 5.28$
28	4.957	$-soc^{0.5} + \left(e^{1.69608626521479e^{-\left(soc^{2.26109}\left(48.658882soc^{2C} - 3.843869\right)^{2} - 0.035951883\right)^{2}}} - 2.58\right)^{3} + 5.44$
29	4.876	$-soc^{C} + \left(e^{1.69174569258972e^{-\left(soc^{2.2781082}\left(48.660423soc^{2C} - 3.8312032\right)^{2} - 0.032211535\right)^{2}}} - 2.55\right)^{3} + 5.33$

C.3 Expressions for C_1

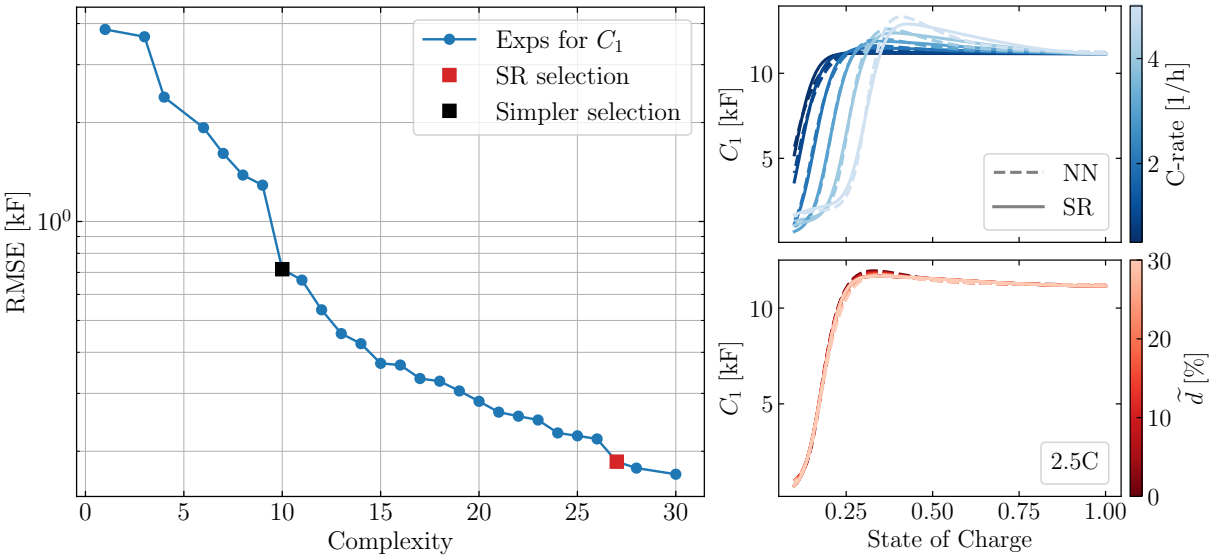


FIG C.3: RMSE loss versus complexity for C_1 symbolic expressions. The chosen expression by SR and an additional simpler expression are marked in red and green respectively.

