
Universidad Nacional de
Educación a Distancia

Escuela Superior de
Ingeniería Informática


Universidad
Complutense de Madrid

Facultad de
Informática

# Modeling and evaluation of a battery balancing library in Modelica

Iván Arenas Kelbelova

Director: Alfonso Urquía Moraleda
Co-director: José Manuel Díaz Martínez

Trabajo de Fin de Master

Máster Universitario en Ingeniería de Sistemas y Control

Curso 2024/2025, convocatoria ordinaria

**Universidad Nacional de Educación a Distancia (UNED)**

**MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL**

# *Modeling and evaluation of a battery balancing library in Modelica.*

**Autor:** Iván Arenas Kelbelova
**Director:** Alfonso Urquía Moraleda
**Co-Director:** José Manuel Díaz Martínez

**Proyecto de tipo B:** Proyecto específico propuesto por el alumno

# Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Iván Arenas Kelbelova

Iván Arenas

Firma del alumno

# Abstract

The increased availability of second-life lithium-ion cells has created new opportunities for cost-effective stationary energy storage. However, these applications require careful management of cell-to-cell variations and aging effects, particularly in balancing and control systems. This thesis introduces a Modelica-based simulation library, named `Battery_Balancing Library`, designed to support the development, integration, and evaluation of battery management system (BMS) functionalities tailored to second-life scenarios.

The library enables modular implementation of aging cell models, simplified charging/discharging logic, and configurable balancing strategies. Its architecture prioritizes reusability, transparency, and ease of extension, making it suitable for both academic research and early-stage control prototyping. A four-cell case study is used to demonstrate the framework's capabilities, where active and passive balancing modules are applied under realistic degradation profiles and operational constraints.

The results confirm that the library can reproduce key behaviors associated with balancing under second-life conditions and serve as a reliable platform for algorithm validation.

**Keywords**: Second-life batteries, Battery simulation, Modelica, BMS development, Cell balancing, Aging models

# Resumen

La creciente disponibilidad de celdas de ion-litio de segunda vida ha abierto nuevas oportunidades para soluciones de almacenamiento estacionario de energía más rentables. No obstante, estas aplicaciones requieren una gestión cuidadosa de las variaciones entre celdas y de los efectos del envejecimiento, especialmente en lo que respecta a los sistemas de equilibrado y control. Esta tesis presenta una biblioteca de simulación basada en Modelica, denominada `Battery_Balancing Library`, diseñada para facilitar el desarrollo, integración y evaluación de funcionalidades del sistema de gestión de baterías (BMS) adaptadas a escenarios de segunda vida.

La biblioteca permite implementar de forma modular modelos de celdas envejecidas, lógicas simplificadas de carga/descarga y estrategias de equilibrado configurables. Su arquitectura está orientada a la reutilización, la transparencia y la facilidad de extensión, lo que la hace adecuada tanto para la investigación académica como para la prototipación temprana de algoritmos de control. Se utiliza un estudio de caso con cuatro celdas para demostrar la capacidad del entorno, aplicando módulos de equilibrado pasivo y activo bajo perfiles de degradación realistas y restricciones operativas.

Los resultados confirman que la biblioteca puede reproducir los comportamientos clave asociados al equilibrado en condiciones de segunda vida y servir como una plataforma fiable para la validación de algoritmos.

**Palabras clave**: Baterías de segunda vida, Simulación de baterías, Modelica, Desarrollo de BMS, Equilibrado de celdas, Modelos de envejecimiento

# Acknowledgments

To mom and dad.

# Glossary

**BMS** Battery Management System; the electronic system that manages battery safety, performance, and balancing.

**C-Rate** Charge/discharge rate relative to a battery's nominal capacity; for example, 1C means charging or discharging in one hour.

**C1** Capacitance in ECMs; paired with R1 to form an RC branch representing transient response.

**CC** Constant Current; a charging/discharging mode where the current is held constant.

**CCCV** Constant Current–Constant Voltage; a common charging strategy that applies a fixed current until a voltage limit is reached, followed by a constant voltage phase.

**CtCVs** Cell-to-Cell Variations; differences in capacity, resistance, or performance characteristics among individual cells in a pack.

**CV** Constant Voltage; a charging mode where the voltage is held constant while current decreases.

**DC** Direct Current; the unidirectional flow of electric charge used in battery systems.

**ECM** Equivalent Circuit Model; a simplified electrical representation of a battery used for modeling and simulation.

**EoC** End of Charge; the point at which the battery voltage reaches its maximum allowable value or the current drops below a cutoff threshold.

**EoD** End of Discharge; the point at which the battery voltage reaches its minimum allowable value.

**EVs** Electric Vehicles; vehicles powered by electric motors using energy stored in batteries.

**LFP** Lithium Iron Phosphate; a lithium-ion battery chemistry known for thermal stability and long cycle life.

**NMC** Nickel Manganese Cobalt; a common lithium-ion battery chemistry with high energy density.

**OCV** Open Circuit Voltage; the voltage of a battery cell when it is at rest and no current is flowing.

**R0** Ohmic resistance in ECMs; models the instantaneous voltage drop due to internal resistance.

**R1** Polarization resistance in ECMs; part of the RC network modeling the dynamic behavior of the battery.

**SEI** Solid Electrolyte Interphase; a passivation layer that forms on the anode during battery operation, crucial for performance and safety.

**SL** Second-Life; refers to battery applications after their first use cycle, typically involving repurposing EV batteries.

**SOC** State of Charge; represents the current charge level of a battery relative to its maximum capacity.

**SoH** State of Health; indicates the overall condition of a battery, typically as a percentage of its original capacity or performance.

**SoP** State of Power; represents the ability of a battery to deliver power under given operating conditions.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction, Goals and Structure

This document serves as the research report for the project **Balancing topologies for a Second-Life battery cell pack**. The study begins with a comprehensive literature review on LiIon batteries, aging, second-life battery applications, challenges, and existing balancing methodologies. Following this, a battery cell model is developed in Modelica, and base parameters and parameters after the effect of aging are derived. Two passive and two active balancing modules are implemented. These balancing models are tested under a given 4-cell aged cell pack. The final step involves validating the simulation results by running and analyzing the models designed, ensuring consistency between circuit simulation and the physical behavior expected.

## 1.1. Motivation

Energy has been a driving force behind societal progress, shaping the way civilizations grow and operate. Historically, the focus has been on increasing energy production while optimizing its distribution and consumption. With the rise of renewable energy sources, energy storage has become a critical element in ensuring stability, efficiency, and sustainability in modern power grids.

Batteries, particularly lithium-ion (Li-ion) batteries, have emerged as a key technology in the energy landscape due to their high energy density, efficiency, and scalability. They have been widely adopted in electric vehicles (EVs), consumer electronics, and grid-scale storage systems. Among these, electric vehicle batteries present a unique opportunity for applications in second life once they degrade below the performance requirements of the automotive industry (Astudillo et al., 2025).

The end-of-life of EV batteries does not necessarily mean the end of their usefulness. Many batteries retain up to 70–90% of their original capacity even after being removed from vehicles, making them suitable for second-life applications. These applications include:

- Stationary energy storage for renewable integration.

- Microgrid and backup power systems.
- Grid frequency regulation.

However, second-life batteries come with significant challenges. Aging-induced degradation, leading to capacity mismatch among cells; increased internal resistance, causing imbalance during charge/discharge cycles; and safety concerns due to uneven heat generation and overcharging risks. To address these challenges, Battery Management Systems (BMS) incorporate cell-balancing techniques to equalize charge levels and extend the lifespan of repurposed battery cells.

Already from the production line, deviations between cell characteristics appear (Beck et al., 2021). Due to many factors (see more Section 2.3.5), these inhomogeneities between cells are increased even further during their lifetime. This leads to less efficient use of the cells and a reduction of their total lifetime. Thus, cell balancing is crucial during the cell's lifetime, especially during the second life of the cell, where the characteristics are already diminished.

Balancing can be achieved through passive, active, or hybrid methods. Although active balancing techniques use energy transfer between cells to reduce losses, they often require complex power electronics, added costs, and efficiency trade-offs. However, passive balancing is a simpler and cost-effective alternative that dissipates excess energy as heat through resistors. With the drawback of energy loss under operation and the bigger need for a cooling system to dissipate this heat, (Khan et al., 2024). Finding the correct solution is necessary for both economic and operational reasons.

## 1.2. Proposition and Goals

The proposition of this project to address the situation above is to develop and validate a Modelica-based library for balancing systems that allows for modular and adaptable testing of different balancing methods. The developed models will be organized within a library named `Battery_Balancing`, which is designed following the principles of object-oriented modeling. This library will constitute the principal outcome of the thesis work. This includes:

- Modeling second-life battery cells using an equivalent circuit model.
- Implementing balancing modules for the different kinds of balancing methods selected.
- Creating and adapting a control algorithm for each of the cases.
- Evaluating and validating models.

The main objective is the development of a Modelica library that allows simulations of second-life lithium-ion batteries and their balancing. This main objective includes a set of smaller milestones to achieve:

- Conduct a literature review on Li-ion batteries, second-life battery applications and challenges, existing balancing methods (passive, active, and hybrid), and Modelica-based approaches to battery modeling.
- Develop battery models based on the electrical equivalent model, which captures the electrical behavior of battery cells. This includes incorporating state-of-charge (SOC), state-of-health (SOH), and aging effects to accurately represent the characteristics of second-life cells.
- Implement and simulate different balancing strategies on a 4-cell series-connected pack:
  - Shunt resistor balancing.
  - Switched resistor balancing.
  - Single inductor balancing.
  - Single capacitor balancing.
- Analyze the behavior of the models and validate the performance and modularity of the developed library.

## 1.3. Document Structure

The document is structured as follows:

- **Chapter 1: Introduction**
  - The project goals and motivation are defined and the general outline of the document is presented.
- **Chapter 2: Literature Review**
  - Introduction of basic concepts of batteries and battery management systems.
  - In-depth review of second-life batteries and balancing strategies
  - Key battery modeling approaches, including ECM and electrochemical models.
- **Chapter 3: Modeling Hypothesis**
  - Modeling strategies followed for the main components.
  - Assumptions and simplifications used in the Modelica simulations.
- **Chapter 4: Implementation Details**
  - Modelica library development and particularities surrounding its implementation.

- **Chapter 5: Library Architecture**
  - The structure and organization of the Modelica library created for this project.
- **Chapter 6: Results and discussion**
  - Validation of Modelica models based on examples where models are applied.
  - Discussion and analysis of simulation results.
  - Discussion of the methods used and the validity of the results.
- **Chapter 7: Conclusions and Future Work**
  - Key findings, possible improvements and future work.
- **Appendix A: Modelica Library Code**
  - Complete library code for Battery_Balancing Modelica library.
- **Appendix B: Library User's Documentation**
  - User's documentation for the Battery_Balancing Library.

# Chapter 2

# Literature Review

## 2.1. Introduction

In this chapter, a general theoretical background is presented on the technologies involved in the project. First, the modeling language Modelica is introduced, followed by an overview of battery technologies, with special attention to lithium-ion cells. Their operation, aging behavior, and the role of the Battery Management System (BMS) are also discussed.

The chapter continues with a review of lithium-ion battery modeling approaches, focusing on equivalent circuit models (ECMs) and outlines their typical components, parameterization, and how aging and cell imbalance can be included. Finally, battery balancing strategies are explored, comparing passive and active topologies, and introducing the control strategies relevant to this work.

## 2.2. Modelica

Modelica is a free-to-use, object-oriented modeling language that is focused on the modeling of complex physical systems. It allows the modeling and simulation of systems in multiple domains, including electrical, mechanical, thermal, hydraulic, control, chemical process, electronic, or electric power; allowing the interconnection between them. Components are modeled by their mathematical behavior and can be grouped forming libraries that can be used by other users to model more complex systems. This is one of the main advantages of the creation of libraries with Modelica; Once a model is created, it can be reused and modified by the end-user without the need to know the details of the mathematical modeling behind it. That, together with the interconnection between different domains, allows the end to be used to develop complex models in a fast way, (Urquía and Martín-Villalba, 2018), (Tiller), (Modelon).

OpenModelica and Dymola are simulation environments based on the Modelica language. They serve as an interface for modeling, compilation, and simulation of models in the Mod-

elica language.

## 2.3. Batteries

An electrochemical device is a device that can transform chemical energy stored in components inside the cell directly into electrical energy.

The history of batteries has its start with the discovery of electrostatic effects, but it was in the 1800s when Alessandro Volta invented the first electrochemical battery. During the 19th century batteries were the main source of electrical power to general grids, but once the large-scale AC grids were implemented, batteries were relegated to back up electrical power systems and a few portable applications. It wasn't until the late twentieth century, with the appearance of mass-produced portable devices, that interest in batteries arose again. With that, a rapid development of new battery technologies was born, developing between them: metal hydride batteries, nickel-cadmium batteries, lithium ion batteries and sodium ion batteries, (Wulandari et al., 2023).

### 2.3.1. Lithium-Ion Batteries

Figure 2.1 gives a visual representation of the relationship between specific power and specific energy for the different technologies. Representation that is useful for illustrating one of the main advantages of Li-Ion technologies, which makes them a preferred option in many industry applications. They offer the best relationship-specific energy/power, leading to smaller and less weight solutions, which in many fields, such as in mobility applications, as a noticeable impact.

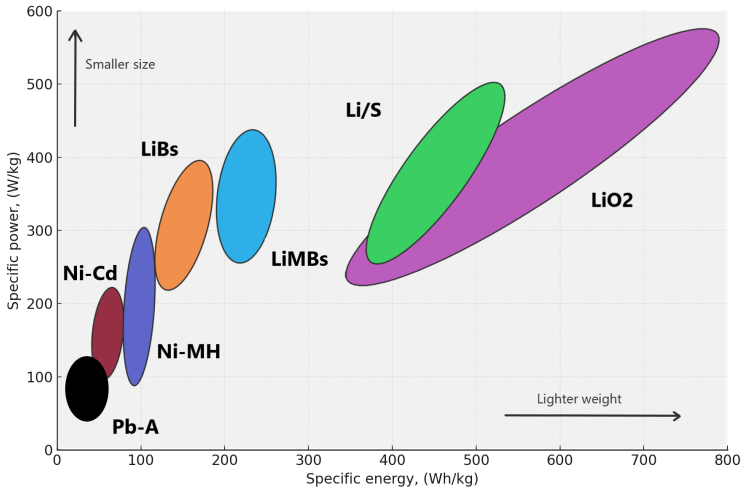



Figure 2.1: Energy density vs Power density for different types of battery technologies. Image inspired by (Olabi et al., 2023).

For more references on the differences between battery technologies, please refer to technical data available online, such as (Epec, 2023a). In general, these are the main advantages for the use of Li-ion batteries (Lu, 2016):

- **High specific energy.** Li-ion batteries present higher specific energy, ranging from 100–300 Wh/kg in most industry models (Waseem et al., 2023). For comparison, the second-best are NiMH batteries, which present performances slightly over 100 Wh/kg. High specific energy leads to smaller batteries for the same energy stored, which is of high value in most industries, including EVs.

- **High voltage.** One of the main characteristics achieved after Goodenough's research (see Section 2.3.1). Li-ion cells present noticeably higher voltage than competitors, reaching 4 V in modern applications (Manthiram, 2020). This means that fewer cells in parallel are needed to reach the required application voltage, making the BMS implementation easier for this type of cell.

- **Low self-discharge rate.** Modern Li-ion cells present a self-discharge rate even lower, between 1.5%–2% per month (Lu, 2016), considerably outperforming the alternatives.

- **Fast charging time.** The structural stability and high conductivity in lithium oxide cathodes allow for noticeably faster charge times than any other technology. This is further explained in Section 2.3.1. It is a highly important characteristic in most portable applications, especially in EVs, where one of the main limitations is the typically slow charging time.

- **Low maintenance.** No scheduled cycling is required to prevent battery life degradation. NiCd and NiMH solutions require periodic discharge to reset the memory effect, while some lead-acid batteries need to be topped up (Lu, 2016).

These are the main characteristics that explain why the Li-ion battery market has grown to be the largest and is expected to continue along that path; see (Waseem et al., 2023). Since they are the market leaders, the focus of the project will be based on this technology, although the work performed in this thesis is easily applicable to other battery technologies.

Within lithium-ion technologies, there are also different types based on the type of cathode material used. The cathode material in today's market is what limits the energy density of the cell and constitutes the largest portion of the cost. Regarding the composition of the anode, graphite appears to be the most appropriate candidate; at the moment, it is used in the majority of Li-ion industrial solutions. For more information on the topic please refer to (Epec, 2023b).

### 2.3.2. Cell behavior

Since one of the bases of the project is obtaining a good model for a Li-ion cell, it is important to present the main behaviors of this kind of cell. Therefore, some of the common responses of cells are presented here. The behavior of a cell can be separated into two parts: static behavior and dynamic behavior.

Static behavior refers to how the cell behaves when it is at rest, which means that no charge is applied and internal variables are in equilibrium. The main parameter that shows the state of a cell under static conditions is the OCV. The OCV is mainly dependent on two factors under normal operation. Figure 2.2 shows graphically how the OCV relates SOC.

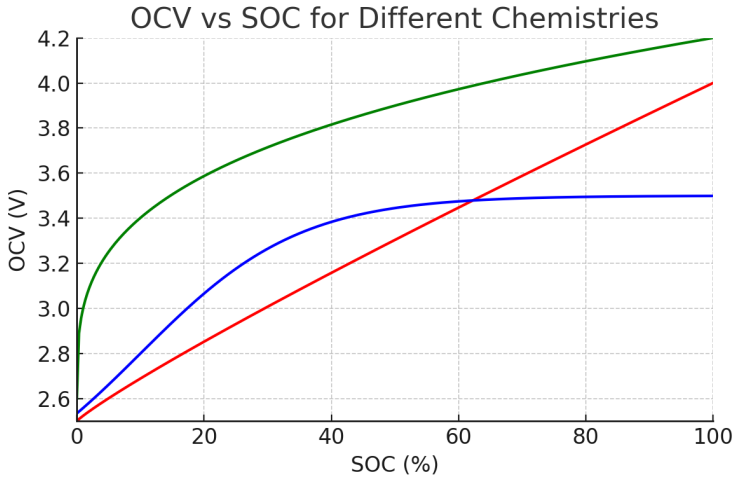



Figure 2.2: OCV depending on SOC of the cell for three different Li-Ion technologies. Image inspired by (Plett, 2015).

As a norm, when cells are fully charged, the OCV is at its maximum value, and it goes down as the cell discharges, reaching its minimum value when the cell is completely discharged. This is shown in Figure 2.2 for three different Li-ion cells. Although the relationship between SOC and OCV differs for different types of technologies, the higher the SOC, the higher the OCV of the cell. Another relevant dependency is the relationship between OCV and temperature. This behavior varies among different types of LIB technologies and should be modeled for each particular application, (Zhang and Xia, 2018).

However, dynamic behavior affects how the cell reacts under charge or discharge, or in the instant after it. These dynamic behaviors explain the difference between the OCV of a cell at a particular SOC and the voltage of the cell during charge or discharge. The influence of charge and discharge rates on cell voltage can be seen in Figures 2.3 and 2.4.

Figure 2.3 shows that the higher the charge rate, the higher the voltage at the cell terminals. For the cell to be charged, the terminal voltage always needs to be higher than the OCV. On the other hand, Figure 2.4 shows that the higher the discharge rate, the lower the voltage of the cell. These differences between charge/discharge voltages and OCV can be

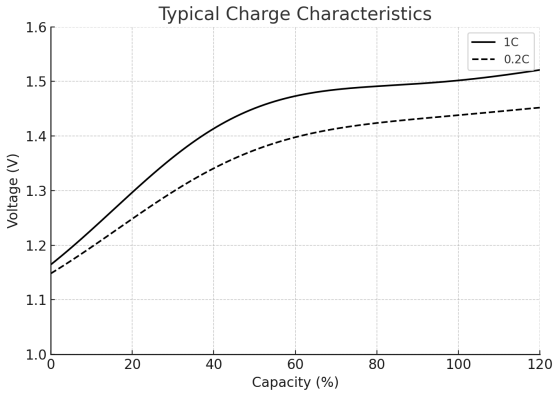



Figure 2.3: Charge curves for two different charge rates, for a cell of NiMH. Image inspired by (Pistoia, 2005)

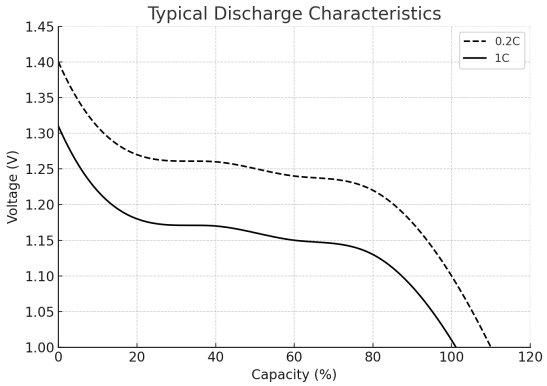



Figure 2.4: Discharge curves for two different discharge rates, for a cell of NiMH. Image inspired by (Pistoia, 2005).

explained based on two phenomena: ohmic losses and polarization of the cell. The voltage for charging and discharging a cell can be expressed as follows (Pistoia, 2005):

$$V_d = \text{OCV} - \eta_+ - \eta_- - I \cdot R \tag{2.1}$$

$$V_c = \text{OCV} + \eta_+ + \eta_- + I \cdot R \tag{2.2}$$

This ohmic behavior is due to the resistance encountered in the different media during the transport of electrons through the electrodes and current collectors. The $\eta_+$ and $\eta_-$ are the polarization on each electrode. And each polarization is composed of two main components: activation polarization, which is an intrinsic property of the electrodes, and concentration polarization, which is related to the accumulation or depletion of ions near the electrodes (Pistoia, 2005). On this topic, see Figure 2.5.

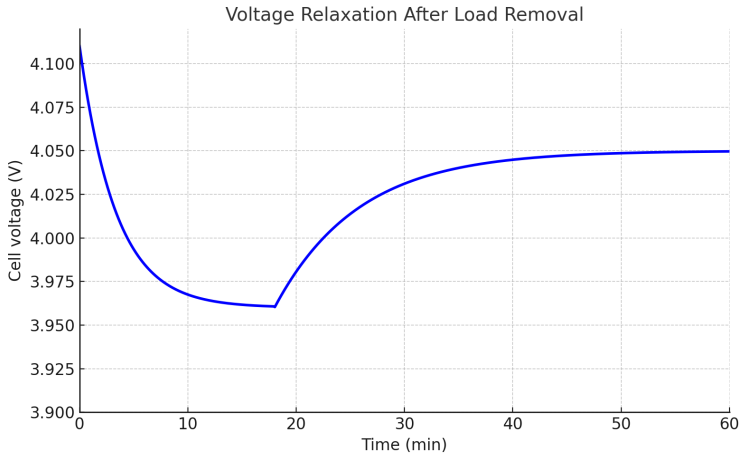



Figure 2.5: Discharge pulse test on Li-Ion cell - Discharge period 15 minutes. Image inspired by (Plett, 2015).

Figure 2.5 shows the voltage response to a pulse test on a Li-ion cell. The test starts with the cell in rest mode, then a load is connected for a total of 15 minutes and subsequently disconnected, making the cell return to a rest state (Plett, 2015). This experiment clearly shows the main behaviors during the discharge of a cell. The voltages at times 0 min and 60 min represent the OCV before and after the discharge, once the cell has stabilized. During discharge, two behaviors can be observed: first, an instant voltage drop that corresponds to the ohmic behavior of the cell (IR); second, the polarization of the cell, which takes place between minutes 5–10. From 10 to 20 minutes, the voltage continues to decrease due to the reduction in SOC.

The same can be observed when the load is disconnected. First, an immediate recovery of the cell voltage occurs at 20 minutes. Afterward, the voltage gradually returns to the OCV as the polarization energy dissipates—this phenomenon is known as diffusion voltage.

There is one last dynamic behavior of Li-ion cells: hysteresis. As explained with the discharge pulse test, after charging a cell, the voltage undergoes a transition period before stabilizing at the new OCV. However, real cells exhibit hysteresis both after charge and discharge. To explain it simply: when we charge a cell to a certain amount, say $X\%$ of SOC, and let it return to a stationary state, the cell will present a voltage higher than the OCV for that SOC (SOC + hysteresis). The opposite occurs after a discharge. When a cell is discharged to an $X\%$ SOC, after stabilization, the resulting voltage will be lower than the OCV at that SOC. Refer to Chapter 2 of (Plett, 2015) and (He, 2020) for a more detailed explanation of this.

Aside from the normal behaviors of Li-ion cells, other types of changes influence cell performance during its lifetime. The main causes for this change in performance are further discussed in Section 2.3.5. Some of the expected changes linked to cell aging include:

- Increased internal impedance.
- Reduced capacity of the cell.
- Increased self-discharge.

The influence of all these variables on the final operation of the cell makes accurate modeling of such systems complex. Physical models offer greater precision and can better simulate the influence of temperature or aging on the cell. ECM models are more limited, but with appropriate considerations, good performance can still be achieved; these are presented in Section 3.2.

### 2.3.3. Functions of a BMS

The Battery Management System (BMS) constitutes the supervisory unit in any lithium-ion battery-powered application, ensuring not only the safe and efficient operation of the

battery pack but also the prolongation of its working life. As lithium-ion cells are particularly sensitive to operating conditions such as temperature, voltage, and current, the BMS is tasked with the constant monitoring and control of these variables to prevent unwanted events such as thermal runaway, overcharging, and deep discharging (Liu et al., 2022).

At a basic level, a BMS is responsible for four primary functions: data acquisition, protection, state estimation, and control. Data acquisition involves continuous sensing of parameters including cell voltage, pack current, and temperature. These measurements provide the necessary basis for safety functions and diagnostics (Uzair et al., 2021). Protection circuits are employed to disconnect the battery from the load or charger under critical conditions such as overvoltage, undervoltage, overcurrent, or excessive temperatures. To give a better idea of the concept of a BMS, a basic BMS block diagram is included in Figure 2.6.

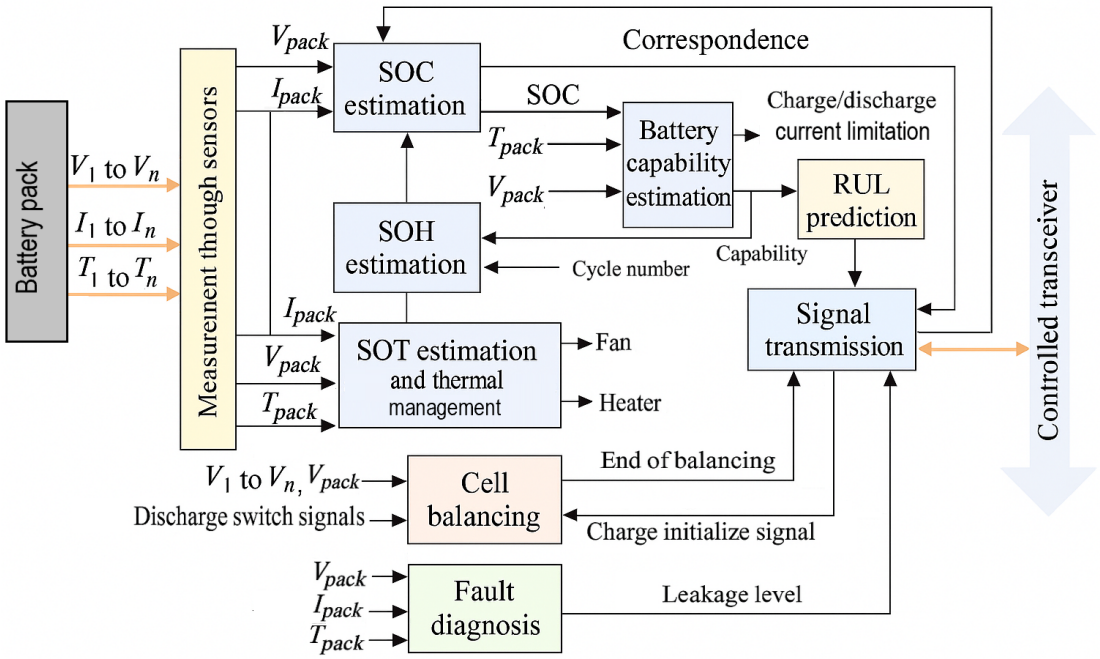



Figure 2.6: Basic BMS logic schematics, showing basic inputs and outputs as well as interactions between internal function blocks. Image inspired by (Liu et al., 2022).

Another essential task of the BMS is to estimate key battery states, notably the State of Charge (SoC), State of Health (SoH), and State of Power (SoP). These estimates are essential for optimizing energy usage, enabling predictive maintenance, and supporting system-level diagnostics. SoC estimation is typically derived from open-circuit voltage (OCV), Coulomb counting, or model-based observers, while SoH is inferred from indicators such as capacity fade and internal resistance growth over time (Uzair et al., 2021).

Cell balancing is a further critical function, particularly in multi-cell configurations. Due to manufacturing variances and aging, individual cells may exhibit different capacities or

impedance characteristics, leading to voltage divergence during charge and discharge cycles. The BMS addresses this issue using either passive or active balancing strategies to equalize cell voltages, thereby avoiding premature cutoffs and ensuring uniform utilization of cell capacity (Khan et al., 2024), (Uzair et al., 2021). Passive methods dissipate excess energy as heat through resistive elements, while active methods redistribute energy among cells using converters or switched capacitor/inductor networks. This topic will be further explored in Section 2.3.

Beyond these core functions, modern BMS architectures may incorporate thermal management controls, fault diagnostics, communication interfaces, and data logging capabilities. These enhancements are particularly relevant in second-life applications, where cells have been subject to prior degradation and may present non-uniform aging profiles. Advanced algorithms, such as those based on artificial intelligence or digital twin concepts, are increasingly being integrated into BMS frameworks to improve their adaptability and predictive accuracy in such scenarios (Pérez et al., 2024), (Reiter et al., 2023).

### Charging and Discharging Control

One of the fundamental functions of a Battery Management System is to supervise and regulate the charging and discharging processes, ensuring that all cells operate within their safe operating area. Lithium-ion cells, while offering high energy density and efficiency, are susceptible to degradation and safety risks if operated outside their range, normally between 2.5 V to 4.2 V (Samaddar et al., 2020). The BMS continuously monitors cell voltage and pack current, interrupting the process in cases of overvoltage, undervoltage, or excessive current output (Uzair et al., 2021).

Charging and discharging strategies can be categorized as follows:

- **Constant Current / Constant Voltage (CCCV):** This is the most used charging protocol for lithium-ion batteries. Initially, a constant current—typically around 0.5C to 1C—is applied until the cell voltage reaches the upper limit (e.g., 4.2 V for standard Li-ion cells). Then, a constant voltage is applied, and the current goes down exponentially until it falls below a cutoff threshold. This method balances speed and safety, reducing stress on the battery while ensuring full capacity utilization (Samaddar et al., 2020).

- **Multistage Constant Current–Constant Voltage (CC–CV):** An improvement of CCCV, this approach applies varying current stages, optimizing the balance between charging speed and thermal/electrochemical stability (Samaddar et al., 2020).

- **Pulse Charging:** In this technique, the charge is delivered in pulses rather than as a continuous current. Between pulses, the system can monitor voltage recovery, which provides insights into internal resistance and thermal behavior. While beneficial

for reducing charge time and thermal issues, the control is complex and less widely implemented in automotive applications.

- **Taper Charging:** Mostly used in cells with specific chemistry or aged battery scenarios, this method gradually reduces the current as the terminal voltage approaches the upper threshold, minimizing cell stress near full charge.

Typical current rates for both charging and discharging are expressed as C-rate. In automotive applications, discharge currents can exceed 1C during high-power acceleration, while charging currents for fast-charging applications may reach up to 2C, although thermal limitations may arise in this case.

The BMS also manages the end-of-charge (EOC) and end-of-discharge (EOD) events, halting current flow when the voltage of the most charged or discharged cell crosses the limits. These limits are essential for maintaining long-term capacity and avoiding irreversible damage caused by lithium plating during overcharge or copper dissolution during deep discharge.

In general, by enforcing charge and discharge limits, selecting optimal charging strategies, and actively monitoring cell behavior during these phases, the BMS improves and protects battery longevity, operational safety, and performance.

### 2.3.4. Second-life Applications

Lithium-ion cells are typically considered suitable for second-life (SL) use when their State of Health (SoH) drops below a threshold, often around 70–80% of their nominal capacity, and the cell is no longer viable for its original high-performance application, such as in electric vehicles. While not yet fully degraded, the cell's reduced capacity and increased internal resistance limit its power delivery and energy throughput, and power/energy density is consequently diminished. However, its residual capacity and lowered characteristics can still be utilized in less demanding applications, marking the transition into its second life (Waseem et al., 2023).

Second-life applications aim to find a new use for these cells to extend their overall lifespan, reduce environmental impact, and increase resource efficiency. Common use cases include stationary energy storage systems (ESS) for residential or commercial solar installations, grid peak shaving, backup power systems, and low-power mobility platforms such as electric bicycles or forklifts (Liu et al., 2022), (Waseem et al., 2023). These applications demand lower power and energy density, are commonly dedicated to static solutions, and are well-suited for cells with decreased performance.

From a performance perspective, second-life batteries can still offer useful life. Depending on their chemistry, operating conditions, and usage history, lithium-ion cells can undergo up to 1000–2000 full equivalent cycles or accumulate 3000–6000 operating hours before reaching end-of-first-life thresholds. Their second-life duration depends heavily on further aging

behavior, but additional usage of 3–7 years is commonly achievable when proper control is implemented (Uzair et al., 2021).

Despite their economic and environmental advantages, second-life battery systems present numerous technical challenges. Heterogeneity of cell conditions is the most concerning issue, arising from different aging rates due to temperature gradients, cycling history, and cell-to-cell manufacturing variations. This inhomogeneity complicates cell grouping, SoH estimation, and safety validation. Moreover, second-life applications often require re-parameterization of BMS algorithms, thermal management adaptations, and additional circuitry for safety (Beck et al., 2021), (Reiter et al., 2023).

Another important consideration is the difficulty of characterization and classification. Cells recovered from electric vehicles or other systems present wide variability in capacity, internal resistance, and degradation. Accurate screening—often involving impedance spectroscopy, capacity testing, or data-driven estimation methods—is essential to ensure reliability and extend the effective second-life duration.

The general degradation profile of a lithium-ion battery is depicted in Figure 2.7, which shows how capacity declines over time and defines the transition from first-life to second-life operation.

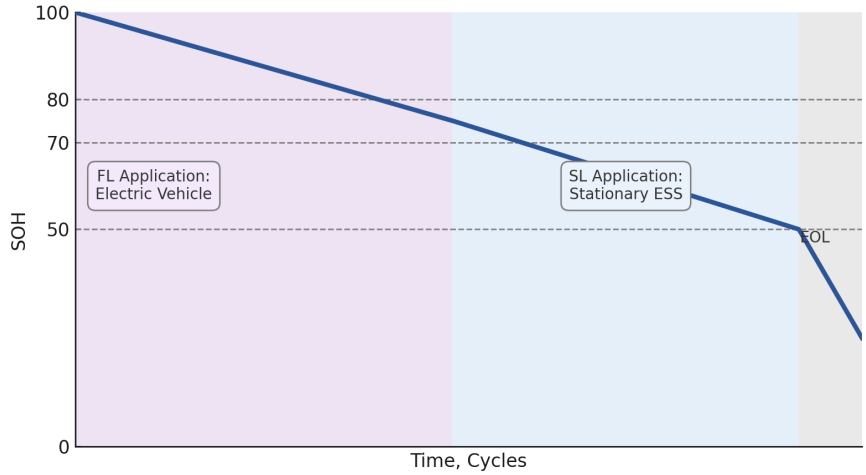



Figure 2.7: Typical lifecycle representation of a lithium-ion battery showing the transition from first-life to second-life based on SoH degradation. Image inspired by (Pérez et al., 2024).