Tab C.3: List of all symbolic expressions for C_1

Comp.	Loss [kF]	Expression
1	3.839	9.45
3	3.647	$soc + 8.95$
4	2.389	$14.0soc^{0.5}$
6	1.929	$soc\,(3.29 - soc)^3$
7	1.611	$(soc - 1.35)^9 + 11.5$
8	1.384	$(1.31 - e^{-13.857677soc})^9$
9	1.290	$\left(2.27 - e^{C-8.129967soc}\right)^3$
10	0.716	$\left(2.25 - e^{(C-7.5335298soc)^3}\right)^3$
11	0.663	$e^{\left(1.3476671-e^{3C-19.575846soc}\right)^3}$
12	0.539	$e^{\left(1.3460115-e^{6C-28.1116302soc}\right)^3}$
13	0.456	$11.5e^{-3e^{(C-5.5889444soc-0.6183074)^3}}$
14	0.425	$11.4e^{-3e^{2(C-5.1820164soc-0.45555556)^3}}$
15	0.370	$C + \left(1.3 - e^{(C-4.8343606soc-0.94116664)^3}\right)^9$
16	0.366	$C + 11.0e^{-8.391672e^{(C-4.8474994soc-0.96651226)^3}}$
17	0.333	$C - soc + \left(e^{(C-5.002297soc-0.78875285)^3} - 1.5\right)^6$
18	0.327	$C - soc + 11.6e^{-5.9967823e^{(C-4.9323034soc-0.87454927)^3}}$
19	0.305	$C\,(1.4 - soc) + \left(1.31 - e^{(C-4.81508soc-0.93363065)^3}\right)^9$
20	0.284	$C\,(2.56 - e^{soc}) + \left(1.31 - e^{(C-4.774492soc-0.93593603)^3}\right)^9$
21	0.263	$C\,(1.13 - soc^3)^3 + \left(1.31 - e^{(C-4.729318soc-0.94193393)^3}\right)^9$
22	0.256	$\left(1.31 - e^{(C-4.7014947soc-0.9478456)^3}\right)^9 + (C^{0.5} - soc^3 + 0.157)^3$
23	0.249	$6.41C\,(soc^{0.5} - soc) + 11.0e^{-10.197502e^{(C-4.6292124soc-1.0248253)^3}}$
24	0.227	$C\,(5.42soc^{0.5} - 5.42soc)^2 + 11.1e^{-9.759867e^{(C-4.5825844soc-1.016391)^3}}$
25	0.223	$-141.0C^{1.5}\,(-soc^{0.5} + soc)^3 + 11.2e^{-9.861933e^{(C-4.4634314soc-1.0388385)^3}}$
26	0.218	$(C - 0.141)\,(5.36soc^{0.5} - 5.36soc)^3 + 11.2e^{-9.756742e^{(C-4.4659915soc-1.036344)^3}}$
27	0.186	$C^2\,(-5.97soc^{0.5} + 5.97soc)^2 + 11.2e^{-9.877789e^{\left(C^{1.2646277}-4.6134715soc-0.9607453\right)^3}}$
28	0.178	$C^2\,(5.46soc^{0.5} - 5.46soc)^3 + 11.2e^{-11.1214905e^{\left(C^{1.2714428}-4.511421soc-0.99477786\right)^3}}$
30	0.170	$C^2\,(5.32soc^{0.5} - 5.32soc)^3 + 0.174 + 11.0e^{-11.137805e^{\left(C^{1.2671312}-4.5832295soc-0.97955734\right)^3}}$

C.4 Expressions for k

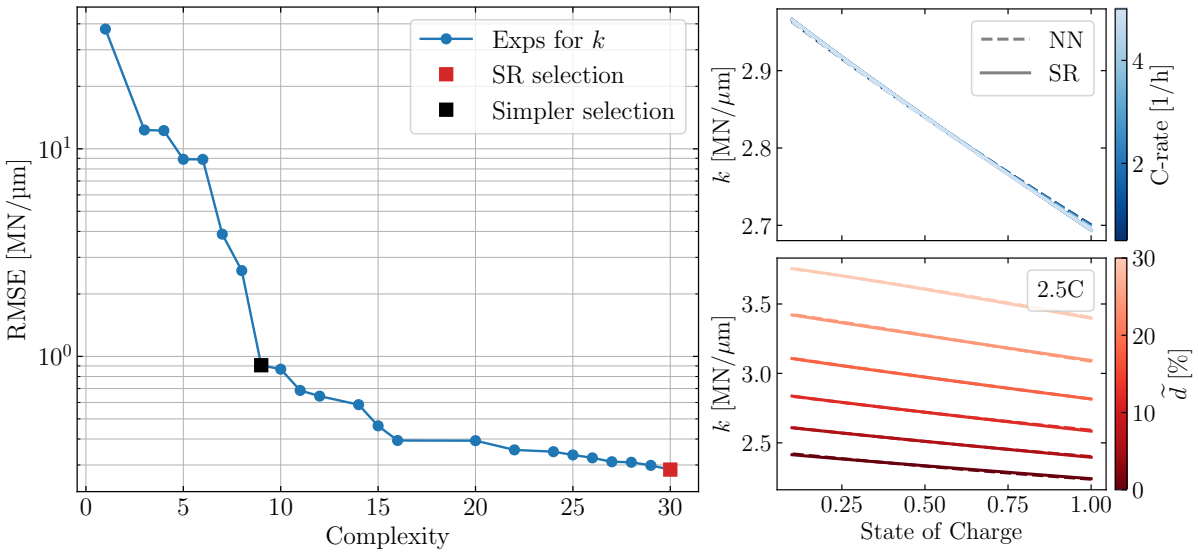


FIG C.4: Caption

TAB C.4: List of all symbolic expressions for k

Comp.	**Loss** [MN/µm]	**Expression**
1	37.740	2.89
3	12.320	$d + 2.39$
4	12.241	$d^2 + 2.55$
5	8.905	$\frac{6.59}{2.82 - d}$
6	8.895	$2.28^d + 1.33$
7	3.881	$\frac{7.28 - soc}{2.88 - d}$
8	2.590	$\frac{7.31 - soc^{0.5}}{2.84 - d}$
9	0.907	$\frac{6.58}{-d + 0.23soc + 2.7}$
10	0.868	$\frac{10.0 - soc}{(3.25 - d)^{1.2}}$
11	0.686	$-0.399 + \frac{8.49}{-d + 0.233soc + 3.0}$
12	0.644	$0.416 + \frac{37.2}{(-d + 0.233soc + 4.29)^2}$
14	0.586	$d - \left(-0.148d^2 + 0.148soc\right) e^{\sqrt{d}} + 2.42$
15	0.463	$0.497d + 0.497\left(-d - 0.789\right)\left(-d + 0.469soc\right) + 2.43$
16	0.394	$0.211\left(-3.41d^{0.855} + soc\right)\left(-d - 0.93\right) + 2.43$
20	0.393	$0.371d + 0.21\left(-2.46d + soc^{0.974}\right)\left(-d - 0.943\right) + 2.44$
22	0.355	$0.503d + 0.229\left(soc - e^d\right)\left(0.146d^6 - d - 0.836\right) + 2.24$
24	0.348	$0.527d + 0.228\left(soc - e^{0.963124560995786d}\right)\left(0.111d^9 - d - 0.83\right) + 2.24$
25	0.335	$0.562d + 0.229\left(soc - e^{0.938291654261669d}\right)\left(-d + 0.76\left(d - 0.504\right)^3 - 0.814\right) + 2.23$
26	0.325	$0.52d + 0.228\left(soc - e^d\right)\left(-d + \left(d - 0.477d^{soc}\right)^3 - 0.817\right) + 2.24$
27	0.311	$0.515d + \left(0.228soc - 0.228e^d\right)\left(-d + \left(d - 0.478\left(d^{soc}\right)^{0.5}\right)^3 - 0.821\right) + 2.24$
28	0.309	$0.559d + 0.228\left(soc - e^{0.944962189227922d}\right)\left(-d + soc^{soc}\left(d - 0.5\right)^3 - 0.817\right) + 2.23$
29	0.299	$0.523d + 0.228\left(soc - e^d\right)\left(-d + \left(d - 0.475\left(d^{soc}\right)^{soc}\right)^3 - 0.807\right) + 2.24$
30	0.285	$0.505d + 0.227\left(soc^{0.935} - e^d\right)\left(0.191d^6 soc^{soc} - d - 0.86\right) + 2.24$