The adoption of second-life batteries is increasing but is still limited by regulatory, economic, and technical uncertainties. Market studies indicate growing interest, especially in grid applications and renewable energy integration, but large-scale deployment has been limited and remains in the early phases (Waseem et al., 2023).

Understanding and managing second-life battery systems requires an understanding of the aging mechanisms that lead to capacity degradation and performance decline. The transition from first-life to second-life use is defined by these aging processes. The following

section explores the principal aging phenomena in relation to lithium-ion batteries, their underlying causes, and their implications for modeling and system design.

### 2.3.5. Aging

As mentioned above, aging is a key factor that determines the performance, safety, and usable life of lithium-ion batteries. Over time, batteries lose capacity and internal resistance increases, which directly influences how much energy they can store and deliver. This aging process occurs both when the battery is in use and at rest. Battery aging is typically classified into two categories:

- **Calendar aging**, which occurs when the battery is at rest, especially at high State of Charge (SoC) and elevated temperatures. It can be seen as the aging of the cell over time.
- **Cycle aging**, which results from repeated charging and discharging cycles—especially under high current rates, deep depth-of-discharge (DoD), or extreme temperatures. In general, conditions near the operational limits.

Both types of aging are caused by physical and chemical changes inside the cell. Among the most common degradation mechanisms are solid electrolyte interphase (SEI) growth, lithium plating, electrode cracking, and electrolyte decomposition. These mechanisms vary depending on battery chemistry and usage. A more detailed analysis of these degradation modes is beyond the scope of this work but can be found in dedicated literature such as (Beck et al., 2021).

The effects of aging include a gradual loss of capacity, known as capacity fade; a rise in internal resistance, known as power fade; and increased heat generation during operation. These changes reduce efficiency and available energy, and in extreme cases, can affect cell safety. Aging is influenced by several factors, including temperature, average SoC, C-rate (charge/discharge current), and depth of discharge. High temperatures and high SoC levels tend to accelerate calendar aging, while large current loads and deep cycles increase the rate of cycle aging. These effects can be mitigated by avoiding high SoC storage, limiting exposure to high temperatures, using moderate C-rates, and applying thermal management strategies.

In any battery pack, individual cells differ slightly in properties due to variations introduced during the manufacturing process. These cell-to-cell variations (CtCVs) include differences in capacity, internal resistance, and self-discharge rate. Even in new cells, the variability in capacity can range between 1–3%, and internal resistance can vary by up to 5% (Reiter et al., 2023).

As the battery ages, these differences tend to increase. Cells degrade at different rates due to localized conditions such as temperature gradients, uneven current distribution, or

minor defects. For example, a cell placed closer to a heat source may experience faster SEI growth, leading to accelerated capacity fade. This results in diverse SoC levels within the pack and makes balancing more challenging over time (Beck et al., 2021), (He and Chen, 2023).

This phenomenon is especially relevant for second-life battery packs, where cells come with a history of use under different conditions. The BMS must take these variations into account to ensure safe operation and energy distribution. In this context, aging and cell-to-cell inhomogeneities are directly connected to the goal of this work, which focuses on evaluating balancing methods for reused cells with non-uniform degradation profiles.

## 2.4. Modeling of Lithium-Ion Batteries

The idea of a model is to represent the behaviors of a given system under certain circumstances. In that sense, G.L. Plett presents in his book a distinction in the way models can be classified: "One is based on understanding the underlying physics that govern the cell's operation and building a model of the cell dynamics from the inside out." He continues, "... the second, simpler approach to modeling cell operation, which uses electrical-circuit analogs to define a behavioral or phenomenological approximation to how a cell's voltage responds to different input-current stimuli" (Plett, 2015). When he made this distinction, he was attempting to separate the two most common types of cell modeling used in the industry.

The first approach refers to the physical modeling of the cell. The basis of this approach is to simulate the physical phenomena occurring inside the cell—mostly electrochemical and thermal phenomena. By modeling the reactions inside the cell, the model should respond similarly to general conditions as a real cell would. This is a more complex type of modeling since the reaction behaviors are influenced by many factors. These models are usually more accurate and also allow us to understand what is happening in reality, rather than treating the model as a black box. Nonetheless, due to their complexity, these models tend to be associated with high computational costs.

The goal of the second approach is not to simulate the physicochemical behavior occurring inside the cell, but rather to mimic how the cell behaves under certain conditions. When describing this type of model, he refers to ECM models, which characterize cell behavior using a more or less simple equivalent circuit. The equivalent circuit has no direct relation to the internal reactions, but by tuning the ECM parameters, it is possible to replicate the cell's external behavior. The biggest advantage of this type of model is fast computational time. Even though the model is not as precise as physical models, the ECM remains a preferred option in many applications. This is the type of model this project is focused on, and more about ECMs is presented in Section 2.4.1.

Another modeling approach that has been gaining relevance in recent years is data-driven methods. This approach uses input data about the cell to predict the battery's state during

operation. This can be done in stand-alone mode—where the entire model is simulated based on data—or combined with other models to assist in estimating complex parameters such as aging. These methods can be implemented using various learning strategies: neural networks, support vector machines, or other machine learning techniques. This topic lies outside the scope of this project and will not be explored further.

### 2.4.1. ECM Components

As mentioned, ECMs are a behavioral type of modeling that aim to represent the behavior of a battery cell using a simple equivalent circuit. The majority of BMS implementations in today's EVs use some kind of ECM to predict or estimate the internal state of the battery. That is why, despite their simplicity, ECMs are highly important in the industry. In this chapter, the different components used in ECMs are presented and linked to the behaviors discussed in Section 2.3.2. In Section 2.4.2, these components will be used to introduce the most common ECM structures. This section is primarily based on the description provided in (Plett, 2015); please refer to it for further detail on the topic.

The first and most common component of most ECMs is a voltage source. In the case of ECMs, the voltage source represents the OCV of the cell. As introduced in Section 2.3.2, the OCV of a cell is highly dependent on the SOC and the operating temperature. In that sense, OCV functions are typically obtained experimentally from tests on the particular cell, and they can be implemented either as look-up tables (with linear interpolation) or as fitted polynomials derived using regression techniques.

Associated with the OCV voltage source is a tracker of the SOC. Since the OCV-SOC dependence is highly relevant to the cell's functioning, this must be implemented in the model. The modeling of the voltage source may thus include the following equations:

$$z = SOC \tag{2.3}$$

$$OCV(t) = f(z(t), T(t)) \tag{2.4}$$

$$\dot{z}(t) = -\eta \cdot \frac{i(t)}{Q} \tag{2.5}$$

Here, $\dot{z}(t) = \frac{d(SOC)}{dt}$, $i(t)$ is the current in amperes (positive during discharge), $Q$ is the total capacity in ampere-seconds, and $\eta$ is the Coulombic efficiency. The OCV is expressed as a function of SOC and operating temperature. If temperature dependence is not required or data is unavailable, this function can be simplified.

The next component typically found in ECMs is a series resistance, which represents the ohmic effect of the cell as introduced in Section 2.3.2. It affects the cell voltage and is modeled as:

$$v(t) = OCV(t) - i(t) \cdot R_0 \tag{2.6}$$

As mentioned, the ohmic resistance is also dependent on the operating temperature, and including this dependency can improve model fidelity.

To model the dynamic behavior of the cell, polarization is commonly represented using one or more RC groups. Including one RC group alongside the ohmic resistance adds two new equations and modifies Equation 2.6:

$$v(t) = OCV(t) - i(t) \cdot R_0 - i_{R1}(t) \cdot R_1 \tag{2.7}$$

$$i(t) = i_{R1}(t) + i_{C1}(t) \tag{2.8}$$

$$i_{C1}(t) = C_1 \cdot \dot{v}_{C1}(t) \tag{2.9}$$

The modification to Equation 2.6 adds the voltage drop across the RC group expressed in terms of the current through the resistance. $R_1$ and $C_1$ are known as the transient resistance and transient capacitance, which model the dynamic polarization effect and the time it takes for the voltage to stabilize after a step input (Tekin and Karamangil, 2024). The two new equations represent current balance at the node and the capacitor's characteristic behavior. Like ohmic resistance, the polarization response also depends on SOC and temperature, and including this dependence improves accuracy.

With these components, all the main external behaviors of a cell can be modeled. Other components may be added in more complex ECMs, particularly if frequency-dependent behavior is of interest. For example, Warburg impedance elements can be used for such modeling. However, for the scope of this project, the components presented here are sufficient for the development of a fairly accurate ECM.

### 2.4.2. Typical ECM Structure

Depending on the battery and the desired model accuracy, ECMs of varying complexity can be employed. If more detailed dynamic behavior is to be captured, the use of Thevenin-based structures is a better approach. Below, two basic ECMs are introduced, shown in Figures 2.8 and 2.9.

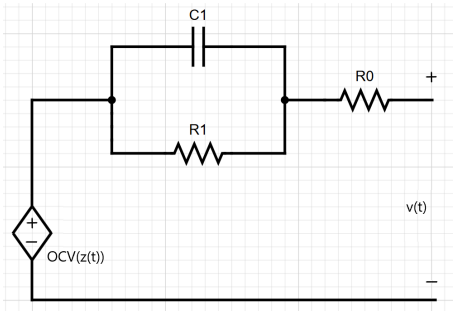



Figure 2.8: ECM with one RC branch.

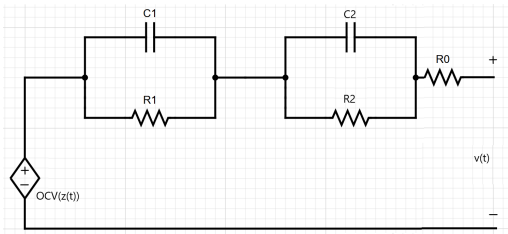



Figure 2.9: ECM with two RC branches.

Figure 2.8 and Figure 2.9 present the so-called first-order and second-order Thevenin

ECMs, respectively. In these models, $R_1$, $R_2$, $C_1$, and $C_2$ represent the polarization behavior of the cell, as discussed in Section 2.3.2. Although higher-order Thevenin ECMs can be found in the literature to increase modeling accuracy, their complexity significantly complicates parameter identification.

The selection of a particular ECM structure depends primarily on the level of precision required by the application. In many cases, overly complex structures increase parameterization difficulty and computational cost without providing significant benefits. In electric vehicles, more complex ECMs are often justified, as higher modeling precision can noticeably enhance BMS performance. However, since no specific cell is targeted in this project, a first-order Thevenin model is deemed sufficient. It reduces computational effort, simplifies parameter identification, and provides an adequate representation of generic cell behavior. Employing a more complex ECM would not yield additional benefits and, in combination with the approximate aging model, would result in unnecessary effort; see Section 2.4.3.

### 2.4.3. Aging in ECM Models

Aging modeling in cells is generally highly dependent on experimental data. The same holds for ECMs, where aging is often introduced via derating functions applied to the model parameters. These functions typically depend on variables such as cycle count, calendar time, or depth of discharge (DoD). In practice, a series of experimental tests is conducted, and the parameters of the derating functions are fitted to match the observed degradation behavior.

Since this project does not focus on modeling a specific cell, implementing complex empirical models with parameter fitting is unnecessary. Therefore, a static aging approach has been adopted. In this method, the main parameters of interest—cell capacity and series resistance—are derated at the beginning of the simulation and remain constant throughout. While derating formulas for the RC parameters are also included, their effect on cell behavior is secondary. The static aging derating is described by the following formulas:

$$Q_{\text{aged}} = Q_{\text{initial}} \cdot (1 - \text{capacity loss percentage}) \tag{2.10}$$

$$R_{s,\text{aged}} = R_{s,\text{initial}} \cdot (1 + \text{resistance increase percentage}) \tag{2.11}$$

$$R_{1,\text{aged}} = R_{1,\text{initial}} \cdot (1 + \text{polarization resistance increase percentage}) \tag{2.12}$$

$$C_{1,\text{aged}} = C_{1,\text{initial}} \cdot (1 - \text{transient capacitance loss percentage}) \tag{2.13}$$

In these expressions, $Q_{\text{initial}}$ and $R_{s,\text{initial}}$ are the nominal cell capacity and initial series resistance before aging, while $Q_{\text{aged}}$ and $R_{s,\text{aged}}$ are their aged values after accounting for capacity fade and resistance growth. Similarly, $R_{1,\text{initial}}$ and $C_{1,\text{initial}}$ represent the initial polarization resistance and transient capacitance of the cell, with $R_{1,\text{aged}}$ and $C_{1,\text{aged}}$ being

the corresponding values after degradation.

Reasonable values for these derating parameters, according to literature, suggest that second-life lithium-ion cells typically exhibit a capacity reduction of 10–30% and a series resistance increase of 30–100%, see (Pérez et al., 2024), (He and Chen, 2023). Additionally, aging effects on the transient elements, such as polarization resistance and capacitance, have been observed, with typical increases in polarization resistance of 20–80% and decreases in transient capacitance of 5–30%, see (Ecker et al., 2014), (Keil and Jossen, 2016).

When modeling a specific cell where detailed conclusions are required, a dedicated aging model with time-dependent behavior becomes necessary. Modeling aging accurately is particularly important when the objective is to track the evolution of ECM characteristics over the full life of the cell. In this project, however, the primary focus is not on the aging process itself, but rather on establishing a realistic model of an aged cell suitable for evaluating the efficiency of different balancing methods. Therefore, a set of static aged parameters has been assumed and applied unequally across the different cells to introduce inhomogeneity typical of real-world second-life battery systems.

### 2.4.4. Unbalance in ECMs

In addition to aging effects, cell packs typically exhibit inherent unbalance, originating from manufacturing tolerances, different operational histories, or early-stage degradation (as introduced in Section 2.3.5) (Reiter et al., 2023). To represent this phenomenon, an unbalance parameter is introduced into the model, independently of the global aging parameters. The unbalance is modeled as a deviation applied to the capacity and series resistance of each cell:

$$Q_i = Q_{\text{aged}} \cdot (1 + \delta Q_i) \tag{2.14}$$

$$R_{s,i} = R_{s,\text{aged}} \cdot (1 + \delta R_{s,i}) \tag{2.15}$$

Here, $\delta Q_i$ and $\delta R_{s,i}$ are the relative deviations of the cell's capacity and resistance, respectively, with respect to the aged nominal values. As mentioned in Section 2.3.5, the variability in capacity can range between 1–3%, and internal resistance can vary by up to 5% (Reiter et al., 2023).

### 2.4.5. Parameter Identification

Parameter identification is the process of obtaining the values of the components of the equivalent circuit model (ECM), including the internal resistance, transient resistances, and capacitances. These parameters cannot be obtained directly from manufacturer datasheets; they must be extracted from experimental cell voltage responses under controlled current excitations. This means that each particular cell will have an associated set of parameters

that approximate the electrical behavior of the cell at the time of the experiment.

Among the available techniques, time-domain identification based on current pulse tests offers a practical balance between accuracy and implementation cost. In this method, a known current pulse is applied to the cell, and the voltage response is recorded. The immediate voltage drop is attributed to the ohmic resistance $R_0$, while the subsequent relaxation behavior is fitted to exponential decay functions, from which the polarization resistances and capacitances can be obtained (Tekin and Karamangil, 2024). An outline of the procedure is summarized in Table 2.1.

Table 2.1: Summary of identification procedure for ECM parameters using time-domain analysis of pulse tests.

| **Step** | **Procedure** | **Extracted Parameters** | **Notes** |
|---|---|---|---|
| 1 | Apply a constant current pulse (discharge or charge) | – | Typically a step current of known amplitude and duration |
| 2 | Measure instantaneous voltage drop at pulse onset | $R_0 = \Delta V / I$ | Reflects internal ohmic resistance |
| 3 | Analyze voltage relaxation during the pulse | $R_1$, $C_1$ (or $R_2$, $C_2$) | Fit exponential decay using time constant $\tau = RC$ |
| 4 | Fit terminal voltage response using nonlinear least-squares | All parameters | Optional refinement over visual inspection |
| 5 | Validate model against pulse and rest periods | – | Confirm transient match under dynamic current |

For the purposes of this work, the ECM parameters $R_0, R_1, C_1$, and $E_m$ are implemented as SOC-dependent parameters, obtained by using pulse tests at different SOC levels. This approach reflects the dynamic cell behavior under realistic operating conditions. To maintain computational feasibility in Modelica, interpolation tables are used, and the aging and unbalance effects are applied as multiplicative derating factors on top of the SOC-dependent curves.

## 2.5. Battery Balancing

As discussed in Section 2.3.5, inhomogeneities between battery cells result from both manufacturing deviations and aging. These deviations lead to uneven capacity usage, thermal stress, and premature cutoffs during charge or discharge. The origin of imbalances may lie in variations in internal resistance, capacity, or purely in SOC. While SOC imbalance typically manifests as voltage spread during operation, resistive and capacitive imbalances

affect heat generation, dynamic response, and usable energy range. To mitigate these effects and ensure safe and efficient utilization of the entire battery pack, balancing strategies are implemented.

### 2.5.1. Balancing Methodologies

Battery balancing methods are generally classified into passive, active, and hybrid approaches. Each aims to reduce SoC imbalances by redistributing or dissipating energy, with trade-offs in complexity, efficiency, and cost (Khan et al., 2024), (Uzair et al., 2021).

In passive balancing, excess energy from higher-voltage cells is dissipated as heat using resistive elements. Common implementations of this type include:

- Shunt resistor balancing, where fixed resistors are used to continuously discharge cells, both with higher and lower voltages. The theory behind this is that cells with higher voltage will discharge faster than those with lower voltage, balancing the energy between cells.
- Switched resistor balancing, which introduces MOSFETs to selectively control energy dissipation.

Passive balancing is normally selected in low-cost or low-complexity systems, although it inherently results in energy loss and longer balancing times (Perişoară et al., 2018), (Uzair et al., 2021).

Active balancing redistributes energy among cells using transfer components. This includes:

- Switched capacitor networks, for voltage equalization between cells.
- Inductive or transformer-based circuits, enabling charge transfer of different types depending on the desired result.

Active balancing improves energy efficiency and balancing speed but requires more complex circuitry and control logic. It is better suited for high-energy systems or applications where efficiency is critical (Khan et al., 2024).

Hybrid strategies combine both approaches, for instance using passive balancing at the end of charge and active methods during dynamic operation. Some hybrid systems include auxiliary DC-DC converters with resistive paths to optimize performance across varying conditions (Uzair et al., 2021). To illustrate and show the wide range of balancing methods available in the market, Figure 2.10 has been added.

As shown in Figure 2.10, within the different categories mentioned above, multiple types of solutions can be implemented. This allows for adaptation of the technology to the specific requirements. Beyond circuitry, balancing can also be classified by energy transfer directionality:

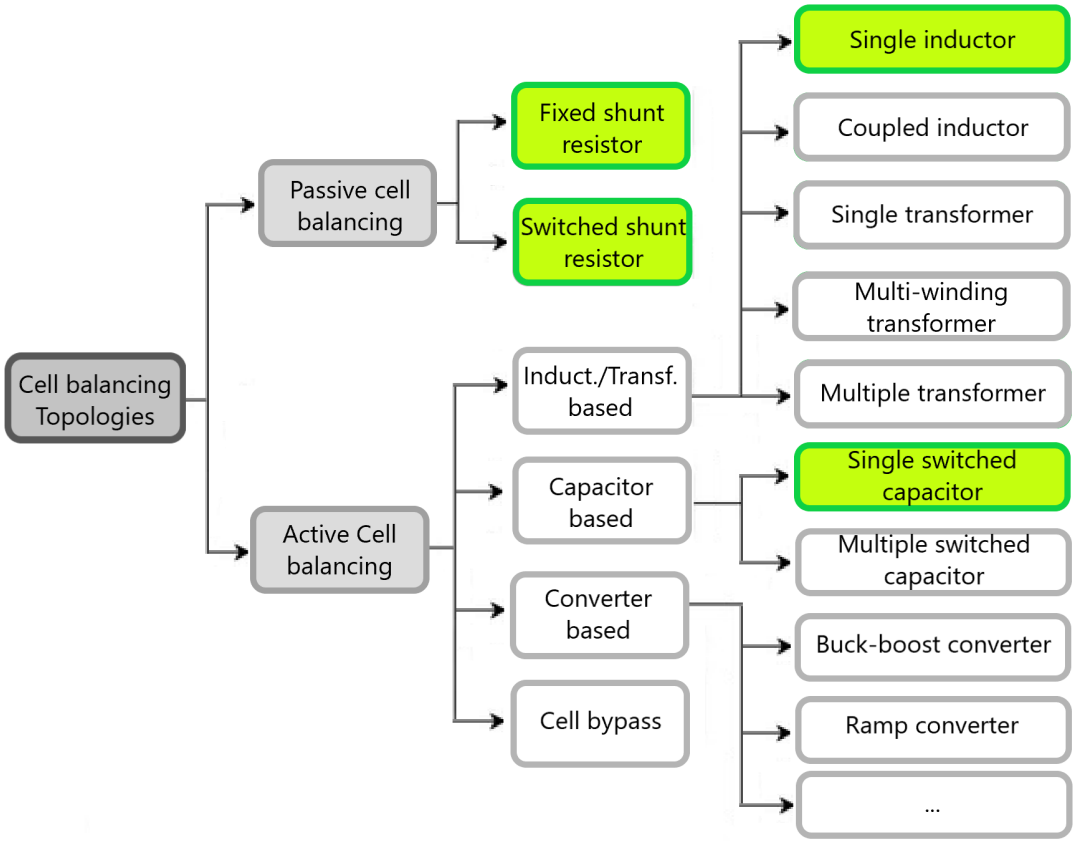



Figure 2.10: Classification of most common balancing topologies. Image inspired by (Ashraf et al., 2024).

- Cell-to-cell: Direct redistribution between adjacent or non-adjacent cells.
- Cell-to-pack: Charging a common bus from high-energy cells to the lowest charged pack.
- Pack-to-cell: Recharging low-voltage cells in low-voltage packs from other packs with higher voltage.
- Pack-to-pack: Inter-module balancing, especially in modular or second-life systems.

Some of these configurations provide increasing flexibility but demand more coordination and system-level communication (Uzair et al., 2021). This study focuses on four balancing methods, both in passive and active categories:

- Passive: Shunt resistor and switched resistor.
- Active: Single capacitor and single inductor.

Their schematic structures are summarized in Figures 2.11, 2.12, and 2.13. The methods will be evaluated in a controlled four-cell pack, offering realistic imbalance scenarios while remaining computationally manageable.

### 2.5.2. Balancing Topologies

To explore the behavior of these balancing strategies in depth, a four-cell lithium-ion pack was selected. This configuration allows for meaningful imbalance dynamics while limiting simulation and hardware complexity. Each method is described below in terms of principle, schematic, pros and cons, and general control considerations.

#### Passive Balancing Methods

Regarding passive balancing methods, this project explores two of them, shunt resistors and switched resistors. A schematic of each topology is included in Figure 2.11 and 2.12.

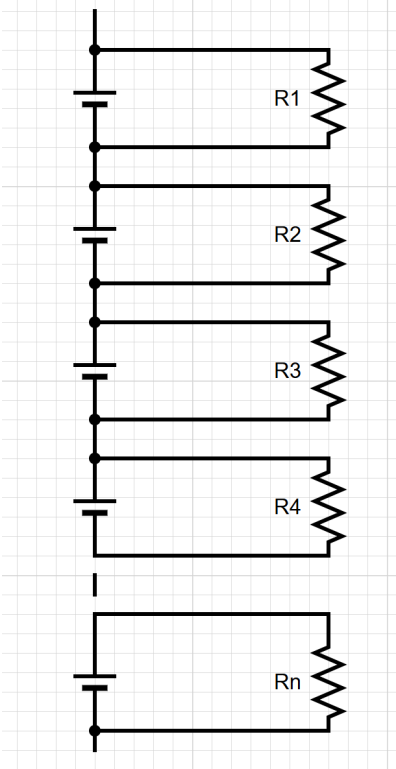



Figure 2.11: Schematic of shunt resistor balancing method.

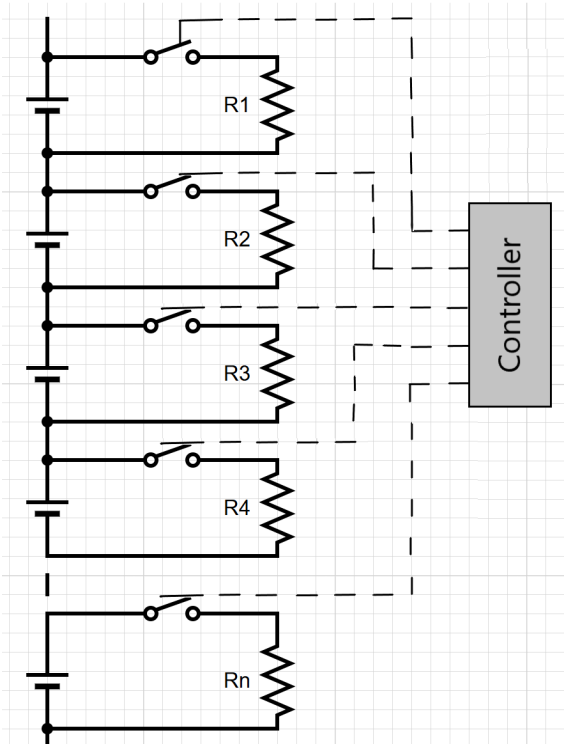



Figure 2.12: Schematic of switched resistor balancing method.

Figure 2.11 shows the schematic of the shunt resistor balancing. This method continuously or periodically discharges cells through fixed resistors to match the voltage of the lowest cell. The highest-voltage cell tends to discharge faster than the lower-voltage cells, equalizing over time to the same voltage. Advantages of this method include low cost, minimal complexity, and simple control logic. Disadvantages include low energy efficiency, slow balancing response, and the need for thermal design under extended operation.

Figure 2.12 shows the schematic of the switched resistor balancing. In this case, MOSFETs are used to enable or disable the balancing resistors based on cell voltage differences. The control strategy can vary depending on the desired results. The most common strategies

are either charging the cell up to the maximum and later balancing, or continuous balancing using the cell with the lowest voltage as reference. The advantages of this method include controlled dissipation, better energy usage than the shunt resistor method, and integration into basic BMS platforms. On the other hand, this method is still inherently lossy and requires more components and switching logic.

### Active Balancing Methods

Regarding active balancing methods, this project explores two of them: single capacitor balancing and single inductor balancing. A schematic of the single capacitor topology is included in Figure 2.13; the schematic for the single inductor topology is identical but uses an inductor instead.

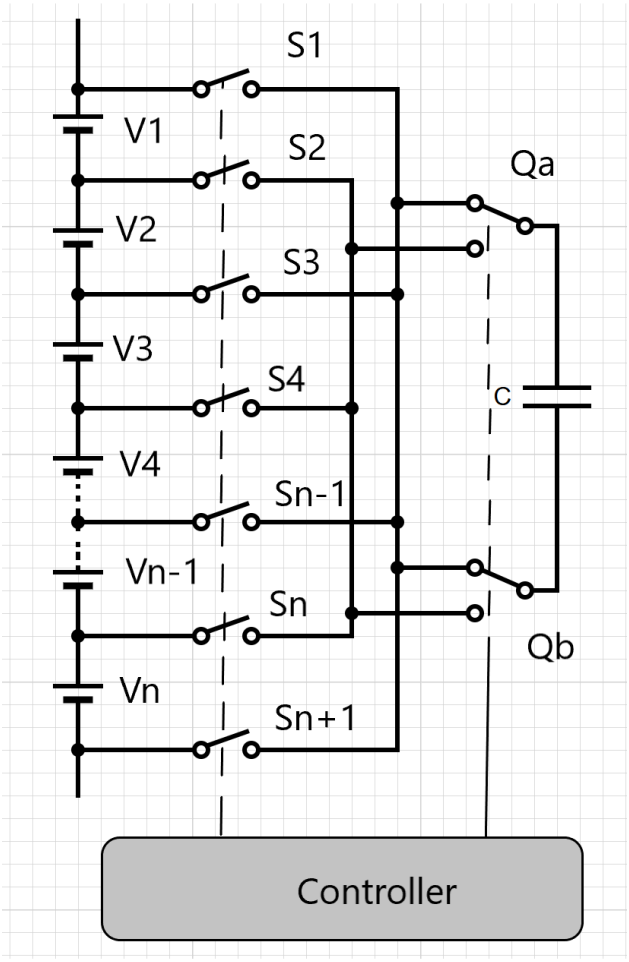



Figure 2.13: Schematic of single capacitor balancing method.

In single-capacitor balancing, a capacitor is used per pack of cells. Charge from the most charged cell is continuously transferred through the capacitor to the cell with the lowest charge, using switches coordinated by a PWM pulse. This balancing can be performed

during charging, discharging, or static operation, contributing to a more effective use and distribution of energy. The single inductor method is similar in most aspects.

Advantages of these methods include higher energy efficiency and the ability to balance during both charge and discharge, improving the utilization range of all cells. Disadvantages include limitations to packs with a small number of cells and timing-critical control for effective operation (Khan et al., 2024).

### 2.5.3. Control Strategies

Balancing topologies are defined not only by their circuit design but also by the control strategy applied. The control algorithm determines when and how balancing is activated and has a direct impact on efficiency, response time, and safety. Depending on the control variable selected, the following strategies can be distinguished:

- **Voltage-Based Control.** This method monitors cell voltages and initiates balancing when deviations exceed a defined threshold. It is the simplest to implement and widely used in passive BMS designs. However, it may be inaccurate during dynamic load conditions, where terminal voltage does not directly reflect SoC (Khan et al., 2024).
- **SoC-Based Control.** This approach estimates the SoC of each cell through current integration or lookup tables. It provides better long-term balancing accuracy but depends on accurate SoC estimation, which itself can be subject to drift and calibration issues (Khan et al., 2024).
- **Model-Based Control.** The most advanced strategy uses internal cell models—equivalent circuit or electrochemical—to estimate SoC, internal resistance, and even aging states. These systems can optimize energy transfer under varying conditions and are well-suited to complex or second-life applications, though they demand significantly more computation and parameter knowledge (Khan et al., 2024), (Uzair et al., 2021).

This study adopts voltage-based control to prioritize simplicity, robustness, and comparability across passive and active implementations. The effects of aging introduce differences between the OCV of different cells for the same SoC, making control through SoC more complex. Extensions to SoC-based or model-based strategies are identified as future work. For each of the topologies presented in Section 2.5.2, different strategies can be selected for the control of each structure. The selections for this project will be presented and justified in Sections 4.3 and 4.5.

## 2.6. Conclusion

This chapter introduced the key concepts and technologies used throughout the project. Modelica was presented as the chosen modeling tool for its flexibility and reuse capabilities.

A general review of lithium-ion battery behavior was provided, including static and dynamic characteristics, aging effects, and second-life considerations.

The main modeling approaches were discussed, with the focus placed on ECMs due to their simplicity and suitability for fast simulations. The different components of ECMs and their parameterization were explained in detail, along with how aging and imbalance can be represented.

Balancing methods were classified and compared, with four specific topologies selected for implementation and testing. Voltage-based control was chosen as the base strategy for this project due to its simplicity and suitability for second-life applications. Together, these topics form the theoretical foundation for the modeling and testing work developed in the next chapters.

# Chapter 3

# Modeling Hypotheses

## 3.1. Introduction

This chapter explains the modeling approach used to build the battery system library. The main goal is to describe how each part of the model was designed, including the electrical cell models, the balancing methods, and the control logic.

The chapter is divided into two parts. First, Section 3.2 focuses on the ECM (Equivalent Circuit Model), including how the cell voltage, resistances, and capacitance are modeled based on SOC data. It also explains how cells are grouped into packs. Then, Section 3.3 introduces the balancing systems used in this work, along with the charge/discharge module and control unit. The structure and logic of each balancing method are shown with diagrams and code examples.

## 3.2. ECM Modeling

This section shows how the different parts needed for the ECM have been modeled. Starting from the basic components, specifying the data structure required for them and the final result forming a complete ECM model cell. Only structures with certain particularities are described; the complete code for the library with annotations can be found in Appendix A for reference. Details about the implementation of these models are presented in Chapter4.

### 3.2.1. Open Circuit Voltage

As mentioned, the open-circuit voltage is commonly modeled as a function of the SOC and the temperature, see 2.4.1. For the given model, only the SOC will be considered. With that being said, the formulas used to model the SOC are Formulas 3.1 and 3.2; and the main equations that model this behavior are shown in Listing 3.1.

$$OCV(t) = f(z(t)) \tag{3.1}$$

$$\dot{z}(t) = -\eta i(t)/Q \tag{3.2}$$

```
model OCV_source "OCV source"
  extends Interfaces.UnPuerto(activo=true);

  input Real OCV;
  parameter SI.ElectricCharge Q;
  parameter Real eta(min=0, max=1)=1;

  Real z(start=1, max=1, min=0);

equation
  u = OCV;
  der(z) = -eta * i / Q;
end OCV_source;
```

Listing 3.1: Modelica code for OCV component

Where, in Figure 3.1 OCV will be the function of the SOC as given by laboratory tests and the internal estimation of SOC is done by Coulomb counting. The value of the Coulombic efficiency will be given a preset value of 1, which is the most normal approximation if unknown. And the OCV is loaded as a dataset and interpolated linearly between points depending on the OCV. To avoid stiffness in the interpolation the use of the native Modelica "CombiTable1Ds" class is used, this is introduced in the following Section 3.2.2.

### 3.2.2. Parameter Loader

This project will focus on parameterizing one RC branch ECM from experimental data, as well as the ohmic resistance $R_0$, polarization resistance $R_1$, and transient capacitance $C_1$ are implemented as SOC-dependent quantities. The parametrization done from a set of data of impulse tests on a real cell is presented in Section 4.2. Once the parametrization is done a curve representing each parameter in relation to the SOC of the cell is obtained, see Figure 4.2. These curves arise from a set of points, in order to use it in the simulation the missing data in between points will be linearly interpolated. The code in Listing 3.2 shows the loading and interpolation for one of the parameters as a function of the SOC.

```
model ECM_ParameterLoader_1Thv "ECM Parameter loader from .mat"
  input Real SOC;  // External input

  CombiTable1Ds table_OCV(
    tableName="ECMdata",
    fileName="dataECM.mat",
    columns={2},
    tableOnFile=true,
    table = [0, 0],
    smoothness=Smoothness.LinearSegments) ;

  output Real OCV;
  ... //Needs to be repeated for each variable

equation
  // Connect SOC input to table inputs
  table_OCV.u = SOC;

  // Read interpolated outputs
  OCV = table_OCV.y[1];
  ...

end ECM_ParameterLoader_1Thv;
```

Listing 3.2: Paramter loading and interpolation with CombiTable1Ds

Note that all the parameters and the OCV curve are given in the form of a vector, containing a value for every SOC stored. To obtain data between data points smoothened in-built interpolation fuctions are used. Avoiding this way possible problems with stiffness and non-linearities. For more information on this function refer to (Association, 2024a).

### 3.2.3. ECM Structure

In this section, the structure used for creating ECM is presented. More than one ECM structure has been created for different applications. In this section the model for the second life single Thevenin ECM is presented. Variations of the model mentioned can be found in A. Figure 3.1 shows the Diagram view of the model.

As seen in Figure 3.1 the internal structure of the cell is mainly created from previous components from the diagram view. It is worth noticing that the parameters for the OCV curve, and the resistances and capacitance will be derated when introducing them in the model. There are two main types of deratings, by age and by unbalance. Part of the

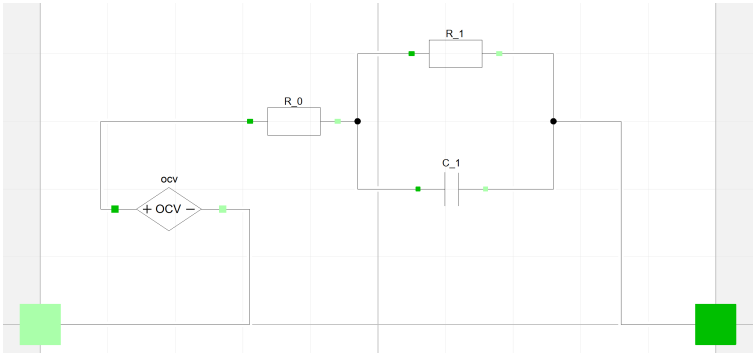



Figure 3.1: Graphic view of the internal structure of a single Thevenin ECM

structure used to derate this values is shown in Listing 3.3.

```
model ECM_Thevenin_SL
  ...
  //Derating factors
  parameter Real alpha_R0(min = 0, max = 1) = 1.0 "R0 aging derating factor";
  parameter Real beta_R0(min = 0.5, max = 1.5) = 1.0 "R0 unbalance derating factor";
  //Effective parameters after derating
  SI.Resistance R0_eff "Effective series resistance";
  ...
equation
  // Derating equations for parameters
  R0_eff = R0 * alpha_R0 * beta_R0;
  ...
end ECM_Thevenin_SL;
```

Listing 3.3: Derating factors for 1 cell single Thevenin ECM

The idea, as expressed in Section 2.4.3, is to adjust each parameter by two factors, alpha and beta, to simulate the aging conditions of the cell. The specific parameters used for the test examples are presented in Section 4.2.

### 3.2.4. Cell packs structure

As in the previous chapter, several models of cell packs are available in the library for different uses. They offer a similar set up and to ilustrate the main characteristics of them the model for Battery4Cell_Ball is further explained. The diagram set-up is shown in the following Figure 3.2:

As shown in Figure 3.2, the set up consists of four ECM cell models in series with a dedicated pin between each cell to allow for balancing if needed. Regarding the particularities of the code, two matrixes of derating values are needed to fill every cell and the model expects

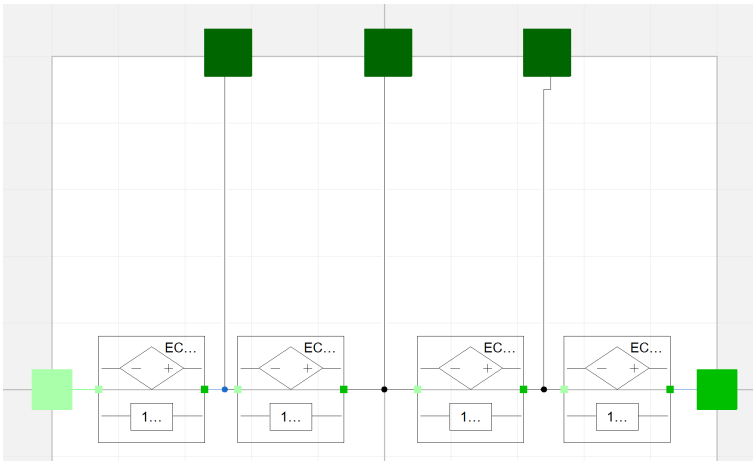



Figure 3.2: Graphic view of the diagram of model Battery4Cell_Bal.

the parameter-SOC curves to be loaded from a higher level. No voltage feedback or control is added to this standard module, these functions will be introduced in the Balancing module instead as explain in the following Section 3.3.

## 3.3. Balancing Modeling

This section shows how the different parts of the balancing systems have been modeled. It introduces the balancing models of the four selected structures, together with the charge/load module and the respective control units. Specifics about the later implementation of the models for the tests are commented in Sections 4.3, 4.5.

### 3.3.1. Shunt Resistor Modeling

The structure for the shunt resistor module has been modeled as shown in Figure 3.3.

Figure 3.3 shows the graphic diagram of the model. It has one resistance and one MOSFET per cell, and the MOSFETs are activated by a step in voltage on the gate. The activation signal comes through the connector, which sends one signal for all the voltages. And the second connector sends the voltage feedback data for each cell.

### 3.3.2. Switched Resistor Modeling

The structure for the switched resistor module has been modeled with an almost identical structure to the one shown for the Shunt balancing module in Figure 3.3. The main difference comes in the connector. Instead of having only one signal coming from the control module that controls every switch, each switch presents an individual switching signal, allowing every swicht to act independently.

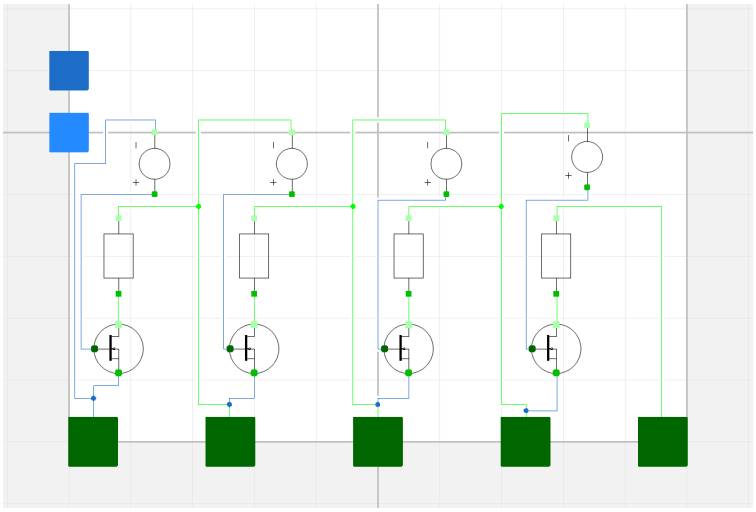


Figure 3.3: Graphic view of the diagram of the shunt resistor balancing module.

### 3.3.3. Single Capacitor Modeling

The structure for the single capacitor module has been modeled as shown in Figure 3.4.

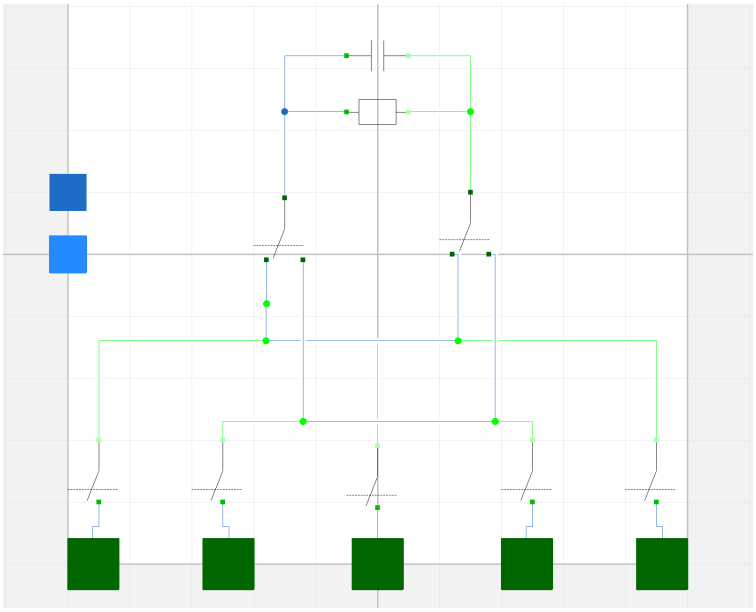


Figure 3.4: Graphic view of the diagram of the single capacitor balancing module.

The module consists of five controlled switches and two controlled commutators. The layout allows for connecting any cell to the balancing capacitor with a set of input signals controlling the commutators and switches. Then this set of signals is periodically switched to transfer charge from one cell to the capacitor and from the capacitor to the subsequent cell. Similarly to the switched balancing the module has two connectors; one with seven signals (one for each switching component), and a second connector to retrieve the voltage feedback of each cell. More details on the control structure are given in Section 3.3.6.

### 3.3.4. Single Inductor Modeling

The balancing structure for the single inductor balancing module is almost identical to the one presented in 3.4. The main difference in the balancing module is that a inductor is used instead of a capacitor. Due to it's physical nature some changes need to be also implemented in the switching structure. And the bleed resistance is in series with the inductor, since having a parallel resistance would create an RL group with unwanted discharging dynamics.

### 3.3.5. CC Charging model

With the purpose of running the tests, a module that consistently handles the charging and discharging of the cell has been created. For charging, as it was mentioned in Section 2.3.3, there are different methods, being the Constant current, constant voltage the most common one, reffer to the given section for more details. Nonetheless the simple constant charging with safety margin is considered to be sufficient for out application, thus constant voltage charging has not been implemented. In the other hand, for discharging, there is no preferred method in the industry. In this case, constant current discharge has been selected.

The model for the charge/discharge block is simple from the component side. It includes a variable current source and a load in series. To be able to select the operating mode a connector that communicates with the control unit is included. Particularities about the model are presented in Listing 3.4.

```
model ChargeDischarge
  // Parameter declaration
  ...
equation
  if controlInput.mode == 1 then
    currentSource.i_discharge = +I_charge;
  elseif controlInput.mode == -1 then
    currentSource.i_discharge = -I_discharge;
  else
    currentSource.i_discharge = 0;
  end if;
  ...
end ChargeDischarge;
```

Listing 3.4: Charge/Discharge model code.

As it can be seen in Listing 3.4, the functionality of the Charge/Discharge module is easily defined by an if statement. Depending on the signal from the control unit "mode", the source current imposes a certain current through the module.The particular parameters

and limits for charging/discharging are assessed from the control module and will be stated in the implementation details, see Section 4.4.

### 3.3.6. Control Unit model

The control unit module is responsible of processing all the information from the state of the cells and executing both the charging/discharging and the adequate balancing. A general view of how the Control Unit interacts with the other modules is shown in Figure 3.5.

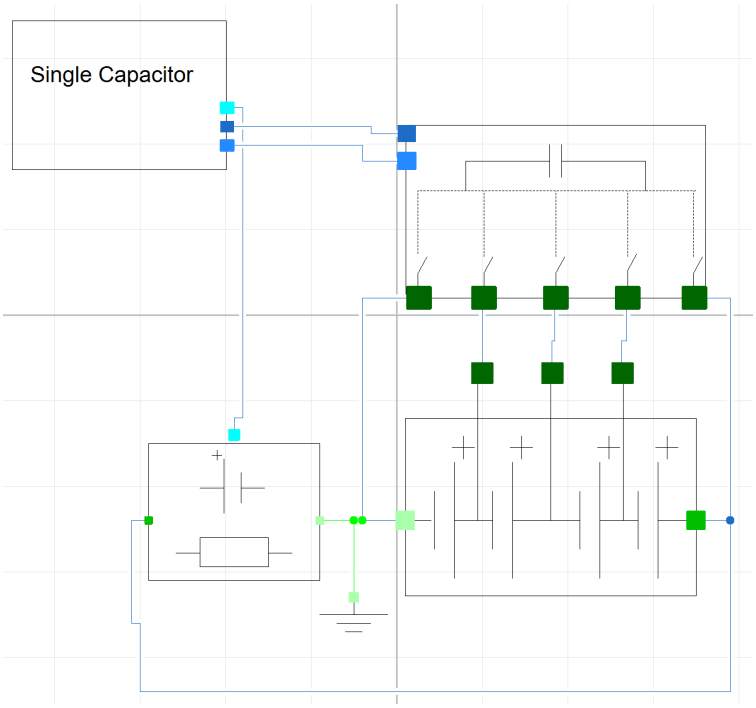



Figure 3.5: Graphic view of the diagram of single capacitor balancing example.

As presented in Figure 3.5, the control unit is normally connected by two connectors to the balancing system; one connection for the voltageFeedback and a second one for the balancing control signals; and with one connector to the Charge/Discharge module to allow for mode selection. The necessary control sequence is case-specific, which means that the control unit module for each case is different. For this sequence control the native Modelica library StateGraph2 has been used. This library allows for the creation of Grafcet diagrams integrated with code. For more information about the library, refer to (Association, 2024b). The modeling details for each of the main cases are described below, and a schematic version for the 4 cases can be found in Appendix A for clarification.

### Control Unit - Shunt Balancing

The control logic for shunt resistor balancing follows a simple logic loop that operates only during the charging phase. The internal control loop logic is defined by the following steps:

- Initial Idle
- Discharge until $min(V_{\text{cell}}) < V_{\text{min}} + margin$
- Idle to bring back cell to stationary state.
- Charge until $max(V_{\text{cell}}) > V_{\text{max}} - margin$
- Go back to initial step.

In the code section the necessary control signals are processed and sent to the respective modules. Listing 3.5 shows the main parts of this model code.

```
model Controller_ShuntResistor_SG
  // Parameter declaration
  ...
equation
  ...
  //Charge/discharge mode sequence
  controlLoad.mode = if Discharge.active then -1
    elseif Charge.active then 1
    else 0;
  //Balancing signal
  controlBalancing.OnOff = Charge.active;
  ...
end Controller_ShuntResistor_SG;
```

Listing 3.5: Charge/Discharge model code.

As introduced in Section 2.5.2, the shunt effect of the module will be only and always activated while charging. This can be seen in Listing above 3.5, the control balancing signal is active while the Charge step in the grafcet is active. In the same Listing, the charge discharge control signal equations are set. Depending on which step of the sequence is active, it activates either charge, discharge or idle on the charge/discharge module. This is a structure that remains the same for all the control structures. Being, the main changes about how the balancing is performed and not on the charge/discharge cycle.

### Control Unit - Switched Resistor

The control unit for the switched resistor module extends the logic of the shunt resistor with individual and selective cell balancing. Unlike the simple shunt where all dissipation happens simultaneously in all the cells at the same time and during the whole chargin cycle, the switched resistor approach allows per-cell decision making with individual control over each dissipative path.

The control steps regarding charging discharging are similar. It being a circular sequence idle, discharge, idle, charge, idle. The main difference has to do with the balancing while charging functionality. For that two new states are created, "CheckBalancing" and "ExecuteBalancing". Where the need of balancing is checked and stored, and later executed. Please see Appendix A and the library in Modelica for more details on the control sequence. Listing 3.6 shows the particularities of the shunt control module.

```
model Controller_SwitchedResistor_SG
  //parameter declaration
  ...
equation
  ...
  // Per-cell balancing state machine with hysteresis
  for i in 1:4 loop
    OnOffMem[i] =
      if CheckBalancing.active then
        if voltageFeedback.V_cell[i] > min(voltageFeedback.V_cell) + dV_off then
          true
        elseif voltageFeedback.V_cell[i] < min(voltageFeedback.V_cell) + dV_off -
        eps then
          false
        else
          pre(OnOffMem[i])
      elseif ExecuteBalancing.active then
        pre(OnOffMem[i])
      else
        false;
  end for;
  ...
  //Assign balancing signal
  controlSignalBalancing_Switched.OnOff = OnOffMem;
end Controller_SwitchedResistor_SG;
```

Listing 3.6: Control unit for switched balancing.

The main condition deciding to go from Charge step to CheckBalancing is whether

$min(V_{\text{cell}}) < max(V_{\text{min}}) - dV_{on}$. In Listing 3.6 the conditions for updating the control signal memory and the actual control signal are presented. Once in balancing states this code is executed to see what cells need to be kept the same or switched to a different state. The use of dV_off and eps ensures avoiding chattering (instant multiple switching) in the simulation.

### Control Unit - Single Capacitor

The single capacitor control unit logic manages energy redistribution between adjacent cells using a flying capacitor topology. This method operates via rapid switching sequences, transferring charge from the highest to the lowest voltage cell through controlled capacitor charging and discharging cycles. This compensation is done during the whole operating time, which means that it is always active during charge, discharge and idle. This makes it convenient to use a parallel cycle structure, both for this topology as well as for the single inductor topology. The first cycle is identical to the one explained in the shunt capacitor control, Section 3.3.6. And it controls solely the charge and discharge of the module. While the second cycle is only focused in balancing. This second internal control loop logic is defined by the following steps:

- Precharge. To initialize the voltage on the capacitor.
- Check voltage. Non-action state until balancing is needed. Balancing condition: $max(V_{\text{cell}}) - min(V_{\text{cell}}) > dV_{on}$.
- Execute balance. Updates the cells that need to be compensated and update the control signal to the balancing module. After a given t_Balance returns to CheckVoltage.

Besides the parallel control sequences, this control module presents other particularities. As mentioned in Section 3.3.3, there is a particular set of switches/commutators to correctly engage each cell during balancing. These are represented by the switch and com matrices in Listing 3.7.

Listing 3.7 gives the combination of switches necessary to engage this cell. These addresses are used in the equation section to select the correct switching addresses depending on the index of the cells to engage and execute periodic flipping of these addresses, see Listing 3.8.

Listing 3.8 shows the logic behind the flipping of switch patterns between the highest an lowest voltage cells. First identification of addresses of the above-mentioned cells, second the storing of the switching patterns for such addresses; and third, switching execution between maximum and minimum voltage switching patterns every period.

### Control Unit - Single Inductor

This single inductor control unit has a similar logic to the single capacitor one. It has identical parallel control sequences but the approach for assigning, updating, and flipping

```
model Controller_SingleCapacitor_SG
  ...
  constant Boolean switchPattern[4, 5] =
  [ true,  true,  false, false, false;
    false, true,  true,  false, false;
    false, false, true,  true,  false;
    false, false, false, true,  true];
  constant Integer comPattern[4, 2] =
  [ 0, 1;
    1, 0;
    0, 1;
    1, 0];
equation
  ...
end Controller_SingleCapacitor_SG;
```

Listing 3.7: Control unit for single capacitor balancing, switching patterns.

switching patterns is slightly different. In this case to ensure that the balancing inductance has a net-zero voltage/current curve, the cells to balance need to be connected with opposite polarity each cycle. See Listing 3.9 for the main differences.

As presented in Listing 3.9 there is no "fixed" switching pattern for the commutators. In this case, the decision is whether or not the commutator pattern needs to be flipped every cycle or not. Depending on the index of the cells to balance, this is evaluated, and a flip condition is added to the actual commutator flipping during the balancing step.

## 3.4. Conclusion

In this chapter, the full modeling structure for the battery system was described. The ECM was built using parameter curves taken from experimental data, and extra features like aging and cell imbalance were added using derating factors. These models were combined into cell packs for testing.

Four different balancing systems were modeled, each with its own control unit. The control logic was built using state diagrams that activate charging, discharging, and balancing actions. While the models work well for small systems, each balancing method needs its own custom control, and larger systems may require extra improvements. This chapter provides the complete setup needed to run simulations and test the results in the next chapters.

```
model Controller_SingleCapacitor_SG
  ...
equation
  ...
  // Argument of min and max voltage cells
  when Check_Execute.fire then
    idx_max =Functions.argmax(voltageFeedback.V_cell);
    idx_min =Functions.argmin(voltageFeedback.V_cell);
  // Assing switching patterns for max and min
    for i in 1:5 loop
      OnOff_max[i] = switchPattern[idx_max, i];
      OnOff_min[i] = switchPattern[idx_min, i];
    end for;
    for j in 1:2 loop
      com_max[j] = comPattern[idx_max, j];
      com_min[j] = comPattern[idx_min, j];
    end for;
  end when;
  // Periodic commutation switching
  when sample(0, T_commute) and ExecuteBalancing.active then
    comm_flip = not pre(comm_flip);
  end when;
    // Activate control signal depending on state and comm_flip
  for i in 1:5 loop
    controlSignalBalancing_Single.OnOff[i] =
      if initial() then (i == 5) else
      if Precharge.active then (i == 5) else
      if ExecuteBalancing.active then
      if comm_flip then OnOff_max[i] else OnOff_min[i]
      else false;
  end for;
  controlSignalBalancing_Single.Com[1] = if ExecuteBalancing.active then
     if comm_flip then com_max[1] else com_min[1] else 1;
  controlSignalBalancing_Single.Com[2] = if ExecuteBalancing.active then
      if comm_flip then com_max[2] else com_min[2] else 0;
end Controller_SingleCapacitor_SG;
```

Listing 3.8: Control unit for single capacitor balancing, equations.

```
model Controller_SingleInductor_SG
 ...
equation
  when Check_Execute.fire then
    ...
    // Assign base commutator pattern
    com_max =  {1, 0};
    com_min =  {0, 1};
  // Determine whether commutator flipping is needed
    flipCommutators =  abs(idx_max - idx_min) == 2;
    ...
  end when;
    ...
  for i in 1:5 loop
    controlSignalBalancing_Single.OnOff[i] =
      if initial() then (i == 5 or i == 4) else
      if Precharge.active then (i == 5) else
      if ExecuteBalancing.active then
      if comm_flip then OnOff_max[i] else OnOff_min[i]
      else false;
  end for;
controlSignalBalancing_Single.Com[1] =
    if initial() then 1
    else if ExecuteBalancing.active then
      if flipCommutators then
        if comm_flip then com_max[1] else com_min[1]
      else com_max[1]
    else 0;
    //Same for Com[2]
    ...
end Controller_SingleInductor_SG;
```

Listing 3.9: Control unit for single inductor balancing, equations.

# Chapter 4

# Implementation Details

## 4.1. Introduction

In this chapter, the implementation details of the modeling and simulation framework are presented. Building on the model structures introduced in Chapter 3, the specific parameter values, component selections, and control configurations used in the simulations are explained.

The chapter begins with the parametrization of the equivalent circuit model (ECM), including aging and unbalance factors to replicate second-life battery behavior. It continues with the sizing of components for each balancing topology—resistive, capacitive, and inductive—and justifies the design choices based on literature and practical constraints. The charge and discharge module setup is also described, along with the logic behind the balancing control algorithms. Altogether, these details define the simulation environment used to evaluate the performance of the selected balancing methods.

## 4.2. Parametrization of ECM

As previously introduced, a single Thevenin ECM was selected to represent the behavior of each lithium-ion cell. As presented in Figure 3.1, the structure includes an ohmic resistance ($R_0$), a parallel RC branch defined by a polarization resistance ($R_1$) and a transient capacitance ($C_1$), along with a voltage source representing the open-circuit voltage (OCV). This section explores the parametrization of these variables for an example cell and the modeling of aging and unbalance within a four-cell pack, all starting from the same nominal values.

### 4.2.1. Base Parameters

The parametrization of a cell arises from experimental testing on a physical cell. Typically, this is performed through a pulse discharge test, in which the battery is subjected

to current pulses at various states of charge (SOC), as described in Section 2.4.5. For simplification purposes, the OCV is considered SOC-dependent, while the remaining model parameters are taken as independent of SOC. Effects such as temperature variation and hysteresis—although relevant in high-fidelity modeling—are omitted here to reduce complexity and focus the analysis on balancing efficiency under defined conditions.

The base data used for parameterization are derived from experimental results available in the MATLAB battery library, specifically from a pulse discharge test. Figure 4.1 presents the discharge current, voltage, and SOC over the duration of the test.

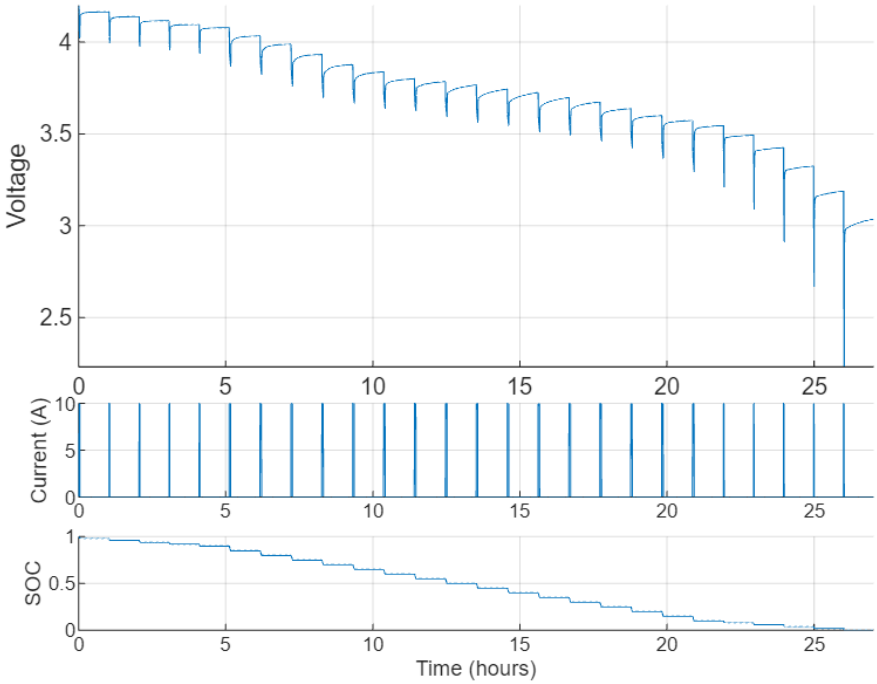



Figure 4.1: Voltage response under pulse discharge test.

From the voltage response shown in Figure 4.1, the dynamic characteristics of the cell can be analyzed. By applying an optimization procedure, the parameterization of the model can be performed. In this project, the parameters $R_0$, $R_1$, $C_1$, and OCV are all considered SOC-dependent and are stored as lookup tables. Equation 4.1 models the voltage response for the single Thevenin configuration.

$$V = \text{OCV} - i \cdot R_0 - \frac{dV_{C1}}{dz} \cdot \tau_1 \tag{4.1}$$

By running an optimization algorithm on Equation 4.1 to fit the experimental data, parameter values for each component are extracted. The result is a set of parameters—OCV(SOC), $R_0$, $R_1$, and $C_1$—as shown in Figure 4.2.

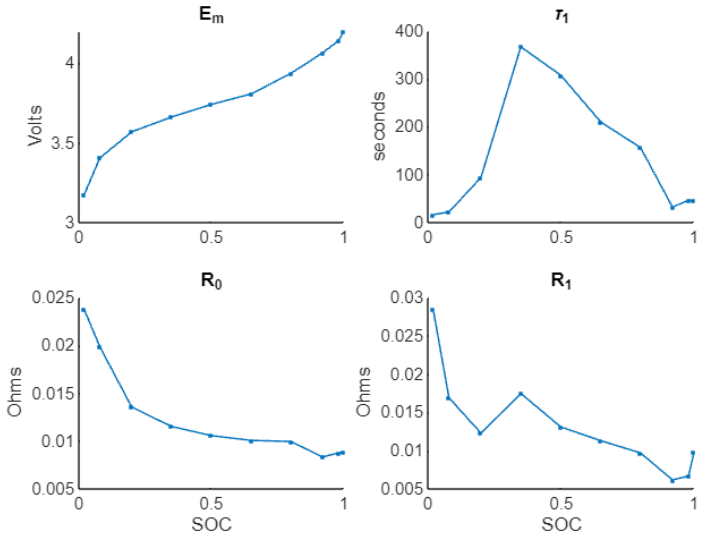



Figure 4.2: OCV and ECM parameters as functions of SOC.

These ECM parameters characterize the tested physical cell. All parameters are implemented as SOC-dependent using linear interpolation between sampled points (by Example). This strategy enables a more representative modeling of cell behavior while still allowing aging and unbalance effects to be applied through parameter derating.

### 4.2.2. Aging and Unbalance

To approximate the behavior of second-life cells, aging is represented by introducing parameter variation in each of the four cells in the pack. These derated values reflect typical degradation patterns observed in aged lithium-ion batteries: an increase in internal resistance, a decrease in polarization capacitance, and a reduction in capacity. Refer to Section 2.4.3 for more details. The derating parameters used for implementation are summarized in Table 4.1.

Table 4.1: ECM derating parameters

| **Parameter** | $R_0$ | $R_1$ | $C_1$ | $Q$ **[Ah]** |
|---|---|---|---|---|
| **Value** | +50% | +50% | -20% | -20% |

In addition to aging effects, unbalance is introduced across the cells to emulate realistic cell-to-cell variations. The unbalance parameters summarized in Table 4.2 define the relative percentage deviations applied to each cell with respect to the uniformly aged baseline. The capacity ($Q$) and polarization capacitance ($C_1$) are symmetrically varied across the four cells, while the resistances ($R_0$, $R_1$) are symmetrically increased and decreased. This configuration enables a controlled and systematic distribution of inhomogeneities.

Table 4.2: Unbalance parameters applied to each cell

| Parameter | Cell 1 | Cell 2 | Cell 3 | Cell 4 |
|---|---|---|---|---|
| $R_0$ (%) | +5% | +1.67% | -1.67% | -5% |
| $R_1$ (%) | +5% | +1.67% | -1.67% | -5% |
| $C_1$ (%) | -3% | -1% | +1% | +3% |
| $Q$ (%) | -3% | -1% | +1% | +3% |

This systematic setup ensures a controlled and repeatable evaluation of the selected balancing strategies. By applying uniform deviations, the performance of each balancing approach can be analyzed under realistic aging and unbalance conditions. Both aging and unbalance factors are applied to the SOC-dependent parameter curves shown in Figure 4.2.

## 4.3. Balancing Module Parameters

In the implementation of the balancing system, four distinct methodologies were incorporated, each adapted to allow for control and monitoring of the cells. These configurations were selected based on their documented use in second-life applications, their suitability for a four-cell pack, and the feasibility of modeling them in Modelica (Khan et al., 2024), (Uzair et al., 2021).

As discussed in the modeling section, all switching elements within the balancing modules are implemented using MOSFET components rather than ideal logical switches. This design choice serves two main purposes. First, modeling with MOSFETs captures non-ideal effects such as finite on-resistance, gate delays, and switching losses—factors that are relevant for estimating both power losses and balancing effectiveness. Second, using component-based switches avoids the creation of algebraic loops and numerical stiffness, which often pose challenges to solvers in Modelica environments. These advantages make MOSFET-based switching a technically appropriate and practical choice for the simulation framework.

### Resistor sizing

In passive balancing systems, an important part of the implementation is the resistor sizing. Modules with lower resistance allow for a higher balancing speed, at the drawback of more heat generation. According to (Szczepaniak et al., 2022), passive balancing currents for EVs typically range between 50 mA and 300 mA. Given that this project targets a second-life application, a current value in the middle of the range, 150 mA, is selected. This avoids fast balancing with excessive dissipation but ensures a reasonable response time to accommodate the higher balancing needs characteristic of aged cells.

The value of the balancing resistor can be calculated using the following formula:

$$R_{\text{bal}} = \frac{V_{\text{cell}}}{I_{\text{bal}}} \tag{4.2}$$

where:

- $R_{\text{bal}}$ is the balancing resistor value (Ω),
- $V_{\text{cell}}$ is the nominal cell voltage (V),
- $I_{\text{bal}}$ is the desired balancing current (A).

Considering the nominal cell voltage of 4.2 V and a target balancing current of 150 mA, the resulting resistor value is approximately 28 Ω. A commercially available value of 27 Ω or 30 Ω could be selected. Continuing with the conservative approach, a resistor of 30 Ω will be selected and applied for both passive balancing methods. It could be argued that a switched resistor topology allows for a higher balancing current since dissipation is selective; however, for consistency and to facilitate a comparative study, the same balancing conditions are maintained across all methods.

### Capacitor sizing

The sizing of the capacitor has an inverse dependence on the maximum voltage difference expected and the desired balancing current. Additionally, the switching frequency directly influences the sizing. The formula for the sizing of the balancing capacitor is given by (Cao et al., 2020):

$$C = \frac{2 \cdot E}{(\Delta V_{\text{cap}})^2} \tag{4.3}$$

The energy transferred per switching cycle is:

$$E = \frac{P}{f_{\text{sw}}} \tag{4.4}$$

Balancing power can be approximated as:

$$P = \Delta V \cdot I_{\text{bal}} \tag{4.5}$$

where:

- $C$ is the required capacitance (F),
- $E$ is the energy transferred per switching event (J),
- $\Delta V_{\text{cap}}$ is the allowed voltage ripple across the capacitor (V),
- $P$ is the balancing power (W),
- $f_{\text{sw}}$ is the switching frequency (Hz),
- $\Delta V$ is the maximum voltage difference between cells (V),

- $I_{\text{bal}}$ is the target balancing current (A).

It is important to note that the energy calculated per cycle assumes ideal instantaneous charging and discharging phases. In more accurate modeling, considering the current waveform as triangular due to the switching behavior, the effective energy transferred per cycle can be corrected by a factor of one-half. In this case, the energy per cycle is given by:

$$E = \frac{1}{2} \cdot \frac{P}{f_{\text{sw}}} \tag{4.6}$$

This correction accounts for the fact that energy transfer occurs during two distinct phases (charging and discharging) within each full switching cycle and that the average current over each phase is lower than the peak balancing current. For conservative sizing, the uncorrected formula is typically employed, while the corrected version can be used for optimized designs (Makowski and Pedram, 2016; Texas Instruments, 2015). Following this more realistic approach, the average balancing current is considered to be half of the peak balancing current, according to Formula 4.6.

The switching frequency selected for the balancing circuit is 50 kHz, within the typical industry range of 30 kHz to 150 kHz (Texas Instruments, 2015). Selecting 50 kHz offers a good compromise between reducing the size of passive components and minimizing switching losses. The target balancing current is set at 1 A, which falls within the industry standard range of 0.5 A to 2.0 A (Szczepaniak et al., 2022; Texas Instruments, 2015). A balancing current of 1 A ensures sufficiently fast redistribution of energy among cells without causing significant thermal or electrical stress on the system components. As mentioned, the nominal OCV of the cell is 4.2 V and a maximum voltage ripple between cells of 0.2 V (0.1 V across the capacitor). Altogether, this gives a result of 800 µF, selecting a 820 µF commercially available.

Overall, these parameter selections ensure that the balancing module is efficient, reliable, and aligned with industry best practices. The calculated values for the capacitor and resistor are summarized in Table 4.3. As will be commented on in Chapter 6, these "industry standard" values introduce a lot of complications when simulating with Modelica. Ensuring that the model is physically realistic will be the priority of this project even if that means having to reduce the switching characteristics to make the simulation possible. Solutions to this issue are also explored in Chapters 6 and 7.

### Inductor sizing

Similar to the single capacitor topology, the single inductor configuration operates based on charging and discharging cycles at a determined switching frequency. The sizing of the inductor is directly dependent on the duty cycle, the switching frequency, the maximum expected voltage difference between cells, and the maximum balancing current allowed. As shown in (Tudorache et al., 2017), the required inductance can be calculated according to:

Table 4.3: Balancing Topologies and Assigned Parameters.

| **Balancing Topology** | **Assigned Parameters** |
|---|---|
| Shunt (resistor only) | $R = 30\,\Omega$ |
| Switched resistor | $R = 30\,\Omega$, controlled by MOSFET |
| Switched capacitor | $C = 820\,\mu\text{F}$, switching at 50 kHz |
| Inductor-based transfer | $L = 2.7\,\mu\text{H}$, switching at 50 kHz |

$$L = \frac{\Delta V \cdot D \cdot (1 - D)}{f_{\text{sw}} \cdot \Delta I} \tag{4.7}$$

where:

- $L$ is the required inductance (H),
- $\Delta V$ is the voltage difference between the two cells (V),
- $D$ is the duty cycle (dimensionless, typically around 0.5),
- $f_{\text{sw}}$ is the switching frequency (Hz),
- $\Delta I$ is the allowed peak-to-peak current ripple (A).

The same assumptions as for the capacitor sizing are taken: a fixed duty cycle of 0.5 and a switching frequency of 50 kHz have been selected. These values fall within the typical industry range of 30 kHz to 150 kHz (Texas Instruments, 2015). By selecting the same values as in Section 4.3, it will be possible to draw more conclusions on the results with each technology. For the same reason, a balancing current of 1 A has been selected, consistent with the values defined in the capacitor sizing section and within the standard range for industrial active balancing applications (Szczepaniak et al., 2022). As with the capacitor, a voltage ripple of 0.2 V and a current ripple of 0.4 A (40% of the balancing current) are allowed. Knowing that the OCV of the cell is 4.2 V, the inductance required is around 2.5 µH, selecting a commercially available inductor of 2.7 µH.

Overall, the inductor sizing methodology ensures consistency across the active balancing strategies developed in this work, and the selected parameters fall within realistic values for industry standards. The same reflection made for the capacitor balancing applies here: the industry-standard switching times and inductance values are hard to handle in Modelica in a straightforward way. Thus, the "correct" physical behavior will be prioritized even if the switching frequencies and inductance levels need to be set higher.

### Sizing results

With the assumptions and formulas introduced in the sections above, the assigned parameters for each topology are calculated and listed in Table 4.3.

All balancing modules interact with a control module that evaluates cell state parameters (voltage, SoC) and activates the balancing logic based on predefined conditions. In Section 4.5, the signal generation and control methodology are addressed in detail.

## 4.4. Charge/Load module

The charging and discharging cycles of the second-life lithium-ion cell pack are governed through a dedicated control unit utilizing a constant current (CC) strategy. As presented in Section 3.3.5, this simplified approach derives from the conventional Constant Current–Constant Voltage (CCCV) method, reflecting a deliberate choice to reduce modeling complexity. This implementation is justified by the relatively moderate energy throughput and absence of high-precision SoC calibration requirements typical in second-life evaluations.

In the charging phase, a 2 C rate—equivalent to 2 A, given the nominal pre-derated capacity of 3600 C—is applied across the cell pack. This elevated rate enables rapid energy replenishment while remaining within the safe operating limits of the cell. To mitigate risks associated with overvoltage and reduce the likelihood of accelerated degradation phenomena such as lithium plating, the charging process is terminated once any individual cell reaches 4.2 V. To enhance robustness, a safety margin below this limit is introduced, creating a voltage guard band that ensures operational safety (Perişoară et al., 2018; Khan et al., 2024).

Discharging is carried out at a constant current of 0.8 A, corresponding to approximately 0.8 C. This conservative discharge rate is suitable for second-life evaluations (He and Chen, 2023). The discharging process is interrupted when the terminal voltage of the lowest cell reaches 3.5 V. This voltage threshold incorporates a protective margin, as with the maximum voltage, protecting the cells against over-discharge.

The Modelica-based implementation of this logic integrates the CC-only charge and discharge operations directly into the simulation framework. Unlike CCCV protocols, no transition to voltage-hold mode is implemented. Instead, the system operates purely in current-controlled modes, with transitions governed by cell-level voltage comparisons to the defined upper and lower bounds. This approach is consistent with the reduced complexity required for experimental analysis and simulation of passive balancing strategies. The overall charge/load module interacts dynamically with the control logic outlined in Section 5.8.

## 4.5. Balancing Algorithm

The balancing algorithm serves as the monitoring system and the direct actuator of the balancing process on the cell. Its primary objective is to reduce inter-cell voltage deviations, thereby improving the effective capacity and safety of the pack, particularly in second-life configurations. The algorithm monitors the instantaneous voltage of each cell and makes decisions based on predefined threshold conditions.

For passive topologies, the balancing is done during the charging phase, consistent with standard battery management practices. This approach ensures that excess energy from higher-voltage cells is redistributed or dissipated without risking deep discharge or additional degradation during other phases. Discharge and idle periods are excluded from balancing to avoid thermal or efficiency penalties (Khan et al., 2024), (Uzair et al., 2021).

For active methods, the balancing will be done continuously if needed. That means, during charge, discharge, and idle. In the case of active balancing systems, efficiency is not achieved through dissipation, thus balancing during discharge periods can actually improve the effective capacity of the battery by passing charge from less aged cells to more degraded cells. Since dissipation is almost not present (besides that induced in the MOSFET logic), balancing in any stage does not impact the efficiency of the balancing itself differently.

The control logic continuously compares the voltage of each cell against the average pack voltage. A balancing control signal is triggered when the deviation between a cell's voltage and the mean exceeds a 20 mV threshold, a commonly adopted limit that balances convergence speed and avoids noise (Chavan et al., 2025). Once triggered, the specific balancing method (resistive or active) is applied. In the case of passive methods, energy is dissipated through controlled resistors, while in active approaches, energy is redistributed. For active methods, the selection of which cells to balance is determined by the highest deviation in magnitudes (cells with the highest and lowest voltages).

Both for active and passive methods, the algorithm generally follows a cyclic structure:

- Read cell values.
- Identify the cell with the lowest voltage.
- Compare voltages from other cells and evaluate the need for balancing.
- Activate balancing in all cells that need it (passive); activate balancing paths sequentially (active).
- Wait and start again.

This logic cycle repeats continuously under operation. The control unit continuously updates the status of all four cells and issues binary activation signals to each balancing module accordingly. For the active methods, the control module additionally includes a PWM signal to regulate the switching frequency between charging/discharging, for this project set to 50 kHz as explained in Section 4.3.

For clarity, it is recommended for the reader to open and review the control unit modules. These visual representations formalize the operational logic and facilitate validation and future extensions. The values and thresholds influencing the control can be accessed in the code or directly from models graphic views.

## 4.6. Conclusion

This chapter presented the technical implementation of the battery model and its balancing systems. The parametrization process for the ECM was outlined, including a realistic setup of aged and unbalanced cells to simulate second-life battery behavior. Component values for the balancing modules were sized according to standard equations and adjusted to reflect practical and industry-relevant conditions.

The control logic and operating limits for both passive and active balancing strategies were also introduced. Special attention was given to maintaining consistent testing conditions across all methods to allow for a fair comparison in the following validation and results chapter. The decisions made here—on parameter values, control structure, and operating modes—establish the foundation for the performance evaluation and model validation that follow.

# Chapter 5

# Battery_Balancing Library Architecture

## 5.1. Introduction

This chapter presents the internal architecture and design methodology of the custom Modelica library developed for modeling and simulating second-life lithium-ion battery packs with various balancing strategies. The development follows a modular and hierarchical approach, characteristic of object-oriented modeling, where components are defined in terms of reusable classes.

The library is organized into eight primary packages: Interfaces, Components, Functions, ECM_Structures, Cell_Packs, Balancing_Structures, Control_Structures, and Examples. Each package encapsulates a specific aspect of the modeling framework, facilitating transparency and scalability.

Figure 5.1 illustrates the structural hierarchy of the library. This visual representation serves both as a topological map of the library. The architecture supports key modeling requirements outlined in earlier chapters, including the representation of equivalent circuit models (ECMs), the modular assembly of cell packs, and the implementation of diverse balancing topologies.

The library design is grounded in the use of partial classes and connector interfaces to enforce consistent integration rules and reduce redundancy. For example, all ECM-based components inherit from a base model that includes standardized positive and negative electrical pins. This ensures uniformity across models and simplifies replacement or extension, a critical feature when evaluating the performance of balancing methods under varying second-life degradation scenarios.

Furthermore, the use of structured test cases in the Examples package provides a reproducible environment. These simulations employ parameter sets and controlled current/voltage profiles to compare the behavior of different balancing algorithms in ideal and degraded cell configurations.

In subsequent sections, each package is presented in detail, highlighting its purpose,

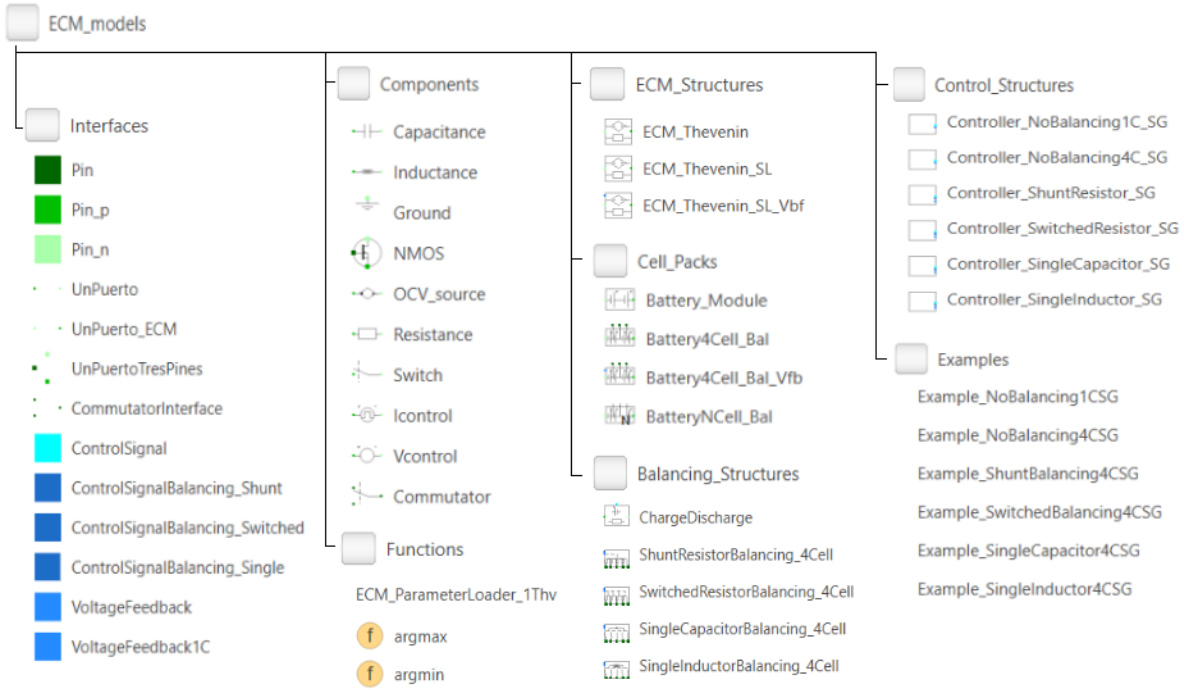



Figure 5.1: Diagram of library architecture, including packages and classes.

internal organization, and contribution to the overall simulation environment.

## 5.2. Package Interfaces

This package defines the component structures necessary to describe and connect different elements within the library. By using a set of well-defined connectors and partial base models, the architecture supports the modular design approach characteristic of Modelica and ensures proper connection between elements.

The interfaces included in this package establish the fundamental contract between electrical components, controllers, and auxiliary modules. For example, the use of standardized Pin_p and Pin_n connectors ensures a uniform definition of current and voltage directionality, which is essential for correctly assembling ECM structures and balancing networks. The following models are included in this package:

- Pin. Standard structure for neutral electrical port.
- Pin_p. Standard structure for a positive electrical port.
- Pin_n. Standard structure for a negative electrical port.
- UnPuerto. Partial model for two-port electrical components using one positive and one negative pin.

- UnPuerto_ECM. Partial model specifically structured for ECM models using one positive and one negative pin.
- UnPuertoTresPines. Partial model includes an additional signal pin.
- CommutatorInterface. Partial model structured for commutator model.
- ControlSignal. Connector to transfer non-physical control signal for communication between the control and the charge/discharge module.
- ControlSignalBalancing_Shunt. Connector to transfer non-physical control signal for shunt resistor balancing.
- ControlSignalBalancing_Switched. Connector to transfer non-physical control signal in switched resistor topology.
- ControlSignalBalancing_Single. Connector to transfer non-physical control signal in single component (capacitor or inductor) balancing topologies.
- VoltageFeedback. Connector to transfer voltage feedback for control purposes.
- VoltageFeedback1C. Specialization of VoltageFeedback for one-cell configurations.

The abstract and reusable nature of these interface definitions allows different packages in the library to inherit common behavior and integrate between them.

## 5.3. Package Components

This package contains the necessary electrical components used for the modeling of the ECM structures included in Section 5.5, as well as other standard electrical components used for building basic models and performing tests.

The models in this package serve as foundational elements in the construction of the rest of the library. Several components, such as switches and controlled sources, are equipped with control signal inputs, making them directly compatible with the interface structures defined in the Interfaces package. The following models are included in this package:

- Capacitance. Ideal capacitor model using a constant value.
- Inductance. Ideal inductor model.
- Ground. Electrical ground reference point.
- NMOS. Simplified model of an NMOS transistor used in switching configurations.

- OCV_Source_SOC. Voltage source dependent on SOC, representing the open circuit voltage characteristic.
- Resistance. Ideal resistor component.
- Switch. Controlled switch with internal resistance.
- Icontrol. Current source defined by time-varying input profile, used in pulse discharge simulations.
- Vcontrol. Voltage control source defined by external input profile.
- Commutator. Switch-matrix or multiplexer structure to route signals between two entries to base.

These components collectively support the creation of test benches, ECMs, and complete battery packs with integrated balancing strategies. In its core, this library could have been substituted by a standard Modelica library. The action of creating this library from scratch has the intent to have control over the whole library creation process.

## 5.4. Package Functions

This package contains auxiliary functions used in more complex structures such as balancing and control modules. These functions encapsulate computational logic and data-handling operations that are reused across various packages. This approach also facilitates rapid updates to mathematical routines without altering the structural composition of higher-level models. The following functions are included in this package:

- ECM_ParameterLoader_1Thv. Loads pre-calculated ECM parameters from file for the single Thevenin branch model.
- argmax. Returns the index of the maximum element in a vector.
- argmin. Returns the index of the minimum element in a vector.

These functions are typically used during model initialization or within control blocks. The parameter loader, in particular, enables the application of externally charged ECM data for second-life battery configurations.

## 5.5. Package ECM_structures

This package contains various structures of equivalent circuit models (ECMs). These models are used to simulate the electrical behavior of lithium-ion cells under different conditions. Each ECM structure has a different purpose and allows for its use in different cases depending on the needs of the user. Furthermore, the parameterization of resistive and capacitive elements allows to represent both nominal and degraded second-life behavior. The following models are included in this package:

- ECM_Thevenin. Standard single-branch Thevenin ECM model including resistance and RC network.
- ECM_Thevenin_SL. Extended ECM_Thevenin for second-life applications with individual parameter loading.
- ECM_Thevenin_SL_Vbf. Variant of the second-life Thevenin model including direct voltage feedback to the control.

Each of these variants supports integration with higher-level cell and pack models, allowing for the possibility to test on different scenarios. This includes: uniform degradation, cell-to-cell variability, or control feedback for balancing strategies.

## 5.6. Package Cell_Packs

This package contains configurations of 4-cell series-connected ECM packs. These are used to evaluate performance in balancing and degradation conditions. These models are designed to be compatible with the control and balancing modules defined in subsequent packages, ensuring consistent simulation scenarios across different balancing methods. The following models are included in this package:

- Battery_Module.N-Cell pack connected in series.
- Battery4Cell_Bal. 4-Cell pack with intermediate electrical nodes between cells for balancing.
- Battery4Cell_Bal_Vbf.4-Cell pack with intermediate electrical nodes between cells for balancing and voltage feedback.
- BatteryNCell_Bal. N-Cell pack with intermediate electrical nodes between cells for balancing.

These models serve both as experimental setups for case studies and as reusable structures. The setups selected are based on the needs observed in scientific literature, where testing balancing on 4 cells in series is the most common case.

## 5.7. Package Balancing_Structures

This package contains the models of the Balancing model structures described in Section 2.5. As introduced previously, the balancing is going to be performed in packs of 4 cells. Each structure in this package implements a distinct balancing methodology, including both passive and active techniques.

All balancing models are built using modular components. The 4-cell configuration enables focused study on cell dynamics while maintaining manageable model complexity for simulation and analysis. The following models are included in this package:

- ChargeDischarge. Core module managing pack-level charge and discharge boundary conditions.
- ShuntResistorBalancing_4Cell. Applies passive resistive balancing by shunting excess energy under the charging cycle.
- SwitchedResistorBalancing_4Cell. Applies switching control to limit losses in individual resistor-based balancing.
- SingleCapacitorBalancing_4Cell. Active balancing using charge redistribution through a single capacitor.
- SingleInductorBalancing_4Cell. Active balancing using charge redistribution through a single inductor.

## 5.8. Package Control_Structures

This package includes the controller blocks that regulate charging, discharging, and balancing decisions. The controllers are designed using a signal-based architecture compatible with the connector definitions in the `Interfaces` package. They receive voltage feedback from the pack and issue control signals to the switching components. Each controller is parameterized for a specific balancing topology.

Internally, the models apply basic Grafcet logic characterized by steps and transitions. While the control logic is relatively simple, it enables fast simulation and modular integration with balancing structures. The following models are included in this package:

- Controller_NoBalancing1C_SG. Controller for one-cell pack with no balancing.
- Controller_NoBalancing4C_SG. Controller for 4-cell pack with no balancing.
- Controller_ShuntResistor_SG. Controller for 4-cell pack with shunt balancing resistor under charging.

- Controller_SwitchedResistor_SG. Controller for 4-cell pack with individual switched resistor balancing.

- Controller_SingleCapacitor_SG. Controller for 4-cell pack with active single capacitor balancing.

- Controller_SingleInductor_SG. Controller for 4-cell pack with active single inductor balancing.

The structure of these controllers is tailored 1-to-1 to each balancing methodology. Although the modularity of the balancing and control modules allows them to be replaced or extended with more advanced logic, such as PID-based or reinforcement learning controllers, without altering the core balancing or cell pack models.

## 5.9. Package Examples

This package contains test examples demonstrating the performance of different balancing strategies under a defined charge/discharge routine. Each example assembles components from the library—ECM cells, balancing structures, control units, and boundary conditions—into a full simulation system. The following models are included in this package:

- Example_NoBalancing1CSG. One-cell system without balancing, used for model validation.

- Example_NoBalancing4CSG. Four-cell system without balancing.

- Example_ShuntBalancing4CSG. Four-cell passive shunt balancing test case.

- Example_SwitchedBalancing4CSG. Four-cell switched balancing test case.

- Example_SingleCapacitor4CSG. Four cell active balancing using a single capacitor test case.

- Example_SingleInductor4CSG. Four cell active balancing using an inductor test case.

These examples are critical for validating the functionality of the library and demonstrating its applicability to second-life battery analysis.

## 5.10. Conclusion

This chapter presented the structure and contents of the custom Modelica library developed for the implementation and testing of the proposed battery balancing strategies. The library is organized in a modular way, with each package addressing a specific aspect of the modeling task—from defining basic connectors and components to implementing ECM structures, control units, and full test examples.

This modular approach supports easy reusability, parameter adjustment, and further extension of the models. The architecture is also designed to ensure clarity and consistency between simulations, enabling a structured workflow from individual components to system-level behavior. The library forms the backbone of the simulations and serves as a reusable toolset for further experimentation or future projects.

# Chapter 6

# Model Validation and Results

## 6.1. Introduction

In this section, the validation of the developed battery library is presented based on the results obtained through a series of simulations. The primary objective is to assess whether the behavior of the models aligns with expected results, both from a physical and numerical standpoint. Different test cases are executed to evaluate individual cell behavior, aging effects, and the performance of various balancing strategies.

Additionally, practical challenges that arose during the simulations are discussed, including issues related to computational complexity and switching dynamics. These insights provide context for the strengths and limitations of the implemented library and inform suggestions for future improvements.

## 6.2. Single Cell Validation

The first validation step focuses on the behavior of a single cell, modeled using the Thevenin equivalent structure described previously. To evaluate the behavior of the model, the main aspect to analyze is the voltage evolution with regard to the SOC of the cell. Figure 6.1 plots the voltage and SOC of one cell over a charge/discharge cycle.

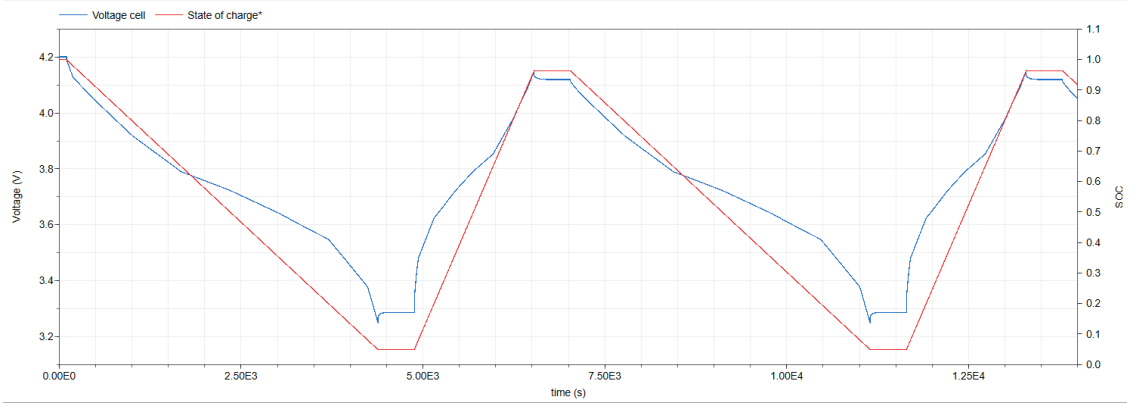



Figure 6.1: Voltage evolution of one cell under charge/discharge cycle.

As shown in Figure 6.1, cell voltage follows a reasonable response for the given discharge current. There is a reduction of the voltage at the terminals that is accentuated close to full charge and full discharge states, as expected. A clear stabilization effect is observed during idle periods following charging and discharging cycles, which reflects the influence of the RC time constant present in the single-cell Thevenin model. This relaxation confirms the suitability of the model in capturing transient behavior. Overall, the single-cell implementation behaves consistently with the expected electrochemical characteristics and provides a valid basis for system-level modeling. Figure 6.3 shows that the evolution of parameters with respect to SOC behaves correctly in relation to the parametrization data shown in Figure 6.2.

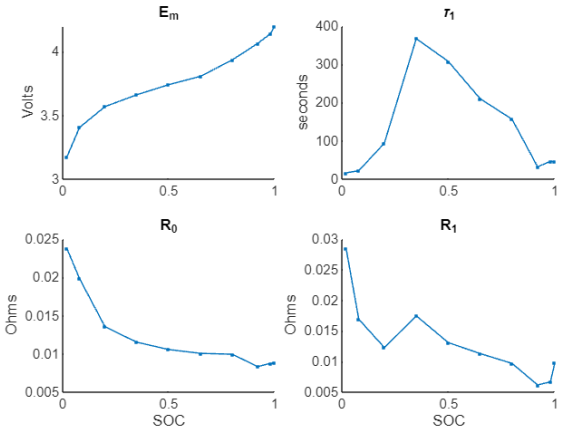



Figure 6.2: SOC-OCV curve obtained from MATLAB reference data.

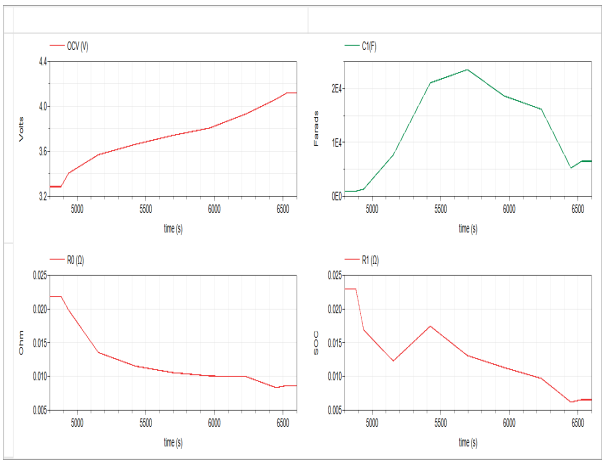



Figure 6.3: SOC-OCV response from Modelica single cell simulation.

As shown in Figures 6.2 and 6.3, the model behavior with respect to SOC appears correct and confirms a good basis for the rest of the project. The slight differences between the curves are due to the safety voltage margins implemented—the curve does not reach 0 or 1 SOC but stays within a safety margin.

## 6.3. Cell Pack Derated Validation

To evaluate the behavior of a second-life pack, a system of four cells was simulated under charge and discharge conditions without any balancing mechanism. This allows investigation of the inherent cell-to-cell variations and their impact on pack behavior.

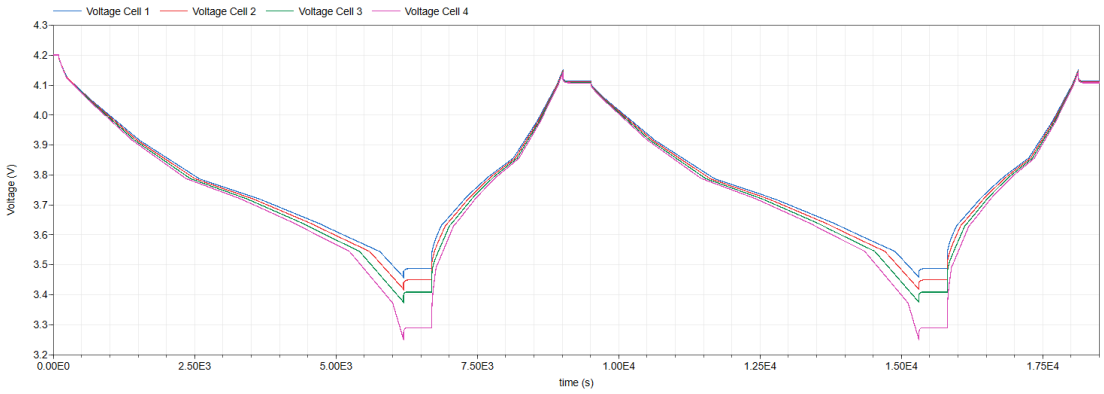



Figure 6.4: Voltage profiles of four derated cells during charge/discharge cycle.

Figure 6.4 illustrates the voltage evolution of the four cells. Each cell has a different degree of capacity derating, with Cell 1 being the least and Cell 4 the most affected, as presented in Table 4.2. As a result, the voltage of Cell 4 rises and falls more rapidly compared to the others under load, since the cell is more degraded. This is a direct consequence of its lower effective capacity. The observed behavior validates the capacity scaling implemented in the model and confirms that the pack representation correctly reflects second-life aging effects.

## 6.4. Passive Balancing Methods

Building on the previously validated cell pack, passive balancing methods were introduced to study their effect on mitigating voltage disparities. Two passive balancing topologies were tested: a simple shunt resistor and a switched resistor configuration.

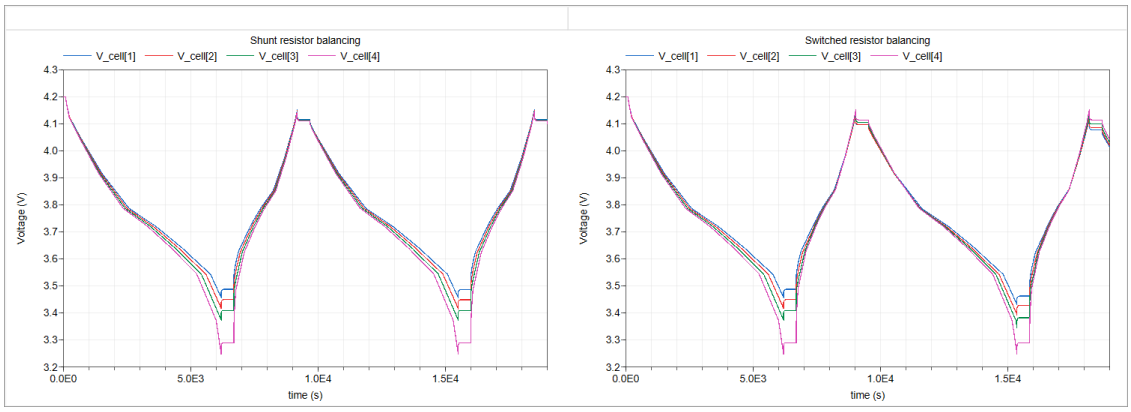



Figure 6.5: Voltage profiles of four-cell charge/discharge cycle with Shunt (Left) and Switched (Right) balancing.

Figure 6.5 illustrates that both balancing methods effectively reduce the voltage spread among the cells during the charging phase. A minor difference in charging duration is observed, with the shunt-balanced system requiring slightly more time to reach full charge. Moreover, the switched balancing method demonstrates faster and more selective balancing behavior, although this distinction is not immediately evident from the voltage curves alone.

Notably, after the charging phase during the idle period, the shunt-balancing strategy appears to achieve a more uniform voltage distribution across the cells. This outcome can be attributed to the continuous operation of the shunt mechanism throughout the charging process. In contrast, the switched balancing method stops the balancing sequence once the voltage difference between cells falls below a predefined threshold. As a result, a residual voltage disparity may remain, depending on the magnitude of the set threshold.

To gain further insight into the activation dynamics and balancing intensity of both methods, the individual cell balancing currents were monitored and are presented in Figure 6.6.

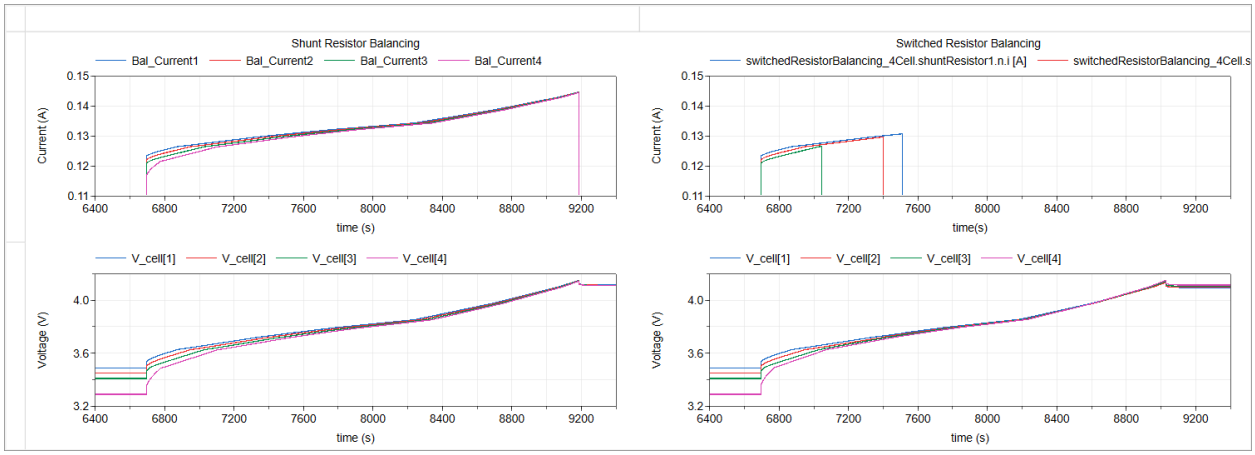



Figure 6.6: Balancing currents and voltage profiles under full charge with Shunt (Left) and Switched (Right) balancing.

From Figure 6.6, it is evident that the switched resistor topology results in more selective and lower amplitude balancing currents, thereby reducing power dissipation. However, this configuration also demands more complex control logic. During simulation, some issues with control signal chattering were observed, which led to the implementation of hysteresis or time delay elements. A further point of interest is the comparison between the expected theoretical balancing current (based on resistance value) and the actual simulated values, which match within an acceptable tolerance. This confirms both the physical consistency of the model and the correctness of its electrical implementation.

## 6.5. Active Balancing Methods

To address the power losses inherent in passive balancing, active methods were introduced, starting with a single capacitor topology. While the physical behavior aligns with theoretical expectations, challenges were encountered when operating under high-frequency switching conditions. Results for both capacitor and inductor balancing have been similar, and the conclusions are the same for both systems. Thus, for clarity, some figures will show results for only one of them. Refer to Appendix A to execute and analyze the full results. Figure 6.7 shows a charge-discharge cycle for the capacitor balancing structure.

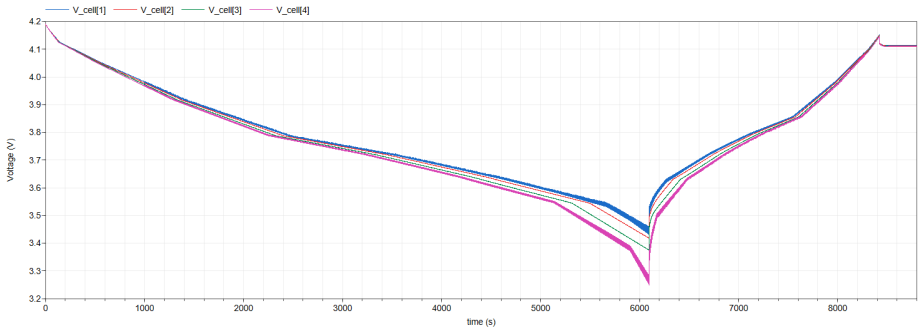



Figure 6.7: Voltage profile of the balancing capacitor during a transfer cycle.

Figure 6.7 illustrates the cell voltages under a charge-discharge cycle. Since the simulation became computationally intensive when attempting to reach realistic industrial switching frequencies, a capacitance of 10e-3 F and a frequency of 25 Hz have been used to successfully run the simulations. This limitation led to suboptimal balancing performance due to an insufficient balancing current. To ensure that the model behavior aligns with physical reality, Figure 6.8 shows a close-up view with high step resolution of the balancing capacitor voltage and current when balancing is active.

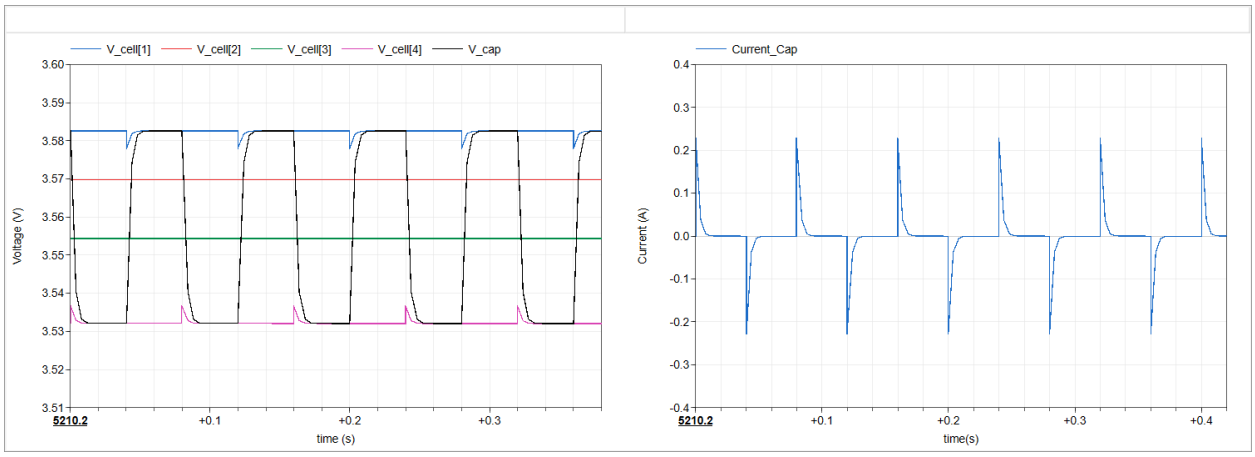



Figure 6.8: Voltage (Left) and current profile (Right) of the balancing capacitor.

The left panel of Figure 6.8 shows the voltage evolution of the capacitor while balancing. Each period when a switch flip is performed, the capacitor undergoes a clear change in voltage—charging when connected to the highest-voltage cell and discharging when the lowest-voltage cell is connected instead. This behavior aligns with physical expectations. On the other hand, the right panel reveals that due to the high capacitance and relatively low switching frequency (relative to industry standards), the balancing currents were considerably lower than required for effective cell balancing.

A similar behavior is observed when repeating the exercise with the inductor balancing example. Figure 6.9 shows a close-up view with high step resolution of the balancing inductor voltage and current when balancing is active. In this case, an inductance of 0.04 H and a frequency of 25 Hz were used.

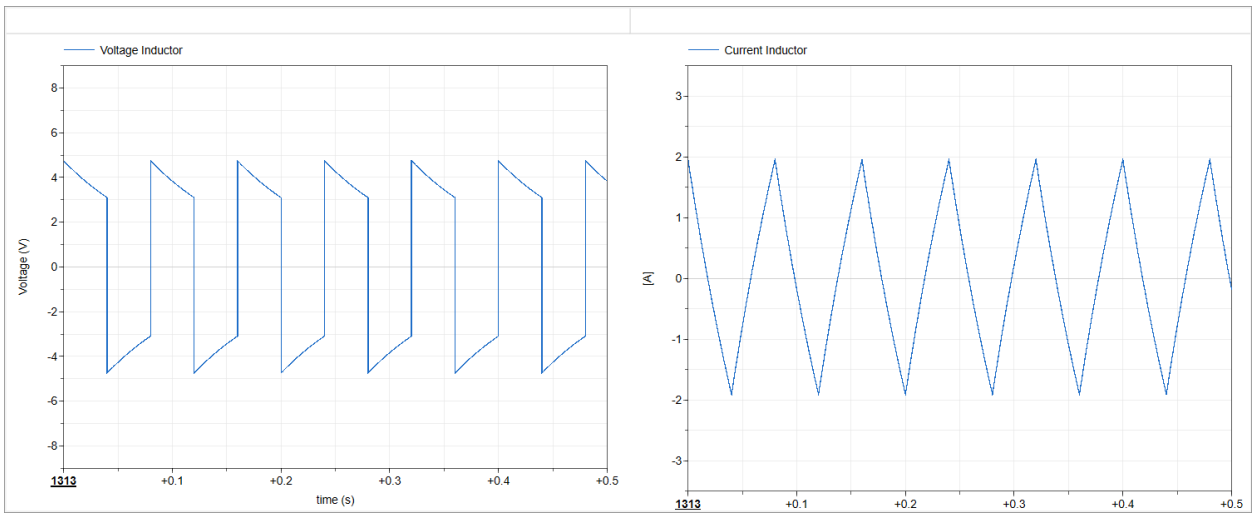



Figure 6.9: Voltage (Left) and current profile (Right) of the balancing inductor.

Figure 6.9 confirms similar behavior to that described for the capacitor case. The left panel shows a reasonable voltage evolution of the inductor while balancing. The voltage

jumps from one cell connection polarity to the opposite every period, aligning with expected physical behavior. Contrary to the capacitor case, the inductor allows for significant balancing currents even at lower switching frequencies. Nonetheless, in real applications, the switching frequency would be considerably higher, and the balancing inductance much lower.

In both cases, this confirms the conceptual validity of the model, even if execution was constrained by simulation performance. To overcome the computational burden and achieve results that better reflect industrial operation, several strategies can be implemented:

- Divide the total simulation time into smaller time slices using the `simulate->continue` function in Modelica.
- Reduce the number of stored variables and output parameters to decrease memory usage.
- Introduce simplified switch models or equivalent average models for high-frequency switching behavior.
- Apply fixed-step solvers with coarser granularity for long transients, combined with high-resolution submodels.

Although the balancing model is physically accurate, a dedicated simulation strategy is required to validate it under industrial operating frequencies and constraints.

As a final comment, it is relevant to reflect on how the control system executes different switch pattern flips and how this directly translates into the physical behavior of the system. Figure 6.10 shows the evolution of the five switch signals and two commutators, together with the evolution of the capacitor voltage.

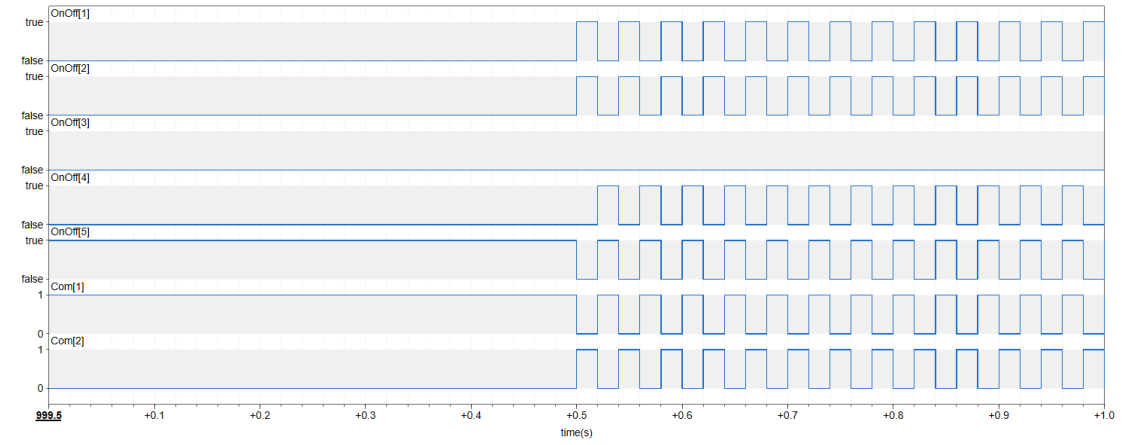



Figure 6.10: Switch patterns and physical behavior.

In this case, cells 1 and 4 are engaged in the balancing process. Cell 1 requires switches 1, 2 and commutator 2 to be active for engagement. In contrast, Cell 4 requires switches 4, 5

and commutator 1. Figure 6.10 shows how each period alternates the switch pattern between that for Cell 1 and Cell 4, engaging one at a time. This is directly reflected in the physical behavior of the capacitor voltage: when the pattern for Cell 1 is active, the capacitor charges toward the higher voltage; when the pattern for Cell 4 is active, it discharges toward the lower voltage.

## 6.6. Conclusions

The simulation results confirm that the developed battery models and control strategies reflect realistic cell- and pack-level behavior. The single-cell model accurately reproduces SOC-OCV dynamics, and the derated pack demonstrates correct voltage divergence without balancing. Passive methods were effective in reducing imbalance, with switched topologies offering better efficiency at the cost of increased control complexity. Active balancing methods showed correct physical operation but revealed computational limitations when simulating high-frequency switching events.

Overall, the library is validated as functionally correct. However, further work is required to address numerical challenges such as chattering and to develop simulation methodologies that enable high-fidelity execution of active balancing under realistic operational conditions.

# Chapter 7

# Conclusions

## 7.1. Conclusions

This section summarizes the results obtained from the development, implementation, and testing of the battery modeling and balancing library. Overall, the library successfully fulfills its intended purpose. The base components were correctly implemented and enabled the graphical construction of more complex battery systems. The use of the graphical Modelica environment proved advantageous in building and visualizing structures based on reusable elements.

The implemented ECM models have shown satisfactory performance in validation tests. The single-cell model reproduces the SOC-OCV characteristics with good accuracy, including transient effects influenced by RC time constants. These models offer sufficient precision for simulation tasks and decision-making processes related to cell balancing strategies. Should a higher-fidelity or chemistry-specific cell model be required in future applications, the modular structure allows for its incorporation and subsequent reuse within the same balancing framework.

With regard to the balancing systems and associated control logic, the implemented methods behaved as expected and supported various configurations of practical and theoretical interest. The flexibility of the structure permits further development and integration of additional balancing techniques. A few important observations can be highlighted from the simulation results:

- **Passive balancing methods:** These methods were effective in reducing voltage imbalances, particularly near full charge, allowing all cells to reach almost full SOC. However, since balancing occurs only during the charging phase, usable capacity at low SOC remains unoptimized, with some cells not fully discharged. Furthermore, the shunt-based approaches suffered from significant energy inefficiency due to resistive dissipation. The switched resistor method improved selectivity and energy use but introduced control-related issues such as signal chattering, which led to the development

of a hysteresis-based control strategy.

- **Active balancing methods:** These demonstrated physically reasonable behavior in transferring energy from high-voltage to low-voltage cells. Nevertheless, complications arose in achieving industrially relevant performance, notably due to fast switching requirements and low-valued capacitive or inductive components. Additionally, insufficient precharging of transfer elements could result in undesired voltage drops, requiring careful adjustment on a case-by-case basis. Memory and computational load during simulations increased significantly with higher switching frequencies, hindering extensive performance validation.
- **Control systems:** Control structures were found to be highly case-dependent. The amount and type of control signals vary for each simulation scenario, limiting the reusability and generalization of controllers. A more adaptive or self-tuning approach could potentially reduce development time for new configurations.
- **Scalability:** While the library performed well for the tested 4-cell configuration, its scalability remains limited. Applying the current structure to larger packs would demand substantial rework in control design, component instancing, and simulation stability, particularly under real-time constraints.

In summary, the developed library is validated as a sound and functional simulation platform for small-scale second-life battery systems. It permits systematic testing of balancing methods and their control under realistic operating conditions. However, limitations such as energy inefficiency, chattering, simulation resource demands, and control adaptability highlight important topics for further improvement and exploration.

## 7.2. Future Work

This project opens a wide range of possibilities for continued investigation, both in simulation and physical experimentation. Several promising directions are outlined below:

- **Experimental validation:** Implement the developed models alongside real lithium-ion cells and experimentally verify their dynamic response. This would include conducting charge-discharge cycling tests with and without balancing mechanisms and comparing measured data with simulated behavior. Special focus could be placed on obtaining SoC-dependent ECM parameters through empirical identification techniques.
- **Aging analysis:** Extend the model to include dynamic aging effects and perform long-term simulations to evaluate how balancing systems influence capacity retention and imbalance development. Understanding how passive and active strategies perform under degradation could guide design choices for second-life applications.

- **Larger-scale modeling:** Expand the library to support configurations with a larger number of cells and/or multiple parallel branches. This would enable simulation of full battery modules and the study of inter-pack balancing approaches (e.g., cell-to-pack or pack-to-pack energy redistribution).
- **Advanced balancing topologies:** With larger systems, more sophisticated balancing architectures become relevant. This includes multi-stage capacitor or inductor networks, transformer-coupled active systems, or hybrid passive-active configurations. Implementing and benchmarking these strategies would deepen understanding of their trade-offs in second-life battery contexts.
- **Optimization and automation:** Develop adaptive control algorithms or machine learning–based decision strategies that can automatically adjust balancing parameters based on pack behavior. This would reduce the manual tuning currently required for each test case and improve the robustness of simulations.
- **Simulation efficiency:** Investigate methods for running high-frequency simulations more efficiently. This could include simulating shorter time slices with continuation, reducing logged variables, simplifying models for high-frequency switching, or employing multi-rate or co-simulation approaches.

These are only a subset of the possibilities that emerged during the course of this project. Both the field of battery balancing and the domain of second-life applications offer fertile ground for research. The approach taken in this work—focusing on physical modeling with practical simulation capability—remains a valuable tool not only for design but also for understanding and predicting the behavior of real battery systems. Whether through further library development or experimental integration, the insights gained here open a broad field of future investigations.

# Bibliography

A. Ashraf, B. Ali, M. S. A. Alsunjury, H. Goren, H. Kilicoglu, F. Hardan, and P. Tricoli. Review of cell-balancing schemes for electric vehicle battery management systems. *Energies*, 17(6):1271, 2024. doi: 10.3390/en17061271.

Modelica Association. Modelica.blocks.tables.combitable1ds documentation. `https://build.openmodelica.org/Documentation/Modelica.Blocks.Tables.CombiTable1Ds.html`, 2024a. Accessed: 2025-06-03.

Modelica Association. Modelica.stategraph2 documentation. `https://build.openmodelica.org/Documentation/Modelica_StateGraph2.html`, 2024b. Accessed: 2025-06-03.

G. D. Astudillo, H. Beiranvand, F. Cecati, C. Werlich, A. Würsig, and M. Liserre. Integrated strategy for optimized charging and balancing of lithium-ion battery packs. *IEEE Transactions on Transportation Electrification*, 11(1):4980–4991, 2025.

D. Beck, P. Dechent, D. U. Sauer, and M. Dubarry. Inhomogeneities and cell-to-cell variations in lithium-ion batteries: A review. *Energies*, 2021.

Modelica by Example. Interpolation, modelica by example. URL `https://mbe.modelica.university/behavior/functions/interpolation/`. Accessed 2025-06-03.

J. Cao, Y. Wang, and S. Zhang. Single lc-series balancer for li-ion packs. *Bulletin of Electrical Engineering and Informatics (BEEI)*, 9(6):2397–2404, 2020.

S. L. Chavan, M. A. Kanawade, and R. S. Ankushe. An effective passive cell balancing technique for lithium-ion battery. *Next Energy*, 8, 2025. ISSN 2949-821X.

M. Ecker, J. B. Gerschler, J. Vogel, S. Käbitz, F. Hust, P. Dechent, and D. U. Sauer. Calendar and cycle life study of li(nimnco)o2 based 18650 lithium-ion batteries. *Journal of Power Sources*, 248:839–851, 2014.

Epec. Battery cell comparison, 2023a. URL `https://www.epectec.com/batteries/cell-comparison.html`. Published by Epec Engineered Technologies. Accessed May 2025.

Epec. Lithium battery technologies, 2023b. URL `https://www.epectec.com/batteries/lithium-battery-technologies.html`. Published by Epec Engineered Technologies. Accessed May 2025.

H. He and X. Chen. Analysing unbalanced ageing in ev battery packs using the low-cost lumped single particle model (lspm): The impact of temperature gradients among parallel-connected cells. *Transportation Research Procedia*, 70:406–413, 2023.

Y. He. Modeling of dynamic hysteresis characters for the lithium-ion battery. *Journal of The Electrochemical Society*, 167(9), 2020.

P. Keil and A. Jossen. Charging protocols for lithium-ion batteries and their impact on cycle life—an experimental study with different 18650 high-power cells. *Journal of Energy Storage*, 6:125–141, 2016.

N. Khan, C. A. Ooi, A. S. Alturki, M. Amir, and T. Alharbi. A critical review of battery cell balancing techniques, optimal design, converter topologies, and performance evaluation for optimizing storage system in electric vehicles. *Energy Reports*, 11:4999–5032, 2024. doi: 10.1016/j.egyr.2024.04.041.

W. Liu, T. Placke, and K. T. Chau. Overview of batteries and battery management for electric vehicles. *Energy Reports*, 8:4058–4084, 2022.

D. Lu. *Identifying Physical Model Parameter Values for Lithium-Ion Cells*. PhD thesis, Colorado State University-Pueblo, 2016.

M. Makowski and M. Pedram. Low-power battery management ic using switched capacitor circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(5):1996–2007, 2016.

A. Manthiram. A reflection on lithium-ion battery cathode chemistry. *Nature Communication*, 11, 2020.

Modelon. Modelica: What is it and why is it important? URL `https://modelon.com/what-is-modelica/`. Accessed 2025-06-03.

A. G. Olabi, Q. Abbas, P. Shinde, and M. A. Abdelkareem. Rechargeable batteries: Technological advancement, challenges, current and emerging applications. *Energy*, 266, 2023.

L. A. Perişoară, I. C. Guran, and D. C. Costache. A passive battery management system for fast balancing of four lifepo4 cells. *IEEE 24th International Symposium for Design and Technology in Electronic Packaging*, pages 390–393, 2018.

G. Pistoia. *Batteries for Portable Devices*, chapter 1. Elsevier Science, 2005.

G. L. Plett. *Battery Management Systems, Volume 1 - Battery Modeling*. Artech House, 2015. ISBN 978-1-63081-023-8.

A. Pérez, I. San Martín, P. Sanchis, and A. Ursúa. Lithium-ion second-life batteries: Aging modeling and experimental validation. In *Proceedings of the 2024 International Conference on Renewable Energies and Smart Technologies (REST)*, pages 193–197, 2024.

A. Reiter, S. Lehner, O. Bohlen, and D. U. Sauer. Electrical cell-to-cell variations within large-scale battery systems—a novel characterization and modeling approach. *Journal of Energy Storage*, 2023. ISSN 2352-152X.

N. Samaddar, N. S. Kumar, and R. Jayapragash. Passive cell balancing of li-ion batteries used for automotive applications. *Journal of Physics: Conference Series*, 2020.

A. Szczepaniak, S. Gajewski, and A. Tomczewski. Characteristics of battery management systems of electric vehicles with consideration of the active and passive cell balancing process. *Energies*, 15(5):1681, 2022.

M. Tekin and M. I. Karamangil. Comparative analysis of equivalent circuit battery models for electric vehicle battery management systems. *Journal of Energy Storage*, 86:111327, 2024.

Texas Instruments. Cell balancing design guidelines for the bq769x0 family, 2015.

M. M. Tiller. Modelica by example. URL `https://mbe.modelica.university/`. Accessed 2025-06-03.

T. Tudorache, A. Bitoleanu, and L. N. Tutelea. Modeling of single inductor based battery balancing circuit for hybrid electric vehicles. *ResearchGate*, 2017.

A. Urquía and C. Martín-Villalba. Modelado orientado a objetos y simulación de sistemas físicos, Segunda edición, 2018.

M. Uzair, G. Abbas, and S. Hosain. Characteristics of battery management systems of electric vehicles with consideration of the active and passive cell balancing process. *World Electric Vehicle Journal*, page 120, 2021. ISSN 2032-6653.

M. Waseem, M. Ahmad, A. Parveen, and M. Suhaib. Battery technologies and functionality of battery management system for evs: Current status, key challenges, and future prospectives. *Journal of Power Sources*, 580, 2023.

T. Wulandari, D. Fawcett, S. B. Majumder, and G. E. J. Poinern. Lithium-based batteries, history, current status, challenges, and future perspectives. *Battery Energy*, 2:834–854, 2023.

R. Zhang and B. Xia. A study on the open circuit voltage and state of charge characterization of high capacity lithium-ion battery under different temperature. *Energies*, 11(9):2408, 2018.

# Appendix A

# Battery_Balancing Library Code

This appendix provides a comprehensive overview of the source code constituting the custom `Battery_Balancing` Modelica library, developed for modeling second-life lithium-ion battery packs with various balancing strategies. The code is modularly organized into several packages, each fulfilling a specific functional role. What follows is a section-wise listing of key components with brief introductions to their content and purpose. The beginning and end of the main library looks as follows, and the rest of the packages are contained within it.

```
package Battery_Balancing
  "Electrochemical Model Library for Lithium-Ion Cell and Pack Simulation with
  Balancing and Aging Dynamics."
  import SI = Modelica.Units.SI;
  import Modelica.Constants;
  ...
end Battery_Balancing;
```

## A.1. Code Package `Interfaces`

The `Interfaces` package defines standardized connectors and base models used across the library to ensure compatibility and modularity in component design.

```
package Interfaces
  "Interface package for external connection between components."
  connector Pin "Neutral electrical connector."
    SI.Voltage u;
    flow SI.Current i;
  end Pin;
```

```
connector Pin_p "Positive electrical connector."
  SI.Voltage u;
  flow SI.Current i;
end Pin_p;

connector Pin_n "Negative electrical connector."
  SI.Voltage u;
  flow SI.Current i;
end Pin_n;

partial model UnPuerto
  "Two-terminal base model with optional current direction logic."
  parameter Boolean activo=false "Active/inactive component";
protected
  SI.Voltage u "Voltage between pins (= p.u - n.u)";
  SI.Current i "Current through component";
public
  Pin_p p "Positive port";
  Pin_n n "Negative port";
equation
  u = p.u - n.u;
  if (activo) then
    i = n.i;
  else
    i = p.i;
  end if;
  p.i = -n.i;
end UnPuerto;

partial model UnPuerto_ECM
  "Two-terminal interface for ECM components."
public
  Pin_p p "Positive pin";
  Pin_n n "Negative pin";
end UnPuerto_ECM;

partial model UnPuertoTresPines
  "Three-terminal interface for gate-controlled devices"
  import SI = Modelica.Units.SI;

protected
  SI.Voltage u_DS(start=0) "Voltage drain-source";
  SI.Voltage u_GS(start=0) "Voltage gate-source";
  SI.Current i_D "Drain current";
  SI.Current i_G "Gate current";
```

```
  //Graphic declaration of components
public
  Pin g "Gate";
  Pin_p p "Drain";
  Pin_n n "Source";
equation
  // Voltage equations
  u_DS = p.u - n.u;
  u_GS = g.u - n.u;
  //Current equations
  i_D = p.i;
  i_G = g.i;
  // Kirchhoff  currents
  n.i = -(p.i + g.i);
end UnPuertoTresPines;

partial model CommutatorInterface
  "Three-terminal interface for commutator components."
  Pin b "Commute pin 1";
  Pin c "Commute pin 2";
  Pin a "Entry Pin";
protected
  SI.Voltage u_ab(start=0) "Voltage A to B";
  SI.Voltage u_ac(start=0) "Voltage A to C";
  SI.Current i_a "Current into terminal A";
  SI.Current i_b "Current into terminal B";
  SI.Current i_c "Current into terminal C";

equation
  //Voltage definition
  u_ab = a.u - b.u;
  u_ac = a.u - c.u;
  //Current definition
  i_a = a.i;
  i_b = b.i;
  i_c = c.i;
  //Kirchoff law for currents
  a.i + b.i + c.i = 0;
end CommutatorInterface;

connector ControlSignal
  "Connector for mode control signal in charge/discharge modules."
  Integer mode "mode: -1/discharge, 0/idle, +1/charge CC";
end ControlSignal;
```

```
  connector ControlSignalBalancing_Shunt
    "Control signal connector for shunt resistor balancing."
    Boolean OnOff "Signal On/Off all switches";

  end ControlSignalBalancing_Shunt;

  connector ControlSignalBalancing_Switched
    "Control signal connector for switched resistor balancing."
    Boolean OnOff[4] "Control signal for each switch";

  end ControlSignalBalancing_Switched;

  connector ControlSignalBalancing_Single
    "Control signal connector for single capacitor or inductor balancing."

    Boolean OnOff[5] "Control signal for each switch";
    Integer Com[2] "Control signal for each commutator";
  end ControlSignalBalancing_Single;

  connector VoltageFeedback
    "Voltage feedback connector for 4-cell pack."
    Real V_cell[4] "Voltage of each cell (V)";

  end VoltageFeedback;

  connector VoltageFeedback1C
    "Voltage feedback connector for single cell."
    Real V_cell "Voltage of cell (V)";
  end VoltageFeedback1C;

end Interfaces;
```

## A.2. Code Package Components

The Components package includes basic electrical elements such as resistors, capacitors, switches, and control sources, which serve as the building blocks for higher-level models.

```
package Components
  "Models of fundamental electrical components for ECM-based battery modeling."
  model Capacitance "Ideal linear capacitor."
    extends Interfaces.UnPuerto;
    input SI.Capacitance C=10^(-6) "Capacitance";
```

```
  SI.Voltage V_cap;
equation
  //Capacitance equations
  C*der(u) = i;
  V_cap = u;
end Capacitance;

model Inductance "Ideal inductor."
  extends Interfaces.UnPuerto;
  input SI.Inductance L=10^(-3) "Inductance";
  SI.Voltage V_ind;
initial equation
  i = 0;
equation
  //Inductance equations
  L*der(i) = u;
  V_ind = u;
end Inductance;

model Ground "Electrical ground"
  Interfaces.Pin_n p "Ground pin";
equation
  //Voltage reference
  p.u = 0;
end Ground;

model NMOS "Non-ideal NMOS transistor with finite on-resistance"
  extends Interfaces.UnPuertoTresPines;
  parameter Real Kn=0.1 "Transconductance parameter";
  parameter Real Lambda=0.02 "Channel length modulation factor";
  parameter SI.Voltage Vth=2.0 "Threshold voltage";
  parameter SI.Resistance Rds_on=0.1 "Drain-source on resistance";
protected
  SI.Voltage u_channel "Effective channel voltage (V)";
equation
  // Gate displacement current
  i_G = 0;
  // Effective voltage
  u_channel = u_DS - i_D*Rds_on;
  // Drain current definition
  if u_GS < Vth then
    // Cut-off
    i_D = 0;
  else
    if u_channel <= (u_GS - Vth) then
```

```
      // Linear region
      i_D = Kn*(2*(u_GS - Vth)*u_channel - u_channel^2);
    else
      // Saturation region
      i_D = Kn*(u_GS - Vth)^2*(1 - Lambda*u_channel);
    end if;
  end if;
end NMOS;

model OCV_source
  "Open-circuit voltage (OCV) source with SOC tracking."
  extends Interfaces.UnPuerto(activo=true);
  input Real OCV "OCV Voltage";
  parameter SI.ElectricCharge Q "Cell capacity";
  parameter Real eta(
    min=0,
    max=1) = 1 "Coulomb counting efficiency";
  Real z(
    start=1,
    max=1,
    min=0) "State of charge (SOC)";
  //Initially charged
equation
  u = OCV;
  //From parameter loader
  der(z) = -eta*i/Q;
  // Coulomb counting equation
end OCV_source;

model Resistance "Ideal linear resistor."
  extends Interfaces.UnPuerto;
  input SI.Resistance R "Resistance";
equation
  // Resistance equations
  u = R*i;
end Resistance;

model Switch "Idealized controlled switch with on-resistance."
  extends Interfaces.UnPuerto;
  input Boolean OnOff "Control signal, true when conducting";
  parameter SI.Resistance R_on=1e-3 "Resistance (Ohm)";
equation
  if OnOff then
    u = R_on*i;
    // When conducting, resistance
```

```
    else
      i = 0;
      // When open, no current
    end if;
  end Switch;

  model Icontrol "Externally controlled current source."
    extends Interfaces.UnPuerto(activo=false);
    input SI.Current i_discharge(start=0) "Discharge current";
  equation
    i = -i_discharge;
  end Icontrol;
  model Vcontrol "Externally controlled voltage source."
    extends Interfaces.UnPuerto(activo=true);
    input SI.Voltage V_set "Controlled voltage";
  equation
    u = V_set;
  end Vcontrol;

  model Commutator "Two-way controlled commutator switch"
    extends Interfaces.CommutatorInterface;
    input Integer select "If 0, a to b; if 1, a to c";
    parameter SI.Resistance R_on=1e-3 "On-state resistance";
  equation
    if select == 0 then
      u_ab = R_on*i_a;
      // a to b
      i_c = 0;
    else
      u_ac = R_on*i_a;
      // a to c
      i_b = 0;
    end if;
  end Commutator;
end Components;
```

## A.3. Code Package `Functions`

This package provides auxiliary functions and interpolation tools, including look-up tables for ECM parameter estimation and custom utility functions like `argmax` and `argmin`.

```
package Functions
  "Auxiliary function blocks for parameter interpolation and control logic"
  import Modelica.Blocks.Tables.CombiTable1Ds;
  import Modelica.Blocks.Types.Smoothness;

  model ECM_ParameterLoader_1Thv
    "Lookup model for loading SOC-dependent ECM parameters from external data
    file."
    input Real SOC;
    // External input
    CombiTable1Ds table_OCV(
      tableName="ECMdata",
      fileName="dataECM.mat",
      columns={2},
      tableOnFile=true,
      table=[0, 0],
      smoothness=Smoothness.LinearSegments) "Table smooth interpolator, OCV";
    CombiTable1Ds table_R0(
      tableName="ECMdata",
      fileName="dataECM.mat",
      columns={3},
      tableOnFile=true,
      table=[0, 0],
      smoothness=Smoothness.LinearSegments) "Table smooth interpolator, R0";
    CombiTable1Ds table_R1(
      tableName="ECMdata",
      fileName="dataECM.mat",
      columns={4},
      tableOnFile=true,
      table=[0, 0],
      smoothness=Smoothness.LinearSegments) "Table smooth interpolator, R1";
    CombiTable1Ds table_C1(
      tableName="ECMdata",
      fileName="dataECM.mat",
      columns={5},
      tableOnFile=true,
      table=[0, 0],
      smoothness=Smoothness.LinearSegments) "Table smooth interpolator, C1";
    output Real OCV "Output OCV voltage /SOC";
    output Real R0 "Output R0 /SOC";
    output Real R1 "Output V0 /SOC";
    output Real C1 "Output C1/SOC";
  equation
    // Connect SOC input to table inputs
```

```
  table_OCV.u = SOC;
  table_R0.u = SOC;
  table_R1.u = SOC;
  table_C1.u = SOC;
  // Read interpolated outputs
  OCV = table_OCV.y[1];
  R0 = table_R0.y[1];
  R1 = table_R1.y[1];
  C1 = table_C1.y[1];
end ECM_ParameterLoader_1Thv;

function argmax
  "Returns the index of the maximum value in a real-valued vector."
  input Real vec[:] "Vector of given size";
  output Integer index "Highest value index in vector";
protected
  Real maxVal=vec[1] "Maximum value in vector";
algorithm
  index := 1;
  //Loop and check which is highest index
  for i in 2:size(vec, 1) loop
    if vec[i] > maxVal then
      maxVal := vec[i];
      index := i;
    end if;
  end for;
end argmax;

function argmin
  "Returns the index of the minimum value in a real-valued vector."
  input Real vec[:] "Vector of given size";
  output Integer index "Lowest value index in vector";
protected
  Real minVal=vec[1] "Minimum value in vector";
algorithm
  index := 1;
  //Loop and check which is lowest index
  for i in 2:size(vec, 1) loop
    if vec[i] < minVal then
      minVal := vec[i];
      index := i;
    end if;
  end for;
end argmin;
```

```
end Functions;
```

## A.4. Code Package ECM_Structures

Contains various equivalent circuit model (ECM) architectures for lithium-ion cells, incorporating parameterization, aging effects, and feedback mechanisms.

```
package ECM_Structures
  "Package of ECM structures for lithium-ion cell modeling."
  model ECM_Thevenin
    "Basic Thevenin model with SOC-based input parameters."
    extends Interfaces.UnPuerto_ECM;
    parameter SI.ElectricCharge Q=3600 "Cell Capacity";
    parameter Real eta(
      min=0,
      max=1) = 1 "Coulomb counting efficiency";
    input SI.Voltage OCV "Cell OCV";
    input SI.Resistance R0 "Series Resistance";
    input SI.Resistance R1 "Parallel Resistance";
    input SI.Capacitance C1 "Parallel Capacitance";
    Components.Resistance R_0(R=R0);
    Components.Resistance R_1(R=R1);
    Components.Capacitance C_1(C=C1);
    Components.OCV_source ocv(
      Q=Q,
      eta=eta,
      OCV=OCV);
  equation
    connect(R_0.n, R_1.p);
    connect(R_0.n, C_1.p);
    connect(R_1.n, p);
    connect(C_1.n, p);
    connect(ocv.p, R_0.p);
    connect(ocv.n, n);

  end ECM_Thevenin;

  model ECM_Thevenin_SL
    "Extended model with aging and unbalance derating factors."
    extends Interfaces.UnPuerto_ECM;

    parameter SI.ElectricCharge Q=3600 "Cell Capacity";
    parameter Real eta(
```

```
    min=0,
    max=1) = 1 "Coulomb counting efficiency";

  input SI.Voltage OCV "Cell OCV";
  input SI.Resistance R0 "Series Resistance";
  input SI.Resistance R1 "Parallel Resistance";
  input SI.Capacitance C1 "Parallel Capacitance";

  // Derating factors with extended ranges
  parameter Real alpha_Q(
    min=1,
    max=5) = 1.0 "Q aging derating factor";
  parameter Real alpha_R0(
    min=0,
    max=1) = 1.0 "R0 aging derating factor";
  parameter Real alpha_R1(
    min=0,
    max=1) = 1.0 "R1 aging derating factor";
  parameter Real alpha_C1(
    min=1,
    max=5) = 1.0 "C1 aging derating factor";

  parameter Real beta_Q(
    min=0.5,
    max=1.5) = 1.0 "Q unbalance derating factor";
  parameter Real beta_R0(
    min=0.5,
    max=1.5) = 1.0 "R0 unbalance derating factor";
  parameter Real beta_R1(
    min=0.5,
    max=1.5) = 1.0 "R1 unbalance derating factor";
  parameter Real beta_C1(
    min=0.5,
    max=1.5) = 1.0 "C1 unbalance derating factor";
  // Effective parameters after derating
  final parameter SI.ElectricCharge Q_eff=Q*alpha_Q*beta_Q
    "Effective cell capacity";
  SI.Resistance R0_eff "Effective series resistance";
  SI.Resistance R1_eff "Effective parallel resistance";
  SI.Capacitance C1_eff "Effective parallel capacitor";
  Components.Resistance R_0(R=R0_eff);
  Components.Resistance R_1(R=R1_eff);
  Components.Capacitance C_1(C=C1_eff);
  Components.OCV_source ocv(
    Q=Q_eff,
```

```
    eta=eta,
    OCV=OCV);
equation
  // Derating equations for parameters
  R0_eff = R0*alpha_R0*beta_R0;
  R1_eff = R1*alpha_R1*beta_R1;
  C1_eff = C1*alpha_C1*beta_C1;

  connect(R_0.n, R_1.p);
  connect(R_0.n, C_1.p);
  connect(R_1.n, p);
  connect(C_1.n, p);
  connect(ocv.p, R_0.p);
  connect(ocv.n, n);
end ECM_Thevenin_SL;

model ECM_Thevenin_SL_Vbf
  "Same as ECM_Thevenin_SL with an added voltage feedback connector."
  extends Interfaces.UnPuerto_ECM;

  parameter SI.ElectricCharge Q=3600 "Cell Capacity";
  parameter Real eta(
    min=0,
    max=1) = 1 "Coulomb counting efficiency";

  input SI.Voltage OCV "Cell OCV";
  input SI.Resistance R0 "Series Resistance";
  input SI.Resistance R1 "Parallel Resistance";
  input SI.Capacitance C1 "Parallel Capacitance";

  // Derating factors with extended ranges
  parameter Real alpha_Q(
    min=1,
    max=5) = 1.0 "Q aging derating factor";
  parameter Real alpha_R0(
    min=0,
    max=1) = 1.0 "R0 aging derating factor";
  parameter Real alpha_R1(
    min=0,
    max=1) = 1.0 "R1 aging derating factor";
  parameter Real alpha_C1(
    min=1,
    max=5) = 1.0 "C1 aging derating factor";

  parameter Real beta_Q(
```

```
    min=0.5,
    max=1.5) = 1.0 "Q unbalance derating factor";
  parameter Real beta_R0(
    min=0.5,
    max=1.5) = 1.0 "R0 unbalance derating factor";
  parameter Real beta_R1(
    min=0.5,
    max=1.5) = 1.0 "R1 unbalance derating factor";
  parameter Real beta_C1(
    min=0.5,
    max=1.5) = 1.0 "C1 unbalance derating factor";

  // Effective parameters after derating
  final parameter SI.ElectricCharge Q_eff=Q*alpha_Q*beta_Q
    "Effective cell capacity";
  SI.Resistance R0_eff "Effective parallel resistance";
  SI.Resistance R1_eff "Effective series resistance";
  SI.Capacitance C1_eff "Effective series capacitance";

  Components.Resistance R_0(R=R0_eff);
  Components.Resistance R_1(R=R1_eff);
  Components.Capacitance C_1(C=C1_eff);
  Components.OCV_source ocv(
    Q=Q_eff,
    eta=eta,
    OCV=OCV);

  Interfaces.VoltageFeedback1C voltageFeedback
    "Pin for voltage feedback 1 Cell";
equation
  // Derating equations
  R0_eff = R0*alpha_R0*beta_R0;
  R1_eff = R1*alpha_R1*beta_R1;
  C1_eff = C1*alpha_C1*beta_C1;

  //Feedback to connector
  voltageFeedback.V_cell = p.u - n.u;

  connect(R_0.n, R_1.p);
  connect(R_0.n, C_1.p);
  connect(R_1.n, p);
  connect(C_1.n, p);
  connect(ocv.p, R_0.p);
  connect(ocv.n, n);
end ECM_Thevenin_SL_Vbf;
```

```
end ECM_Structures;
```

## A.5. Code Package `Cell_Packs`

Provides configurations of multiple ECM-modeled cells into modules and packs, including internal node access and scaling capabilities.

```
package Cell_Packs "Create packs of ECM modeled cells"

  model Battery_Module
    "Battery module formed by N ideal cells in series."
    extends Interfaces.UnPuerto_ECM;
    parameter Integer N(
      min=2,
      max=100) = 4 "Cells in series";

    // Constant parameters for all cells
    parameter SI.ElectricCharge Q=3600 "Capacity of each cell";
    parameter Real eta(
      min=0,
      max=1) = 1 "Coulomb counting efficiency";

    // SOC-dependent inputs (must be provided from top level)
    input SI.Voltage OCV[N] "OCV Voltage for each cell";
    input SI.Resistance R0[N] "Series resistance for each cell";
    input SI.Resistance R1[N] "Parallel resistance for each cell";
    input SI.Capacitance C1[N] "Parallel capacitor for each cell";

    // Cells array
    replaceable ECM_Structures.ECM_Thevenin Cell[N](each Q=Q, each eta=eta)
      "Cell";

  equation
    //Connections between different elements
    connect(n, Cell[1].n);
    for i in 1:N - 1 loop
      connect(Cell[i].n, Cell[i + 1].p);
    end for;

    connect(Cell[N].p, p);

    // Provide SOC-dependent inputs to each cell
    for i in 1:N loop
```

```
    Cell[i].OCV = OCV[i];
    Cell[i].R0 = R0[i];
    Cell[i].R1 = R1[i];
    Cell[i].C1 = C1[i];
  end for;

end Battery_Module;

model Battery4Cell_Bal
  "4-cell battery module with second-life ECMs and exposed balancing nodes."
  extends Interfaces.UnPuerto_ECM;

  parameter SI.ElectricCharge Q[4]=fill(3600, 4) "Capacity of each cell";
  parameter Real eta[4](
    each min=0,
    max=1) = fill(1, 4) "Coulomb counting efficiency";

  input SI.Voltage OCV[4] "OCV Voltage of each cell";
  input SI.Resistance R0[4] "Series resistance of each cell";
  input SI.Resistance R1[4] "Parallel resistance of each cell";
  input SI.Capacitance C1[4] "Parallel capacitance of each cell";

  parameter Real alpha[4, 4]=fill(1.0, 4, 4)
    "Aging derating matrix: [Q, R0, R1, C1]";
  parameter Real beta[4, 4]=fill(1.0, 4, 4)
    "Unbalance derating matrix: [Q, R0, R1, C1]";
  ECM_Structures.ECM_Thevenin_SL Cell1(
    Q=Q[1],
    eta=eta[1],
    alpha_Q=alpha[1, 1],
    alpha_R0=alpha[1, 2],
    alpha_R1=alpha[1, 3],
    alpha_C1=alpha[1, 4],
    beta_Q=beta[1, 1],
    beta_R0=beta[1, 2],
    beta_R1=beta[1, 3],
    beta_C1=beta[1, 4]);

  ECM_Structures.ECM_Thevenin_SL Cell2(
    Q=Q[2],
    eta=eta[2],
    alpha_Q=alpha[2, 1],
    alpha_R0=alpha[2, 2],
    alpha_R1=alpha[2, 3],
    alpha_C1=alpha[2, 4],
```

```
    beta_Q=beta[2, 1],
    beta_R0=beta[2, 2],
    beta_R1=beta[2, 3],
    beta_C1=beta[2, 4]);
  ECM_Structures.ECM_Thevenin_SL Cell3(
    Q=Q[3],
    eta=eta[3],
    alpha_Q=alpha[3, 1],
    alpha_R0=alpha[3, 2],
    alpha_R1=alpha[3, 3],
    alpha_C1=alpha[3, 4],
    beta_Q=beta[3, 1],
    beta_R0=beta[3, 2],
    beta_R1=beta[3, 3],
    beta_C1=beta[3, 4]);
  ECM_Structures.ECM_Thevenin_SL Cell4(
    Q=Q[4],
    eta=eta[4],
    alpha_Q=alpha[4, 1],
    alpha_R0=alpha[4, 2],
    alpha_R1=alpha[4, 3],
    alpha_C1=alpha[4, 4],
    beta_Q=beta[4, 1],
    beta_R0=beta[4, 2],
    beta_R1=beta[4, 3],
    beta_C1=beta[4, 4]);
  Interfaces.Pin node_3_4;
  Interfaces.Pin node_2_3;
  Interfaces.Pin node_1_2;
equation
  // Set each cell OCV/R0/R1/C1 done cell by cell since it was preffered
  to declare them graphically
  Cell1.OCV = OCV[1];
  Cell1.R0 = R0[1];
  Cell1.R1 = R1[1];
  Cell1.C1 = C1[1];
  Cell2.OCV = OCV[2];
  Cell2.R0 = R0[2];
  Cell2.R1 = R1[2];
  Cell2.C1 = C1[2];
  Cell3.OCV = OCV[3];
  Cell3.R0 = R0[3];
  Cell3.R1 = R1[3];
  Cell3.C1 = C1[3];
  Cell4.OCV = OCV[4];
```

```
    Cell4.R0 = R0[4];
    Cell4.R1 = R1[4];
    Cell4.C1 = C1[4];
    connect(n, Cell4.n);
    connect(Cell4.p, node_3_4);
    connect(node_3_4, Cell3.n);
    connect(node_2_3, Cell3.p);
    connect(node_2_3, Cell2.n);
    connect(node_1_2, Cell2.p);
    connect(node_1_2, Cell1.n);
    connect(Cell1.p, p);
  end Battery4Cell_Bal;

  model Battery4Cell_Bal_Vfb
    "4-cell battery module with voltage feedback connector."
    extends Interfaces.UnPuerto_ECM;
    parameter SI.ElectricCharge Q[4]=fill(3600, 4) "Capacity of each cell";
    parameter Real eta[4](
      each min=0,
      max=1) = fill(1, 4) "Coulomb counting efficiency";
    input SI.Voltage OCV[4] "OCV Voltage of each cell";
    input SI.Resistance R0[4] "Series resistance of each cell";
    input SI.Resistance R1[4] "Parallel resistance of each cell";
    input SI.Capacitance C1[4] "Parallel capacitance of each cell";
    parameter Real alpha[4, 4]=fill(1.0, 4, 4)
      "Aging derating matrix: [Q, R0, R1, C1]";
    parameter Real beta[4, 4]=fill(1.0, 4, 4)
      "Unbalance derating matrix: [Q, R0, R1, C1]";
    ECM_Structures.ECM_Thevenin_SL Cell1(
      Q=Q[1],
      eta=eta[1],
      alpha_Q=alpha[1, 1],
      alpha_R0=alpha[1, 2],
      alpha_R1=alpha[1, 3],
      alpha_C1=alpha[1, 4],
      beta_Q=beta[1, 1],
      beta_R0=beta[1, 2],
      beta_R1=beta[1, 3],
      beta_C1=beta[1, 4]);
    ECM_Structures.ECM_Thevenin_SL Cell2(
      Q=Q[2],
      eta=eta[2],
      alpha_Q=alpha[2, 1],
      alpha_R0=alpha[2, 2],
      alpha_R1=alpha[2, 3],
```

```
    alpha_C1=alpha[2, 4],
    beta_Q=beta[2, 1],
    beta_R0=beta[2, 2],
    beta_R1=beta[2, 3],
    beta_C1=beta[2, 4]);
  ECM_Structures.ECM_Thevenin_SL Cell3(
    Q=Q[3],
    eta=eta[3],
    alpha_Q=alpha[3, 1],
    alpha_R0=alpha[3, 2],
    alpha_R1=alpha[3, 3],
    alpha_C1=alpha[3, 4],
    beta_Q=beta[3, 1],
    beta_R0=beta[3, 2],
    beta_R1=beta[3, 3],
    beta_C1=beta[3, 4]);
  ECM_Structures.ECM_Thevenin_SL Cell4(
    Q=Q[4],
    eta=eta[4],
    alpha_Q=alpha[4, 1],
    alpha_R0=alpha[4, 2],
    alpha_R1=alpha[4, 3],
    alpha_C1=alpha[4, 4],
    beta_Q=beta[4, 1],
    beta_R0=beta[4, 2],
    beta_R1=beta[4, 3],
    beta_C1=beta[4, 4]);
  Interfaces.Pin node_3_4;
  Interfaces.Pin node_2_3;
  Interfaces.Pin node_1_2;
  Interfaces.VoltageFeedback voltageFeedback;
equation
  // Set each cell OCV/R0/R1/C1 done cell by cell since it was preffered
  to declare them graphically
  Cell1.OCV = OCV[1];
  Cell1.R0 = R0[1];
  Cell1.R1 = R1[1];
  Cell1.C1 = C1[1];
  Cell2.OCV = OCV[2];
  Cell2.R0 = R0[2];
  Cell2.R1 = R1[2];
  Cell2.C1 = C1[2];
  Cell3.OCV = OCV[3];
  Cell3.R0 = R0[3];
  Cell3.R1 = R1[3];
```

```
    Cell3.C1 = C1[3];
    Cell4.OCV = OCV[4];
    Cell4.R0 = R0[4];
    Cell4.R1 = R1[4];
    Cell4.C1 = C1[4];
    //Voltage feedback equation for each cell
    voltageFeedback.V_cell[1] = p.u - node_1_2.u;
    voltageFeedback.V_cell[2] = node_1_2.u - node_2_3.u;
    voltageFeedback.V_cell[3] = node_2_3.u - node_3_4.u;
    voltageFeedback.V_cell[4] = node_3_4.u - n.u;
    connect(n, Cell4.n);
    connect(Cell4.p, node_3_4);
    connect(node_3_4, Cell3.n);
    connect(node_2_3, Cell3.p);
    connect(node_2_3, Cell2.n);
    connect(node_1_2, Cell2.p);
    connect(node_1_2, Cell1.n);
    connect(Cell1.p, p);
  end Battery4Cell_Bal_Vfb;

  model BatteryNCell_Bal
    "Configurable N-cell battery module with intermediate balancing nodes."
    extends Interfaces.UnPuerto_ECM;
    parameter Integer N(min=2) = 4 "Number of cells in series";
    parameter SI.ElectricCharge Q[N]=fill(3600, N) "Capacity of each cell";
    parameter Real eta[N](
      each min=0,
      max=1) = fill(3600, N) "Coulomb counting efficiency";
    input SI.Voltage OCV[4] "OCV Voltage of each cell";
    input SI.Resistance R0[4] "Series resistance of each cell";
    input SI.Resistance R1[4] "Parallel resistance of each cell";
    input SI.Capacitance C1[4] "Parallel capacitance of each cell";
    parameter Real alpha[N, 4]=fill(1.0, N, 4)
      "Aging derating matrix: [Q, R0, R1, C1]";
    parameter Real beta[N, 4]=fill(1.0, N, 4)
      "Unbalance derating matrix: [Q, R0, R1, C1]";
    ECM_Structures.ECM_Thevenin_SL Cell[N](
      Q=Q,
      eta=eta,
      alpha_Q=alpha[:, 1],
      alpha_R0=alpha[:, 2],
      alpha_R1=alpha[:, 3],
      alpha_C1=alpha[:, 4],
      beta_Q=beta[:, 1],
      beta_R0=beta[:, 2],
```

```
      beta_R1=beta[:, 3],
      beta_C1=beta[:, 4]) "Cell";
    Interfaces.Pin node[N - 1] "Intermediate nodes between cells";
  equation
    // Connect first terminal
    connect(n, Cell[1].n);
    // Connect series cells and expose nodes
    for i in 1:N - 1 loop
      connect(Cell[i].p, node[i]);
      connect(node[i], Cell[i + 1].n);
    end for;
    // Connect last terminal
    connect(Cell[N].p, p);
    // Provide SOC-dependent inputs
    for i in 1:N loop
      Cell[i].OCV = OCV[i];
      Cell[i].R0 = R0[i];
      Cell[i].R1 = R1[i];
      Cell[i].C1 = C1[i];
    end for;
  end BatteryNCell_Bal;

end Cell_Packs;
```

## A.6. Code Package Balancing_Structures

This package introduces models of passive and active cell balancing methods implemented on a 4-cell module, including charge/discharge infrastructure.

```
package Balancing_Structures
  "Balancing structures for 4-cell modules and Charge/Discharge behavior."

  model ChargeDischarge
    "Charge and discharge module for battery cell stacks."
    parameter SI.Current I_charge=2.0 "Charge current";
    parameter SI.Current I_discharge=0.8 "Discharge current";
    parameter SI.Resistance R_load=0.1 "Series load";

    Interfaces.ControlSignal controlInput "Control Input charge/discharge";
    Components.Icontrol currentSource;
    Components.Resistance loadResistor(R=R_load);
    Interfaces.Pin_p p "Positive pin";
    Interfaces.Pin_n n "Negative pin";
```

```
equation
  connect(currentSource.n, loadResistor.p);
  connect(currentSource.p, p);
  connect(loadResistor.n, n);

  // Depending on mode charge/discharge or nothing
  if controlInput.mode == 1 then
    currentSource.i_discharge = +I_charge;
  elseif controlInput.mode == -1 then
    currentSource.i_discharge = -I_discharge;
  else
    currentSource.i_discharge = 0;
  end if;
end ChargeDischarge;

model ShuntResistorBalancing_4Cell
  "Shunt resistor balancing topology for 4-cell battery packs"

  parameter SI.Resistance R_shunt=10 "Shunt resistor value (Ohm)";
  parameter SI.Resistance Rds_on=0.05 "MOSFET on-resistance (Ohm)";
  parameter SI.Voltage Vth=2.0 "MOSFET threshold voltage (V)";
  parameter SI.Voltage V_gate=10 "Gate voltage when ON (V)";

  Interfaces.Pin pin3;
  Interfaces.Pin pin1;
  Interfaces.Pin pin5;
  Interfaces.Pin pin4;
  Interfaces.Pin pin2;
  Interfaces.ControlSignalBalancing_Shunt controlInput;

  Components.NMOS nMOS4(Vth=Vth, Rds_on=Rds_on);
  Components.NMOS nMOS3(Vth=Vth, Rds_on=Rds_on);
  Components.NMOS nMOS2(Vth=Vth, Rds_on=Rds_on);
  Components.NMOS nMOS1(Vth=Vth, Rds_on=Rds_on);

  Components.Resistance shuntResistor4(R=R_shunt);
  Components.Resistance shuntResistor3(R=R_shunt);
  Components.Resistance shuntResistor2(R=R_shunt);
  Components.Resistance shuntResistor1(R=R_shunt);

  Components.Vcontrol vcontrol4;

  Interfaces.VoltageFeedback voltageFeedback;
  Components.Vcontrol vcontrol3;
```

```
  Components.Vcontrol vcontrol2;
  Components.Vcontrol vcontrol1;
equation
  connect(shuntResistor4.n, pin4);
  connect(shuntResistor3.n, pin3);
  connect(shuntResistor2.n, pin2);
  connect(shuntResistor1.n, pin1);

  // Control of MOSFETs by activating gate voltage
  vcontrol1.V_set = if controlInput.OnOff then V_gate else 0;
  vcontrol2.V_set = if controlInput.OnOff then V_gate else 0;
  vcontrol3.V_set = if controlInput.OnOff then V_gate else 0;
  vcontrol4.V_set = if controlInput.OnOff then V_gate else 0;

  //Voltage feedback equations
  voltageFeedback.V_cell[1] = pin1.u - pin2.u;
  voltageFeedback.V_cell[2] = pin2.u - pin3.u;
  voltageFeedback.V_cell[3] = pin3.u - pin4.u;
  voltageFeedback.V_cell[4] = pin4.u - pin5.u;
  connect(nMOS4.n, shuntResistor4.p);
  connect(nMOS3.p, pin4);
  connect(nMOS3.n, shuntResistor3.p);
  connect(nMOS2.n, shuntResistor2.p);
  connect(nMOS2.p, pin3);
  connect(nMOS1.p, pin2);
  connect(nMOS1.n, shuntResistor1.p);
  connect(pin5, vcontrol4.n);
  connect(vcontrol4.p, nMOS4.g);
  connect(vcontrol3.p, nMOS3.g);
  connect(vcontrol3.n, pin4);
  connect(vcontrol2.p, nMOS2.g);
  connect(vcontrol2.n, pin3);
  connect(vcontrol1.p, nMOS1.g);
  connect(vcontrol1.n, pin2);
  connect(pin5, nMOS4.p);
end ShuntResistorBalancing_4Cell;

model SwitchedResistorBalancing_4Cell
  "Switched Resistor Balancing model for 4 cell battery pack."
  parameter SI.Resistance R_shunt=10 "Shunt resistor value (Ohm)";
  parameter SI.Resistance Rds_on=0.05 "MOSFET on-resistance (Ohm)";
  parameter SI.Voltage Vth=2.0 "MOSFET threshold voltage (V)";
  parameter SI.Voltage V_gate=10 "Gate voltage when ON (V)";
  Interfaces.ControlSignalBalancing_Switched controlInput;
  Interfaces.VoltageFeedback voltageFeedback;
```

```
    Components.NMOS nMOS4(Vth=Vth, Rds_on=Rds_on);
    Components.NMOS nMOS3(Vth=Vth, Rds_on=Rds_on);
    Components.NMOS nMOS2(Vth=Vth, Rds_on=Rds_on);
    Components.NMOS nMOS1(Vth=Vth, Rds_on=Rds_on);
    Components.Resistance shuntResistor4(R=R_shunt);
    Components.Resistance shuntResistor3(R=R_shunt);
    Components.Resistance shuntResistor2(R=R_shunt);
    Components.Resistance shuntResistor1(R=R_shunt);
    Components.Vcontrol vcontrol4;
    Components.Vcontrol vcontrol3;
    Components.Vcontrol vcontrol2;
    Components.Vcontrol vcontrol1;
    Interfaces.Pin pin3;
    Interfaces.Pin pin1;
    Interfaces.Pin pin5;
    Interfaces.Pin pin4;
    Interfaces.Pin pin2;
  equation
    // Control of MOSFETs by activating gate voltage
    vcontrol1.V_set = if controlInput.OnOff[1] then V_gate else 0;
    vcontrol2.V_set = if controlInput.OnOff[2] then V_gate else 0;
    vcontrol3.V_set = if controlInput.OnOff[3] then V_gate else 0;
    vcontrol4.V_set = if controlInput.OnOff[4] then V_gate else 0;
    // Voltage feedback equations
    voltageFeedback.V_cell[1] = pin1.u - pin2.u;
    voltageFeedback.V_cell[2] = pin2.u - pin3.u;
    voltageFeedback.V_cell[3] = pin3.u - pin4.u;
    voltageFeedback.V_cell[4] = pin4.u - pin5.u;
    connect(shuntResistor4.n, pin4);
    connect(shuntResistor3.n, pin3);
    connect(shuntResistor2.n, pin2);
    connect(shuntResistor1.n, pin1);
    connect(nMOS4.n, shuntResistor4.p);
    connect(nMOS3.p, pin4);
    connect(nMOS3.n, shuntResistor3.p);
    connect(nMOS2.n, shuntResistor2.p);
    connect(nMOS2.p, pin3);
    connect(nMOS1.p, pin2);
    connect(nMOS1.n, shuntResistor1.p);
    connect(pin5, vcontrol4.n);
    connect(vcontrol4.p, nMOS4.g);
    connect(vcontrol3.p, nMOS3.g);
    connect(vcontrol3.n, pin4);
    connect(vcontrol2.p, nMOS2.g);
    connect(vcontrol2.n, pin3);
```

```
  connect(vcontrol1.p, nMOS1.g);
  connect(vcontrol1.n, pin2);
  connect(pin5, nMOS4.p);
end SwitchedResistorBalancing_4Cell;

model SingleCapacitorBalancing_4Cell
  "Single Capacitor Balancing model for 4 cell battery pack."
  parameter SI.Capacitance C=1e-3 "Inductance for energy transfer";
  parameter SI.Resistance R_on=0.5 "On-state resistance for switches";
  parameter SI.Resistance R_bleed=1e3 "Bleed resistance";
  parameter SI.Voltage V_start=3.8 "Start Voltage capacitor";
  Interfaces.Pin pin3;
  Interfaces.Pin pin1;
  Interfaces.Pin pin5;
  Interfaces.Pin pin4;
  Interfaces.Pin pin2;
  Components.Switch switch1(R_on=R_on);
  Components.Switch switch2(R_on=R_on);
  Components.Switch switch3(R_on=R_on);
  Components.Switch switch4(R_on=R_on);
  Components.Switch switch5(R_on=R_on);
  Components.Commutator com2;
  Components.Commutator com1;
  Interfaces.VoltageFeedback voltageFeedback;
  Interfaces.ControlSignalBalancing_Single controlSignal;
  Components.Capacitance capacitance(C=C, V_cap(start=V_start, fixed=true));
  Components.Resistance Resistance_bleed(R=R_bleed);
equation

  // Control assignments
  switch1.OnOff = controlSignal.OnOff[1];
  switch2.OnOff = controlSignal.OnOff[2];
  switch3.OnOff = controlSignal.OnOff[3];
  switch4.OnOff = controlSignal.OnOff[4];
  switch5.OnOff = controlSignal.OnOff[5];
  com1.select = controlSignal.Com[1];
  com2.select = controlSignal.Com[2];

  // Control of MOSFETs by activating gate voltage
  voltageFeedback.V_cell[1] = pin1.u - pin2.u;
  voltageFeedback.V_cell[2] = pin2.u - pin3.u;
  voltageFeedback.V_cell[3] = pin3.u - pin4.u;
  voltageFeedback.V_cell[4] = pin4.u - pin5.u;

  connect(switch1.n, com1.b);
```

```
    connect(switch3.n, com1.b);
    connect(switch5.n, com1.b);
    connect(com2.b, com1.b);
    connect(switch4.n, com1.c);
    connect(switch2.n, com1.c);
    connect(com2.c, com1.c);
    connect(switch5.p, pin5);
    connect(switch4.p, pin4);
    connect(switch3.p, pin3);
    connect(switch2.p, pin2);
    connect(switch1.p, pin1);
    connect(com1.a, capacitance.p);
    connect(capacitance.n, com2.a);
    connect(Resistance_bleed.n, com2.a);
    connect(Resistance_bleed.p, com1.a);
  end SingleCapacitorBalancing_4Cell;

  model SingleInductorBalancing_4Cell
    "Single Inductor Balancing model for 4 cell battery pack."

    parameter SI.Inductance L=2e-3 "Inductance for energy transfer";
    parameter SI.Resistance R_on=0.2 "On-state resistance for switches";
    parameter SI.Resistance R_bleed=1e-2 "Bleed resistance";
    parameter SI.Time t_precharge=1 "Precharge duration";
    Interfaces.Pin pin3;
    Interfaces.Pin pin1;
    Interfaces.Pin pin5;
    Interfaces.Pin pin4;
    Interfaces.Pin pin2;
    Components.Switch switch1(R_on=R_on);
    Components.Switch switch2(R_on=R_on);
    Components.Switch switch3(R_on=R_on);
    Components.Switch switch4(R_on=R_on);
    Components.Switch switch5(R_on=R_on);
    Components.Commutator com2;
    Components.Commutator com1;
    Components.Inductance inductance(L=L);
    Interfaces.VoltageFeedback voltageFeedback;
    Interfaces.ControlSignalBalancing_Single controlSignal;
    Components.Resistance resistance_Bleed(R=R_bleed);
    Components.Vcontrol dummy_voltage(V_set=0);
    Components.Switch dummy_switch;
  equation
    switch1.OnOff = controlSignal.OnOff[1];
    switch2.OnOff = controlSignal.OnOff[2];
```

```
    switch3.OnOff = controlSignal.OnOff[3];
    switch4.OnOff = controlSignal.OnOff[4];
    switch5.OnOff = controlSignal.OnOff[5];
    com1.select = controlSignal.Com[1];
    com2.select = controlSignal.Com[2];
    voltageFeedback.V_cell[1] = pin1.u - pin2.u;
    voltageFeedback.V_cell[2] = pin2.u - pin3.u;
    voltageFeedback.V_cell[3] = pin3.u - pin4.u;
    voltageFeedback.V_cell[4] = pin4.u - pin5.u;
    dummy_switch.OnOff = time < t_precharge;
    connect(switch1.n, com1.b);
    connect(switch3.n, com1.b);
    connect(switch5.n, com1.b);
    connect(com2.b, com1.b);
    connect(switch4.n, com1.c);
    connect(switch2.n, com1.c);
    connect(com2.c, com1.c);
    connect(switch5.p, pin5);
    connect(switch4.p, pin4);
    connect(switch3.p, pin3);
    connect(switch2.p, pin2);
    connect(switch1.p, pin1);
    connect(resistance_Bleed.n, com2.a);
    connect(inductance.p, com1.a);
    connect(inductance.n, resistance_Bleed.p);
    connect(dummy_voltage.n, inductance.n);
    connect(dummy_switch.n, dummy_voltage.p);
    connect(dummy_switch.p, inductance.p);
  end SingleInductorBalancing_4Cell;

end Balancing_Structures;
```

## A.7. Code Package `Control_Structures`

Contains state-graph-based controllers for managing charge/discharge and activating appropriate balancing strategies under given operating conditions.

```
package Control_Structures "Package containing charge/discharge infrastructure
and cell balancing architectures for a 4-cell lithium-ion battery module."

  model Controller_NoBalancing1C_SG
    "No balancing 1 cell controller State graph based."
```

```
    Interfaces.VoltageFeedback1C voltageFeedback "Voltage feedback from cell.";
    Interfaces.ControlSignal controlLoad
      "Control signal to Charge Discharge module.";

    parameter SI.Time t_Start=100 "Idle time between phases";
    parameter SI.Time t_Idle=500 "Idle time between phases";
    parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
    parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
    parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";


    Modelica_StateGraph2.Step Init(initialStep=true, nOut=1);
    Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
    Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
    Modelica_StateGraph2.Transition Init_Discharge(delayedTransition=true,
        waitTime=t_Start);
    Modelica_StateGraph2.Transition
    Discharge_Idle(condition=voltageFeedback.V_cell
          <= Vmin + margin);
    Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
        waitTime=t_Idle);
    Modelica_StateGraph2.Step Charge(nIn=1, nOut=1);
    Modelica_StateGraph2.Transition
    Charge_CBalancing(condition=voltageFeedback.V_cell
          >= Vmax - margin);
    Modelica_StateGraph2.Step idle2(nOut=1, nIn=1);
    Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
        waitTime=t_Idle);
  equation
    // Charge discharge mode sequence
    controlLoad.mode = if Discharge.active then -1 elseif Charge.active then 1
       else 0;

    connect(Init.outPort[1], Init_Discharge.inPort);
    connect(Init_Discharge.outPort, Discharge.inPort[1]);
    connect(Discharge.outPort[1], Discharge_Idle.inPort);
    connect(Discharge_Idle.outPort, Idle1.inPort[1]);
    connect(Idle1.outPort[1], Idle_Charge.inPort);
    connect(Idle_Charge.outPort, Charge.inPort[1]);
    connect(Charge_CBalancing.inPort, Charge.outPort[1]);
    connect(idle2.outPort[1], Idle_Discharge.inPort);
    connect(Idle_Discharge.outPort, Discharge.inPort[2]);

    connect(Charge_CBalancing.outPort, idle2.inPort[1]);
  end Controller_NoBalancing1C_SG;
```

```
model Controller_NoBalancing4C_SG
  "No Balancing 4 cell pack controller State graph based."

  Interfaces.VoltageFeedback voltageFeedback
    "Voltage feedback from cell pack.";
  Interfaces.ControlSignal controlLoad
    "Control signal to charge/discharge module.";

  parameter SI.Time t_Start=100 "Idle time between phases";
  parameter SI.Time t_Idle=500 "Idle time between phases";
  parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
  parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
  parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";

  Modelica_StateGraph2.Step Init(initialStep=true, nOut=1);
  Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
  Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Init_Discharge(delayedTransition=true,
      waitTime=t_Start);
  Modelica_StateGraph2.Transition Discharge_Idle(condition=min(
        voltageFeedback.V_cell) <= Vmin + margin);
  Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
      waitTime=t_Idle);
  Modelica_StateGraph2.Step Charge(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Charge_CBalancing(condition=max(
        voltageFeedback.V_cell) >= Vmax - margin);
  Modelica_StateGraph2.Step idle2(nOut=1, nIn=1);
  Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
      waitTime=t_Idle);
equation
  //Charge/discharge mode sequence
  controlLoad.mode = if Discharge.active then -1 elseif Charge.active then 1
     else 0;

  connect(Init.outPort[1], Init_Discharge.inPort);
  connect(Init_Discharge.outPort, Discharge.inPort[1]);
  connect(Discharge.outPort[1], Discharge_Idle.inPort);
  connect(Discharge_Idle.outPort, Idle1.inPort[1]);
  connect(Idle1.outPort[1], Idle_Charge.inPort);
  connect(Idle_Charge.outPort, Charge.inPort[1]);
  connect(Charge_CBalancing.inPort, Charge.outPort[1]);
  connect(idle2.outPort[1], Idle_Discharge.inPort);
  connect(Idle_Discharge.outPort, Discharge.inPort[2]);
```

```
  connect(Charge_CBalancing.outPort, idle2.inPort[1]);
end Controller_NoBalancing4C_SG;

model Controller_ShuntResistor_SG
  "Shunt Resistor Balancing controller State graph based."

  Interfaces.VoltageFeedback voltageFeedback
    "Voltage feedback from balancing module.";
  Interfaces.ControlSignalBalancing_Shunt controlBalancing
    "Control signal to balancing module.";
  Interfaces.ControlSignal controlLoad
    "Control signal to charge/discharge module.";

  parameter SI.Time t_Start=100 "Idle time between phases";
  parameter SI.Time t_Idle=500 "Idle time between phases";
  parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
  parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
  parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";

  Modelica_StateGraph2.Step Init(initialStep=true, nOut=1);
  Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
  Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Init_Discharge(delayedTransition=true,
      waitTime=t_Start);
  Modelica_StateGraph2.Transition Discharge_Idle(condition=min(
        voltageFeedback.V_cell) <= Vmin + margin);
  Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
      waitTime=t_Idle);
  Modelica_StateGraph2.Step Charge(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Charge_CBalancing(condition=max(
        voltageFeedback.V_cell) >= Vmax - margin);
  Modelica_StateGraph2.Step idle2(nOut=1, nIn=1);
  Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
      waitTime=t_Idle);
equation
  //Charge/discharge mode sequence
  controlLoad.mode = if Discharge.active then -1 elseif Charge.active then 1
     else 0;

  //Balancing signal
  controlBalancing.OnOff = Charge.active;

  connect(Init.outPort[1], Init_Discharge.inPort);
  connect(Init_Discharge.outPort, Discharge.inPort[1]);
  connect(Discharge.outPort[1], Discharge_Idle.inPort);
```

```
  connect(Discharge_Idle.outPort, Idle1.inPort[1]);
  connect(Idle1.outPort[1], Idle_Charge.inPort);
  connect(Idle_Charge.outPort, Charge.inPort[1]);
  connect(Charge_CBalancing.inPort, Charge.outPort[1]);
  connect(idle2.outPort[1], Idle_Discharge.inPort);
  connect(Idle_Discharge.outPort, Discharge.inPort[2]);

  connect(Charge_CBalancing.outPort, idle2.inPort[1]);
end Controller_ShuntResistor_SG;

model Controller_SwitchedResistor_SG
  "Switched Resistor Balancing controller State graph based."

  Interfaces.VoltageFeedback voltageFeedback
    "Voltage feedback from balancing module.";
  Interfaces.ControlSignal controlLoad
    "Control signal to charge/discharge module.";

  parameter SI.Time t_Start=100 "Idle time between phases";
  parameter SI.Time t_Idle=500 "Idle time between phases";
  parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
  parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
  parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";
  parameter SI.Voltage dV_on=0.1 "Start balancing when cell exceeds min+X";
  parameter SI.Voltage dV_off=0.05 "Stop balancing when all within min+Y";
  parameter SI.Voltage eps=0.04 "Hysteresis buffer to avoid chattering";
  parameter SI.Time t_Bal=10 "Duration to hold balancing state";

  discrete Boolean OnOffMem[4](start={false,false,false,false});

  Modelica_StateGraph2.Step Init(initialStep=true, nOut=1);
  Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
  Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Init_Discharge(delayedTransition=true,
      waitTime=t_Start);
  Modelica_StateGraph2.Transition Discharge_Idle(condition=min(
        voltageFeedback.V_cell) <= Vmin + margin);
  Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
      waitTime=t_Idle);
  Modelica_StateGraph2.Step Charge(nIn=2, nOut=2);
  Modelica_StateGraph2.Transition Charge_CBalancing(condition=max(
        voltageFeedback.V_cell) - min(voltageFeedback.V_cell) >= dV_on);
  Modelica_StateGraph2.Step idle2(nOut=1, nIn=2);
  Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
      waitTime=t_Idle);
```

```
  Modelica_StateGraph2.Step CheckBalancing(nIn=2, nOut=1);
  Modelica_StateGraph2.Transition CBalancing_Charge(
    condition=(max(voltageFeedback.V_cell) - min(voltageFeedback.V_cell) <=
        dV_off) and max(voltageFeedback.V_cell) < Vmax - margin,
    delayedTransition=true,
    waitTime=1);
  Interfaces.ControlSignalBalancing_Switched controlSignalBalancing_Switched
    "Control signal to balancing module";
  Modelica_StateGraph2.Transition Charge_Idle(condition=
  max(voltageFeedback.V_cell)
        >= Vmax - margin);
  Modelica_StateGraph2.Step ExecuteBalancing(nOut=3, nIn=1);
  Modelica_StateGraph2.Transition T1(delayedTransition=true, waitTime=1);
  Modelica_StateGraph2.Transition Execute_Check(delayedTransition=true,
      waitTime=t_Bal);
  Modelica_StateGraph2.Transition Execute_Idle(condition=
  max(voltageFeedback.V_cell)
        >= Vmax - margin);
equation

  // Charge/discharge mode sequence
  controlLoad.mode = if Discharge.active then -1 elseif Charge.active or
    CheckBalancing.active or ExecuteBalancing.active then 1 else 0;

  // Per-cell balancing state machine with hysteresis
  for i in 1:4 loop
    OnOffMem[i] = if CheckBalancing.active then if voltageFeedback.V_cell[i] >
      min(voltageFeedback.V_cell) + dV_off then true elseif
      voltageFeedback.V_cell[
      i] < min(voltageFeedback.V_cell) + dV_off - eps then false else pre(
      OnOffMem[i]) elseif ExecuteBalancing.active then pre(OnOffMem[i])
      else false;
    // Only update state during CheckBalancing, hold previous value during
    balancing, disable outside balancing modes
  end for;

  // Assign balancing signal
  controlSignalBalancing_Switched.OnOff = OnOffMem;

  connect(Init.outPort[1], Init_Discharge.inPort);
  connect(Init_Discharge.outPort, Discharge.inPort[1]);
  connect(Discharge.outPort[1], Discharge_Idle.inPort);
  connect(Discharge_Idle.outPort, Idle1.inPort[1]);
  connect(Idle1.outPort[1], Idle_Charge.inPort);
  connect(Idle_Charge.outPort, Charge.inPort[1]);
```

```
  connect(Charge_CBalancing.inPort, Charge.outPort[1]);
  connect(idle2.outPort[1], Idle_Discharge.inPort);
  connect(Idle_Discharge.outPort, Discharge.inPort[2]);
  connect(Charge_CBalancing.outPort, CheckBalancing.inPort[1]);
  connect(CBalancing_Charge.outPort, Charge.inPort[2]);
  connect(Charge.outPort[2], Charge_Idle.inPort);
  connect(Charge_Idle.outPort, idle2.inPort[1]);
  connect(ExecuteBalancing.outPort[1], Execute_Check.inPort);
  connect(ExecuteBalancing.outPort[2], Execute_Idle.inPort);
  connect(ExecuteBalancing.outPort[3], CBalancing_Charge.inPort);
  connect(Execute_Check.outPort, CheckBalancing.inPort[2]);
  connect(Execute_Idle.outPort, idle2.inPort[2]);
  connect(T1.outPort, ExecuteBalancing.inPort[1]);
  connect(CheckBalancing.outPort[1], T1.inPort);
end Controller_SwitchedResistor_SG;

model Controller_SingleCapacitor_SG
  "State-graph controller for single-capacitor balancing."

  parameter SI.Time t_Start=100 "Idle time between phases";
  parameter SI.Time t_Idle=500 "Idle time between phases";
  parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
  parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
  parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";
  parameter SI.Voltage dV_on=0.01 "Start balancing when cell exceeds min+X";
  parameter SI.Time t_Balance=100 "Idle time between phases";
  discrete Boolean comm_flip(start=true);
  discrete Integer idx_max "Maximum value index", idx_min
    "Minimum value index";
  parameter SI.Time T_commute=0.2 "Commuter switching interval";
  discrete Boolean OnOff_max[5] "Switch pattern of high voltage cell",
    OnOff_min[5] "Switch pattern of low voltage cell";
  discrete Integer com_max[2] "Commutator pattern of high voltage cell",
    com_min[2] "Commutator pattern of low voltage cell";

  constant Boolean switchPattern[4, 5]=[true, true, false, false, false; false,
      true, true, false, false; false, false, true, true, false; false, false,
      false, true, true] "Switch pattern per cell index";

  constant Integer comPattern[4, 2]=[0, 1; 1, 0; 0, 1; 1, 0]
    "Commutator pattern per cell index";

  Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
  Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Discharge_Idle(condition=min(
```

```
        voltageFeedback.V_cell) <= Vmin + margin);
  Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
      waitTime=t_Idle);
  Modelica_StateGraph2.Step Charge(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Charge_CBalancing(condition=max(
        voltageFeedback.V_cell) >= Vmax - margin);
  Modelica_StateGraph2.Step idle2(nOut=1, nIn=1);
  Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
      waitTime=t_Idle);
  Interfaces.VoltageFeedback voltageFeedback
    "Voltage feedback from balancing module.";
  Interfaces.ControlSignal controlLoad
    "Control signal to charge/discharge module.";
  Interfaces.ControlSignalBalancing_Single controlSignalBalancing_Single
    "Control signal to balancing module.";
  Modelica_StateGraph2.Step CheckVoltage(nOut=1, nIn=3);
  Modelica_StateGraph2.Transition Check_Execute(condition=(max(
        voltageFeedback.V_cell) - min(voltageFeedback.V_cell) >= dV_on) and (
        max(voltageFeedback.V_cell) < Vmax - margin));
  Modelica_StateGraph2.Step ExecuteBalancing(nIn=1, nOut=2);
  Modelica_StateGraph2.Transition Idle_Balance(delayedTransition=true,
      waitTime=t_Balance);
  Modelica_StateGraph2.Parallel step1(
    initialStep=true,
    use_inPort=false,
    nEntry=2);
  Modelica_StateGraph2.Step Precharge(
    initialStep=false,
    nIn=1,
    nOut=1);
  Modelica_StateGraph2.Transition T1(delayedTransition=true, waitTime=1000);
  Modelica_StateGraph2.Transition T2(
    condition=max(voltageFeedback.V_cell) >= Vmax - margin,
    delayedTransition=true,
    waitTime=0.1);
equation
  // Control to charge/discharge
  controlLoad.mode = if Discharge.active then -1 elseif Charge.active or
    ExecuteBalancing.active then 1 else 0;

  // Update max/min logic and assign patterns before balancing state
  when Check_Execute.fire then
    idx_max = Functions.argmax(voltageFeedback.V_cell);
    idx_min = Functions.argmin(voltageFeedback.V_cell);
```

```
    // Assing switching patterns for max and min
    for i in 1:5 loop
      OnOff_max[i] = switchPattern[idx_max, i];
      OnOff_min[i] = switchPattern[idx_min, i];
    end for;

    for j in 1:2 loop
      com_max[j] = comPattern[idx_max, j];
      com_min[j] = comPattern[idx_min, j];
    end for;
  end when;

  // Enable switch 5 during Precharge and assign control signal when balancing
  for i in 1:5 loop
    controlSignalBalancing_Single.OnOff[i] = if initial() then (i == 5) else
      if Precharge.active then (i == 5) else if ExecuteBalancing.active then
      if comm_flip then OnOff_max[i] else OnOff_min[i] else false;
  end for;

  //Switch commuter with given period
  controlSignalBalancing_Single.Com[1] = if ExecuteBalancing.active then if
    comm_flip then com_max[1] else com_min[1] else 1;
  controlSignalBalancing_Single.Com[2] = if ExecuteBalancing.active then if
    comm_flip then com_max[2] else com_min[2] else 0;
  // Periodic commutation switching
  when sample(0, T_commute) and ExecuteBalancing.active then
    comm_flip = not pre(comm_flip);
  end when;

  connect(Discharge.outPort[1], Discharge_Idle.inPort);
  connect(Discharge_Idle.outPort, Idle1.inPort[1]);
  connect(Idle1.outPort[1], Idle_Charge.inPort);
  connect(Idle_Charge.outPort, Charge.inPort[1]);
  connect(Charge_CBalancing.inPort, Charge.outPort[1]);
  connect(idle2.outPort[1], Idle_Discharge.inPort);
  connect(Idle_Discharge.outPort, Discharge.inPort[1]);
  connect(Charge_CBalancing.outPort, idle2.inPort[1]);
  connect(CheckVoltage.outPort[1], Check_Execute.inPort);
  connect(Check_Execute.outPort, ExecuteBalancing.inPort[1]);
  connect(ExecuteBalancing.outPort[1], Idle_Balance.inPort);
  connect(Idle_Balance.outPort, CheckVoltage.inPort[1]);

  connect(step1.entry[1], Discharge.inPort[2]);
  connect(step1.entry[2], Precharge.inPort[1]);
  connect(T1.outPort, CheckVoltage.inPort[2]);
```

```
  connect(Precharge.outPort[1], T1.inPort);
  connect(ExecuteBalancing.outPort[2], T2.inPort);
  connect(T2.outPort, CheckVoltage.inPort[3]);
end Controller_SingleCapacitor_SG;

model Controller_SingleInductor_SG
  "State-graph controller for single-inductor balancing."

  parameter SI.Time t_Start=100 "Idle time between phases";
  parameter SI.Time t_Idle=500 "Idle time between phases";
  parameter SI.Voltage Vmin=3.2 "Minimum allowed voltage";
  parameter SI.Voltage Vmax=4.2 "Maximum allowed voltage";
  parameter SI.Voltage margin=0.05 "Voltage marging for Vmax/Vmin";
  parameter SI.Voltage dV_on=0.01 "Start balancing when cell exceeds min+X";
  parameter SI.Time t_Balance=100 "Idle time between phases";
  discrete Boolean comm_flip(start=true);
  discrete Boolean flipCommutators
    "True if commutation pattern needs to alternate.";
  discrete Integer idx_max "Maximum value index", idx_min
    "Minimum value index";
  parameter SI.Time T_commute=0.2 "Commuter switching interval";
  discrete Boolean OnOff_max[5] "Switch pattern of high voltage cell",
    OnOff_min[5] "Switch pattern of low voltage cell";
  discrete Integer com_max[2] "Commutator pattern of high voltage cell",
    com_min[2] "Commutator pattern of low voltage cell";

  constant Boolean switchPattern[4, 5]=[true, true, false, false, false; false,
      true, true, false, false; false, false, true, true, false; false, false,
      false, true, true] "Switch pattern per cell index";

  constant Integer comPattern[4, 2]=[1, 0; 1, 0; 1, 0; 1, 0]
    "Commutator pattern per cell index";

  Modelica_StateGraph2.Step Discharge(nIn=2, nOut=1);
  Modelica_StateGraph2.Step Idle1(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Discharge_Idle(condition=min(
        voltageFeedback.V_cell) <= Vmin + margin);
  Modelica_StateGraph2.Transition Idle_Charge(delayedTransition=true,
      waitTime=t_Idle);
  Modelica_StateGraph2.Step Charge(nIn=1, nOut=1);
  Modelica_StateGraph2.Transition Charge_CBalancing(condition=max(
        voltageFeedback.V_cell) >= Vmax - margin);
  Modelica_StateGraph2.Step idle2(nOut=1, nIn=1);
  Modelica_StateGraph2.Transition Idle_Discharge(delayedTransition=true,
      waitTime=t_Idle);
```

```
    Interfaces.VoltageFeedback voltageFeedback
      "Voltage feedback from balancing module.";
    Interfaces.ControlSignal controlLoad
      "Control signal to charge/discharge module.";
    Interfaces.ControlSignalBalancing_Single controlSignalBalancing_Single
      "Control signal to balancing module.";
    Modelica_StateGraph2.Step CheckVoltage(nOut=1, nIn=3);
    Modelica_StateGraph2.Transition Check_Execute(condition=(max(
          voltageFeedback.V_cell) - min(voltageFeedback.V_cell) >= dV_on) and (
          max(voltageFeedback.V_cell) < Vmax - margin));
    Modelica_StateGraph2.Step ExecuteBalancing(nIn=1, nOut=2);
    Modelica_StateGraph2.Transition Idle_Balance(delayedTransition=true,
        waitTime=t_Balance);
    Modelica_StateGraph2.Parallel step1(
      initialStep=true,
      use_inPort=false,
      nEntry=2);
    Modelica_StateGraph2.Step Precharge(
      initialStep=false,
      nIn=1,
      nOut=1);
    Modelica_StateGraph2.Transition T1(delayedTransition=true, waitTime=1000);
    Modelica_StateGraph2.Transition T2(
      condition=max(voltageFeedback.V_cell) >= Vmax - margin,
      delayedTransition=true,
      waitTime=0.1);
  equation
    // Control to charge/discharge
    controlLoad.mode = if Discharge.active then -1 elseif Charge.active or
      ExecuteBalancing.active then 1 else 0;

    // Update max/min logic and assign patterns before balancing state
    when Check_Execute.fire then
      idx_max = Functions.argmax(voltageFeedback.V_cell);
      idx_min = Functions.argmin(voltageFeedback.V_cell);

      // Assing switching patterns for max and min
      for i in 1:5 loop
        OnOff_max[i] = switchPattern[idx_max, i];
        OnOff_min[i] = switchPattern[idx_min, i];
      end for;

      // Assign base commutator pattern
      com_max = {1,0};
      com_min = {0,1};
```

```
      // Determine whether commutator flipping is needed
      flipCommutators = abs(idx_max - idx_min) == 2;

    end when;

    // Enable switch 5 during Precharge and assign control signal when balancing
    for i in 1:5 loop
      controlSignalBalancing_Single.OnOff[i] = if initial() then (i == 5 or i ==
        4) else if Precharge.active then (i == 5) else if ExecuteBalancing.active
         then if comm_flip then OnOff_max[i] else OnOff_min[i] else false;
    end for;

    //Switch commuter with given period
    controlSignalBalancing_Single.Com[1] = if initial() then 1 else if
      ExecuteBalancing.active then if flipCommutators then if comm_flip then
      com_max[1] else com_min[1] else com_max[1] else 0;
    controlSignalBalancing_Single.Com[2] = if initial() then 0 else if
      ExecuteBalancing.active then if flipCommutators then if comm_flip then
      com_max[2] else com_min[2] else com_max[2] else 0;
    // Periodic commutation switching
    when sample(0, T_commute) and ExecuteBalancing.active then
      comm_flip = not pre(comm_flip);
    end when;

    connect(Discharge.outPort[1], Discharge_Idle.inPort);
    connect(Discharge_Idle.outPort, Idle1.inPort[1]);
    connect(Idle1.outPort[1], Idle_Charge.inPort);
    connect(Idle_Charge.outPort, Charge.inPort[1]);
    connect(Charge_CBalancing.inPort, Charge.outPort[1]);
    connect(idle2.outPort[1], Idle_Discharge.inPort);
    connect(Idle_Discharge.outPort, Discharge.inPort[1]);
    connect(Charge_CBalancing.outPort, idle2.inPort[1]);
    connect(CheckVoltage.outPort[1], Check_Execute.inPort);
    connect(Check_Execute.outPort, ExecuteBalancing.inPort[1]);
    connect(ExecuteBalancing.outPort[1], Idle_Balance.inPort);
    connect(Idle_Balance.outPort, CheckVoltage.inPort[1]);
    connect(step1.entry[1], Discharge.inPort[2]);
    connect(step1.entry[2], Precharge.inPort[1]);
    connect(T1.outPort, CheckVoltage.inPort[2]);
    connect(Precharge.outPort[1], T1.inPort);
    connect(T2.outPort, CheckVoltage.inPort[3]);
    connect(ExecuteBalancing.outPort[2], T2.inPort);
  end Controller_SingleInductor_SG;
end Control_Structures;
```

## A.8. Code Package Examples

Demonstrates the use of the library via testbenches that simulate real-world operation of second-life battery packs under different control and balancing scenarios.

```
package Examples "Collection of simulation testbenches demonstrating the behavior
and performance of various battery management and balancing strategies
for a 4-cell lithium-ion pack."

  model Example_NoBalancing1CSG
    "Simulation of a single-cell system with no cell balancing mechanism."

    // Add parameter loaders for each cell
    Functions.ECM_ParameterLoader_1Thv paramLoader1;

    parameter Real alpha[4]={1,1,1,1} "Derating aging values";
    parameter Real beta[4]={1,1,1,1} "Derating unbalance values";


    Components.Ground ground;
    ECM_Structures.ECM_Thevenin_SL_Vbf battery(
      Q=3600,
      eta=1,
      alpha_Q=alpha[1],
      alpha_R0=alpha[2],
      alpha_R1=alpha[3],
      alpha_C1=alpha[4],
      beta_Q=beta[1],
      beta_R0=beta[2],
      beta_R1=beta[3],
      beta_C1=beta[4]);
    Control_Structures.Controller_NoBalancing1C_SG controller_NoBalancing1C_SG;

    Balancing_Structures.ChargeDischarge chargeDischarge;

  equation

    // Connect SOC feedback to param loaders
    paramLoader1.SOC = battery.ocv.z;

    // Connect outputs to battery inputs
    battery.OCV = paramLoader1.OCV;
    battery.R0 = paramLoader1.R0;
    battery.R1 = paramLoader1.R1;
    battery.C1 = paramLoader1.C1;
```

```
  connect(chargeDischarge.n, ground.p);
  connect(battery.n, ground.p);
  connect(battery.p, chargeDischarge.p);
  connect(controller_NoBalancing1C_SG.controlLoad, chargeDischarge.controlInput);
  connect(controller_NoBalancing1C_SG.voltageFeedback, battery.voltageFeedback);


end Example_NoBalancing1CSG;

model Example_NoBalancing4CSG
  "Simulation of a 4-cell battery pack without any balancing strategy."

  // Add parameter loaders for each cell
  Functions.ECM_ParameterLoader_1Thv paramLoader1;
  Functions.ECM_ParameterLoader_1Thv paramLoader2;
  Functions.ECM_ParameterLoader_1Thv paramLoader3;
  Functions.ECM_ParameterLoader_1Thv paramLoader4;

  parameter Real alpha[4, 4]=[1.5, 1.5, 0.8, 0.8; 1.5, 1.5, 0.8, 0.8; 1.5, 1.5,
      0.8, 0.8; 1.5, 1.5, 0.8, 0.8] "Derating aging values";
  parameter Real beta[4, 4]=[1.05, 1.05, 0.97, 0.97; 1.0167, 1.0167, 0.99, 0.99;
      0.9833, 0.9833, 1.01, 1.01; 0.95, 0.95, 1.03, 1.03]
    "Derating unbalance values";


  Components.Ground ground;
  Cell_Packs.Battery4Cell_Bal_Vfb battery(
    Q={3600,3600,3600,3600},
    eta={1,1,1,1},
    alpha=alpha,
    beta=beta);
  Control_Structures.Controller_NoBalancing4C_SG controller_NoBalancing4C_SG;

  Balancing_Structures.ChargeDischarge chargeDischarge;

equation

  // Connect SOC feedback to param loaders
  paramLoader1.SOC = battery.Cell1.ocv.z;
  paramLoader2.SOC = battery.Cell2.ocv.z;
  paramLoader3.SOC = battery.Cell3.ocv.z;
  paramLoader4.SOC = battery.Cell4.ocv.z;

  // Connect outputs to battery inputs
```

```
  battery.OCV = {paramLoader1.OCV,paramLoader2.OCV,paramLoader3.OCV,
    paramLoader4.OCV};
  battery.R0 = {paramLoader1.R0,paramLoader2.R0,paramLoader3.R0,
  paramLoader4.R0};
  battery.R1 = {paramLoader1.R1,paramLoader2.R1,paramLoader3.R1,
  paramLoader4.R1};
  battery.C1 = {paramLoader1.C1,paramLoader2.C1,paramLoader3.C1,
  paramLoader4.C1};

  connect(controller_NoBalancing4C_SG.voltageFeedback, battery.voltageFeedback);
  connect(controller_NoBalancing4C_SG.controlLoad, chargeDischarge.controlInput);
  connect(chargeDischarge.n, ground.p);
  connect(chargeDischarge.n, battery.n);
  connect(battery.p, chargeDischarge.p);
end Example_NoBalancing4CSG;

model Example_ShuntBalancing4CSG
  "Simulation of a 4-cell battery pack using passive shunt resistor balancing."

  parameter Real alpha[4, 4]=[1.5, 1.5, 0.8, 0.8; 1.5, 1.5, 0.8, 0.8; 1.5, 1.5,
      0.8, 0.8; 1.5, 1.5, 0.8, 0.8] "Derating aging values";
  parameter Real beta[4, 4]=[1.05, 1.05, 0.97, 0.97; 1.0167, 1.0167, 0.99, 0.99;
      0.9833, 0.9833, 1.01, 1.01; 0.95, 0.95, 1.03, 1.03]
    "Derating unbalance values";


  // Add parameter loaders for each cell
  Functions.ECM_ParameterLoader_1Thv paramLoader1;
  Functions.ECM_ParameterLoader_1Thv paramLoader2;
  Functions.ECM_ParameterLoader_1Thv paramLoader3;
  Functions.ECM_ParameterLoader_1Thv paramLoader4;

  Components.Ground ground;
  Control_Structures.Controller_ShuntResistor_SG controller_ShuntResistor_SG(
      t_Idle=500);
  Balancing_Structures.ChargeDischarge chargeDischarge;
  Cell_Packs.Battery4Cell_Bal battery(
    Q={3600,3600,3600,3600},
    eta={1,1,1,1},
    alpha=alpha,
    beta=beta);
  Balancing_Structures.ShuntResistorBalancing_4Cell balancingBlock(R_shunt=100);

equation
  // Connect SOC feedback to param loaders
```

```
    paramLoader1.SOC = battery.Cell1.ocv.z;
    paramLoader2.SOC = battery.Cell2.ocv.z;
    paramLoader3.SOC = battery.Cell3.ocv.z;
    paramLoader4.SOC = battery.Cell4.ocv.z;

    // Connect outputs to battery inputs
    battery.OCV = {paramLoader1.OCV,paramLoader2.OCV,paramLoader3.OCV,
      paramLoader4.OCV};
    battery.R0 = {paramLoader1.R0,paramLoader2.R0,paramLoader3.R0,
    paramLoader4.R0};
    battery.R1 = {paramLoader1.R1,paramLoader2.R1,paramLoader3.R1,
    paramLoader4.R1};
    battery.C1 = {paramLoader1.C1,paramLoader2.C1,paramLoader3.C1,
    paramLoader4.C1};
    connect(controller_ShuntResistor_SG.voltageFeedback,
    balancingBlock.voltageFeedback);
    connect(controller_ShuntResistor_SG.controlBalancing,
    balancingBlock.controlInput);
    connect(controller_ShuntResistor_SG.controlLoad,
    chargeDischarge.controlInput);
    connect(battery.p, chargeDischarge.p);
    connect(battery.n, chargeDischarge.n);
    connect(ground.p, battery.n);
    connect(balancingBlock.pin2, battery.node_1_2);
    connect(balancingBlock.pin4, battery.node_3_4);

    connect(battery.n, balancingBlock.pin5);
    connect(battery.p, balancingBlock.pin1);
    connect(battery.node_2_3, balancingBlock.pin3);

end Example_ShuntBalancing4CSG;

model Example_SwitchedBalancing4CSG
  "Simulation of a 4-cell battery pack employing switched resistor balancing."

  // Add parameter loaders for each cell
  Functions.ECM_ParameterLoader_1Thv paramLoader1;
  Functions.ECM_ParameterLoader_1Thv paramLoader2;
  Functions.ECM_ParameterLoader_1Thv paramLoader3;
  Functions.ECM_ParameterLoader_1Thv paramLoader4;

  parameter Real alpha[4, 4]=[1.5, 1.5, 0.8, 0.8; 1.5, 1.5, 0.8, 0.8; 1.5, 1.5,
      0.8, 0.8; 1.5, 1.5, 0.8, 0.8] "Derating aging values";
  parameter Real beta[4, 4]=[1.05, 1.05, 0.97, 0.97; 1.0167, 1.0167, 0.99, 0.99;
      0.9833, 0.9833, 1.01, 1.01; 0.95, 0.95, 1.03, 1.03]
```

```
    "Derating unbalance values";

  Components.Ground ground;
  Balancing_Structures.SwitchedResistorBalancing_4Cell
    switchedResistorBalancing_4Cell;
  Control_Structures.Controller_SwitchedResistor_SG
    controller_SwitchedResistor_SG;

  Balancing_Structures.ChargeDischarge chargeDischarge;
  Cell_Packs.Battery4Cell_Bal battery(
    Q={3600,3600,3600,3600},
    eta={1,1,1,1},
    alpha=alpha,
    beta=beta);
equation
  // Connect SOC feedback to param loaders
  paramLoader1.SOC = battery.Cell1.ocv.z;
  paramLoader2.SOC = battery.Cell2.ocv.z;
  paramLoader3.SOC = battery.Cell3.ocv.z;
  paramLoader4.SOC = battery.Cell4.ocv.z;

  // Connect outputs to battery inputs
  battery.OCV = {paramLoader1.OCV,paramLoader2.OCV,paramLoader3.OCV,
    paramLoader4.OCV};
  battery.R0 = {paramLoader1.R0,paramLoader2.R0,paramLoader3.R0,
  paramLoader4.R0};
  battery.R1 = {paramLoader1.R1,paramLoader2.R1,paramLoader3.R1,
  paramLoader4.R1};
  battery.C1 = {paramLoader1.C1,paramLoader2.C1,paramLoader3.C1,
  paramLoader4.C1};

  connect(battery.p, chargeDischarge.p);
  connect(battery.n, chargeDischarge.n);
  connect(ground.p, battery.n);
  connect(battery.node_1_2, switchedResistorBalancing_4Cell.pin2);
  connect(battery.node_2_3, switchedResistorBalancing_4Cell.pin3);
  connect(battery.node_3_4, switchedResistorBalancing_4Cell.pin4);
  connect(switchedResistorBalancing_4Cell.pin5, battery.n);
  connect(switchedResistorBalancing_4Cell.pin1, battery.p);
  connect(controller_SwitchedResistor_SG.voltageFeedback,
    switchedResistorBalancing_4Cell.voltageFeedback);
  connect(controller_SwitchedResistor_SG.controlSignalBalancing_Switched,
    switchedResistorBalancing_4Cell.controlInput);
  connect(controller_SwitchedResistor_SG.controlLoad,
  chargeDischarge.controlInput);
```

```
end Example_SwitchedBalancing4CSG;

model Example_SingleCapacitor4CSG
  "Simulation of a 4-cell battery pack with single-capacitor active balancing."

  // Relevant Parameters
  parameter Real alpha[4, 4]=[1.5, 1.5, 0.8, 0.8; 1.5, 1.5, 0.8, 0.8; 1.5, 1.5,
      0.8, 0.8; 1.5, 1.5, 0.8, 0.8] "Derating aging values";
  parameter Real beta[4, 4]=[1.05, 1.05, 0.97, 0.97; 1.0167, 1.0167, 0.99, 0.99;
      0.9833, 0.9833, 1.01, 1.01; 0.95, 0.95, 1.03, 1.03]
    "Derating unbalance values";
  parameter SI.Capacitance C=1e-3 "Capacitance for energy transfer";
  parameter SI.Resistance R_on=0.1 "On resistance";
  parameter SI.Resistance R_bleed=1e9;
  parameter SI.Time t_Balance=10 "Idle time between phases (s)";
  parameter SI.Time T_commute=0.02 "Commuter switching interval";

  // Add parameter loaders for each cell
  Functions.ECM_ParameterLoader_1Thv paramLoader1;
  Functions.ECM_ParameterLoader_1Thv paramLoader2;
  Functions.ECM_ParameterLoader_1Thv paramLoader3;
  Functions.ECM_ParameterLoader_1Thv paramLoader4;

  Balancing_Structures.ChargeDischarge chargeDischarge;
  Cell_Packs.Battery4Cell_Bal battery(
    Q={3600,3600,3600,3600},
    eta={1,1,1,1},
    alpha=alpha,
    beta=beta);
  Components.Ground ground;
  Balancing_Structures.SingleCapacitorBalancing_4Cell
    singleCapacitorBalancing_4Cell(
    C=C,
    R_on=R_on,
    R_bleed=R_bleed);
  Control_Structures.Controller_SingleCapacitor_SG
    controller_SingleCapacitor_SG(t_Balance=t_Balance, T_commute=T_commute);
equation

  // Connect SOC feedback to param loaders
  paramLoader1.SOC = battery.Cell1.ocv.z;
  paramLoader2.SOC = battery.Cell2.ocv.z;
  paramLoader3.SOC = battery.Cell3.ocv.z;
  paramLoader4.SOC = battery.Cell4.ocv.z;
```

```
  // Connect outputs to battery inputs
  battery.OCV = {paramLoader1.OCV,paramLoader2.OCV,paramLoader3.OCV,
    paramLoader4.OCV};
  battery.R0 = {paramLoader1.R0,paramLoader2.R0,paramLoader3.R0,
  paramLoader4.R0};
  battery.R1 = {paramLoader1.R1,paramLoader2.R1,paramLoader3.R1,
  paramLoader4.R1};
  battery.C1 = {paramLoader1.C1,paramLoader2.C1,paramLoader3.C1,
  paramLoader4.C1};
  connect(battery.p, chargeDischarge.p);
  connect(battery.n, chargeDischarge.n);
  connect(ground.p, battery.n);
  connect(singleCapacitorBalancing_4Cell.pin2, battery.node_1_2);
  connect(singleCapacitorBalancing_4Cell.pin3, battery.node_2_3);
  connect(singleCapacitorBalancing_4Cell.pin4, battery.node_3_4);
  connect(singleCapacitorBalancing_4Cell.pin5, battery.n);
  connect(singleCapacitorBalancing_4Cell.pin1, battery.p);
  connect(controller_SingleCapacitor_SG.controlLoad,
  chargeDischarge.controlInput);
  connect(controller_SingleCapacitor_SG.controlSignalBalancing_Single,
    singleCapacitorBalancing_4Cell.controlSignal);
  connect(controller_SingleCapacitor_SG.voltageFeedback,
    singleCapacitorBalancing_4Cell.voltageFeedback);
end Example_SingleCapacitor4CSG;

model Example_SingleInductor4CSG
  "Simulation of a 4-cell battery pack using single-inductor balancing."

  // Relevant Parameters
  parameter Real alpha[4, 4]=[1.5, 1.5, 0.8, 0.8; 1.5, 1.5, 0.8, 0.8; 1.5, 1.5,
      0.8, 0.8; 1.5, 1.5, 0.8, 0.8] "Derating aging values";
  parameter Real beta[4, 4]=[1.05, 1.05, 0.97, 0.97; 1.0167, 1.0167, 0.99, 0.99;
      0.9833, 0.9833, 1.01, 1.01; 0.95, 0.95, 1.03, 1.03]
    "Derating unbalance values";
  parameter SI.Inductance L=0.04 "Inductance for energy transfer";
  parameter SI.Resistance R_on=0.2 "On resistance";
  parameter SI.Resistance R_bleed=0.01;
  parameter SI.Time t_Balance=10 "Idle time between phases (s)";
  parameter SI.Time T_commute=0.04 "Commuter switching interval";

  // Add parameter loaders for each cell
  Functions.ECM_ParameterLoader_1Thv paramLoader1;
  Functions.ECM_ParameterLoader_1Thv paramLoader2;
  Functions.ECM_ParameterLoader_1Thv paramLoader3;
  Functions.ECM_ParameterLoader_1Thv paramLoader4;
```

```
  Balancing_Structures.ChargeDischarge chargeDischarge;
  Cell_Packs.Battery4Cell_Bal battery(
    Q={3600,3600,3600,3600},
    eta={1,1,1,1},
    alpha=alpha,
    beta=beta);
  Components.Ground ground;
  Balancing_Structures.SingleInductorBalancing_4Cell
    singleInductorBalancing_4Cell(
    L=L,
    R_on=R_on,
    R_bleed=R_bleed);
  Control_Structures.Controller_SingleInductor_SG
    controller_SingleInductor_SG(t_Balance=t_Balance, T_commute=T_commute);
equation

  // Connect SOC feedback to param loaders
  paramLoader1.SOC = battery.Cell1.ocv.z;
  paramLoader2.SOC = battery.Cell2.ocv.z;
  paramLoader3.SOC = battery.Cell3.ocv.z;
  paramLoader4.SOC = battery.Cell4.ocv.z;

  // Connect outputs to battery inputs
  battery.OCV = {paramLoader1.OCV,paramLoader2.OCV,paramLoader3.OCV,
    paramLoader4.OCV};
  battery.R0 = {paramLoader1.R0,paramLoader2.R0,paramLoader3.R0,
  paramLoader4.R0};
  battery.R1 = {paramLoader1.R1,paramLoader2.R1,paramLoader3.R1,
  paramLoader4.R1};
  battery.C1 = {paramLoader1.C1,paramLoader2.C1,paramLoader3.C1,
  paramLoader4.C1};
  connect(battery.p, chargeDischarge.p);
  connect(battery.n, chargeDischarge.n);
  connect(ground.p, battery.n);
  connect(controller_SingleInductor_SG.voltageFeedback,
    singleInductorBalancing_4Cell.voltageFeedback);
  connect(controller_SingleInductor_SG.controlLoad, chargeDischarge.controlInput);
  connect(singleInductorBalancing_4Cell.pin5, battery.n);
  connect(battery.node_3_4, singleInductorBalancing_4Cell.pin4);
  connect(battery.node_2_3, singleInductorBalancing_4Cell.pin3);
  connect(battery.node_1_2, singleInductorBalancing_4Cell.pin2);
  connect(singleInductorBalancing_4Cell.pin1, battery.p);
  connect(controller_SingleInductor_SG.controlSignalBalancing_Single,
  singleInductorBalancing_4Cell.controlSignal);
```

```
  end Example_SingleInductor4CSG;

end Examples;
```

# Appendix B

# User Documentation of Battery_Balancing Library

The following appendix contains the auto-generated documentation of the Battery_Balancing library developed as part of this thesis. The documentation includes graphical representations, structural hierarchy, and brief descriptions of components, extracted using the Dymola export functionality.

# Battery_Balancing

**Electrochemical Model Library for Lithium-Ion Cell and Pack Simulation with Balancing and Aging Dynamics.**

## Information

This package contains a structured set of models for the simulation of lithium-ion battery systems based on Equivalent Circuit Models (ECM). It includes both single-cell and multi-cell pack architectures incorporating thermal and electrical phenomena, aging derating, and various balancing topologies.

Key Components:

- ECM Structures: Parameterizable ECM variants including single-cell Thevenin models with optional second-life degradation modeling (via alpha and beta derating coefficients).
- Control Structures: State machine-based controllers that govern charge-discharge cycles, with or without balancing. Includes variants for shunt resistors, switched resistors, and energy-transfer-based active balancing (capacitor and inductor).
- Balancing Structures: Circuit modules implementing physical topologies for different balancing strategies. Interfaces support voltage sensing, control input, and energy flow routing across the battery.
- Cell Packs: Aggregated battery assemblies including 4-cell configurations with access to internal nodes for balancing integration. Each cell can be individually parameterized to simulate realistic inhomogeneities.
- Examples: Test benches that combine all the above modules for simulation and analysis of different balancing strategies under controlled load cycles and aging conditions.

Purpose:

The ECM_models library enables comprehensive and modular simulation of lithium-ion battery packs under aging and balancing scenarios. It supports comparative evaluation of balancing algorithms, investigates performance degradation due to aging and mismatch, and provides a physical basis for algorithm validation and controller prototyping within a Modelica environment."

## Package Content

| Name | Description |
|---|---|
| Interfaces | Interface package for external connection between components. |
| Components | Models of fundamental electrical components for ECM-based battery modeling. |
| Functions | Auxiliary function blocks for parameter interpolation and control logic |
| ECM_Structures | Package of ECM structures for lithium-ion cell modeling. |
| Cell_Packs | Create packs of ECM modeled cells |
| Balancing_Structures | Balancing structures for 4-cell modules and Charge/Discharge behavior. |
| Control_Structures | Package containing charge/discharge infrastructure and cell balancing architectures for a 4-cell lithium-ion battery module. |
| Examples | Collection of simulation testbenches demonstrating the behavior and performance of various battery management and balancing strategies for a 4-cell lithium-ion pack. |

# Battery_Balancing.Interfaces

**Interface package for external connection between components.**

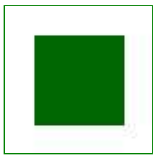
## Information

This package defines standard connector types and partial models used for consistent interconnection of electrical and control components in the battery simulation framework.

## Package Content

| Name | Description |
|---|---|
| Pin | Neutral electrical connector. |
| Pin_p | Positive electrical connector. |
| Pin_n | Negative electrical connector. |
| UnPuerto | Two-terminal base model with optional current direction logic. |
| UnPuerto_ECM | Two-terminal interface for ECM components. |
| UnPuertoTresPines | Three-terminal interface for gate-controlled devices |
| CommutatorInterface | Three-terminal interface for commutator components. |
| ControlSignal | Connector for mode control signal in charge/discharge modules. |
| ControlSignalBalancing_Shunt | Control signal connector for shunt resistor balancing. |
| ControlSignalBalancing_Switched | Control signal connector for switched resistor balancing. |
| ControlSignalBalancing_Single | Control signal connector for single capacitor or inductor balancing. |
| VoltageFeedback | Voltage feedback connector for 4-cell pack. |
| VoltageFeedback1C | Voltage feedback connector for single cell. |

# Battery_Balancing.Interfaces.Pin



**Neutral electrical connector.**

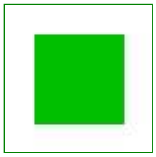
## Information

Represents an unpolarized electrical port carrying voltage u and currentu..

## Contents

| Name | Description |
|---|---|
| u | [V] |
| i | [A] |

# Battery_Balancing.Interfaces.Pin_p



**Positive electrical connector.**

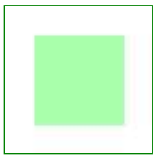
## Information

Standard port for positive terminal in electrical models, carrying voltage u and current i.

## Contents

| Name | Description |
|---|---|
| u | [V] |
| i | [A] |

## Battery_Balancing.Interfaces.Pin_n



**Negative electrical connector.**

### Information

Standard port for negative terminal in electrical models, carrying voltage u and current i.

### Contents

| Name | Description |
|---|---|
| u | [V] |
| i | [A] |

## Battery_Balancing.Interfaces.UnPuerto

**Two-terminal base model with optional current direction logic.**

### Information

Defines a standardized structure for components with a positive and a negative terminal. Includes internal voltage and current variables, and an activo flag to adapt current direction depending on component configuration.

### Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |

### Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

## Battery_Balancing.Interfaces.UnPuerto_ECM

**Two-terminal interface for ECM components.**

### Information

Simplified variant of UnPuerto used to prevent algebraic loops in ECM structures by excluding directional logic.

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |

## Battery_Balancing.Interfaces.UnPuertoTresPines

**Three-terminal interface for gate-controlled devices**

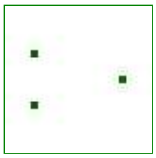
### Information

Defines the structure for MOSFET-style elements with drain, source, and gate connections. Includes internal definitions for voltages and currents across all three terminals.

### Connectors

| Name | Description |
|---|---|
| g | Gate |
| p | Drain |
| n | Source |

## Battery_Balancing.Interfaces.CommutatorInterface



**Three-terminal interface for commutator components.**

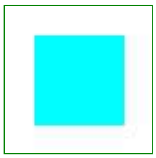
### Information

Provides structure for switch-matrix elements, allowing bidirectional current flow between three ports. Designed for use in balancing commutators or routing switches.

### Connectors

| Name | Description |
|---|---|
| b | Commute pin 1 |
| c | Commute pin 2 |
| a | Entry Pin |

## Battery_Balancing.Interfaces.ControlSignal



**Connector for mode control signal in charge/discharge modules.**

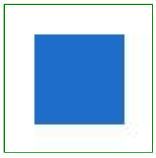
### Information

Transmits an integer mode value to define pack operating status: −1 for discharge, 0 for idle, +1 for charge (CC).

### Contents

| Name | Description |
|---|---|
| mode | mode: -1/discharge, 0/idle, +1/charge CC |

## Battery_Balancing.Interfaces.ControlSignalBalancing_Shunt



**Control signal connector for shunt resistor balancing.**

### Information

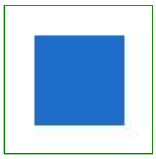
Transfers a Boolean flag to enable or disable all balancing resistors simultaneously.

### Contents

| Name | Description |
|---|---|
| OnOff | Signal On/Off all switches |

## Battery_Balancing.Interfaces.ControlSignalBalancing_Switched

**Control signal connector for switched resistor balancing.**



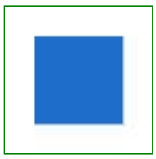
### Information

Transfers an array of Boolean values, each corresponding to the activation state of an individual cell switch.

### Contents

| Name | Description |
|---|---|
| OnOff[4] | Control signal for each switch |

## Battery_Balancing.Interfaces.ControlSignalBalancing_Single



**Control signal connector for single capacitor or inductor balancing.**

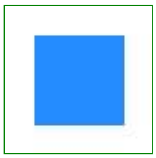
### Information

Carries a Boolean array for switch activation and an integer array for commutator configuration in active balancing systems.

### Contents

| Name | Description |
|---|---|
| OnOff[5] | Control signal for each switch |
| Com[2] | Control signal for each commutator |

## Battery_Balancing.Interfaces.VoltageFeedback



**Voltage feedback connector for 4-cell pack.**

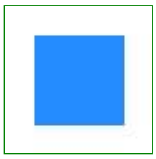
### Information

Transmits an array of cell voltages (in volts) for real-time monitoring and control.

### Contents

| Name | Description |
|---|---|
| V_cell[4] | Voltage of each cell (V) |

## Battery_Balancing.Interfaces.VoltageFeedback1C



**Voltage feedback connector for single cell.**

### Information

Transmits a single cell voltage value for one-cell configurations.

### Contents

| Name | Description |
|---|---|
| V_cell | Voltage of cell (V) |

## Battery_Balancing.Components

**Models of fundamental electrical components for ECM-based battery modeling.**

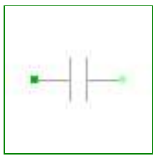
## Information

This package includes ideal and semi-ideal electrical elements required for constructing battery equivalent circuit models (ECMs) and test configurations. It provides implementations of basic components such as resistors, capacitors, inductors, controlled sources, and switches used throughout the library.

## Package Content

| Name | Description |
|---|---|
| Capacitance | Ideal linear capacitor. |
| Inductance | Ideal inductor. |
| Ground | Electrical ground |
| NMOS | Non-ideal NMOS transistor with finite on-resistance |
| OCV_source | Open-circuit voltage (OCV) source with SOC tracking. |
| Resistance | Ideal linear resistor. |
| Switch | Idealized controlled switch with on-resistance. |
| Icontrol | Externally controlled current source. |
| Vcontrol | Externally controlled voltage source. |
| Commutator | Two-way controlled commutator switch |

# Battery_Balancing.Components.Capacitance



**Ideal linear capacitor.**

## Information

Implements a two-terminal capacitor model defined by the relation i=C·du/dt. Used in RC branches of ECMs to represent transient polarization behavior and as a main component in the single capacitor balancing topology.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

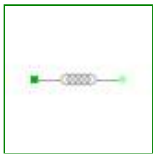
## Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |

## Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

# Battery_Balancing.Components.Inductance



**Ideal inductor.**

## Information

Two-terminal inductor defined by the relation u=L·di/dt. Applied in active balancing models where energy transfer between cells involves magnetic storage.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

### Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |

### Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

## Battery_Balancing.Components.Ground

**Electrical ground**

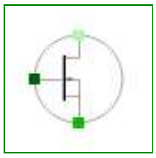
### Information

Provides a fixed zero-voltage reference point for electrical circuits.

### Connectors

| Name | Description |
|---|---|
| p | Ground pin |

## Battery_Balancing.Components.NMOS



**Non-ideal NMOS transistor with finite on-resistance**

### Information

Implements a simplified NMOS transistor model with regions for cutoff, linear, and saturation operation. Intended for realistic switching behavior in balancing circuits.

Extends from Interfaces.UnPuertoTresPines (Three-terminal interface for gate-controlled devices).

### Parameters

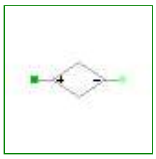
| Name | Description |
|---|---|
| Kn | Transconductance parameter |
| Lambda | Channel length modulation factor |
| Vth | Threshold voltage [V] |
| Rds_on | Drain-source on resistance [Ohm] |

### Connectors

| Name | Description |
|---|---|
| g | Gate |
| p | Drain |
| n | Source |

## Battery_Balancing.Components.OCV_source

**Open-circuit voltage (OCV) source with SOC tracking.**



## Information

Acts as a voltage source controlled by an externally supplied OCV value, with internal state-of-charge (SOC) calculated via Coulomb counting. Used in ECM models for SoC-voltage reference.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

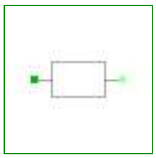
## Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |
| Q | Cell capacity [C] |
| eta | Coulomb counting efficiency |

## Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

# Battery_Balancing.Components.Resistance



**Ideal linear resistor.**

## Information

Implements a basic Ohmic resistor using the linear relation u=R·iu. Used in both passive balancing and ECM internal resistance modeling.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

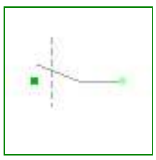
## Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |

## Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

# Battery_Balancing.Components.Switch



**Idealized controlled switch with on-resistance.**

## Information

Two-terminal element that conducts with resistance Ron when the control signal is true, and blocks current otherwise. Used in both resistive and active balancing structures.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

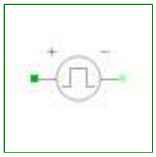
## Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |
| R_on | Resistance (Ohm) [Ohm] |

### Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

## Battery_Balancing.Components.Icontrol



**Externally controlled current source.**

### Information

Imposes a user-defined current on the connected electrical circuit, typically used for charge/discharge testing.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

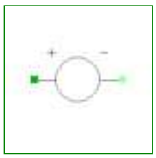
### Parameters

| Name | Description |
|---|---|
| activo | Active/inactive component |

### Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

## Battery_Balancing.Components.Vcontrol



**Externally controlled voltage source.**

### Information

Applies a specified voltage across the terminals of the component.

Extends from Interfaces.UnPuerto (Two-terminal base model with optional current direction logic.).

### Parameters

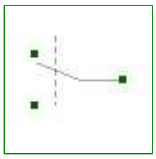
| Name | Description |
|---|---|
| activo | Active/inactive component |

### Connectors

| Name | Description |
|---|---|
| p | Positive port |
| n | Negative port |

## Battery_Balancing.Components.Commutator

**Two-way controlled commutator switch**



## Information

Implements a switch-matrix logic allowing dynamic routing of current from one input to two possible outputs. Selection is controlled via an integer signal. Used in active balancing configurations.

Extends from Interfaces.CommutatorInterface (Three-terminal interface for commutator components.).

## Parameters

| Name | Description |
|---|---|
| R_on | On-state resistance [Ohm] |

## Connectors

| Name | Description |
|---|---|
| b | Commute pin 1 |
| c | Commute pin 2 |
| a | Entry Pin |

# Battery_Balancing.Functions

**Auxiliary function blocks for parameter interpolation and control logic**

## Information

This package contains reusable functional components and utility functions used across the battery modeling framework. It includes an ECM parameter loader for state-of-charge-dependent values and helper functions for index identification in control algorithms.

## Package Content

| Name | Description |
|---|---|
| ECM_ParameterLoader_1Thv | Lookup model for loading SOC-dependent ECM parameters from external data file. |
| f argmax | Returns the index of the maximum value in a real-valued vector. |
| f argmin | Returns the index of the minimum value in a real-valued vector. |

# Battery_Balancing.Functions.ECM_ParameterLoader_1Thv

**Lookup model for loading SOC-dependent ECM parameters from external data file.**

## Information

This model reads four electrical parameters (OCV, R0, R1, and C1) from a .mat file formatted as a lookup table. The values are interpolated based on the input SOC. It is designed to provide real-time access to parameter sets used in the single-branch Thevenin ECM.

Input:

- SOC: State of charge [0–1]

Output:

- OCV: Open-circuit voltage [V]

- R0 : Ohmic resistance [Ohm]

- R1 : Polarization resistance [Ohm]

- C1 : Polarization capacitance [F]

## Battery_Balancing.Functions.argmax

**Returns the index of the maximum value in a real-valued vector.**

### Information

This utility function loops through the input vector and returns the 1-based index of the elementwith the highest value.

Input:

- vec[:] : Real vector of arbitrary size

Output:

- index : Index of the maximum value (1-based)

### Inputs

| Name | Description |
|---|---|
| vec[:] | Vector of given size |

### Outputs

| Name | Description |
|---|---|
| index | Highest value index in vector |

## Battery_Balancing.Functions.argmin

**Returns the index of the minimum value in a real-valued vector.**

### Information

This utility function loops through the input vector and returns the 1-based index of the element with the lowest value.

Input:

- vec[:] : Real vector of arbitrary size

Output:

- index : Index of the minimum value (1-based)

### Inputs

| Name | Description |
|---|---|
| vec[:] | Vector of given size |

### Outputs

| Name | Description |
|---|---|

| | |
|---|---|
| index | Lowest value index in vector |

## Battery_Balancing.ECM_Structures

**Package of ECM structures for lithium-ion cell modeling.**

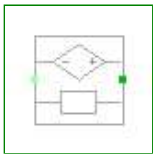
### Information

This package contains a set of equivalent circuit models (ECMs) used to simulate lithium-ion cell behavior under various operational and degradation conditions. The models follow a single-branch Thevenin configuration and incorporate support for second-life cell characterization, including aging and cell-to-cell variation. Optional voltage feedback connectors are included for closed-loop control.

### Package Content

| Name | Description |
|---|---|
| ECM_Thevenin | Basic Thevenin model with SOC-based input parameters. |
| ECM_Thevenin_SL | Extended model with aging and unbalance derating factors. |
| ECM_Thevenin_SL_Vbf | Same as ECM_Thevenin_SL with an added voltage feedback connector. |

## Battery_Balancing.ECM_Structures.ECM_Thevenin



**Basic Thevenin model with SOC-based input parameters.**

### Information

This model simulates a lithium-ion cell using a first-order equivalent circuit consisting of:

- A series resistor (R0),

- A parallel RC branch (R1, C1),

- An OCV voltage source dependent on SOC.

It serves as the base structure for evaluating basic electrochemical response under simplified assumptions. All parameters are externally fed and time-variable, typically via lookup tables.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

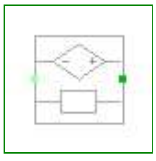
### Parameters

| Name | Description |
|---|---|
| Q | Cell Capacity [C] |
| eta | Coulomb counting efficiency |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |

## Battery_Balancing.ECM_Structures.ECM_Thevenin_SL



**Extended model with aging and unbalance derating factors.**

### Information

This variant of the Thevenin ECM includes multiplicative derating factors to account for degradation (aging) and cell-to-cell unbalance. These factors affect:

- Cell capacity (Q),

- Series and parallel resistances (R0, R1),

- Polarization capacitance (C1).

It enables simulation of aged cells in repurposed or degraded battery packs.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

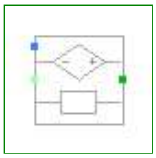
### Parameters

| Name | Description |
|---|---|
| Q | Cell Capacity [C] |
| eta | Coulomb counting efficiency |
| alpha_Q | Q aging derating factor |
| alpha_R0 | R0 aging derating factor |
| alpha_R1 | R1 aging derating factor |
| alpha_C1 | C1 aging derating factor |
| beta_Q | Q unbalance derating factor |
| beta_R0 | R0 unbalance derating factor |
| beta_R1 | R1 unbalance derating factor |
| beta_C1 | C1 unbalance derating factor |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |

## Battery_Balancing.ECM_Structures.ECM_Thevenin_SL_Vbf



**Same as ECM_Thevenin_SL with an added voltage feedback connector.**

### Information

In addition to incorporating aging and unbalance through derating parameters, this model exposes a voltage feedback connector. This enables real-time monitoring of terminal voltage for use in control algorithms and supervisory functions. Useful for integrating the ECM into larger battery management systems with closed-loop balancing control.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

### Parameters

| Name | Description |
|---|---|
| Q | Cell Capacity [C] |
| eta | Coulomb counting efficiency |
| alpha_Q | Q aging derating factor |
| alpha_R0 | R0 aging derating factor |
| alpha_R1 | R1 aging derating factor |

| | |
|---|---|
| alpha_C1 | C1 aging derating factor |
| beta_Q | Q unbalance derating factor |
| beta_R0 | R0 unbalance derating factor |
| beta_R1 | R1 unbalance derating factor |
| beta_C1 | C1 unbalance derating factor |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |
| voltageFeedback | Pin for voltage feedback 1 Cell |

## Battery_Balancing.Cell_Packs

**Create packs of ECM modeled cells**

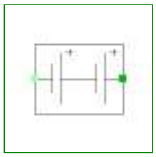
### Information

This package provides modular definitions for assembling battery packs composed of cells modeled using equivalent circuit models (ECMs). It supports different use cases including ideal cell configurations, second-life cell representations with degradation and unbalance effects, and topologies exposing electrical nodes for balancing. These models are used as structural elements in system-level simulations of charge, discharge, and cell balancing under realistic aging and parameter variation.

### Package Content

| Name | Description |
|---|---|
| Battery_Module | Battery module formed by N ideal cells in series. |
| Battery4Cell_Bal | 4-cell battery module with second-life ECMs and exposed balancing nodes. |
| Battery4Cell_Bal_Vfb | 4-cell battery module with voltage feedback connector. |
| BatteryNCell_Bal | Configurable N-cell battery module with intermediate balancing nodes. |

## Battery_Balancing.Cell_Packs.Battery_Module



**Battery module formed by N ideal cells in series.**

### Information

This model instantiates a configurable number of ECM_Thevenin cells connected in series, where each cell shares common parameter structures for capacity and Coulomb efficiency. The SOC-dependent parameters (OCV, R0, R1, C1) are provided externally as arrays. Intended for scalable validation of balancing and pack-level simulation behavior.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

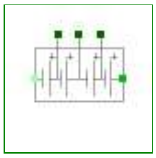
### Parameters

| Name | Description |
|---|---|
| N | Cells in series |
| Q | Capacity of each cell [C] |
| eta | Coulomb counting efficiency |
| Cell[N] | Cell |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |

## Battery_Balancing.Cell_Packs.Battery4Cell_Bal



**4-cell battery module with second-life ECMs and exposed balancing nodes.**

### Information

This model connects four ECM_Thevenin_SL cells in series, each with independent derating parameters for aging and unbalance. Three intermediate electrical nodesbetween the cells are exposed for connection to external balancing components. Ideal for evaluating balancing topologies under realistic degradation conditions.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

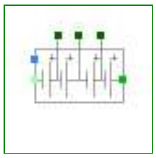
### Parameters

| Name | Description |
|---|---|
| Q[4] | Capacity of each cell [C] |
| eta[4] | Coulomb counting efficiency |
| alpha[4, 4] | Aging derating matrix: [Q, R0, R1, C1] |
| beta[4, 4] | Unbalance derating matrix: [Q, R0, R1, C1] |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |
| node_3_4 | |
| node_2_3 | |
| node_1_2 | |

## Battery_Balancing.Cell_Packs.Battery4Cell_Bal_Vfb



**4-cell battery module with voltage feedback connector.**

### Information

Identical to Battery4Cell_Bal, but with an added connector for real-time voltage feedback. This is used by balancing controllers to assess the individual cell voltages during simulation and determine actuation logic.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

### Parameters

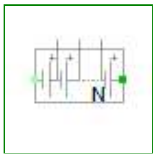
| Name | Description |
|---|---|
| Q[4] | Capacity of each cell [C] |
| eta[4] | Coulomb counting efficiency |
| alpha[4, 4] | Aging derating matrix: [Q, R0, R1, C1] |

| | |
|---|---|
| beta[4, 4] | Unbalance derating matrix: [Q, R0, R1, C1] |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |
| node_3_4 | |
| node_2_3 | |
| node_1_2 | |
| voltageFeedback | |

## Battery_Balancing.Cell_Packs.BatteryNCell_Bal




**Configurable N-cell battery module with intermediate balancing nodes.**

### Information

This model generalizes Battery4Cell_Bal for an arbitrary number of ECM_Thevenin_SL cells in series. It provides intermediate node access between each cell, facilitating the evaluation of general balancing architectures or control strategies over packs of variable size. SOC-dependent electrical parameters and derating coefficients are externally fed as vectors.

Extends from Interfaces.UnPuerto_ECM (Two-terminal interface for ECM components.).

### Parameters

| Name | Description |
|---|---|
| N | Number of cells in series |
| Q[N] | Capacity of each cell [C] |
| eta[N] | Coulomb counting efficiency |
| alpha[N, 4] | Aging derating matrix: [Q, R0, R1, C1] |
| beta[N, 4] | Unbalance derating matrix: [Q, R0, R1, C1] |

### Connectors

| Name | Description |
|---|---|
| p | Positive pin |
| n | Negative pin |
| node[N - 1] | Intermediate nodes between cells |

## Battery_Balancing.Balancing_Structures

**Balancing structures for 4-cell modules and Charge/Discharge behavior.**

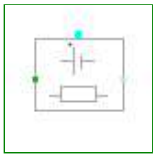
### Information

This package defines structural components used for energy flow management and balancing operations in a 4-cell battery module. It includes several active and passive balancing topologies and one charging/discharging interface. Each model is compatible with 4-cell packs and supports voltage feedback and external digital control signals for switching logic.

### Package Content

| Name | Description |
|---|---|
| ChargeDischarge | Charge and discharge module for battery cell stacks. |
| ShuntResistorBalancing_4Cell | Shunt resistor balancing topology for 4-cell battery packs |
| SwitchedResistorBalancing_4Cell | Switched Resistor Balancing model for 4 cell battery pack. |
| SingleCapacitorBalancing_4Cell | Single Capacitor Balancing model for 4 cell battery pack. |
| SingleInductorBalancing_4Cell | Single Inductor Balancing model for 4 cell battery pack. |

# Battery_Balancing.Balancing_Structures.ChargeDischarge



**Charge and discharge module for battery cell stacks.**

## Information

This model provides a controlled current source capable of charging or discharging a connected battery pack. The direction and magnitude of current are determined by the external control input. A series resistor models the load during discharge.

The mode signal can take values:

- +1: constant current charge

- -1: constant current discharge

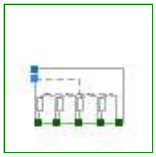
- 0: idle (no current flow).

## Parameters

| Name | Description |
|---|---|
| I_charge | Charge current [A] |
| I_discharge | Discharge current [A] |
| R_load | Series load [Ohm] |

## Connectors

| Name | Description |
|---|---|
| controlInput | Control Input charge/discharge |
| p | Positive pin |
| n | Negative pin |



# Battery_Balancing.Balancing_Structures.ShuntResistorBalancing_4Cell

**Shunt resistor balancing topology for 4-cell battery packs**

## Information

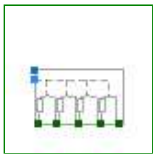
Implements passive balancing by discharging selected cells through fixed shunt resistors. Each resistor is connected in series with a voltage-controlled NMOS transistor. A single control input simultaneously activates all switches. The model supports per-cell voltage monitoring via a voltage feedback connector.

## Parameters

| Name | Description |
|---|---|
| R_shunt | Shunt resistor value (Ohm) [Ohm] |
| Rds_on | MOSFET on-resistance (Ohm) [Ohm] |
| Vth | MOSFET threshold voltage (V) [V] |
| V_gate | Gate voltage when ON (V) [V] |

### Connectors

| Name | Description |
|---|---|
| pin3 | |
| pin1 | |
| pin5 | |
| pin4 | |
| pin2 | |
| controlInput | |
| voltageFeedback | |



## Battery_Balancing.Balancing_Structures.SwitchedResistorBalancing_4Cell

**Switched Resistor Balancing model for 4 cell battery pack.**

### Information

Similar to passive shunt balancing, but allows independent control of each balancing path. The circuit consists of four NMOS-resistor branches, one per cell. Each branch is controlled by a corresponding boolean control signal. Voltage feedback is provided for monitoring cell voltages during balancing.

### Parameters

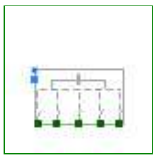
| Name | Description |
|---|---|
| R_shunt | Shunt resistor value (Ohm) [Ohm] |
| Rds_on | MOSFET on-resistance (Ohm) [Ohm] |
| Vth | MOSFET threshold voltage (V) [V] |
| V_gate | Gate voltage when ON (V) [V] |

### Connectors

| Name | Description |
|---|---|
| controlInput | |
| voltageFeedback | |
| pin3 | |
| pin1 | |
| pin5 | |
| pin4 | |
| pin2 | |

## Battery_Balancing.Balancing_Structures.SingleCapacitorBalancing_4Cell

**Single Capacitor Balancing model for 4 cell battery pack.**



## Information

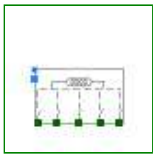
Implements energy redistribution between cells using a single shared capacitor. The energy transfer path is formed by five switches and two commutators, which select the source and destination cells. A bleed resistor across the capacitor models self-discharge and improves simulation stability. Control input consists of five switch signals and two commutator positions. Voltage feedback is provided.

## Parameters

| Name | Description |
|---|---|
| C | Inductance for energy transfer [F] |
| R_on | On-state resistance for switches [Ohm] |
| R_bleed | Bleed resistance [Ohm] |
| V_start | Start Voltage capacitor [V] |

## Connectors

| Name | Description |
|---|---|
| pin3 | |
| pin1 | |
| pin5 | |
| pin4 | |
| pin2 | |
| voltageFeedback | |
| controlSignal | |



# Battery_Balancing.Balancing_Structures.SingleInductorBalancing_4Cell

**Single Inductor Balancing model for 4 cell battery pack.**

## Information

Implements energy redistribution between cells using a single shared inductor. The energy transfer path is formed by five switches and two commutators, which select the source and destination cells. A bleed resistor across the inductor models self-discharge and improves simulation stability. Control input consists of five switch signals and two commutator positions. Voltage feedback is provided.

## Parameters

| Name | Description |
|---|---|
| L | Inductance for energy transfer [H] |
| R_on | On-state resistance for switches [Ohm] |
| R_bleed | Bleed resistance [Ohm] |
| t_precharge | Precharge duration [s] |

## Connectors

| Name | Description |
|---|---|
| pin3 | |
| pin1 | |
| pin5 | |
| pin4 | |
| pin2 | |
| voltageFeedback | |
| controlSignal | |

# Battery_Balancing.Control_Structures

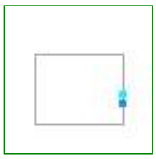
**Package containing charge/discharge infrastructure and cell balancing architectures for a 4-cell lithium-ion battery module.**

## Information

Includes modular implementations of passive and active balancing methods, such as shunt resistor, switched resistor, single capacitor, and single inductor topologies. Each balancing structure is designed for integration with corresponding control models and supports per-cell connectivity via standardized pin interfaces. The `ChargeDischarge` model provides a configurable current source and resistive load for system-level cycling tests.

## Package Content

| Name | Description |
|---|---|
| Controller_NoBalancing1C_SG | No balancing 1 cell controller State graph based. |
| Controller_NoBalancing4C_SG | No Balancing 4 cell pack controller State graph based. |
| Controller_ShuntResistor_SG | Shunt Resistor Balancing controller State graph based. |
| Controller_SwitchedResistor_SG | Switched Resistor Balancing controller State graph based. |
| Controller_SingleCapacitor_SG | State-graph controller for single-capacitor balancing. |
| Controller_SingleInductor_SG | State-graph controller for single-inductor balancing. |



# Battery_Balancing.Control_Structures.Controller_NoBalancing1C_SG

**No balancing 1 cell controller State graph based.**

## Information

Operates a single cell through a discharge–idle–charge sequence based on voltage thresholds. Transitions are timed or condition-triggered. No balancing logic is applied. Designed for baseline testing or reference simulations.

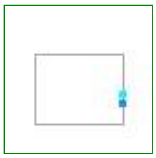
## Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |

| | |
|---|---|
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |

### Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from cell. |
| controlLoad | Control signal to Charge Discharge module. |



## Battery_Balancing.Control_Structures.Controller_NoBalancing4C_SG

**No Balancing 4 cell pack controller State graph based.**

### Information

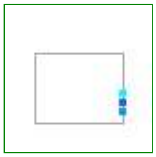
Implements charge and discharge logic using minimum and maximum cell voltages for threshold detection. All cells are controlled collectively. No intra-pack balancing is applied. Suitable for baseline simulations.

### Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |

### Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from cell pack. |
| controlLoad | Control signal to charge/discharge module. |



## Battery_Balancing.Control_Structures.Controller_ShuntResistor_SG

**Shunt Resistor Balancing controller State graph based.**

### Information

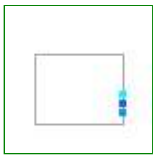
Cycles the battery through discharge–idle–charge phases. Balancing is activated during charging by enabling a single control signal that triggers all shunt paths simultaneously. Designed for shunt passive balancing experiments.

### Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |

### Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from balancing module. |
| controlBalancing | Control signal to balancing module. |
| controlLoad | Control signal to charge/discharge module. |



## Battery_Balancing.Control_Structures.Controller_SwitchedResistor_SG

**Switched Resistor Balancing controller State graph based.**

### Information

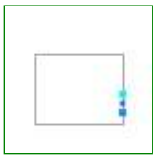
Adds per-cell balancing logic based on voltage deviation from the minimum cell voltage. Balancing is active during charging and uses hysteresis to reduce switching noise. Enables precise and configurable pàssive balancing strategies in systems with discrete switch control.

### Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |
| dV_on | Start balancing when cell exceeds min+X [V] |
| dV_off | Stop balancing when all within min+Y [V] |
| eps | Hysteresis buffer to avoid chattering [V] |
| t_Bal | Duration to hold balancing state [s] |

### Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from balancing module. |
| controlLoad | Control signal to charge/discharge module. |
| controlSignalBalancing_Switched | Control signal to balancing module |



# Battery_Balancing.Control_Structures.Controller_SingleCapacitor_SG

**State-graph controller for single-capacitor balancing.**

## Information

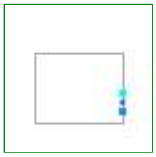
Manages energy redistribution between the most and least charged cells using a shared capacitor. Balancing logic identifies voltage extremes and configures switching patterns and commutator positions. Control includes periodic switching (comm_flip) to alternate capacitor direction.

## Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |
| dV_on | Start balancing when cell exceeds min+X [V] |
| t_Balance | Idle time between phases [s] |
| T_commute | Commuter switching interval [s] |

## Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from balancing module. |
| controlLoad | Control signal to charge/discharge module. |
| controlSignalBalancing_Single | Control signal to balancing module. |



# Battery_Balancing.Control_Structures.Controller_SingleInductor_SG

**State-graph controller for single-inductor balancing.**

## Information

Manages energy redistribution between the most and least charged cells using a shared inductor. Balancing logic identifies voltage extremes and configures switching patterns and commutator positions. Control includes periodic switching (comm_flip) to alternate inductor direction.

## Parameters

| Name | Description |
|---|---|
| t_Start | Idle time between phases [s] |
| t_Idle | Idle time between phases [s] |
| Vmin | Minimum allowed voltage [V] |

| | |
|---|---|
| Vmax | Maximum allowed voltage [V] |
| margin | Voltage marging for Vmax/Vmin [V] |
| dV_on | Start balancing when cell exceeds min+X [V] |
| t_Balance | Idle time between phases [s] |
| T_commute | Commuter switching interval [s] |

### Connectors

| Name | Description |
|---|---|
| voltageFeedback | Voltage feedback from balancing module. |
| controlLoad | Control signal to charge/discharge module. |
| controlSignalBalancing_Single | Control signal to balancing module. |

## Battery_Balancing.Examples

**Collection of simulation testbenches demonstrating the behavior and performance of various battery management and balancing strategies for a 4-cell lithium-ion pack.**

### Information

Each example model integrates the following key elements:

- Electrochemical Equivalent Circuit Model (ECM) with aging and unbalance parameters.

- Parametric derating of internal resistance, capacitance, and OCV for cell aging.

- Charge/discharge cycling via configurable current source module.

- Controller logic based on Modelica_StateGraph2 to emulate operational state transitions.

- Passive and active balancing techniques, including:

  - No balancing (1-cell and 4-cell variants),
  - Shunt resistor balancing,
  - Switched resistor balancing,
  - Single capacitor balancing,
  - Single inductor balancing.

Purpose:

These examples serve to validate balancing algorithms under aging and mismatch scenarios and are intended for comparative simulation studies of energy redistribution and state-of-charge equalization.

### Package Content

| Name | Description |
|---|---|
| Example_NoBalancing1CSG | Simulation of a single-cell system with no cell balancing mechanism. |
| Example_NoBalancing4CSG | Simulation of a 4-cell battery pack without any balancing strategy. |
| Example_ShuntBalancing4CSG | Simulation of a 4-cell battery pack using passive shunt resistor balancing. |
| Example_SwitchedBalancing4CSG | Simulation of a 4-cell battery pack employing switched resistor balancing. |

| | |
|---|---|
| Example_SingleCapacitor4CSG | Simulation of a 4-cell battery pack with single-capacitor active balancing. |
| Example_SingleInductor4CSG | Simulation of a 4-cell battery pack using single-inductor balancing. |

## Battery_Balancing.Examples.Example_NoBalancing1CSG

**Simulation of a single-cell system with no cell balancing mechanism.**

### Information

The model includes an ECM representation in which the detaring factors have been set all to 1 (no aging or unbalance effects included). The graph-based controller governs cyclic charge and discharge behavior with idle intervals and voltage threshold-based transitions. The battery is charged and discharged using a controlled interface, and no active or passive balancing is applied.

Purpose:

This example establishes a baseline for evaluating the performance and degradation behavior of an isolated lithium-ion cell under cyclic operation, without the influence of balancing strategies.

### Parameters

| Name | Description |
|---|---|
| alpha[4] | Derating aging values |
| beta[4] | Derating unbalance values |

## Battery_Balancing.Examples.Example_NoBalancing4CSG

**Simulation of a 4-cell battery pack without any balancing strategy.**

### Information

Each cell is independently modeled using ECM parameters subjected to aging and mismatch effects. Derating factors (`alpha`, `beta`) capture variations in capacity and resistance over time. A shared state graph controller supervises the entire pack's charging and discharging process.

Purpose:

Used as a benchmark case to assess natural imbalance development and the long-term impact of cell mismatch in unbalanced multi-cell systems.

### Parameters

| Name | Description |
|---|---|
| alpha[4, 4] | Derating aging values |
| beta[4, 4] | Derating unbalance values |

## Battery_Balancing.Examples.Example_ShuntBalancing4CSG

**Simulation of a 4-cell battery pack using passive shunt resistor balancing.**

### Information

The model includes ECM-based cells with age-related degradation. The shunt balancing is triggered during the charging phase and is governed by a state graph controller that activates bypass resistors when cell voltages exceed a predefined threshold.

Purpose:

Demonstrates the effectiveness of simple, cost-efficient passive balancing for managing cell voltage deviations during pack operation. Enables comparison with active balancing methods.

### Parameters

| Name | Description |
|---|---|
| alpha[4, 4] | Derating aging values |
| beta[4, 4] | Derating unbalance values |

## Battery_Balancing.Examples.Example_SwitchedBalancing4CSG

**Simulation of a 4-cell battery pack employing switched resistor balancing.**

### Information

The model incorporates voltage-based detection logic for imbalance onset and triggers controlled discharging of individual cells through switchable resistors. The balancing process is hysteresis-controlled with periodic state checks and per-cell logic, ensuring targeted intervention during charging.

Purpose:

Evaluates an active balancing strategy that provides moderate energy redistribution while minimizing controller complexity and hardware cost. Suitable for performance comparison with both passive and more advanced active methods.

### Parameters

| Name | Description |
|---|---|
| alpha[4, 4] | Derating aging values |
| beta[4, 4] | Derating unbalance values |

## Battery_Balancing.Examples.Example_SingleCapacitor4CSG

**Simulation of a 4-cell battery pack with single-capacitor active balancing.**

### Information

Balancing is performed through controlled charge transfer between the most charged and least charged cells using a shared capacitor and a bidirectional commutator. The state graph controller regulates the commutation frequency and switching patterns based on voltage deviation thresholds.

Purpose:

Tests a time-shared balancing strategy offering higher efficiency compared to resistive methods. Useful for analyzing control complexity and energy redistribution effectiveness in capacitor-based topologies.

### Parameters

| Name | Description |
|---|---|

| | |
|---|---|
| alpha[4, 4] | Derating aging values |
| beta[4, 4] | Derating unbalance values |
| C | Capacitance for energy transfer [F] |
| R_on | On resistance [Ohm] |
| R_bleed | [Ohm] |
| t_Balance | Idle time between phases (s) [s] |
| T_commute | Commuter switching interval [s] |

## Battery_Balancing.Examples.Example_SingleInductor4CSG

**Simulation of a 4-cell battery pack using single-inductor balancing.**

### Information

Implements active balancing by transferring energy via an inductor between cells with the highest and lowest voltages. A commutator and switch matrix modulate the transfer path, while a state machine dynamically controls balancing duration and frequency based on real-time cell voltage disparities.

Purpose:

Analyzes the performance of an advanced, energy-efficient balancing method suitable for high-performance applications, highlighting trade-offs between hardware complexity, control logic, and balancing efficiency.

### Parameters

| Name | Description |
|---|---|
| alpha[4, 4] | Derating aging values |
| beta[4, 4] | Derating unbalance values |
| L | Inductance for energy transfer [H] |
| R_on | On resistance [Ohm] |
| R_bleed | [Ohm] |
| t_Balance | Idle time between phases (s) [s] |
| T_commute | Commuter switching interval [s] |

*Automatically generated Mon Jun 2 18:46:06 2025.*