C.5 Expressions for $\dot{s}$

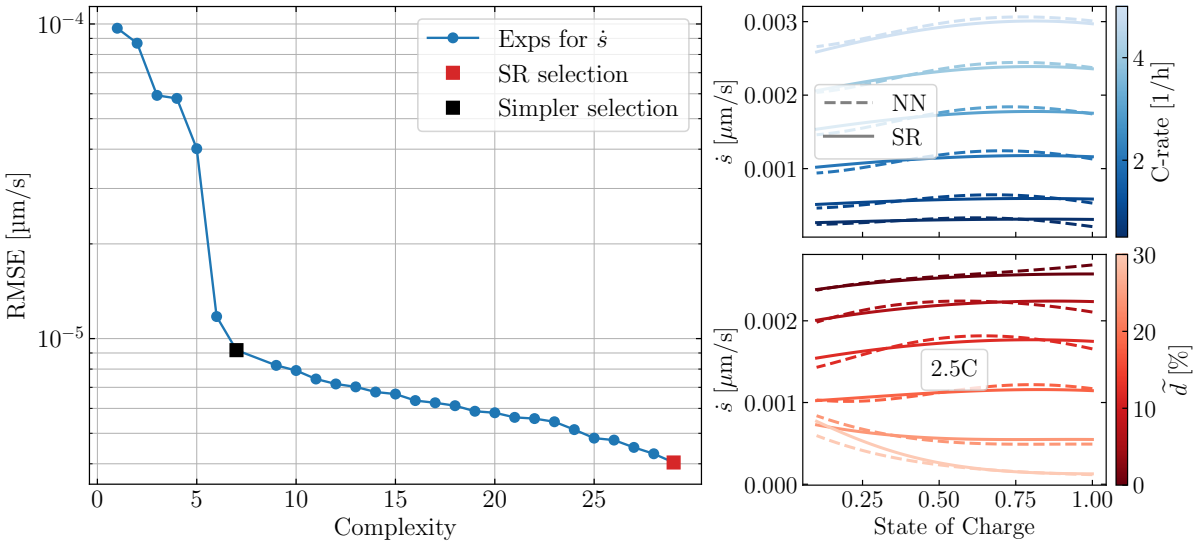


Fig C.5: Caption

Tab C.5: List of all symbolic expressions for C_1

Comp.	Loss [nm/s]	Expression
1	0.09693	C
2	0.08686	e^{C}
3	0.05930	$3.01C$
4	0.05800	$-d + e^{C}$
5	0.04014	e^{2C-2d}
6	0.01175	$\frac{C}{d^3+0.212}$
7	0.00919	$4.9C\,(1.06 - d)$
9	0.00822	$C\,(1.04 - d)\,(5.44 - s)$
10	0.00792	$C\,(1.04 - d)\,(5.22 - s^2)$
11	0.00744	$C\,(5.67 - s)\,e^{(-d-0.3635292)^3}$
12	0.00717	$C\left(-s^2 + 5.49e^{(-d-0.35647807)^3}\right)$
13	0.00702	$5.29Ce^{(-d-0.3348947)^3 e^{s^2}}$
14	0.00676	$5.31Ce^{(-d-0.34372202)^3 \left(e^{s^3}\right)^{\frac{3}{2}}}$
15	0.00666	$(5.19C + 0.0497)\,e^{(-d-0.3347837)^3 e^{s^2}}$
16	0.00635	$(5.21C + 0.0518)\,e^{(-d-0.34372202)^3 \left(e^{s^3}\right)^{\frac{3}{2}}}$
17	0.00625	$(5.19C + 0.0538)\,e^{(-d-0.32842198)^3 e^{3(0.20963338-s)^2}}$
18	0.00612	$(5.21C + 0.0518)\,e^{(-d-0.33902198)^3 e^{2(s-0.16720475)^2}}$
19	0.00588	$(5.27C^{1.05} + 0.0948)\,e^{(-d-0.3427797)^3 \left(e^{s^3}\right)^{\frac{3}{2}}}$
20	0.00581	$(5.24C^{1.05} + 0.0961)\,e^{(-d-0.32747397)^3 e^{3(0.21261285-s)^2}}$
21	0.00562	$(5.26C^{1.05} + 0.0945)\,e^{(-d-0.3367837)^3 e^{2(s-0.17023575)^2}}$
22	0.00557	$(5.24C^{1.06} + 0.0946)\,e^{(-d-0.33246765)^3 e^{3\log(s+0.8339028)^2}}$
23	0.00544	$(5.59C + 0.0548)\,e^{(-d-0.3638188)^3 e^{s^2-\left(d-s^2-soc\right)^3}}$
24	0.00514	$(6.02C + 0.0587)\left(e^{(-d-0.3351321)^3 e^{s^2-(d-s-soc)^3}} - 0.0429\right)$
25	0.00483	$\left((5.17C + 0.141)\,e^{(-d-0.342504)^3 e^{s^2-(d-s-soc)^3}}\right)^{1.08}$
26	0.00476	$(5.81C + 0.0553)\left(e^{(-d-0.3263231)^3 e^{s^2-(d-s-soc+0.098788485)^3}} - 0.0525\right)$
27	0.00451	$\left((5.16C + 0.119)\,e^{(-d-0.3461157)^3 e^{s^2-(d-s-soc+0.07556521)^3}}\right)^{1.06}$
28	0.00430	$(5.74C^{1.06} + 0.104)\left(e^{(-d-0.31522557)^3 e^{s^2-\left(d-s^2-soc\right)^3}} - 0.0573\right)$
29	0.00404	$(5.88C^{1.06} + 0.105)\left(e^{(-d-0.3230624)^3 e^{s^2-(d-s-soc+0.09943366)^3}} - 0.0555\right)$

C.6 GP Model for SOC to OCV

```python
def GP_process():
'''Returns a GP fit for SOC to U_eq'''

    sig = 1
    l = 0.1
    alpha = [l, sig]
    data = get_data('GP_run')

    x_gp = data['soc'].to_numpy().copy()
    y_gp = data['Ue'].to_numpy().copy()

```

```
12      x_gp[x_gp<0] = 0
13      kernel_GP = gp.kernels.RBF(length_scale=alpha[0]) *
        ↪  gp.kernels.ConstantKernel(constant_value=alpha[1])
14      gp_model = gp.GaussianProcessRegressor(kernel=kernel_GP, optimizer=None,
        ↪  normalize_y=False)
15      gp_model.fit(x_gp.reshape(-1,1), y_gp.reshape(-1,1))
16      return gp_model
17
18
19
20  def soc_to_Ue(soc, gp_model):
21      soc = np.asarray(soc)
22      soc[soc<0] = 0; soc[soc>1] = 1
23      return gp_model.predict(soc.reshape(-1,1))
```

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY