Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Energy Storage – EIT InnoEnergy

Comparative study of multiphysics modelling and simulation software for lifetime performance evaluation of battery systems.

Francesco De Marco

Master's thesis
2023

Aalto-yliopisto
Aalto-universitetet
Aalto University

Politecnico di Torino

Author Francesco De Marco		
Title of thesis Comparative study of multiphysics modelling and simulation software for lifetime performance evaluation of battery systems		
Programme Energy Storage – EIT InnoEnergy		
Major Energy Engineering		
Thesis supervisor Prof. Massimo Santarelli, Prof. Kari Tammi		
Thesis advisor(s) Majid Astaneh, Ph.D.		
Collaborative partner Northvolt Systems AB		
Date 10.07.2023	**Number of pages** 90	**Language** English

Abstract

Battery systems play a vital role in numerous applications, ranging from electric vehicles to renewable energy storage. Accurate modelling and simulation of battery systems are crucial for assessing their lifetime performance. Currently, there are various simulation software on the market to assist industry and academia to implement multiphysics models and simulate battery systems. Every tool has unique capabilities, which can suit different needs of users.

This thesis investigates and compares four widely used multiphysics modelling and simulation software tools— Simcenter Amesim, MATLAB/Simulink, MATLAB/Simulink/Simscape and Dymola—with respect to four main aspects, namely computational performance and accuracy, ease of use, features, and licensing. The scope of the study is to highlight peculiarities of every software, to provide insights to simulation engineers, researchers and students that need to select the right simulation software for their investigations.

The very same multiphysics model of a battery pack, composed by an Equivalent Circuit Model (ECM) with two RC branches and liquid cooling will be described in detail and implemented on the four tools. The system includes electric, thermal, and ageing model. The main output of the simulation is battery lifetime degradation in different operating conditions. Accuracy of the four software will be benchmarked against test data, while computational speed will be estimated based on the lifetime simulation.

The main findings of this comparative study is that Dymola proved to be the tool with the best speed performance, requiring only 0.25 seconds to perform a 24h simulation and 288 seconds to perform a 10 years simulation. In terms of ease of use, Amesim resulted as the most user-friendly, with a simple user interface, smooth workflow, and clear documentation. In terms of features and interoperability, Simulink has the great advantage of being closely interconnected with MATLAB environment. The same is true for Simscape, whose speed and accuracy performance were the poorest, but is more user-friendly since it provides multiphysics component.

The findings of this comparative study will assist researchers, engineers, and industry professionals in selecting the most suitable software tool for lifetime performance evaluation of battery systems, based on their specific case.

Keywords Battery Energy Storage System, Lifetime Simulation, Simulation Software, Battery Degradation, Calendar Ageing, Cycle Ageing, Software Comparative Study

The thesis is written as part of a collaborative joint degree programme called Master's Programme in Energy Storage (EIT InnoEnergy).

Aalto University is responsible for the implementation of the programme along with Politecnico di Torino (PoliTo) and EIT InnoEnergy, a knowledge and innovation community of the European Institute of Innovation and Technology.

The thesis is supervised by both partner universities.

Politecnico
di Torino

Aalto University
School of Engineering

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Preface and acknowledgements

I would like to express my deepest gratitude and appreciation to all those who have supported me throughout the completion of my master's thesis. Without their guidance, encouragement, and assistance, this accomplishment would not have been possible.

First and foremost, I would like to thank Northvolt for providing me with the opportunity to undertake my internship. The experience gained during this period was invaluable in shaping my understanding of energy storage. I am especially grateful to my thesis advisor, Majid Astaneh, for his continuous support, valuable insights, and guidance throughout the entire process. I would also like to extend my heartfelt thanks to my manager, Anders Magnusson, for his mentorship and for fostering a conducive environment for learning.

I am deeply grateful to professor Silvia Bodoardo and professor Annukka Santasalo-Aarnio who organized the academic activities of the EIT InnoEnergy Master's degree in Energy Storage. Their expertise and dedication to their field have been instrumental in broadening my knowledge and shaping my professional direction.

I would like to express my sincere appreciation to my ex-supervisor at Edison, Alessandro Perol, for his guidance and support during my previous internship. His mentorship and encouragement have significantly influenced my professional growth and have inspired me to pursue further research in this field.

Many thanks to prof. Massimo Santarelli and prof. Kari Tammi for supervising my thesis work.

Last but not least, I would also like to extend my gratitude to my best friends (in ascending order of height) Clarissa, Giuditta and Alice (parimerito), and Pierfrancesco. Thanks also to Giuliano&Laca. You have been a constant source of encouragement, motivation, and support throughout my master's journey. Thanks also to all my friends from Collegio Onaosi and Random Target.

To all those mentioned above and countless others who have contributed to my growth and success, I offer my heartfelt thanks. Your support has been indispensable, and I am honoured to have had the opportunity to work with and learn from such incredible people.

Abbreviations

BESS	Battery energy storage system
DOD	Depth of Discharge
ECM	Equivalent circuit model
EOL	End of life
FEM	Finite elements model
FMU	Functional mock-up unit
GUI	Graphical user interface
HPPC	Hybrid pulse power characterization
KPI	Key performance indicator
LAM	Loss of Active Material
LC	Loss of Conductivity
LLI	Loss of Lithium Inventory
OCV	Open circuit voltage
SEI	Solid Electrolyte Interface
SOC	State of charge

1 Introduction

1.1 Background and motivation

In recent years, battery systems have become a fundamental technology in modern society. To meet the ambitious goal of net-zero carbon emission by 2050, a strong and global effort must be made. Institutions, researchers, and industry are driving our world toward electrification and clean energy production. One of the strongest trends concerns the development and implementation of renewable energy technologies. The global installed capacity of renewables is expected to double in the next five years and increase by five times in 2050 [1]. On top of that, the other industrial challenge is the complete electrification on road transport by 2035. In this scenario, it is undoubtedly true that energy storage systems, especially batteries, will play a fundamental role. A proper clean and sustainable future cannot be reached without a proficient development of this technology.

Battery modelling and simulation is a powerful tool to enhance battery characteristics in terms of performance, durability, and safety. Through this method, it is possible to optimize the design and the operation of the battery systems to increase efficiency and prolong lifetime, since it is possible to test various scenarios and obtain results rapidly, without the need of conducting expensive and time-consuming experiments. However, battery modelling is a non-trivial work since it involves the detailed study of many phenomena of various sorts, from different physics, such as electrochemistry, thermodynamics, electrotechnics, fluid-dynamics, and mechanics. Because of that, a common approach is to develop a comprehensive multiphysics model, which not only considers all the different phenomena, but also how those effects interact among them.

Recently, many companies are developing software and tools that aims to assist the battery modelling process and allow users to run simulation and obtain multiple physical variables. As the complexity of the model increases, the required computation effort to simulate the system can be significant. Indeed, these software are solving all the physical governing equation (which in most of the cases are differential equations), for each timestep. It is of no surprise that for simulating lifetime operation of a battery system (one or two decades), the computational runtime can be of the order of hours, even days. This represents a bottleneck that harms the proficient development of battery systems that our society requires.

The energy storage industry is rapidly growing and, along with it, an increasing number of software developers propose their own simulation tool, with unique features and specific performance. For this reason, being aware of which software best suits different needs is not obvious. Furthermore, it is

of the utter importance for industry and researchers to properly select the software based on their needs, in terms of accuracy and simulation runtime.

1.2 Comparative studies of simulation software

Many researchers have investigated different simulation software, comparing their performances, features, and ease of use. For instance, Attia et al. [2] have performed a comparative study of ten simulation tools for building performance, conducting a survey with 249 experts. They gathered insights about usability, information management and integration of intelligent design since those factors are agreed to be the most representative to assess "architect friendliness". A comparative study of three tools for Finite Elements Modelling (FEM) has been conducted by Gardner et al. [3]. The authors focused their attention on two qualitative aspects, user friendliness (considering ease of setup and overall control) and features (such as material models and adaptive mesh capabilities). Also, Ross [4] based his study on ease of use and features, comparing simulation software for modelling stability operation. Sousa [5] reviewed and compared five energy simulation software for building, providing a detailed list of their different features and a comparison of the simulation results. This work also includes suggestions for the best use for each tool. A qualitative and quantitative comparison of simulation software has been conducted by Torres-Torriti et al. [6], who investigated mobile robot tools. They showcased the physical modelling capabilities, the solver features, and the accuracy of the results. The computational performance of simulation software has been benchmarked by Magni et al. [7], whose research focused on energy simulation tools for building. Together with the computational cost, this cross-comparison evaluated the different results, provided insights of multiple features, and gave suggestion for choosing the right software. Bernal-Agustín et al. [8] compared 11 simulation tools for optimization of stand-alone renewable energy systems. The author compared the software based on their functionalities, optimization algorithms and licensing.

1.3 Scope of the study

Even though comparative study between software is a common practice, there has never been a comparison between simulation software specifically for multiphysics modelling of battery systems. Therefore, it is not straightforward to select the most suitable software for a specific case via literature review. Given the current and future relevance of battery simulation, this thesis aims to analyse, benchmark, and compare four different simulation tools among the most used ones in research and industry. The investigated software are Simcenter Amesim, MATLAB/Simulink, MATLAB/Simulink/Simscape, and Dymola. The same model of a battery system for stationary

applications will be modelled and implemented in each software and run on the same machine, so that a fair comparison will be made. The scope of each simulation is to estimate battery degradation after lifetime operation. The comparison considers four different categories, namely performance, ease of use, features and licencing. An overall score will be assigned to each category, which will consider both quantitative and qualitative aspects.

Based on personal modelling and simulation experience, Simcenter Amesim proved to be the most user-friendly software, with its intuitive interface, and steep learning curve. In terms of performance, Dymola can perform the fastest simulations. Compared to the second fastest tool, Simulink, Dymola is more than three times faster for simple simulations and more than 5 times faster for lifetime simulations. Both Simulink and Simscape have the great advantage of being closely interconnected with the MATLAB environment, which comes with a number of advantages, described later in this thesis. While Simulink is faster than Simscape, it is the last in terms of ease of use, since there are no multiphysics component in its library. These and more findings will be explained in Section 4.

Given the widely comprehensive framework of this thesis, the scope of this study is multifaceted; the main purposes are the followings:

1. Present a comprehensive multiphysics model of a battery system for stationary application. This model investigates electrical, thermal, and ageing aspects, with the intention of estimating battery degradation after lifetime operation.
2. Develop a methodology that can be used to compare and benchmark simulation software of different kinds, providing insights on Key Performance Indicators (KPI), data collection and results comparison.
3. Evaluate four different simulation software for simulating multiphysics models of battery systems, highlighting advantages and disadvantages of each tool, based on the selected KPIs.
4. Suggest software selection based on different user objectives, both for industry and academia, based on the results of the comparative study.

The thesis is divided in five sections. Section 2 describes in detail how the battery system is modelled. The different use cases are presented, along with the multiphysics phenomena involved. This section ends with an overview of the experimental tests cited in literature to obtain the electrical and thermal parameters of the model. Section 3 presents the methodology adopted. It starts with the reasoning behind software selection, outlining a brief description of each tool. The same section explains how the battery model can be implemented on the selected tools. It also outlines the KPI based on which the software have been compared. Section 4 presents the qualitative and quantitative results of the analysis, providing deeper insights for each

software and recommendation on which tool best suits different requirements. The last section summarizes the key findings.

2 Battery pack multiphysics model

The model represents an electrochemical Battery Energy Storage System (BESS) composed of serially connected battery cells. The purpose of the lifetime simulation is to evaluate overall degradation of the BESS after 10 years of operation. To this aim, a multiphysics simulation will be used to estimate the operating condition of the system over time. The model parameters are calibrated using experimental data and the ageing maps are based on empirical curve fitting. The next sections will provide an overview of the framework adopted, what is the use case power profile, how multiphysics phenomena have been implemented in the model and the experimental approach to determine the parametric maps of the model.

2.1 Simulation framework

Battery degradation is a phenomenon that depends on many kinds of factors, driven by thermal, electrochemical, or mechanical processes. For this reason, lifetime simulation must handle different physics at the same time since they are closely intercorrelated. The workflow adopted to perform the lifetime simulation is shown by the scheme in Figure 1.

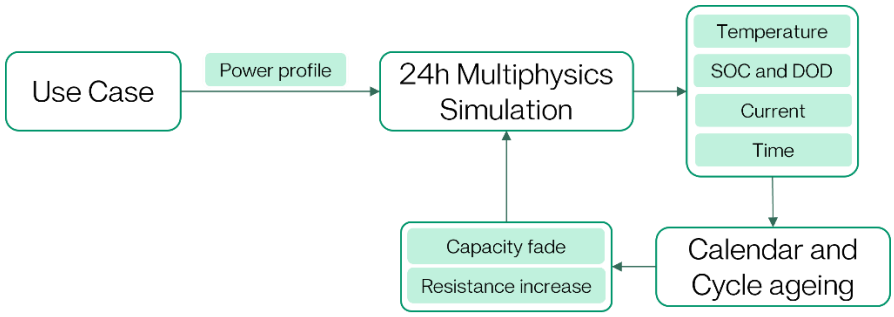

Figure 1 - Lifetime simulation framework

The entire lifetime simulation considers 10 years of daily operation of the battery pack. This period is investigated by means of 24-hours simulations that are performed iteratively. Every daily simulation receives as an input the use case power profile and the updated state of health, which indicates the remaining useful capacity and the increase of internal resistances. The model performs a multiphysics simulation whose output are the daily profiles of electrical and thermal quantities. This output is then implemented in the ageing model, which calculates the degradation associated with the last 24 hours. This is needed to assess the updated state of health, which is given as an input for the next 24-hours simulation. This whole process is repeated 3650 times, to calculate lifetime degradation over 10 years.

2.2 Case studies

One of the inputs of the simulation is the duty cycle, which is usually defined by the user. In this case, the battery system performs two full cycles per day. During daily operation, six different phases can be identified. To investigate the effect of different operating conditions on degradation, lifetime simulation will be performed for four different case studies. The charge/discharge logic in all the four cases is the same; what will vary is the ambient temperature, resting State of Charge (SOC) and Constant Power (CP) rate. We will refer to the standard case as Case 1. The other three cases are based on Case 1, with only one variation in operating conditions. Table 1 summarizes the operating conditions in the four cases.

Table 1 - Operating condition in the four case studies

	Ambient temperature (°C)	Resting SOC (%)	CP Rate (-)
Case 1 (Standard)	20	50	0.33
Case 2 (Ambient)	**40**	50	0.33
Case 3 (State of charge)	20	**20**	0.33
Case 4 (Power rate)	20	50	**1**

Given the operating conditions, the charge/discharge profile is the same. It consists of two full cycles back-to-back, with 1 hour rest between the cycles. The 24-hours power profile can be summarized as follows:

1. The initial phase is the resting phase, while the battery is idle: the SOC is constant at a given value (Resting SOC) without self-discharge effects.
2. After the initial resting time, the battery is discharged at constant power, until one of the following conditions is satisfied:
    - SOC drops under 0.01%.
    - Cell Voltage drops below the lower cut-off voltage set-point.
3. Then, the charging phase begins. An electric current flows in the circuit so that the power output of the battery is constant. Here is the equation that controls the current input:

$$I = \frac{P}{V_t} \tag{1}$$

The charging phase continues until one of the following conditions is satisfied:

- SOC exceeds 100%.
- Cell voltage overcomes the upper cut-off voltage.

4. Once the battery is fully charged, it is discharged with constant power until the resting SOC.
5. The battery is idle for 1 hour.
6. The second cycle begins, following again steps 2, 3 and 4.
7. The battery is idle until the end of the day.

The SOC is obtained by integrating over time the current flowing through the system, divided by the actual cell capacity.

$$SOC(t) = SOC_0 - \int_0^t \frac{I(t)}{\frac{C}{3600}} dt \cdot 100 \tag{2}$$

Where SOC_0 is the State of Charge at the initial time in %, $I(t)$ is the current in A, C is the capacity in Ah and 3600 is the conversion factor from hours to seconds.

The cell Open Circuit Voltage (OCV) is a one-to-one non-linear function of the State of Charge and is obtained from the experimental OCV(SOC) function produced by the tests described in Appendix A.

Power, voltage, current and SOC are interdependent and are concatenated in an algebraic loop. Figure 2 depicts the sequence used by the simulator to compute each variable.

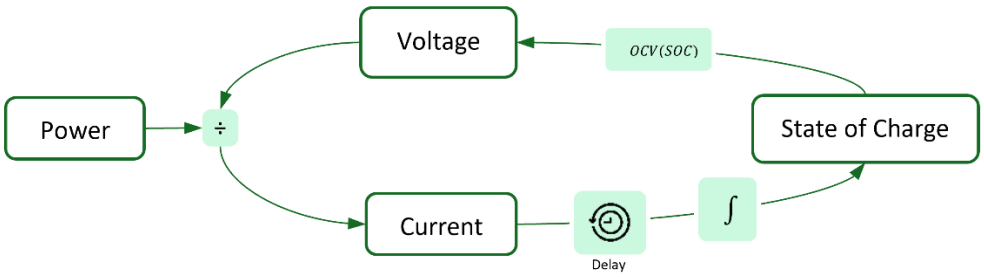

Figure 2 - Interdependency of electric values in the battery model

To break the algebraic loop, a delay can be added after the current is computed, so that all the calculations at the new timestep are performed with the last stored value. The first phase is idle mode, so the initial value of the current is zero.

2.3 Multiphysics model

The system is composed of the battery pack, the liquid cooling system, and an electric circuit to control charge and discharge. The simulation must be able to handle variables from different physics which are interdependent. This section will explain the multiphysics model implementation, highlighting the interdependencies between electric, thermal, and ageing variables and parameters.

2.3.1 Electric model

The battery pack is composed of cells connected in series. To estimate the electric properties of the system, such as current and voltage, it is necessary to identify the electrical parameters of the circuit. The two most spready used approaches to model electric behaviour of battery are the electrochemical models and the Equivalent Circuit Models (ECM) [9]. The former method implements a set of non-linear differential equations that characterize the transport, thermodynamic and kinetic reaction inside the battery. While it is true that those models accurately describe the chemistry and the physics of the reactions, it is also true that for long time-span simulation this approach may lead to unnecessary computational burden, which results in extremely long runtime in lifetime simulations [9]. Because of that, in our case an ECM method is more suitable since it is more straightforward in terms of numerical treatment [10]. This model aims to describe the static and dynamic behaviour of cells through an equivalent circuit containing resistors, capacitors, and a voltage source. With a proper configuration and parameterization, this model can accurately predict the electric variables, such as current, voltage, power and SOC [11]. ECMs are usually composed of three main parts: a static component, usually composed by a series of an Open Circuit Voltage (OCV) and an internal resistance to mimic the thermodynamic behaviour; a dynamic component, which usually includes a series of RC branches; an electric source or load to close the circuit [10], [12].

One of the firstly adopted ECM to describe static and dynamic behaviour of battery is the Thevenin Battery Model [13], whose dynamic part includes one RC parallel branch. He et al. [12] proposed ad Improved Thevenin Model by including a second RC parallel branch. The two RC branches describe the electrochemical polarization effect and the concentration polarization effect, leading to more accurate results. He et al. [9] performed a comparative study of seven battery models. Among those ECMs and electrochemical models, the study investigated the Thevenin and the Improved Thevenin model (DP model). The researchers concluded that the 2-RC branch ECM best describes the dynamic response of the battery, with the lowest error with respect to experimental data. They also proved that increasing the number of RC branches over two leads not only to higher model complexity, but also causes a higher error. Hu et al. [14] compared twelve different battery models,

concluding that second-order RC model increase accuracy of the results. The same results were drawn by Zang et al. [15], that compared first-order with second-order RC model. However, the study highlighted that the slight increase in accuracy comes with higher complexity, which is not always desirable for simple simulation. Nevertheless, for our study we decided to adopt a slightly more complex ECM model with two RC-branches to achieve the highest accuracy possible. Indeed, since the simulation will involve ten years of daily operation, a small error in the model could stack up and lead to inaccurate results.

Figure 3 shows the selected ECM of a single battery cell.

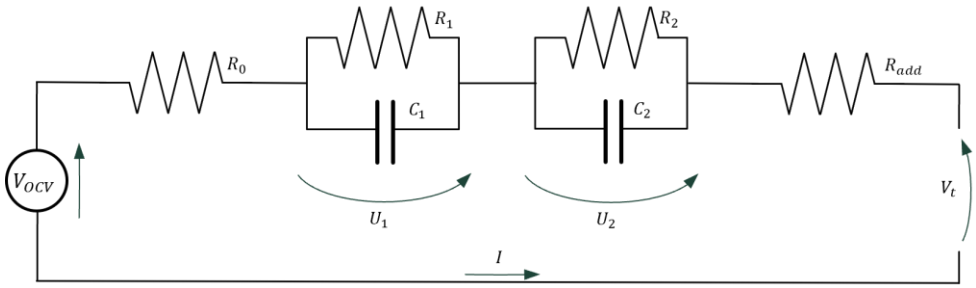

Figure 3 - Equivalent Circuit Model with 2 RC branches

The values of the current and the voltage drops in the ECM are correlated by a set of ordinary differential equations. The following governing equations [9] are implemented in the battery block in each simulation software.

$$\begin{cases} V_t = V_{OCV} + R_0 I + U_1 + U_2 + R_{add} I \\ \dfrac{dU_1}{dt} = -\dfrac{1}{R_1 C_1} U_1 + \dfrac{I}{C_1} & U_1(t=0) = 0 \\ \dfrac{dU_2}{dt} = -\dfrac{1}{R_2 C_2} U_2 + \dfrac{I}{C_2} & U_2(t=0) = 0 \end{cases} \tag{3}$$

All the electric parameters, namely the open circuit voltage OCV, the series resistor R_0, the electrochemical polarization resistance R_1 and capacitance C_1 and the concentration polarization resistance R_2 and capacitance C_2, are not fixed. Their value depends on electrical and thermal variables, such as Temperature, SOC and Current [16]. Table 2 shows each component and on which variable it depends.

Table 2 - ECM parameter dependencies on electric and thermal quantities

Component	Symbol	Dependency
Open Circuit Voltage	OCV	State of Charge
Internal series resistance	R_0	State of Charge, Temperature, Current
First order resistance	R_1	State of Charge, Temperature, Current
First order capacitance	C_1	State of Charge, Temperature, Current
Second order resistance	R_2	State of Charge, Temperature, Current
Second order capacitance	C_2	State of Charge, Temperature, Current

The parametric maps of ECM have been determined through Hybrid Pulse Power Characterization (HPPC) tests that will be further explained in Appendix A. The multiphysics simulation is run in a dynamic way, meaning that at each timestep the values of the electric parameters are calculated based on SOC, current and temperature. These parameters are needed to solve the thermal problem, which will output the cell temperature. Figure 4 graphically explains how the electrical and the thermal model are coupled.

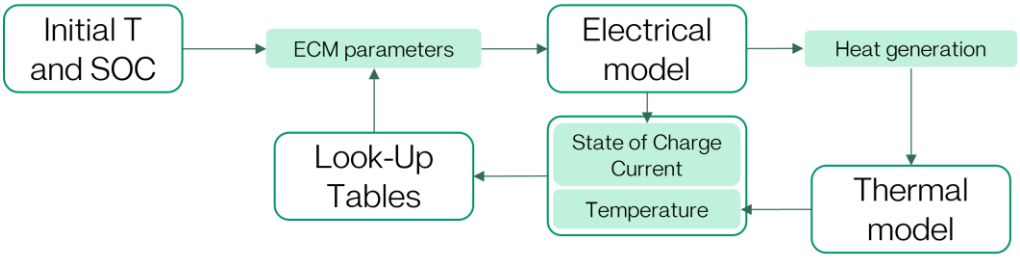

Figure 4 - Electric and thermal interactions in the battery pack model

Every cell is connected in series and the additional resistance R_{add} of the connectors should also be considered. For simplicity, this additional resistance is assumed to be constant and independent from the electrical and thermal quantities.

2.3.2 Thermal model

Battey ageing is closely correlated to electrothermal and thermodynamic phenomena. Therefore, to properly estimate capacity fade and resistance increase with high accuracy, it is fundamental to analyse the thermal behaviour of the system. Battery systems have been thermally modelled in several ways. Gu & Wang [17] developed a multidimensional electrochemical-thermal model of a battery system. They consider the fundamental laws of thermodynamics and integrated the thermal phenomena with the electrochemical heat

generation. Another approach is to couple a Foster network with a CFD simulation, as the one adopted by Hu et al. [18] in their thermal model of battery systems for electric vehicles.

For our purpose, a zero-dimensional, lumped parameter simplified model is the best choice in terms of computational speed and resolution of the results. Through this approach we cannot identify the temperature distribution inside the battery system, but we can obtain the average cell temperature profile with respect to time. This information is sufficient to estimate the thermal stress factors that cause degradation under low C-rate operating conditions such as stationary applications which is the main focus of the current study.

The governing equation of the thermal model is the first law of thermodynamics applied to the battery pack; this law can be simplified with the following assumptions:

- The system is zero-dimensional, and all its properties are evenly distributed.
- The system is closed (no mass-flows) but not thermally insulated.
- The volume is fixed and constant.
- No mechanical work is involved.
- There is no variation of potential and kinetic energy within the system.
- Heat generation inside the system is due to reversible and irreversible phenomena.
- The system exchanges heat with the ambient at a constant temperature T_{amb}.
- The system is cooled with a constant liquid volume flow.

The first law of thermodynamics for the battery system is the following:

$$nMc\frac{dT}{dt} = \dot{Q}_{gen} + \dot{Q}_{amb} + \dot{Q}_{cool} \tag{4}$$

Where n is the number of cells in the pack, M is the mass of one cell in kg, c is the specific heat of one cell in J/(kg K), $\frac{dT}{dt}$ is the rate of change of the temperature of the system in K/s, $\dot{Q}_{gen}$ is the heat generated inside the system in W, $\dot{Q}_{amb}$ is the heat exchanged between the system and the ambient in W, $\dot{Q}_{cool}$ is the heat exchanged between the system and the liquid coolant in W.

The following schematic in Figure 5 represents the thermal model of the battery system, including the heat flows.

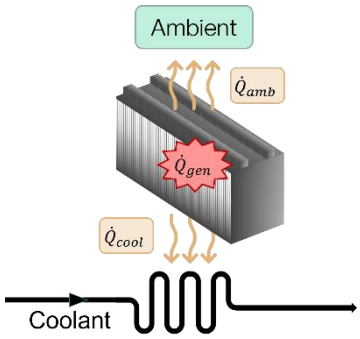

Figure 5 - Schematic of the battery pack thermal model

Eq. (4) is a first-order ordinary differential equation, since on the left-hand side there is the first derivative of the unknown $T(t)$ and on the right-hand side there are the heat flows that depends on the system temperature. The boundary condition is that the system temperature at $t = 0$ is T_0. Every term of Eq. (4) will be explained and modelled in the next section.

Thermal mass

The thermal capacity of the system has been obtained through cooling curve test on the single cell (Appendix A). This experiment provided the thermal capacity of one cell, which is assumed to be constant. When considering a battery pack, this thermal capacity is multiplied by the number of cells in the pack.

Heat generation

The electrical and electrochemical reactions occurring inside the cells during charge and discharge generate heat, hence they should be modelled to correctly apply the first law of thermodynamics. These phenomena [19] are of two kinds:

- Irreversible losses due to the joule effect in the internal electrical resistances.
- Reversible losses due to entropy change resulting from electrochemical reactions within the cell.

In total, four heat component will be considered, namely the ohmic loss, the additional loss, the diffusive loss, and the entropic loss, where the first three are irreversible and the last one is reversible.

The ohmic loss is due to joule dissipation effect in the internal series resistances R_0. It is expressed by the following equation.

$$\dot{Q}_{ohm} = \sum_{i=1}^{N_{cell}} R_{0,i}\, I^2 \tag{5}$$

The additional loss is caused by joule effect in the additional resistances R_{add}. The following equation is used to compute it.

$$\dot{Q}_{add} = \sum_{i=1}^{N_{cell}} R_{add,i}\, I^2 \tag{6}$$

The diffusive loss happens at each RC branch and is evaluated with this equation:

$$\dot{Q}_{diff} = \sum_{i=1}^{N_{cell}} \left(U_{1,i} + U_{2,i}\right) I \tag{7}$$

The reversible heat loss is governed by the entropic coefficient ($\frac{\partial V_{OCV}}{\partial T}$) and it is caused by the reversible entropy change inside the battery due to electrochemical reactions. It is described by the following equation.

$$\dot{Q}_{rev} = I_{cell} T_{cell} \frac{\partial V_{OCV}}{\partial T} \tag{8}$$

Where I_{cell} is the current intensity in Ampere, T_{cell} is the temperature of the cell in Kelvin, $\frac{\partial V_{OCV}}{\partial T}$ is the entropic coefficient in V/K, which is a function of the State of Charge and is calculated based on experimental data. The correlation $\frac{\partial V_{OCV}}{\partial T}(SOC)$ has been determined experimentally through tests.

Ambient cooling

The battery pack is not insulated, therefore it exchanges heat with the ambient via natural convection. The heat flow is estimated using an equivalent thermal resistance between the battery pack and the ambient. The value of the resistance is assumed to be constant and has been determined experimentally through cooling curve test. The following equation describes the heat exchanged between the system and the ambient:

$$\dot{Q}_{amb} = \frac{T_{amb} - T_{cell}}{R_{th,amb}} \tag{9}$$

Where $R_{th,amb}$ is the thermal resistance in K/W, T_{amb} is the ambient temperature in K and T_{cell} is the battery pack temperature in K.

Liquid cooling

The battery pack is liquid cooled with a heat exchanger. The fluid enters the system at atmospheric pressure, ambient temperature, and constant volumetric flow rate. The thermal phenomenon is forced convection and, also in

this case, the heat exchanged is obtained through an experimental thermal resistance, determined by cooling curve test. The thermal resistance is assumed to be function of the volume flow rate: through the tests, an empirical correlation is obtained. The following equation describes the heat exchanged between the battery system and the liquid circuit.

$$\dot{Q}_{cool} = \frac{T_{cool} - T_{cell}}{R_{th,hex}(Q)} \tag{10}$$

Where $R_{th,hex}(Q)$ is the thermal resistance, in K/W, as a function of the volume flow rate Q, in L/min. T_{cool} is the coolant temperature, in K, assumed to be equal to the coolant outlet temperature. Our lifetime simulation is therefore investigating the ageing associated to the battery pack in the most critical condition, so that the degradation is estimated in a conservative way. For this reason, the battery pack temperature is assumed to be equal to the temperature of the most critical cell, i.e., the one with the highest temperature.

To solve the thermal problem, it is necessary to consider an additional equation. The following equation is the first law of thermodynamics applied to the system containing the coolant in the heat exchanger.

$$\frac{dT_{cool}}{dt} V_{hex} \rho_{cool} c_{cool} = \dot{m} c_{cool} (T_{out} - T_{in}) + \dot{Q}_{cool} \tag{11}$$

As already stated, we are assuming that $T_{cool} = T_{out}$ for conservative reasons. V_{hex} is the volume of the heat exchanger in m^3, ρ_{cool} is the liquid density in kg/ m^3, and c_{cool} is the specific heat of the coolant, in J/kgK. $\dot{m}$ is the coolant mass flow rate in kg/s. Being a first-order differential equation, we need an initial condition, being the coolant initial temperature, which is set equal to the ambient.

2.3.3 Ageing model

Battery degradation is a complex phenomenon [20]–[22] since it depends on multiple factors and is caused by several electrochemical reactions. Being able to properly estimate the State of Health (SOH) of a battery is of the utter importance, since low state of health causes poor performances, safety issues and economic losses. Battery SOH is usually expressed in two ways: capacity fade and internal resistance increase [23], [24]. The remaining capacity of the battery is a crucial indicator to assess the SOH and is usually used as an end-of-life (EOL) criteria; in most of the storage applications EOL happens when the remaining capacity is 70-80%. Resistance increase involves all the internal resistances of the battery and is usually caused by parasitic reaction happening within the cell.

Many studies agree that the main causes of battery degradation are Solid Electrolyte Interface (SEI) formation and Lithium plating on the anode surface [21], [22], [24]–[26].

SEI layer grows on the anode-electrolyte interphase, and it is fundamental to passivate the electrode and prevent short circuit. However, an excessive growth of this layer causes a decrease of the electrode porosity and inhibits the kinetic transport of Lithium ions [24]. This results in an increased resistance to the ionic flow which limits the available power and capacity. The formation of the SEI layer leads to contact loss within the anode, causing impedance increase.

Another critical phenomenon is lithium plating, which occurs on the anode surface when the local electric potential drops below zero [24]. In this case, instead of intercalating inside the graphite structure of the anode, Li-ions form a layer of metallic lithium on the anode surface, causing an irreversible loss of lithium ions.

These two reactions, together with several minor parasitic reactions, cause battery degradation through three main modes [25]: Loss of Lithium Inventory (LLI), Loss of conductivity (LC) and Loss of Active Material (LAM). LLI refers to the irreversible reactions that prevent lithium ions to carry charge and intercalate in the electrodes. LC is correlated to the increase of the resistances and the flow limitations. LAM indicate a loss of the active material in the anode and cathode due to cracking, graphite exfoliation, reduced electrical contact, structural changes [27], [28].

Most of those parasitic reactions are enhanced at higher temperature, causing faster degradation of the cell. A highly degraded battery experiences significant resistance increase, which result in higher heat generation during operation. This causes temperature rise which results in a positive feedback thermal loop [29]. To prevent this to happen, temperature of the battery must be controlled with cooling systems.

Battery degradation is commonly modelled with two components: calendar ageing and cycle ageing [20], [22]. The first one occurs when the battery is storing energy without exchanging energy with an external circuit; the second one happens when the battery is charged or discharged. Many studies have modelled these two components empirically, by means of ageing curve and stress factors that depends on multiple conditions. Usually, those models consider either calendar [30], [31] or cycling [32]. The accuracy of the estimation depends on how many degradation factors are considered. Refs. [33], [34] estimated both cycle and calendar ageing but did not consider current rate. Petit et al. [22] did not consider depth of discharge and maximum state of charge as stress factors.

Some studies consider cycle and calendar age separately, without including the calendar component if the battery is cycling. Indeed, the battery may be in cycle mode for a prolonged period, during which the system experiences also calendar ageing. In our model, when the battery is in cycling mode, both the effect of calendar and cycle ageing are considered.

Given that the cited models do not consider some aspect of battery degradation, our approach is one of the most comprehensive one, since it considers the most relevant factors that influence calendar and cycle ageing. In particular, the investigated factors are reported in Table 3.

Table 3 - Stress factors considered for Calendar and Cycle ageing

Calendar Ageing
Resting Temperature
Resting State of Charge

Cycle Ageing
Average Temperature
Maximum State of Charge
Depth of Discharge
Root Mean Square of Current

The next section will present more details on calendar and cycle ageing, how they are influenced by the ageing factors and the equation used.

Calendar ageing

Calendar ageing is the degradation caused by material deterioration over time. This aging component acts while the battery is in resting mode, i.e., not charging nor discharging. During this period, the battery experiences parasitic reactions (mainly SEI formation) that reduce state of health. The rate at which these reactions occur are mainly influenced by the cell temperature and the SOC. Barré et al. [35] highlighted that SEI growth is the prominent degradation reaction in calendar ageing. A comprehensive study on calendar ageing from Naumann et al. [36] investigated the influence of temperature and SOC on degradation. Their results show that the battery experiences stronger degradation for higher temperatures and SOC.

Many studies [32], [37], [38] developed empirical model of calendar ageing based on a power law equation, adjusted with multiplicative stress factors that consider temperature and SOC. In a similar fashion, we adopted the following equation to estimate capacity fade due to calendar ageing.

$$Q_{cal} = D_{q,cal}(T_{rest}, SOC_r)\left[\left(t_{q,eq} + t\right)^{\alpha_{q,cal}} - t_{q,eq}^{\alpha_{q,cal}}\right] \qquad (12)$$

Where Q_{cal} is the capacity loss, experienced during the calendar period t. $D_{q,cal}(T_{rest}, SOC_r)$ is a stress factor that depends on the average temperature T_{rest} during the calendar period and on the resting State of Charge, SOC_r. The exponent $\alpha_{q,cal}$ is an empirical coefficient, obtained by curve fitting in ageing tests. t_{eq} is the equivalent time at the beginning of the calendar period. If the battery is new, its capacity is 100% and the equivalent time is zero. If the battery has experienced degradation, the equivalent time corresponds to the hypothetical amount of time needed to degrade the battery from 100% capacity to its actual capacity, exclusively though calendar ageing at a Temperature and State of charge equal to T_{rest} and SOC_r. It can be calculated with the following equation:

$$t_{q,eq} = \left(\frac{1 - SOH_c}{D_{q,cal}(T_{rest}, SOC_r)} \right)^{\frac{1}{\alpha_{q_{cal}}}} \tag{13}$$

Where SOH_c is the state of health of the battery, in dicating the remaining capacity. After the calendar period t, the battery has experienced a capacity fade equal to Q_{cal}, therefore the new capacity is equal to the remaining capacity at the beginning of the calendar period minus the capacity fade.

The internal resistance increase caused by the calendar ageing can be calculated in a similar fashion, with the difference that the state of health SOH_r will be 100% when the battery is new and will numerically increase when the battery ages. Every resistance in the system, namely the series internal resistance, first charge dynamics resistance and second charge dynamics resistance, is equally affected by this phenomenon. Their actual value is given by the product of their value at Beginning of Life and the resistance state of health, SOH_r. During the calendar period t, the resistance increase is given by the following equation:

$$R_{cal} = D_{r,cal}(T_{rest}, SOC_r)\left[\left(t_{r,eq} + t \right)^{\alpha_{r,cal}} - t_{r,eq}^{\alpha_{r,cal}} \right] \tag{14}$$

Where $D_{r,cal}$ is a calendar stress factor regarding resistance increase and $\alpha_{r,cal}$ is the calendar ageing exponent regarding resistance increase. $t_{r,eq}$ is the equivalent time, defined as the previous equivalent time $t_{r,eq}$, except that indicates the period needed to reach a resistance increase of SOH_r at the beginning of the calendar period. Therefore, it can be obtained with the following equation:

$$t_{r,eq} = \left(\frac{SOH_r - 1}{D_{r,cal}(T_{rest}, SOC_r)} \right)^{\frac{1}{\alpha_{r_{cal}}}} \tag{15}$$

The updated value of the resistance state of health can be obtained by adding R_{cal} to SOH_r at the beginning of the calendar period.

Cyclic ageing

Cyclic aging is the degradation caused by the charge and discharge operation of the battery. Many factors influence this kind of degradation, such as maximum SOC, Depth of Discharge (DOD), average temperature and current rate [22]. Also in this case, the degradation is mainly caused by SEI formation and, in extreme cases, Lithium plating. These parasitic reactions are enhanced when the anode potential is lower [24], which happens during charge or for high SOC. Especially when the battery is charged to high SOC and stores energy for a prolonged period, the degradation is accelerated [20].

DOD indicates the amount of energy withdrawn by the battery during discharge. Huang et al. [26] showed that the deeper the DOD, the faster the battery degrades. Also, fast charging and discharging can decrease SOH, since in causes overheating, mechanical stress, structural changes, uneven SEI formation, anode exfoliation [39].

Temperature has a critical role in cycle ageing. Even in this case, a higher temperature accelerates the degradation process. However, also low temperature can be dangerous for battery health since it reduces capacity and increase internal resistance [29].

A common approach to model capacity loss from cycle ageing is with a power law, corrected with stress factors that consider operating conditions [32]. In our case, the equation adopted is the following.

$$Q_{cyc} = D_{q,cyc}(T_{ave}, I_{RMS}, DOD, SOC_{max}) \left[\left(Ah_{q,eq} + Ah \right)^{\beta_{q,cyc}} - Ah_{q,eq}^{\beta_{q,cyc}} \right] \quad (16)$$

Where $D_{q,cyc}(T_{ave}, I_{RMS})$ is a cycling stress factor that depends on the average temperature during cycling, T_{ave}, and the root mean square value of the current, I_{RMS}. $F_{q,cyc}(DOD, SOC_{max})$ is a cycling stress factor that depends on Depth of Discharge, DOD, (i.e. the difference between maximum and minimum SOC during the cycling period) and the maximum State of Charge during the cycling period, SOC_{max}. Ah is the current throughput in Ampere hours and it corresponds to the time integral of the absolute value of the electric current during the cycling period. $Ah_{q,eq}$ is the equivalent throughput, which corresponds to the electric current throughput needed to degrade the battery from the BOL condition to the actual degradation at beginning of the investigated cycling period, SOH_c. It is expressed by the following equation:

$$Ah_{q,eq} = \left(\frac{1 - SOH_c}{D_{q,cyc}(T_{ave}, I_{RMS}, DOD, SOC_{max})} \right)^{\frac{1}{\beta_{q,cyc}}} \quad (17)$$

Where SOH_c is the state of health of the battery, indicating the remaining capacity. After the cycling period, the battery has experienced a capacity fade equal to Q_{cyc}, therefore the new capacity is equal to the remaining capacity at the beginning of the cycling period minus the capacity fade.

The resistance increase is calculated in a similar fashion, using the following equation:

$$R_{cyc} = D_{r,cyc}(T_{ave}, I_{RMS}, DOD, SOC_{max}) \left[\left(Ah_{r,eq} + Ah \right)^{\beta_{r,cyc}} - Ah_{r,eq}^{\beta_{r,cyc}} \right] \quad (18)$$

Where $D_{r,cal}$ and $F_{r,cyc}$ are stress factor related to resistance increase degradation and $Ah_{r,eq}$ is the equivalent throughput to degrade the battery to SOH_r at the beginning of the cycling period. It can be calculated with the following equation:

$$Ah_{r,eq} = \left(\frac{SOH_r - 1}{D_{r,cyc}(T_{ave}, I_{RMS}, DOD, SOC_{max})} \right)^{\frac{1}{\beta_{r,cyc}}} \quad (19)$$

The new value of the resistance state of health can be obtained by adding R_{cyc} to SOH_r at the beginning of the cycling period.

3 Methodology

This section will explain in detail the methodology adopted to compare and benchmark four simulation software. In the first part we will describe the software, providing insights into their common use, features, and modus operandi. Then, four KPIs will be introduced. The comparison will be based on these four areas, for each software. Section 3.1 will describe the four selected software. Then, the investigated aspects of the comparative study will be showcased in section 3.2. In section 3.3 the hardware is described, and in section 3.4 the battery model implementation is explained, for all of the four tools.

3.1 Software selection

Battery system simulation can be run with a wide variety of tools. For this study, we focused on four of the most common approaches, used in industry and academia to model and simulate battery systems. This section will briefly describe every tool.

3.1.1 Simcenter Amesim

Simcenter Amesim [40] is a system simulation software that allows to model Multiphysics systems. It is used to design and optimize complex systems, taking into consideration various phenomena thanks to a drag-and-drop interface, where many pre-built components are already implemented. This software allows to investigate electrical, fluid-dynamic, mechanical, thermal and control systems with a customizable approach, since the user can modify parameters and input experimental data. However, it is not possible to change equation and functionalities behind each component block directly.

Figure 6 - Simcenter Amesim launch window.

This software is developed by Siemens and is part of the wide pool of software in the Simcenter suite. This means that it is possible to interface system simulation with other projects, built with other Simcenter tools, which include testing and designing tools. Users can integrate it with several Computer-Aided Engineering (CAE) and Computer-Aided Design (CAD) tools and interoperate Functional Mock-up Units (FMU) developed with other simulation tools. This feature guarantees a higher interoperability of simulation tools, allowing models to be operated in different environments.

Amesim interface has four modes, namely "Sketch", "Submodel", "Parameter" and "Simulation". In the "Sketch" mode it is possible to build the multiphysics system by choosing physical components from an embedded library. In the case of the battery system model, the components are mainly electrical (*current source*, *voltage sensor*, *battery pack*, ...), thermal fluid (*pumps*, *heat exchanger*, *fluid properties*, *thermal resistance*, ...) and control signal (*charge/discharge controller*, *property inspectors*, *integrators*, ...). All the components are wisely interconnected to enable the interaction of different effects. In the "Submodel" mode we can choose different modelling characteristics for each component. For example, the *battery pac*k component has three different submodels, namely *Simple ECM*, *Advanced ECM* and *Electrochemical model*, which represents the most common simulation strategies for Li-ion cells.

The "Parameter" mode allows to set the general parameters of the model and to input physical quantities in each component. Parameters can be imported from external files in the format .data, which can contain values and lookup tables. In the case of ECM parameters, they depend on current, temperature and SOC, so these parameters can be input as 3-D lookup tables and Amesim will automatically interpolate the value based on those electrothermal quantities.

The last mode is "Simulation", where it is possible to select the type of solver, the fixed or variable timestep and other run parameters. Then, the simulation is run, and all the logs are displayed through a window. Once the simulation is completed, it is straightforward to visualize the results just by selecting a component and clicking on the quantity that we want to inspect. All results can then be exported or post processed in the "Post processing" window. The simulation can also be run from an external Python script, enabling iterative simulation to run smoothly. The Python library "`amesim`" includes all the commands useful to set parameters, run simulation and extract results. In this way it is possible to run lifetime simulation by implementing the workflow described in Appendix B with only one python script.

3.1.2 MATLAB/Simulink

MATLAB [41] is one of the most widespread programming environments among industry and academia. It is a proprietary high-level programming language initially released in 1984, and is developed by MathWorks, providing an interactive platform to perform numerical computation, data analysis and programming. It is often used by engineers and scientists to develop algorithms and to implement models.

Figure 7 - MATLAB/Simulink logo

Simulink [42] is an Add-on to the MATLAB environment which provides a visual environment for programming and modelling dynamic systems. It can be used to perform simulations, to optimize systems, to build and analyse models based on non-linear and differential equations. For the analysis and simulation of complex systems, MATLAB and Simulink offer a complete collection of tools.

Simulink can be interoperated with MATLAB, since they share a common workspace where all the variables are saved. By implementing a MATLAB script, it is possible to import data and create functions which can directly be used in a Simulink model. Also, the simulation output can be analysed and post-processed in MATLAB, enabling a straightforward link between the two platforms.

Simulink library does not contain multiphysics components; hence, a preliminary step is to implement all the physical governing equations of the system. This requires additional modelling effort because a deep knowledge of the physical phenomena is needed. Once we define the set of equations, these can be implemented in the Simulink model by means of drag-and-drop blocks that perform basic algebraic and logical operation.

3.1.3 MATLAB/Simulink/Simscape

Simscape [43] is an add-on package to the MATLAB/Simulink environment that contains a library with multiphysics components. This enables to use Simulink as a platform to build multiphysics model with a library of pre-

built blocks that mimic the physical behaviour of different part of the system. The foundation library of Simscape includes multiple domains, such as electrical, hydraulic, thermal, mechanical and many more. These libraries contain all the sources, elements, and sensors useful to create such models.

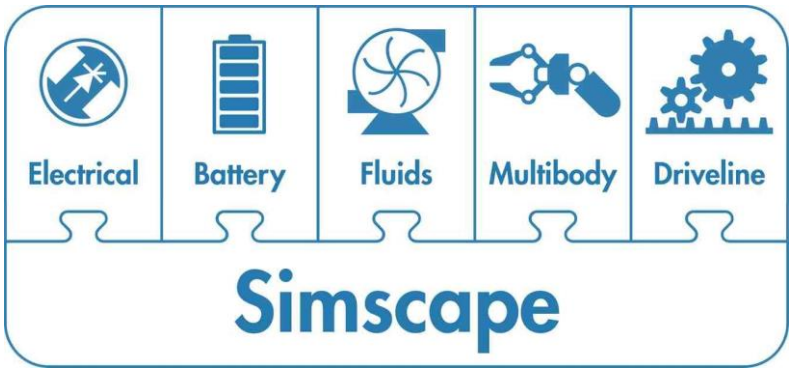

Figure 8 - Simscape logo

Since Simscape is based on a physical modelling approach, is not necessary to know the underlying laws of the system since all the governing equations are already implemented in every block, which simplifies the process of model development and implementation. It is possible to access the source code of the component to understand the underlying equation. It is also possible to modify the script and generate new components in the library, but they can only be developed via the MATLAB programming language. Other than the Simscape foundation library, MathWorks developed several add-on Simscape libraries, specifically designed for a physical domain. In our case, we implemented the battery system model also thanks to the two additional libraries “Simscape Battery” and “Simscape Fluids”.

3.1.4 Dymola

Dymola [44] is a simulation, modelling, testing and post-processing environment developed by the French software house Dassault Systèmes. It is based on Modelica, an open-source modelling language, used in academic and industrial contexts. Systems can be modelled in Dymola with multiphysics component, and many other blocks already implemented in the Dymola library; it is possible to fully access the code behind every block and modify its equations, parameters, and variables by programming with the Modelica language, allowing users to personalize or create new blocks based on their needs. When the simulation is initialized, the Modelica Compiler converts the model to C code and run the script.

Figure 9 - Dymola logo

The Dymola interface shows three main modes: Graphics, Text and Simulation. In the "Graphic" mode it is possible to import blocks and components from the native Modelica library or from additional libraries developed by Dymola. In our model, we mainly used basic Modelica blocks, in combination with more advanced components from the "Battery" and "Cooling" Dymola libraries. Every library contains a vast variety of already built examples, which can be used as a starting point to build customized models. Dymola allows users to extend or duplicate blocks: in the first case, by extending a block it is possible to customize its properties, without changing the default functionalities of the block, which remains the same as the source block in the library; duplicating a block means creating a new library component, which can be fully customized and implemented in the system model.

The model implementation is multi-layered, meaning that every component can be explored at a deeper level to visualize and change the underlying blocks. A different way of building the system is by using the Text mode. The graphic mode and the text mode describe the same system model and the two interfaces are interconnected, meaning that the user can implement the model graphically and see the changes in the script, and vice versa.

In Text mode it is possible to visualize and modify the code behind every block using the Modelica programming language. This mode allows the user to define system variables, parameters, and events. For instance, the ageing model has been implemented partially with the graphics mode (to import the lookup tables), but also at text level, writing the ageing variables and the event triggered when the system switches between resting and cycling mode.

Once the model is built, it can be simulated in the "Simulation" tab. The default solver is the native "dassl" solver, which uses a variable timestep, but it is possible to select different solvers, both with fixed and variable step. Every variable of the system can be displayed on a graph or on a table, which can then be exported in different formats.

The Dymola simulation can be run from an external python script by importing the Dymola library. This library contains several commands that let

the user modify parameters, set initial conditions, run simulation and extract results.

3.2 Key performance indicators

This comparative study has the ambition to analyse different simulation tools in the most comprehensive way, covering multiple aspects of software that can be relevant to various users. Other studies, such as the one conducted in Ref. [45], was focused on aspects that are relevant to business users, professionals, or industry, without considering that other users from academia, such as students or researchers, may put more value in other aspects of a software. Based on our specific goal and on the state-of-the-art software comparison, we intend to evaluate four main aspects of the software: Performance, Ease of Use, Features and Licensing. Some of these have been assessed quantitatively, by conducting tests and experiments and comparing the results; some others, are reviewed qualitatively, by describing the user experience and the distinctive aspects of the tool. In the this the four investigated aspects will be introduced and described, aiming to explain what they address and how they can be evaluated.

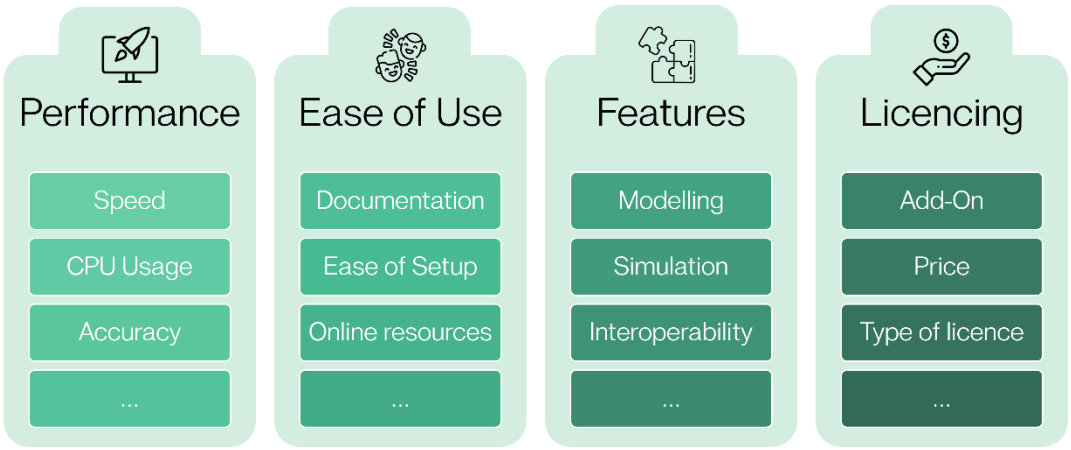

Figure 10 - Investigated key performance indicators.

3.2.1 Performance

One of the key aspects of a software is related to its overall performance. This indicator can be measured by different tests on all the software, and the main two aspects that were investigated are the speed of the simulation and the accuracy of the results. Similarly, many other comparative studies of simulation software compared the different tools based on their accuracy [46]–[50] and speed [7], [46], [49].

Simulation runtime can be a determinant of what software to use, especially for systems with complex and multiple physics, and for lifetime analysis. Increasing the speed of the simulation will allow users to enhance their workflow, obtaining more results faster for a better and more comprehensive analysis. In an industrial context, enabling fast computation can be critical to meet business goals and provide valuable information to customers. In the research context, this enables us to obtain valuable information sooner, especially in fast moving fields, like energy storage. Simulation speed was tested and compared in the studies from Jmai et al. [49] and Magni et al. [7], which also stress the importance of this characteristic.

To make a fair comparison of the software, the very same multiphysics model of a battery system, explained in Section 2, is implemented with the four tools, and a 10-year lifetime simulation are run in the same conditions. All the four use cases depicted in paragraph 2.2 are tested on the four software, giving sixteen simulation outputs. For every use case, a comparison of the simulation runtime of each of the four tools will show the speed performance. To have a more reliable comparison, each simulation is run ten times, and an average runtime is considered. All simulations are run on the same local workstation described in Section 3.3, while all other tasks be idle.

Since there are no real test data regarding ageing in the lifetime simulation, it is not possible to assess the accuracy of the software based on the previous analysis. Judkoff et al. [51], [52] identified three methods to validate simulation results, to assess their accuracy. The output can be compared with real test data (empirical validation), or between the different tools (cross-validation), or with an analytical solution (analytical validation). The last approach is only available for simple systems when it is possible to solve analytically all the equations of the system. In our case, the system contains many non-linearities and parametric maps, which makes this method hardly applicable. Cross-validation has the disadvantage that it is difficult to define a reference against which to assess the accuracy of the other software. Moreover, the same multiphysics model was implemented in all the investigated tools and there is no difference in the comprehensiveness level of the model among the four software. For the third method, empirical validation, it is necessary to have test data available, which we gathered for different operating conditions. Camargos et al. [48] validated the results of two simulation software against catalogue data. Also, the comparative study from Chakrabarti et al. [50] estimates the accuracy of different simulation tools with respect to real data obtained from experiments. In our case, we compared the four software with experimental tests and used empirical validation to assess their accuracy. We used values from two different tests.

For each test, there are information regarding the current, voltage, coolant flow rate and pack temperature. Therefore, the analysis was structured as follows:

- Setup the same parameters and operating conditions of the real experiment on every software.
- Input in the four simulations the same current profile measured during the test. The tools will read the current values from the same experimental timeseries.
- Input the same coolant flow rate timeseries, in a similar way as the current profile.
- Extract from every simulation the voltage and the temperature profiles.
- Compare the simulation output values with the test values, by evaluating their relative error.

The relative error of the simulation output with respect to the test values is given by the following equation:

$$Err = \frac{|X_{sim} - X_{test}|}{|X_{test}|} \tag{20}$$

Where X_{sim} is the simulation output in a timeseries format, X_{test} is the test values. In our case X represents the two investigated outputs, which are the voltage and the temperature profiles.

3.2.2 Ease of use

The second aspect of our comparative study aims to assess the ease of use of the software. Different software may require a higher level of expertise, while others may have a steeper learning curve. Software that requires less modelling effort, or with a more intuitive interface and a smoother workflow will allow users to produce results faster, with a smaller probability of errors and with an overall better experience. Other comparative studies of software focused their attention on this aspect, to guide users in their choice. Gardner et al. [3] compared three FEM simulation software, providing insights on the ease of setup. The software usability was considered as a relevant aspect also by Jafrancesco et al. [47], who compared four simulation software, also based on their Graphical User Interface (GUI) and on their general characteristics regarding ease of use. Attia et al. [2] assessed usability of the software based on 249 interviews to experts and users, summarizing their opinions in tables. In our case, the software comparison will be based on the personal experience of modelling the BESS on different software. As opposed to performance, the ease of use can hardly be quantified by specific indicators based on experiments. The way we intend to compare the different software is by giving a qualitative review of the user experience for each of the four tools, providing insights and suggestions regarding the model implementation. This part will consider all the different factors that determine how easy it is to learn and

use the software. The GUI is a relevant feature of the tool since it is the mean though which the modeler develops the project. Another factor that determines the usability of the software is whether it has pre-built blocks that represent a physical component. With this feature, it is possible to implement complex systems without a deep knowledge of the governing equations, just by drag-and-dropping the block needed. An aspect that was considered by Attia et al. [2] is the flexibility of use, which describes the ability to accommodate both beginners and advanced users on the same software. Indeed, some tools can be properly used only by expert users, while others might be more suitable for unexperienced users, but lack the ability of complex customization of the system when needed. Having a knowledge-based platform, containing detailed documentation, and already built examples, are determining factors in mastering the software, especially in the early stages of use. On top of that, good customer support and online documentation can improve the user experience and simplify the learning process. All these factors, and many more, will be evaluated while using the four tools, and results will be shown in Section 4.2.

3.2.3 Features

The third relevant aspect of a software regards its features and functionalities. Most of the multiphysics system simulation tools have similar capabilities, but every software developer implements unique features to gain competitive advantage to other software houses. These features concern different aspects of the software, which can be related to the solver options, the level of physical detail of the modelling components, the interoperability with other platforms and many others. Also in this case, similarly to the “ease of use” aspect, it is not appropriate to compare software based on experiments and indicators. Nevertheless, a more quantitative comparison can be conducted. Many software comparative studies [2], [3], [7], [47] summarize the relevant features in tables, showcasing a feature in each row and software in each column. We believe that this approach is effective to describe the overall capabilities of the software, and to highlight the uniqueness of each tool. Hence, Section 4.3 provides a summarizing table of the feature of each software, together with a more extensive review for each software. The aim is to highlight the different peculiarity of the tools, so that the user can choose the best approach based on his/her specific requirements. Our description has the ambition to cover all the relevant aspects of the tools, such as the interoperability, the level of physical detail, the simulation options, optimization capabilities, the comprehensiveness of the library and many more.

3.2.4 Licensing

The last but not least aspect of this comparative study is related to the licensing scheme. This factor is relevant in both industry and academia since

the cost of software can represent an important factor in terms of resource management. In a comparative study of simulation software from Jafrancesco et al. [47], the tools are compared also based on the license type, such as open-source, freeware, commercial or home-made. In a similar fashion, we will be describing the licensing scheme of every tool, also giving insights on the costs and license type. On top of that, some software may provide a license for the basic functionalities and require additional purchase for add-ons and libraries specifically built for a particular application, such as battery pack models or thermal cooling systems. Section 4.4 provides deeper insights into this aspect.

3.3 Hardware selection

One of the main performance indicators that we intend to compare is the simulation runtime of every software. To conduct a fair comparison, all the simulations will be run locally on the same workstation, with the same operating conditions. The hardware selected for this study has the following specifications [53]:

Processor	Intel(R) Core(TM) i7-1065G7 CPU
Processor base frequency	1.30GHz
Max Turbo Frequency	3.90 GHz
Total Cores	4
Total Threads	8
Installed RAM	32.0 GB (31.6 GB usable)
Operating System	Windows 10 Enterprise
System type	64-bit operating system

While running the simulation, no other tasks were active, so that all the computational effort is dedicated for our simulation. To guarantee consistency of results, simulation has been run 10 times and an average value of the runtime has been considered.

3.4 Model implementation

This section will briefly explain the implementation process of the multi-physics model of the battery system on the four software. A deeper and more comprehensive description of the implementation process is given in Appendix B. This Section 3.4 aims to provide a glimpse into the different modelling approaches for lifetime simulation on the four tools.

3.4.1 Simcenter Amesim

Simcenter Amesim allows user to build battery systems starting from native multiphysics components. The "battery pack" component contains all the relevant equations comprised in the electric and thermal model described in Section 2.3.1 and Section 2.3.2. The liquid cooling circuit can be implemented with multiphysics component, such as pipes, heat exchangers and temperature sensor, and connected to the battery pack.

Regarding lifetime simulation, the Simcenter Amesim model simulates 24-hour operation. This simulation is run 3650 times (for ten years) from an external python script. After the 24-hour simulation is run from the python script, the current, Temperature, and SOC profiles can be extracted and become the input of the ageing model described in Section 2.3.3, which outputs capacity fade and resistance increase. The updated values of capacity and resistance are then updated in the Simcenter Amesim mode, and anew 24-hour simulation is run. This process is explained more in detail in Appendix B.

Being a multiphysics modelling and simulation tool, Simcenter Amesim presents a clear and intuitive interface, as it shown in Figure 11, which depicts the battery pack component connected to the thermal model.

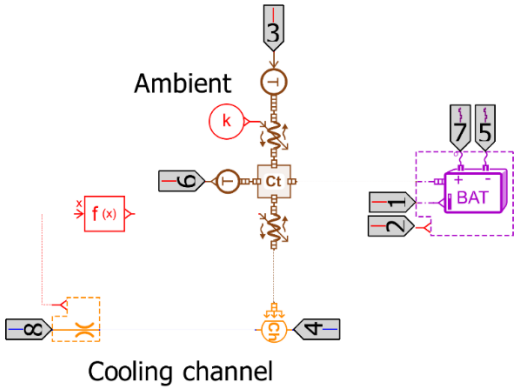

Figure 11 - Simcenter Amesim model of the battery pack connected to the thermal model.

3.4.2 Simscape

The MATLAB/Simulink add-on package "Simscape Battery" comes with the battery builder functionality, which enables users to create a customized battery pack block, by specifying cell geometry, position, connection scheme, ECM parameters and many more features. This battery pack multiphysics block is the added to the Simscape library and can be used in the Simulink environment to simulate a battery pack. Embedded in this block, there are both the ECM electric model and the thermal heat generation for Ohmic effect. The thermal port of the battery pack can then be linked to the thermal

model, developed with Simscape blocks from the “Thermal liquid” library. The system model is displayed in Figure 12.

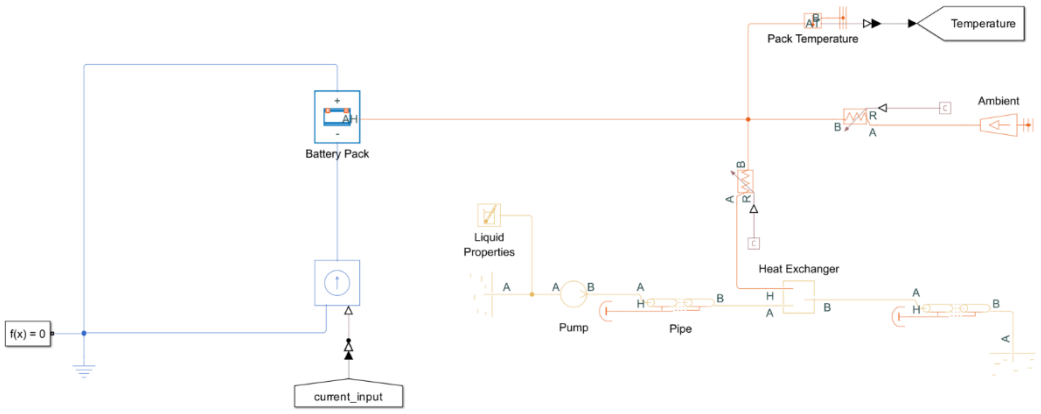

Figure 12 - Simscape model of the battery pack connected to the thermal model.

Similarly to Simcenter Amesim, the Simscape model simulates 24-hours operation. The lifetime simulation is run from an external script that iteratively runs the 24-hour simulation and estimates capacity fade and resistance increase from the output profile. This time, the external script is developed in MATLAB, since it shares the same workspace environment with Simulink/Simscape, hence the interaction between the script and the model is optimized.

3.4.3 Simulink

The implementation of the battery system model in Simulink requires an additional modelling effort, since Simulink library do not contain multiphysics components, but only signal operators. This means that it is necessary to investigate the governing equations of the electric and thermal model (explained in Section 2.3.1 and Section 2.3.2), and implement those relations with basic algebraic and logic blocks in Simulink.

Simulink offers a signal-flow environment that can be used to implement dynamic models based on differential equations. Unlike all the other simulation tools, Simulink does not provide multiphysics components, such as batteries, pipes, voltage sensors and others. This means that the user needs to model the relevant governing equation that represents the system. Figure 13 represents the Simulink model of the system, containing electric model, thermal model, generated heat, and liquid properties.

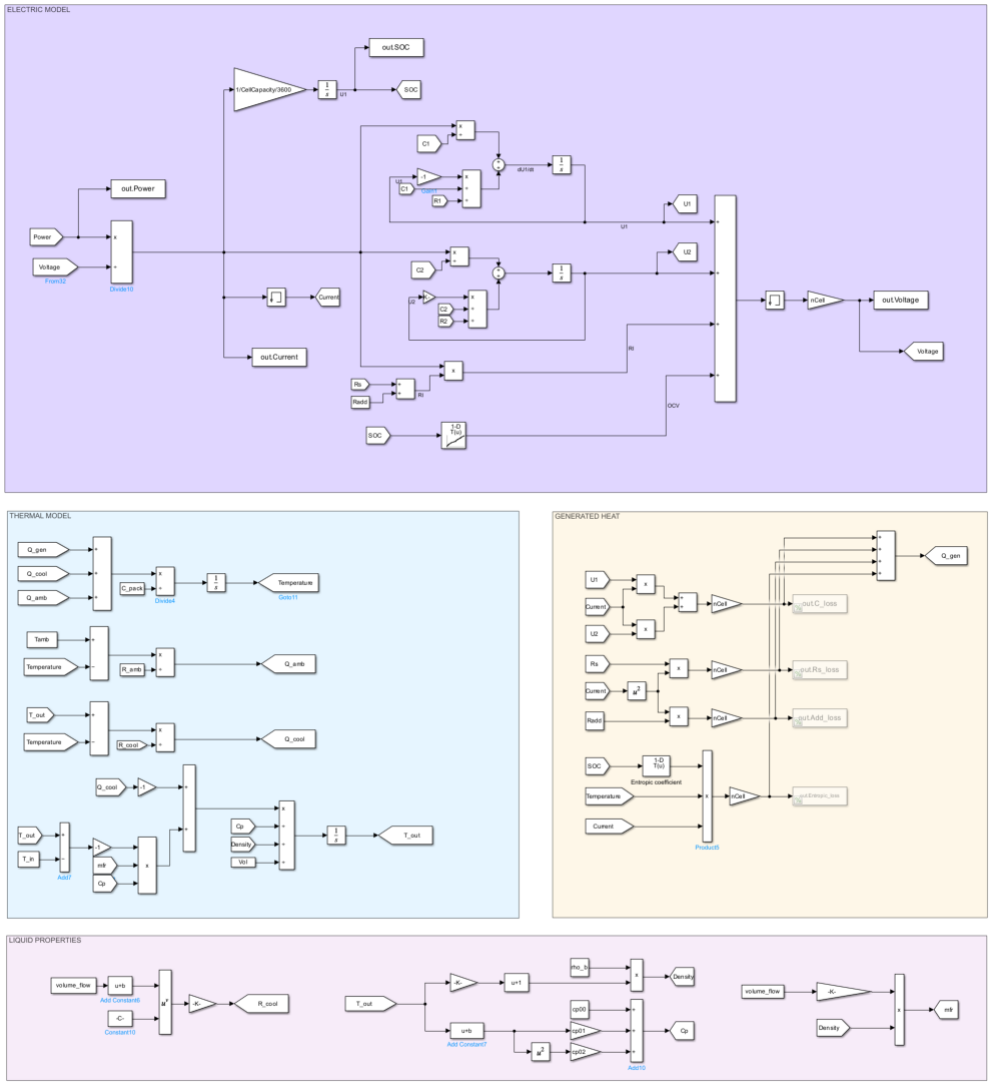

Figure 13 - Simulink implementation of the electric model (top), thermal model (mid-left), generated heat (mid-right), liquid properties (bottom).

Everything must be modelled as set of equations and translated into Simulink blocks. This means that the complexity of the project rises quickly and making changes in the model may require the re-modelling of all equations.

Also in this case, it is convenient to run the lifetime simulation from a MATLAB script, since the Simulink-MATLAB environments share the same workspace.

3.4.4 Dymola

Dymola provides a modelling environment with multiphysics components. Every block is freely accessible, and the source code can be customized. This freedom allows users to modify blocks and model to suit their needs. The battery library provides several “thermal”, “electric” and “aging”

models, which can be combined to form a cell model. The cell model becomes the basis of the battery pack component, which can then be connected to the cooling circuit, as it is shown in Figure 14.

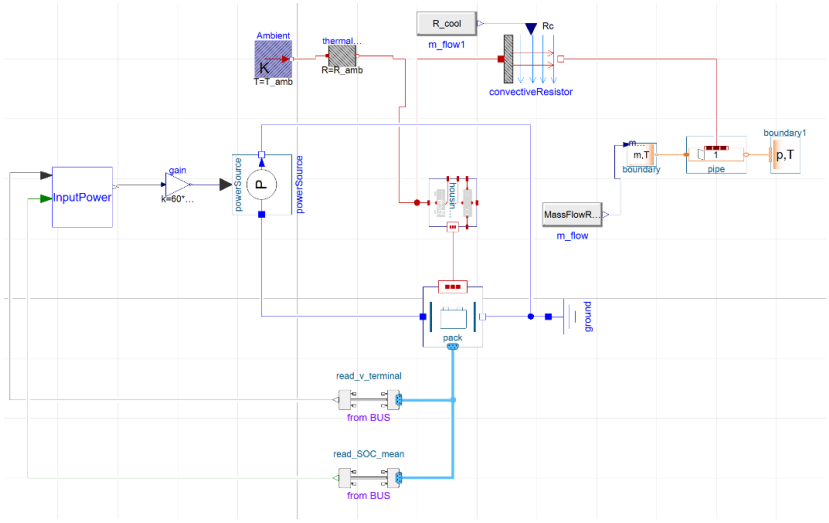

Figure 14 - Dymola model of the battery pack connected to the thermal model.

Since the ageing model has been implemented at cell level in Dymola, there is no need to loop a 24-hour simulation from and external script and calculate degradation accordingly. Instead, the lifetime simulation is run inside the Dymola environment, which ease the model implementation (no need to code externally) and enhance overall performance (no interaction between different platform, but all-in-one solution).

4 Results

This section presents the main findings of the comparative study. The four simulation tools will be compared quantitatively regarding their performance (Speed, CPU Usage and Accuracy), and qualitatively regarding Ease of Use, Features and Licensing.

4.1 Performance

On of the key aspect of a simulation software concerns performance. The same model implemented and simulated on different software may produce different results and with different computational resources. The reason for this lays on the solver configuration, in the different model implementation features and on different block functionalities. Software can be compared based on the required computational resources, which will be measured via the simulation runtime and the CPU usage, and on their accuracy, assessed by the comparison between simulation output and test data.

4.1.1 Runtime

Software simulation runtime will be based on the 24h multiphysics simulation in the four different cases. We believe that this is the fairest comparison to estimate the performance of the simulation. The lifetime simulation runtime, which consists of a loop that runs the 24h simulation 3650 times (10 years), does not represent only the software performance, but also the efficiency of the interaction between the script and the software, which is out of the scope of the study. On top of that, Amesim lifetime simulation has been implemented with a python script, while Dymola the lifetime loop is implemented inside the tool. This different implementation translates in different speed performance, that does not represent the single simulator performance. We will run the 24h simulation 10 times in every case and the average runtime value will be considered. Total elapsed runtime is composed of initialization time and CPU time, which regards the numerical computation period. Table 4 shows the simulation runtime for every software and case. For every software, the table shows the total elapsed runtime to simulate the model and the ratio between the initialization time and the total elapsed runtime, in percentage. For a direct comparison, Figure 15 displays a bar plot of the same results, where every bar is stacked with a brighter bar that represents the initialization time.

Table 4 - Runtime comparison of the four simulation tools for one 24-hour simulation.

	Amesim		Simscape		Simulink		Dymola	
	Elapsed runtime (s)	Initialization time rate (%)	Elapsed runtime (s)	Initialization time rate (%)	Elapsed runtime (s)	Initialization time rate (%)	Elapsed runtime (s)	Initialization time rate (%)
Case 1	**1.047**	2.1	**9.546**	1.6	**0.829**	27.1	**0.260**	9.4
Case 2	**0.901**	2.0	**10.064**	1.6	**0.801**	16.6	**0.243**	9.8
Case 3	**0.851**	1.3	**10.274**	1.5	**0.788**	19.3	**0.259**	9.2
Case 4	**0.779**	1.4	**10.117**	1.6	**0.816**	18.6	**0.254**	10.0
Aver-age	**0.895**	1.7	**10.000**	1.6	**0.809**	20.4	**0.254**	9.6

The results show that Dymola outperformed in terms of simulation runtime, with values ranging from 0.243 seconds in Case 2, to 0.260 seconds in Case 1. Simulink and Amesim showed similar speed, with Simulink performing faster simulation in three out of the four cases. Overall, Simulink performed better than Amesim, with an average runtime of 0.809 seconds versus 0.895 seconds. In terms of elapsed time, Simscape was the slowest, with an average runtime of 10.0 seconds. In terms of initialization, the Dymola simulation took on average 9.6% of the total elapsed time to initialize the simulation, while the same rate on Amesim averaged 1.7%. Simulink took 20.4% average to initialize the simulation, even though it performed better than Amesim. Simscape initialization time represented 1.6% of total time, but in terms of speed it performed 39.3 times slower than Dymola, 12.4 times slower than Simulink and 11.1 times slower than Amesim.

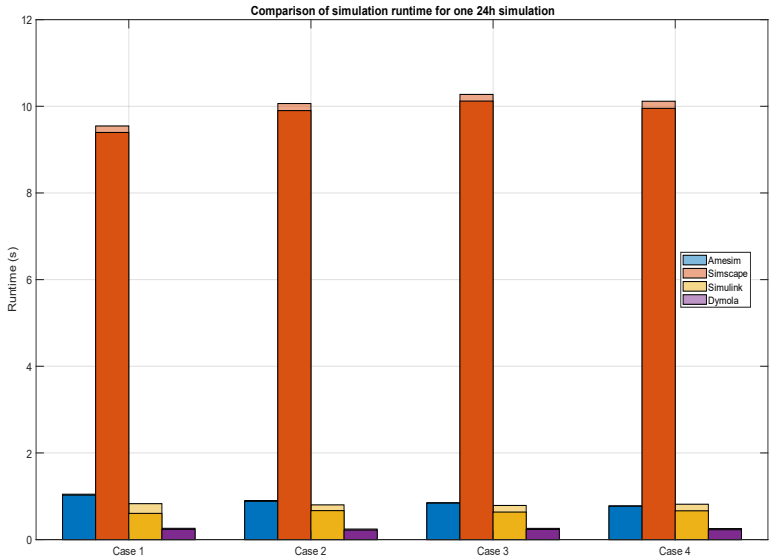

Figure 15 – Runtime comparison of the four simulation tools for the 24-hour simulation.

Figure 15 represents in a bar plot the results shown in Table 4. The lighter bar stacked on top of each bar represents the initialization time.

Runtime of lifetime simulation values are affected not only by the software performance, but also by the implementation method, e.g. external script to run simulation vs simulation run in the same environment. Table 5 shows the simulation runtime of the 10-year lifetime simulation.

Table 5 - Runtime comparison of lifetime simulation on the four tools

	Simcenter Amesim	Simscape	Simulink	Dymola
	Elapsed runtime (s)	Elapsed runtime (s)	Elapsed runtime (s)	Elapsed runtime (s)
Case 1	14953	35770	1013	209
Case 2	16136	36498	1017	158
Case 3	16332	37221	1146	204
Case 4	15439	35627	1077	195
Average	15715	36279	1063	191

It can be noticed that runtime of Simcenter Amesim for 24-hour simulation is only about 3.5 times slower than Dymola (average of 0.895 seconds vs 0.254 seconds), while for lifetime simulation the overall runtime is 82.3 times slower (15715 seconds vs 191 seconds). Most of the computational inefficiency comes from the implementation method, since lifetime simulation in Simcenter Amesim is run from a python script, while everything is embedded in Dymola. A new release of Simcenter Amesim (2304) will allow users to develop a custom ageing model that can be implemented within the battery block. This means that there will be no need of external script, which will enhance the overall computational speed. From the results shown in table 5 it is possible to learn that the interaction between MATLAB and Simulink is significantly more effective than the interaction between python and Simcenter Amesim. For a 24-hor simulation, the two software show similar runtime (on average, 0.895 seconds for Simcenter Amesim and 0.809 seconds for Simulink). However, if we run lifetime simulation for 10 years (3650 iteration of the 24-hor simulation) from an external script (python coupled with Simcenter Amesim and MATLAB coupled with Simulink), the overall runtimes change significantly. The lifetime simulation requires 15715 seconds with Simcenter Amesim, while only 1063 seconds with Simulink, which results 14.8 times faster than Simcenter Amesim. For this reason, the Simcenter Amesim model can be improved by implementing a degradation model within the cell block and avoiding external scripting.

The higher efficiency between MATLAB and Simulink/Simscape is not enough to make Simscape speed performance competitive with the other three tools.

4.1.2 CPU Usage

Another performance metric is the CPU used by the simulation tool. CPU was monitored during the simulation via a python script. While monitoring the CPU used by the simulation task, ten simulations were performed, and the average CPU*seconds value has been considered. The monitoring tool saves the instantaneous CPU percentage used by a specific task. Since simulations require different runtime, the overall computational resources can be obtained by the integration of the CPU usage profile against time. For this reason, the unit used is core*second, which corresponds to the computational resources used by one CPU core at maximum power for one second. Table 5 displays the computational resources used by a 24h simulation for every software. Figure 16 show the same results on a bar plot.

Table 6 - CPU usage comparison for the 24-hour simulation on the four tools

	Amesim (core*seconds)	Simscape (core*seconds)	Simulink (core*seconds)	Dymola (core*seconds)
Case 1	1.62	12.39	2.37	0.87
Case 2	1.44	14.08	1.97	0.86
Case 3	1.44	12.82	2.08	0.91
Case 4	1.44	13.04	2.19	0.94
Average	1.49	13.08	2.15	0.90

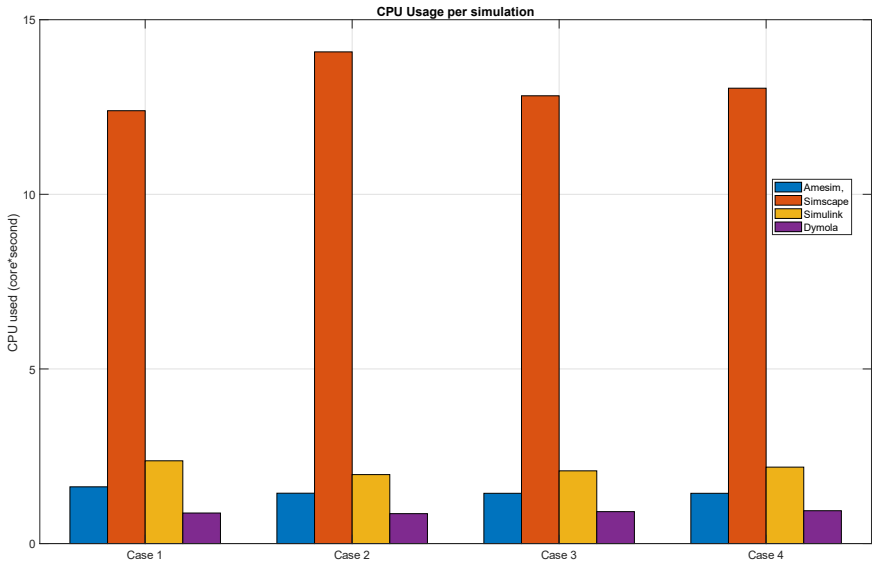

Figure 16 - CPU usage comparison for the 24-hour simulation on the four tools

Also in this case, Dymola required the least computational effort, with an average CPU resource of 0.9 core*seconds. Even though Amesim runtime was slightly higher than Simulink, it required less computational resources, with an average of 1.49 core seconds against 2.15 core seconds of Simulink. Simscape was the simulation tool that required the highest amount of computational effort, with an average value of 13.08 core seconds. This is probably because this simulation has been set with a fixed timestep, while all the other ones use a variable timestep. We could not implement a variable timestep with Simscape because the simulation runtime was too long compared to the other software. We believe that, with further optimization of the model, the software might perform better. This additional analysis can be expanded in a future work.

4.1.3 Accuracy

Simulation outputs have been validated empirically, via comparison with real test data. Two tests were performed, cycling the battery with different CP rate. Every simulation was fed with the current profile and the investigated output is the thermal response of the system. Also, voltage profiles will be compared to assess the accuracy of the software. Regarding the temperature profile, the most accurate software was Simulink, whose relative error was 2.41% for Test 1 and 1.77% for Test 2, However, both Simcenter Amesim and Dymola performed well, with a slightly higher error, as reported in Table 7. On the other hand, Simscape relative error was 4.48% for Test 1 and 5.86% for Test 2, mainly since the battery block does not model entropic losses. Also regarding voltage profile, Simcenter Amesim, Simulink and Dymola performed well, with Simcenter Amesim being the best one. Simscape showed higher relative error, mainly because the ECM parameters can depend only on the temperature and SOC, but not on the current.

Table 7 - Relative error of every simulation tool with respect to experimental test data.

Relative error (%)	Amesim	Simscape	Simulink	Dymola
Test 1 (Temperature)	2.41	4.48	2.41	2.45
Test 2 (Temperature)	1.78	5.86	1.77	1.84
Test 1 (Voltage)	0.72	0.95	0.73	0.72
Test 2 (Voltage)	0.57	0.69	0.57	0.57

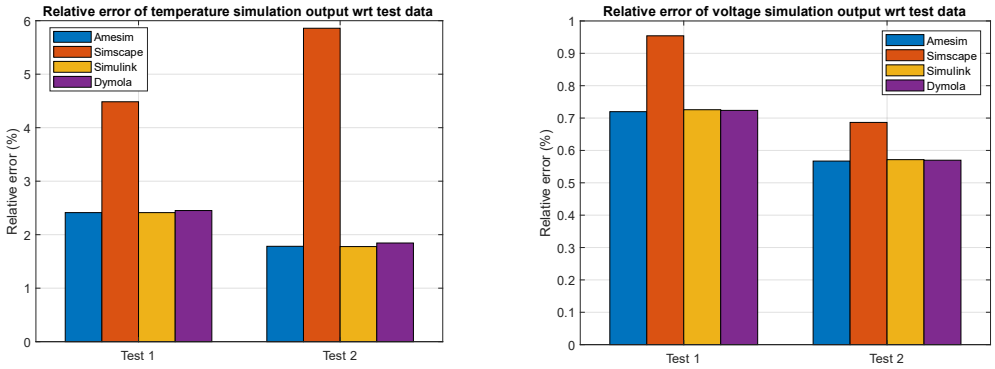

Figure 17 - Accuracy comparison between the four tools.

The reason why Simcenter Amesim resulted as the most accurate software is because the model optimization of the parameters was performed with this tool. The same parametrization on other software may produce small inaccuracies, due to the different solver or diverse block functionalities.

4.2 Ease of use

This section will present the results regarding ease of use and a comparison of the user experience during the modelling and simulation process. Each software will be briefly reviewed, aiming to describe the learning process and to enhance user awareness about what difficulties they might encounter while using the tool. This aspect cannot be quantified by numeric indicators, hence we agreed that the most effective way to give a sense of user-friendliness was to write a “user-experience” review for each tool, so that we could assess each tool in a qualitative way. These reviews are based on personal experience, acquired while modelling, implementing, and simulating the battery system on the four software.

4.2.1 Amesim

Thanks to the vast library of multiphysics components, the modelling of complex systems requires very low effort. In most cases, to implement the model, the user only needs to search for the right block in the library and drag-and-drop in the project environment, in sketch mode. These components can then be parametrized with specific values or tables, just by entering the number, the equation, or the path of the table file. Ease of setup is the key strength of Amesim, and it is probably the most user-friendly software among the ones analysed. The user is not required to have a physical knowledge of the system, or advanced modelling skills. Also, all the blocks are mimicking the most relevant physical phenomena, with a black-box approach. This gives less freedom to the user, but eases the learning process,

since it is not required to model and implement the different physical effects. This makes the tool particularly suited for beginners, who will reach a decent level of expertise in a short time. After the two steps of model implementation (drag-and-drop plus connect blocks) and parametrization (specify the general and block parameters), the model is ready to be simulated; this whole learning process requires little effort from the modeler, which can allocate more energy for the later stage of data processing and visualization. Also, this last process of post-processing and data visualization is very user-friendly, thanks to the Amesim intuitive interface. After the simulation has run, the user can click on a specific component and drag-and-drop a physical quantity to visualize its time-profile. Plots can then be overwritten on the same figure, compared in a new subplot, and exported in .csv format. All these quantities are automatically displayed with their physical unit, decreasing errors coming from misinterpretation of results. In a dedicated window, user can import different results and perform post processing, which can be very useful to extract additional information or to validate the model.

When starting a simulation, the tool opens a log window, which provides all the useful information about the model, including warnings and errors. The debug experience is smooth, since the log window provides detailed information about the possible errors, so that the user can easily identify the sources of error.

As was explained, the components are used with a block-box approach, meaning that the user cannot access the script. However, every block comes with a very detailed documentation, which gives insights on the block functionalities, properties, inputs, and outputs, and all the underlying equation. After double-clicking on the block, an icon with a question mark appears, which is linked to the block documentation. In our personal experience, every documentation provided the necessary information to well understand the block functionalities, easing the modelling process. While on one hand the documentation is comprehensive, on the other hand, for more high-level issues, the online resources, such as forums or tutorials, are not very exhaustive. This means that if the solution cannot be found in the documentation, it is unlikely to solve the issue by online research; the alternative is online support from the tool developer.

In our lifetime simulation model, it was required to interface the Simcenter Amesim model with a python script; this implementation required an intermediate coding ability with python, which increases the otherwise simple implementation of the degradation model. The following release of Simcenter Amesim, v2304, introduced an “ageing identification tool”, that incorporates a degradation model in the battery block, starting from test data. This means that degradation is calculated inside Simcenter Amesim, making external scripting non necessary anymore.

Overall, the learning and modelling process with Amesim is very smooth, without any blocker. This tool is easily accessible to most of users since there is no need for a deeper understanding of the system. As te system complexity increases, little effort is needed to implement the changes in the virtual model. However, for our lifetime simulation, a good knowledge of python scripting was required.

4.2.2 Simscape

As in the case of Amesim, Simscape library also contains a vast variety of multiphysics components. This means that the user is not required to have a deep understanding of the physical governing equation, since the most relevant ones are already incorporated in the physical components. Simscape has good ease of setup, since the user only needs to select the right components, connect them, and simulate the model. Because of that, the learning process does not require much effort, and this tool can fairly accommodate less experienced user. Nevertheless, Simscape shares the same environment with Simulink, whose advanced functionalities can allow more experienced users to implement more complex systems, with a higher level of detail. Indeed, Simscape blocks are easy to use but have low level of detail, which can represent a limitation for advanced users that want to create complex models.

The Simscape documentation is detailed, providing insights about functionalities and equations, including examples. In most of the cases, this resource is enough to model the system. Nevertheless, in the case that the documentation cannot provide help for a specific problem, the online community of Simscape users is quite large, and solutions can be found online.

The log window displays information about model warnings and errors, assisting the user to debug the simulation. However, the tool does not always provide suggestions on actions, making the debugging experience more challenging. Also, because data cannot be immediately displayed without proper blocks (such as Scope, ToWorkspace), finding errors requires more effort. While in Amesim, after every simulation, it is possible to immediately visualize every physical value of every block, in Simscape it is necessary to link a Scope block to the investigate value, run the simulation, and then visualize the profile. Especially for long runtime simulation, this aspect could slow down the modelling process.

In our case, most of the parameters and lookup tables were defined in the MATLAB environment. To do this, a basic knowledge of the MATLAB programming language is needed. Post-processing and data visualization can also be performed in MATLAB, with more advanced coding abilities. This close connection with the MATLAB environment can enable more advanced and complex analysis but adds an additional aspect that requires learning effort. However, the MATLAB language has been designed specifically for

engineers and is one of the most widespread languages in academia and industry. This means that most users will be able to interact with Simscape, knowing the basics of MATLAB.

For our lifetime simulation, a more extensive knowledge of MATLAB was necessary to run the simulation, process data and estimate degradation, which complicates the ease of implementation.

Overall, the Simscape implementation has the advantage of already built multiphysics component, which allows fast implementation of complex systems without a high modelling effort. However, postprocessing and visualization requires basic/intermediate knowledge of the MATLAB coding language. The debug experience is decent, since the tool identifies the error location, without suggesting possible actions.

4.2.3 Simulink

Unlike the other simulation tools, Simulink does not provide multiphysics components in its library, but it is designed to simulate signal flows and dynamic systems. It provides the flexibility to model and implement all sorts of phenomena if the modeler knows all the governing equations of the system. Because of this, building a Simulink model of a battery system requires the additional step of better understanding all the physical effects of the system, describe them with set of equations and represent these equations with Simulink blocks. As a result, the model setup requires more modelling efforts and investigation of the physical effects, which increases the probability of committing errors and the time required to build the system. It goes without saying that this modelling approach requires higher expertise from the user, who needs to investigate the system first, and implement the equations then.

Even though the modelling process takes more time, the implementation happens with basic algebraic and logical blocks, which are easy to use and connect. Simulink is a lower-level programming environment, while the other tools provide an already-built component environment, where the algebraic and logical level is hidden behind the blocks. This different approach gives more freedom to the modeller to customize every block, but it requires higher level of experience.

All Simulink blocks are well documented, enabling a smoother self-learning process. Simulink is by far the most used simulation tool, compared to the other three. This means that most of the warnings and errors encountered during the model implementation can be issued online, by leveraging the vast online MATLAB/Simulink community and the tutorials. A meaningful metric of the widespread of each tool can be the number of results found on Google. By entering “Simulink battery model” in Google, the search engine provides roughly 2.75 million results (research conducted on the 12th June 2023). If instead of “Simulink” we type “Simscape”, the number of results

drops to 200 thousand, while both “Amesim” and “Dymola” produce 50 thousand results each. These values give a sense of how popular each software is, with Simulink resulting as the most popular. This does not imply that other software are not well documented or that errors cannot be issued by an online research. However, in personal modelling experience, it was often the case that errors encountered in Simulink could be debugged by online research in a very easy way. This aspect increases the ease of use and the self-learning process, without the need to contact the vendor support.

Simulink, as well as Simscape, is highly interconnected with the MATLAB environment. Hence, also in this case, a basic knowledge of the MATLAB programming language is needed. Data visualization, post-processing and analysis can be better performed in MATLAB, so the user needs familiarity with MATLAB. This lowers the learning curve, since the user should understand how all these environments are interoperated and how results can be processed and visualized.

Also, the modification or the improvement of a system towards a more complex model requires more time and effort than the other tools. While for other multiphysics simulation tools the user can easily add component or integrate new library, with Simulink the new components must be modelled, meaning that the set of equation of the old system might not be valid anymore and the entire model needs to be rebuilt or heavily modified.

Overall, Simulink proved to be a very flexible and customizable tool to build the battery system. However, it was the only tool that required the user to write all the equations, understand the electrical, thermal, and fluid dynamics phenomena, and translate these equations with basic algebraic and logic blocks. The overall modelling phase requires more knowledge, time, and effort, but support can be easily found online thanks to the vast pool of Simulink user in academia and industry.

4.2.4 Dymola

The fourth and last simulation tool, Dymola, contains a large library of multiphysics components, coming both from the Open Modelica library and the Dymola library. Not only do these libraries contain blocks and components, but also a wide pool of already-built examples. They offer a good starting point for beginners to start building their model. Every example is described in the documentation, which eases the learning phase of the software. The software comes with an extensive user guide, which describes all the functionalities of Dymola. However, in most cases, reading the source code was a more straightforward strategy to understand block functionalities. Also the online resources from the Dymola users community was not suited to debug errors, which in this case were mainly issued via the source code access or the vendor support.

The great advantage of this software is that unexperienced users can implement models starting from the library examples, and more advanced users can easily access the source code and customize the functionalities. Blocks can be duplicated, customized, and added to the library. These libraries can then be shared with other users, so that they can drag-and-drop the new blocks and integrate the new functionalities in their models. For the three other software, sharing work between users mainly happens via the sharing of the model file, while in this case it is easier to import the library.

The source code can be modified easily, but this process requires knowledge of the OpenModelica language, which, unlike most of the other programming languages, is based on acausal modelling. Also known as equation-based approach, with this method users define the relationship between variables, rather than performing the right-to-left assignment of variables. Once defined, the equations are always satisfied in both directions, so there is no need of explicating the variables.

In the personal experience of model implementation with Dymola, mastering this modelling method required effort, but enables an easier description of complex physical systems. Understanding acausal modelling is a necessary step to add complexity of details to the basic models. Once mastered, every block can be customized in the “text” layer, while also having a graphical representation of the system in the “graph” layer.

Like in Amesim, after the model simulation it is possible to click on components and easily visualize the output profiles. Physical quantities can be displayed on plots, subplots or tables, and are always linked to a physical unit. The Check icon can be used to verify that the model is well built and, if it is not the case, the log window displays warning, errors, and possible actions to debug the model. The debugging experience is well supported by the software suggestion on finding and correcting the error.

Unlike the other three software, to run lifetime simulation there was no need for an external python/MATLAB script, since the ageing is calculated at cell level. While it is true that the degradation model needed to be written in the OpenModelica language, this let us avoid writing an additional script, easing the overall implementation of the simulation.

To sum up, Dymola offers great customizability of the blocks, but requires learning effort to understand the OpenModelica approach. However, beginners can start building their models based on the examples provided in the libraries, which are a good starting point to understand the basic functionalities of the software. The user can be overwhelmed by the modelling opportunities that the software offers, but can leverage this possibility with the vendor’s support, which can suggest best practices to properly implement the model.

4.3 Features

This section aims to present and compare the features of the four tools, regarding three main categories, namely modelling, simulation and interoperability. We will present the software so that the user can understand the uniqueness of each tool and choose the one that best suits his/her needs. Table 8 will summarize this paragraph for a rapid and direct comparison.

Table 8 - Feature comparison between the four software.

	Simcenter Amesim	Simscape	Simulink	Dymola
Modelling features				
Fluid property library	Wide, expandible	Moderate, not expandible	No	Wide, expandible
Drag-and-drop multiphysics components	Yes	Yes	No	Yes
Access the source code of the block	No	Yes	Yes	Yes
Modification of the source code of the block	No	Yes, but not easy	Yes, but not easy	Yes
Customizability	Low	Intermediate	High	High
CC-CV charge block	No	Yes	No	Yes
ECM parameter dependency	SOC, T, Current	SOC, T	Free model	SOC, T, Current
Reversible heat generation (entropic coefficient)	Yes	No	Can be modelled	Yes
Integration of ageing model in the battery block	Yes, but not custom (bf v2304)	Yes, but not custom	Can be modelled	Yes, customizable
Crate battery pack based on custom cell model	Yes	Yes	Yes	Yes
Electrochemical battery model	Yes	No	Can be modelled	Yes
Multi-dynamics Equivalent Circuit Model	Yes, no RC limit	Yes, up to 4 RC branches	Can be modelled	Yes, up to 3 RC
Simulation features				
Skip re-initialization in iterative simulation	No	Yes	Yes	No
Variety of solvers	2 variable, 3 fixed	1 variable, 8 fixed	10 variable, 8 fixed	15 variable, 4 fixed
Optimization	Yes	Yes	Yes	Yes
Sweep parameters	No	No	No	Yes
Simulation until steady state	No	No	No	Yes
Ease of results postprocessing	High	Intermediate	Intermediate	High
Ease of output visualization	High	Intermediate	Intermediate	High
Interoperability				
Accommodate beginners and advanced users	Mainly beginners	Both beginner and experts	Mainly advanced	Both beginners and experts
FMU integration	Yes	Yes	Yes	Yes
Run simulation from external script	Yes, Python	Yes, MATLAB	Yes, MATLAB	Yes, Python

Add blocks to the library	No	Yes, but not easy	Yes, but not easy	Yes

4.3.1 Modelling features

Every software provides different tools to describe a physical system. However, the level of detail, the fidelity and the functionalities can vary greatly between tools. For our specific case, we will focus more on the features that concern the battery system modelling experience, neglecting many other features that regards other physical systems. It is important to state that if a modelling feature is not available by default in a tool, it does not imply that this feature cannot be implemented with turnarounds or additional modelling effort. We want to showcase the modelling features that can be easily implemented without adding any extra work from the modeler.

Cells can be modelled with an Equivalent Circuit Model or with an electrochemical mode. Both Simcenter Amesim and Dymola offer the two modelling approaches, while Simscape Battery provides only ECM with up to 4 RC branches. Dymola, by default, provides ECM with maximum 3 RC branches, while in Simcenter Amesim there is no limit. Since Simulink offers the possibility to model every possible equation, there are no modelling limits, hence all the features can be implemented. However, the modelling process requires much more effort, since the user must know and implement all the equations involved. The ECM parameters depend on SOC, Current and Temperature in both Simcenter Amesim and Dymola, while only on SOC and Temperature for Simscape. The irreversible heat losses due to Ohmic effects are present in all the tools, while the reversible heat due to entropic effects is only modelled in Simcenter Amesim and Dymola. All the tools provide ageing models embedded with the cell component, with some differences. Simscape only models Calendar ageing, with two approaches, equation-based or table-based. Simcenter Amesim degradation model can be user-defines static/dynamic or semi-empirical dynamic, meaning that the user can input ageing model based on specific relationships between derivative of state of health and ageing stress factors. In Dymola there are already build ageing model for different battery chemistries, but it is also possible to build a custom ageing model with blocks and code, giving more freedom to the user.

A charging block with an already defined CC-CV charging protocol can be very useful when modelling battery systems. Dymola and Simscape Battery contains a CCCV block, while it is not included in Simcenter Amesim.

More advanced users may need to access the source code of a component to visualize and modify its functionalities. Simcenter Amesim does not provide this functionality by default, while it is possible to visualize the script in Simscape and Dymola. This last tool also offers the possibility to easily modify the script with the Modelica language, which enables users to customize

blocks. This can also be done with Simulink/Simscape, but it requires more effort.

Regarding the fluid library, every tool provides a list of already implemented liquid and gas properties, depending on temperature and pressure. This feature is commonly used when modelling cooling circuits of liquid flows. Simscape provides eight liquids/mixtures, where the user can specify the volume fraction. In Simcenter Amesim, more than 50 media are already modelled and stored in the native library, while for Dymola there are 50 modelled liquids, but many more can be modelled and imported. Also Simcenter Amesim gives the possibility to import user-defined fluids, while in Simscape the user can enter table-based properties.

As already mentioned, Simulink does not natively contain most of the features, but all of them can be modelled and implemented manually. Indeed, it is the only tool that does not provide multiphysics drag-and-drop components.

4.3.2 Simulation features

The simulation features concern all those aspects regarding the post-modelling phase, which allows users to run simulations in different ways, visualize/process the results and perform system analysis and optimization. In terms of solvers, every tool gives the ability to simulate the model with a fixed or a variable timestep. Every software offers a wide variety of solvers, with Dymola containing the higher number of variable-step solver (around 15), while Simulink has around ten, Simcenter Amesim two (regular or cautious) and Simscape only one. Regarding fixed-step solver, Amesim contains three options: Euler, Adam-Bashforth and Runge-Kutta. Dymola has Euler and three types of Runge-Kutta and Simscape/Simulink have eight different fixed-step solvers.

In every simulation tool, when running a simulation, the software first initializes and compiles the model, then runs the simulation. When running a simulation from an external script, the same process happens. For lifetime simulation it is often the case that simulations are run multiple times, meaning that at every iteration the same model is re-initialized and re-built. This process can highly increase the overall simulation runtime. Simulink and Simscape have the "*Fast Restart*" feature, which skips the initialization and compilation step if the simulation is run again without major changes. This feature significantly decreased the runtime, since the model was initialized only for the first day and quickly run for the other days, just with a different state of health. It was not the case for Amesim, for which the lifetime simulation took longer because at every step the model was rebuilt, which took around additional 3 seconds.

All the simulation software contain an optimization tool that can help users to design parameters to fit experimental data. They also give the possibility to stack simulation for different parameters, to conduce sensitivity analysis based on swept parameters. With Dymola it is also possible to run a simulation until steady state, without the need to specify the end time.

In terms of output visualization and post-processing, both Amesim and Dymola have a user-friendly interface, where it is possible to plot and process the results immediately. This feature is not implemented in Simulink and Simscape, which however leverage the close interaction with the MATLAB environment to manage the data.

4.3.3 Interoperability

System simulations are usually based on multiple system components, which can be developed by other users. In this case, the simulation software should be able to integrate in the same model multiple functionalities developed on different platforms. Functional Mockup Interfaces and Functional Mockup Units (FMU) are the most common way of sharing these functionalities. All the software are able to import FMU in their models. Also, FMU can be created and exported starting from each of the investigated simulation tool. Interoperability can also mean the ability of the software to accommodate both beginner and advanced users. In this sense, Dymola can be easily operated by beginners thanks to the graphical view and allows advanced users to customize models by coding in Modelica. Amesim is even simpler to use but lacks the feature of deep and flexible customization of the blocks. Simulink and Simscape interfaces can accommodate beginner users, also providing some advanced features to build more complex and customizable models.

If more than one user is working on the same platform on different parts of the system, with Dymola they can share their jobs by creating a new custom library and share it with others. For instance, the thermal modeler can develop a cell thermal model that can then be shared with the simulator, who can easily select the newly created thermal model into the system. With the other three tools, the job can be shared as a model file, and the user can copy and paste the blocks between different projects. The difference is that the newly developed model is not included in a shared library, so it is needed to open the specific project to import the work. However, Simcenter Amesim can be interoperated with “Simcenter System Simulation Client for Git”, a shared platform where multiple users can upload and access other simulation project, to develop in parallel new models.

In terms of running simulation from an external script, Amesim and Dymola can be operated from python script, even though this process is not

computationally efficient. Simscape and Simulink simulations can be run from a MATLAB script in a more efficient way, since all the data are managed in the same workspace, without the need of continuously reading and writing data on files.

4.4 Licensing

All the tools require a license purchase to be operated. In all cases, licenses can be perpetual (one-time buy), or rented annually. For this comparison, we considered floating licenses with one year of activation period. MATLAB/Simulink resulted as the cheapest one [54], costing around 860 EUR for MATLAB and 1300 EUR for Simulink. In the Simscape case, two additional licenses are required: “Simscape” and “Simscape Battery”, making necessary the purchase of four distinct products. The overall annual price is around 4000 EUR. Also Simcenter Amesim requires a rental license, but in this case there is no need to purchase add-ons. All the functionalities needed for this study were included in the standard license of Simcenter Amesim, which costs around double the annual total Simscape price. The most expensive software is Dymola, which requires the Dymola standard configuration, plus the two libraries: Battery and Cooling. Its overall price is around double than Simcenter Amesim. However, it must be said that the different vendor may provide exclusive discounts, based on number of licenses purchased, type of license required (Standard/Home/Academic). The prices reported in this thesis are a rough estimate, to give a sense of magnitude to the reader.

5 Conclusions

This thesis conducted a comparative study of four simulation software, benchmarking different aspects associated with the modelling and simulation experience. The main investigated indicators regard the software performance (speed, accuracy, CPU usage), ease of use, features, and licensing. The four tools have been compared based on the same multiphysics model of a battery system, liquid cooled. The model has been described in detail, especially the interactions between the electric, thermal, and ageing phenomena. Ease of use was evaluated based on personal experience of modelling, implementation, and simulation of the battery system. Thanks to its simple interface, the very comprehensive library of multiphysics components, clear documentation, and the ease of setup and postprocessing, Simcenter Amesim performed as one of the best in the ease-of-use category. Also Dymola proved to be user-friendly, especially thanks to the many model examples already available, based on which the user can master the software and build more complex systems. However, it has a less steep learning curve because the user might need to learn the Modelica language to customize the block functionalities. Thanks to the multiphysics library and the battery builder functionality, Simscape supports the modelling experience, making it smoother. However, visualization and post processing require basic MATLAB language knowledge. In terms of customizability, both Simulink and Dymola are optimal, for different reasons. With Simulink, the user can model every sort of effect, provided that the underlying equations of the system are known. Dymola gives the possibility both to select multiphysics component, with all the equations already implemented, but also to modify these blocks by accessing the source code in Modelica language. Regarding performance, Dymola is the software that uses the least computational resources and run simulation in the least amount of time. For the 24-hour simulation, it is followed by Simulink and Simcenter Amesim, which are around three times slower. Simscape was the slowest, probably due to implementation inefficiencies that require more investigation. For lifetime simulation Dymola was mor than 5 times faster than Simulink and more than 80 times faster than Simcenter Amesim. This last tool can was slower for lifetime simulation because of the implementation with a python script. In a future release, Simcenter Amesim will be able to evaluate degradation without the need of an external script. From the licensing perspective, Simulink has the lower price, followed by Simscape, which is an add-on to the basic Simulink package.

Overall, every software showed benefits under a particular aspect, hence we believe that the software choice mainly depends on how the user prioritizes the different aspects. Indeed, this thesis aimed to showcase advantages and disadvantages of every tool, based on which the user could make a more

conscious selection. Table 9 summarizes which are the ideal usage of the software for different user requirements.

Table 9 - Ideal use cases for every software

Software	Use case
Simcenter Amesim	Given the high user-friendliness, with a high level of physical detail, this tool is optimal for users that need to build comprehensive models with a smooth workflow. The ease of setup and the inability to customize blocks makes it more appropriate for early design phase and preliminary studies, rather than complex customized system simulations. Also, it well suits parameter fitting and system calibration.
Simulink	Simulink guarantees good performance and interoperability with the MATLAB environment. It best suits advanced users with a good physical knowledge of the system and programming skills to model and run dynamic systems. Given the high flexibility, the model can be detailed as desired. It is also the most economic tool among the ones selected.
Simscape	It is ideal for users that want to gain the benefits coming from the MATLAB/Simulink environment, but value more ease of use than performance. The graphical representation of multiphysics components can please a wider range of users that are not required to study the physical phenomena. It can also accommodate users that needs to understand the general behaviour of a complex system, since it is faster to build and modify complex models, albeit with a lower level of detail.
Dymola	It is the ideal tool for users that needs to run or parallelize computational heavy simulation, with the lowest runtime possible. Model complexity and level of detail can be increased as desired for deep investigation, provided that the user is able to describe and implement physical phenomena with the Modelica language. Ease of setup, starting from components or examples, makes suitable for preliminary investigation, while high flexibility enables also very detailed analysis.

These are suggestion based on the comparative analysis, but it must be acknowledged that no tools showed limitation in the modelling and simulation experience. Some of them may need extra modelling effort or specific turnarounds, but they do not prevent users to build and simulate battery systems. Therefore, the final choice depends on the user preferences.

6 References

[1] I. International Energy Agency, “Renewables 2022,” 2022. [Online]. Available: www.iea.org

[2] S. Attia, L. Beltrán, A. De Herde, and J. Hensen, “‘Architect friendly’: a comparison of ten different building performance simulation tools,” in *Eleventh International IBPSA Conference*, 2009.

[3] J. D. Gardner, A. Vijayaraghavan, and D. A. Dornfeld, “Comparative Study of Finite Element Simulation Software,” *UC Berkeley*, 2005. [Online]. Available: https://escholarship.org/uc/item/8cw4n2tf

[4] J. L. Ross, “A Comparative Study of Simulation Software for Modeling Stability Operations,” 2012.

[5] J. Sousa, “Energy Simulation Software for Buildings: Review and Comparison,” 2010.

[6] M. Torres-Torriti, T. Arredondo, and P. Castillo-Pizarro, “Survey and comparative study of free simulation software for mobile robots,” *Robotica*, vol. 34, no. 4, pp. 791–822, Apr. 2016, doi: 10.1017/S0263574714001866.

[7] M. Magni, F. Ochs, S. de Vries, A. Maccarini, and F. Sigg, “Detailed cross comparison of building energy simulation tools results using a reference office building as a case study,” *Energy Build*, vol. 250, Nov. 2021, doi: 10.1016/j.enbuild.2021.111260.

[8] J. L. Bernal-Agustín and R. Dufo-López, “Simulation and optimization of stand-alone hybrid renewable energy systems,” *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8. pp. 2111–2118, Oct. 2009. doi: 10.1016/j.rser.2009.01.010.

[9] H. He, R. Xiong, H. Guo, and S. Li, “Comparison study on the battery models used for the energy management of batteries in electric vehicles,” in *Energy Conversion and Management*, Dec. 2012, pp. 113–121. doi: 10.1016/j.enconman.2012.04.014.

[10] B. Y. Liaw, G. Nagasubramanian, R. G. Jungst, and D. H. Doughty, “Modeling of lithium ion cells - A simple equivalent-circuit model approach,” in *Solid State Ionics*, Nov. 2004, pp. 835–839. doi: 10.1016/j.ssi.2004.09.049.

[11] M. K. Tran, A. Dacosta, A. Mevawalla, S. Panchal, and M. Fowler, “Comparative study of equivalent circuit models performance in four common lithium-ion batteries: LFP, NMC, LMO, NCA,” *Batteries*, vol. 7, no. 3, Sep. 2021, doi: 10.3390/batteries7030051.

[12] H. He, R. Xiong, X. Zhang, F. Sun, and J. Fan, “State-of-charge estimation of the lithium-ion battery using an adaptive extended Kalman filter based on an improved Thevenin model,” *IEEE Trans Veh Technol*, vol. 60, no. 4, pp. 1461–1469, May 2011, doi: 10.1109/TVT.2011.2132812.

[13] Z. Salameh, M. Casacca, and W. Lynch, “A Mathematical Model for Lead-Acid Batteries,” *IEEE Transactions on Energy Conversion*, 1992.

[14] X. Hu, S. Li, and H. Peng, “A comparative study of equivalent circuit models for Li-ion batteries,” *J Power Sources*, vol. 198, pp. 359–367, 2012, doi: https://doi.org/10.1016/j.jpowsour.2011.10.013.

[15] L. Zhang, H. Peng, Z. Ning, Z. Mu, and C. Sun, “Comparative research on RC equivalent circuit models for lithium-ion batteries of electric vehicles,” *Applied Sciences (Switzerland)*, vol. 7, no. 10, Sep. 2017, doi: 10.3390/app7101002.

[16] X. Lin *et al.*, “A lumped-parameter electro-thermal model for cylindrical batteries,” *J Power Sources*, vol. 257, pp. 1–11, 2014, doi: https://doi.org/10.1016/j.jpowsour.2014.01.097.

[17] W. B. Gu and C. Y. Wang, “Thermal-Electrochemical Modeling of Battery Systems,” *J Electrochem Soc*, vol. 147, no. 8, p. 2910, Aug. 2000, doi: 10.1149/1.1393625.

[18] X. Hu, S. Lin, S. Stanton, and W. Lian, “A Foster Network Thermal Model for HEV/EV Battery Modeling,” *IEEE Trans Ind Appl*, vol. 47, no. 4, pp. 1692–1699, 2011, doi: 10.1109/TIA.2011.2155012.

[19] J. Wang, C. Pan, H. Xu, and X. Xu, “Thermo-Electric Model of the Power Battery and Module Based on AMESim,” in *2017 5th International Conference on Mechanical, Automotive and Materials Engineering, CMAME 2017*, Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 223–228. doi: 10.1109/CMAME.2017.8540167.

[20] M. Ecker *et al.*, “Calendar and cycle life study of Li(NiMnCo)O2-based 18650 lithium-ion batteries,” *J Power Sources*, vol. 248, pp. 839–851, 2014, doi: 10.1016/j.jpowsour.2013.09.143.

[21] J. Vetter *et al.*, “Ageing mechanisms in lithium-ion batteries,” *J Power Sources*, vol. 147, no. 1–2, pp. 269–281, Sep. 2005, doi: 10.1016/j.jpowsour.2005.01.006.

[22] M. Petit, E. Prada, and V. Sauvant-Moynot, “Development of an empirical aging model for Li-ion batteries and application to assess the impact of Vehicle-to-Grid strategies on battery lifetime,” *Appl Energy*, vol. 172, pp. 398–407, Jun. 2016, doi: 10.1016/j.apenergy.2016.03.119.

[23] A. S. Mussa *et al.*, “Fast-charging effects on ageing for energy-optimized automotive LiNi1/3Mn1/3Co1/3O2/graphite prismatic lithium-ion cells,” *J Power Sources*, vol. 422, pp. 175–184, May 2019, doi: 10.1016/j.jpowsour.2019.02.095.

[24] S. Atalay, M. Sheikh, A. Mariani, Y. Merla, E. Bower, and W. D. Widanage, “Theory of battery ageing in a lithium-ion battery: Capacity fade, nonlinear ageing and lifetime prediction,” *J Power Sources*, vol. 478, Dec. 2020, doi: 10.1016/j.jpowsour.2020.229026.

[25] E. Teliz, C. F. Zinola, and V. Díaz, “Identification and quantification of ageing mechanisms in Li-ion batteries by Electrochemical impedance spectroscopy.,” *Electrochim Acta*, vol. 426, Sep. 2022, doi: 10.1016/j.electacta.2022.140801.

[26] W. Huang *et al.*, “Evolution of the Solid-Electrolyte Interphase on Carbonaceous Anodes Visualized by Atomic-Resolution Cryogenic Electron Microscopy,” *Nano Lett*, vol. 19, no. 8, pp. 5140–5148, Aug. 2019, doi: 10.1021/acs.nanolett.9b01515.

[27] P. Iurilli, C. Brivio, and V. Wood, “On the use of electrochemical impedance spectroscopy to characterize and model the aging phenomena of lithium-ion batteries: a critical review,” *Journal of Power Sources*, vol. 505. Elsevier B.V., Sep. 01, 2021. doi: 10.1016/j.jpowsour.2021.229860.

[28] S. L. Hahn, M. Storch, R. Swaminathan, B. Obry, J. Bandlow, and K. P. Birke, “Quantitative validation of calendar aging models for lithium-ion batteries,” *J Power Sources*, vol. 400, pp. 402–414, Oct. 2018, doi: 10.1016/j.jpowsour.2018.08.019.

[29] F. Zhou and C. Bao, “Analysis of the lithium-ion battery capacity degradation behavior with a comprehensive mathematical model,” *J Power Sources*, vol. 515, Dec. 2021, doi: 10.1016/j.jpowsour.2021.230630.

[30] I. Bloom, B. G. Potter, C. S. Johnson, K. L. Gering, and J. P. Christophersen, “Effect of cathode composition on impedance rise in high-power lithium-ion cells: Long-term aging results,” *J Power Sources*, vol. 155, no. 2, pp. 415–419, Apr. 2006, doi: 10.1016/j.jpowsour.2005.05.008.

[31] R. Spotnitz, “Simulation of capacity fade in lithium-ion batteries.”

[32] J. Wang *et al.*, “Cycle-life model for graphite-LiFePO4 cells,” *J Power Sources*, vol. 196, no. 8, pp. 3942–3948, Apr. 2011, doi: 10.1016/j.jpowsour.2010.11.134.

[33] M. Ecker *et al.*, “Development of a lifetime prediction model for lithium-ion batteries based on extended accelerated aging test data,” *J Power Sources*, vol. 215, pp. 248–257, Oct. 2012, doi: 10.1016/j.jpowsour.2012.05.012.

[34] J. Belt, V. Utgikar, and I. Bloom, “Calendar and PHEV cycle life aging of high-energy, lithium-ion cells containing blended spinel and layered-oxide cathodes,” *J Power Sources*, vol. 196, no. 23, pp. 10213–10221, Dec. 2011, doi: 10.1016/j.jpowsour.2011.08.067.

[35] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, “A review on lithium-ion battery ageing mechanisms and estimations for automotive applications,” *Journal of Power Sources*, vol. 241. pp. 680–689, 2013. doi: 10.1016/j.jpowsour.2013.05.040.

[36] M. Naumann, M. Schimpe, P. Keil, H. C. Hesse, and A. Jossen, “Analysis and modeling of calendar aging of a commercial LiFePO4/graphite cell,” *J Energy Storage*, vol. 17, pp. 153–169, Jun. 2018, doi: 10.1016/j.est.2018.01.019.

[37] I. Bloom *et al.*, “An accelerated calendar and cycle life study of Li-ion cells.”

[38] C. Guenther, B. Schott, W. Hennings, P. Waldowski, and M. A. Danzer, “Model-based investigation of electric vehicle battery aging by means of vehicle-to-grid scenario simulations,” *J Power Sources*, vol. 239, pp. 604–610, 2013, doi: 10.1016/j.jpowsour.2013.02.041.

[39] X. Gao, H. Zhou, S. Li, S. L. Chang, Y. Lai, and Z. Zhang, “The fast-charging properties of micro lithium-ion batteries for smart devices,” *J Colloid Interface Sci*, vol. 615, pp. 141–150, Jun. 2022, doi: 10.1016/j.jcis.2022.01.105.

[40] “Simcenter Amesim,” *Siemens Digital Industries Software*. https://plm.sw.siemens.com/en-US/simcenter/systems-simulation/amesim/ (accessed Jul. 03, 2023).

[41] MathWorks, “MATLAB”, Accessed: Jul. 03, 2023. [Online]. Available: https://it.mathworks.com/products/matlab.html

[42] MathWorks, “Simulink”, Accessed: Jul. 03, 2023. [Online]. Available: https://it.mathworks.com/products/simulink.html

[43] MathWorks, “Simscape Battery.” https://it.mathworks.com/products/simscape-battery.html (accessed Jul. 03, 2023).

[44] Dassault Systems, “Dymola.” https://www.3ds.com/products-services/catia/products/dymola/ (accessed Jul. 03, 2023).

[45] M. A. A. Khan and A. K. Sheikh, “A comparative study of simulation software for modelling metal casting processes,” *International Journal of Simulation Modelling*, vol. 17, no. 2, pp. 197–209, Jun. 2018, doi: 10.2507/IJSIMM17(2)402.

[46] E. Weingärtner, H. Vom Lehn, and K. Wehrle, “A performance comparison of recent network simulators,” in *IEEE International Conference on Communications*, 2009. doi: 10.1109/ICC.2009.5198657.

[47] D. Jafrancesco *et al.*, “Optical simulation of a central receiver system: Comparison of different software tools,” *Renewable and Sustainable Energy Reviews*, vol. 94. Elsevier Ltd, pp. 792–803, Oct. 01, 2018. doi: 10.1016/j.rser.2018.06.028.

[48] P. H. Camargos, F. E. N. Campos, and B. I. L. Fuly, “Motor Starting Direct-on-line: Performance Analysis in ATP and MATLAB/SIMULINK Environments,” in *Proceedings - 2021 IEEE 3rd Global Power, Energy and Communication Conference, GPECOM 2021*, Institute of Electrical and Electronics Engineers Inc., Oct. 2021, pp. 85–90. doi: 10.1109/GPECOM52585.2021.9587781.

[49] B. Jmai, A. Rajhi, and A. Gharsallah, “Software comparative study for RF power coupler,” in *Mediterranean Microwave Symposium*, IEEE Computer Society, Apr. 2015. doi: 10.1109/MMS.2014.7088968.

[50] S. Chakrabarti, H. N. Saha, Čhulālongkoṇmahāwitthayālai. Sasin Graduate Institute of Business Administration, Institute of Engineering &

Management, University of Engineering & Management, and Institute of Electrical and Electronics Engineers, *2017 8th Industrial Automation and Electromechanical Engineering Conference (IEMECON) : 16-18 August, 2017, Sasin Graduate Institute of Business Administration of Chulalongkorn University, Bangkok, Thailand*.

[51] R. Judkoff, D. Wortman, B. O'Doherty, and J. Burch, "A methodology for validating building energy analysis simulations, NRELTechnical Rep. 550–42059 1–192.," 2008.

[52] R. Judkoff, D. Wortman, and J. Burch, "Empirical Validation of Building Analysis Simulation Programs: A Status Report ," in *Solar Energy Research Institute*, Golden, Colorado, 1982.

[53] "Intel® Core™ i7-1065G7 Processor," 2023. https://ark.intel.com/content/www/us/en/ark/products/196597/intel-core-i71065g7-processor-8m-cache-up-to-3-90-ghz.html (accessed Jun. 25, 2023).

[54] MathWorks, "MATLAB Price list." https://it.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=comm (accessed Jul. 04, 2023).

[55] T. Bruen and J. Marco, "Modelling and experimental evaluation of parallel connected lithium ion cells for an electric vehicle battery system," *J Power Sources*, vol. 310, pp. 91–101, Apr. 2016, doi: 10.1016/j.jpowsour.2016.01.001.

[56] W. Storr, "Tau - the time constant of an RC Circuit," *Basic Electronics Tutorials*. https://www.electronics-tutorials.ws/rc/time-constant.html (accessed Jun. 29, 2023).

APPENDIX A

Parametric Maps

In this section is meant to provide deeper insights on the tests and methodologies adopted to obtain the parametric maps and the functions that describe electrical and thermodynamical quantities. All those values will be dynamically calculated in the simulation using empirical correlation or multidimensional lookup tables. These tables have been set up to linearly interpolate within the operating range, and to select the nearest value as an extrapolation method. The following paragraphs will describe how to determine OCV function, the ECM parameters, and the thermal.

State of Charge – Open Circuit Voltage map

One of the most common approaches to evaluate the correlation between SOC and OCV is by cycling the cell with a low current rate [10], [11], [16], [55]. The battery is initially fully charged and discharged to determine its capacity. Then a low current is used, so that the voltage drops across the internal impedances are negligible and the OCV can be approximated to the terminal voltage. This constant current typically ranges between C/50 and C/20 and the measurement is performed in a thermally controlled environment. The result is a non-linear curve that covers the entire SOC range 0%-100%, where the OCV reaches its minimum and maximum values.

ECM parameters (HPPC Test)

While the OCV is assumed to be solely dependent on the SOC, the other ECM component must be characterized with respect to SOC, Temperature and current. A commonly used approach [9], [11], [12], [14]–[16], [55] to determine the values of the resistances and capacitances is through Hybrid Pulse Power Characterization (HPPC). The typical experimental setup is as follows:

- The system is kept at a fixed temperature in a thermal chamber.
- The battery is charged to 100% SOC with a CC-CV protocol.
- The battery is kept in this condition until thermal and electrochemical equilibrium is reached.
- An electric impulse discharges the battery for a short period of time at constant power. Current and voltage are monitored.
- The current is cut for a relaxation period to reach equilibrium at a lower SOC.
- A series of pulse-relaxation periods are repeated until the battery is fully discharged.

- The experiment is repeated with different power intensities.
- The experiment is repeated at different temperature.

At the end of the experiments, we will have a series of voltage relaxation profiles, each of which corresponds to a SOC, a current intensity and a temperature. Based on the shape of these voltage profiles it is possible to determine the electric parameters through curve fitting.

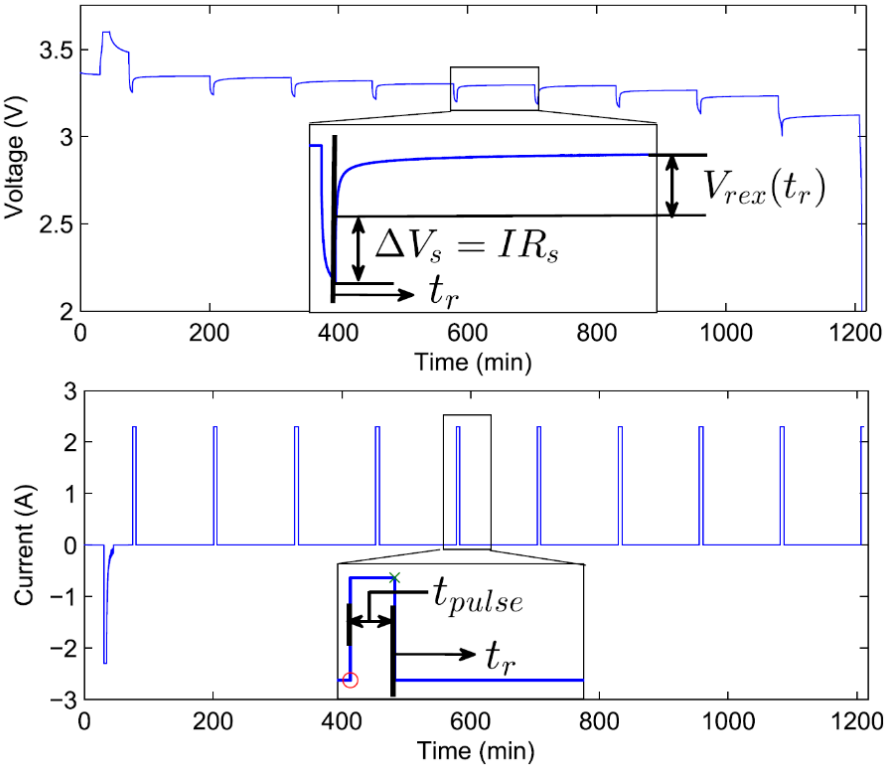

Figure 18 - HPPC Test to detect Electric Circuit Model parameters [16]

After the pulse period t_p, the cell voltage reaches an equilibrium value during the relaxation period. The voltage experiences an initial instant rise, ΔV_s, caused by the series Ohmic resistance R_s. Its value can be obtained with the following equation.

$$R_s = \frac{\Delta V_s}{I} \tag{20}$$

The values of the dynamic RC branches are determined with the relaxation voltage profile after the initial jump. Their values are determined to fit the following equation with experimental data, so that the square error is the minimum.

$$V_{rex}(t_r) = \sum_{i=1}^{n} IR_i \left(1 - e^{-\frac{t_p}{R_i C_i}}\right)\left(1 - e^{-\frac{t_r}{R_i C_i}}\right) \tag{21}$$

Where n is the number of RC branches, in our case 2. R_i and C_i are the resistance and capacitance of the $i-th$ RC branch, t_p is the pulse duration and t_r is the time variable of the relaxation period. With one specific pulse we can obtain R_s, R_1, R_2, C_1 and C_2 that correspond to the imposed temperature, state of charge and current intensity. After repeating the experiment at different condition, the output will be a three-dimensional parametric map for each electric component, depending on SOC, Current and Temperature.

Thermal parameters

The thermal model of the system must be defined in terms of thermal capacity of the cells, thermal resistances with the cooling system and with the ambient, entropic coefficient. To the determine those parameters, cooling tests based on Newton's law of cooling have been adopted. Newton's law of cooling states that the rate of heat flow of an object is directly proportional to the difference between the current temperature of the body and the surrounding temperature.

$$\frac{dQ}{dt} = -k\Delta T \tag{22}$$

Where $\frac{dQ}{dt}$ is the heat flow from the body to the surroundings in Watt, ΔT is the temperature difference between the body and the surroundings in Kelvin and k is a heat transfer coefficient in W/K. In the case of a 0-D constant properties body, after differentiating with respect to time the equation $Q = mc\Delta T$, we obtain the following equation:

$$\frac{dQ}{dt} = -mc\frac{dT}{dt} \tag{23}$$

Where m is the mass of the body in kg and c is the specific heat coefficient in J/kgK. By combining eq. (22) and eq. (23), we obtain a first-order ordinary differential equation, whose solution is the following.

$$\Delta T = (T_0 - T_s)e^{-t/\tau} \tag{24}$$

Where ΔT is the temperature difference between the body and the surroundings, T_0 is the initial temperature of the body, T_s is the temperature of the surroundings, t is the time variable and τ is a characteristic time of the cooling process. In this case τ corresponds to

$$\tau = \frac{mc}{k} \tag{25}$$

By using the analogy between a thermal cooling and an RC electric circuit, it can be proved that in both cases:

$$\tau = RC \tag{26}$$

In the thermal circuit, R is the thermal resistance in K/W between the mass and the surroundings, expressed by the following equation.

$$R = \frac{\Delta T}{\dot{Q}} \tag{27}$$

C represents the thermal capacity of the mass, i.e. the product mc. By measuring the cooling curve, τ corresponds to the time at which the ΔT equals 36.8% of its initial value.

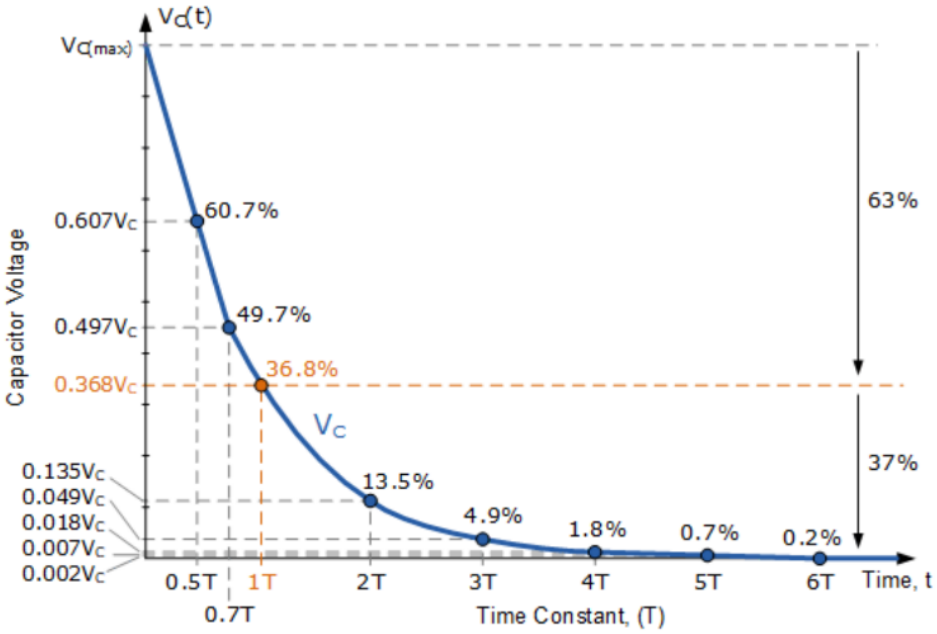

Figure 19 - Rate of Voltage decay to identify time constant T [56]

The experimental setup consists in several cooling curve tests, initially run with a constant coolant flow rate. The battery is heated with a sequence of constant current pulses to reach a steady state temperature. Once the upper temperature is reached, the current is cut off and the temperature profile is detected. With this method it is possible to identify τ. The global thermal resistance can be obtained with eq. (27), where ΔT is the temperature difference between the battery and the coolant (as the average between inlet and outlet). The heat flow $\dot{Q}$ can be calculated with the following equation.

$$\dot{Q} = c_p \dot{m} \left(T_{cool,out} - T_{cool,in}\right) \tag{28}$$

Where c_p is the specific heat of the coolant in J/kgK, $\dot{m}$ is the mass flow rate of the coolant in kg/s and $\left(T_{cool,out} - T_{cool,in}\right)$ is the temperature

difference between the coolant outlet and the inlet in K. Once τ and C are calculated at a given coolant flow rate, it is possible to determine the total thermal resistance with eq. (26).

The test is repeated with different coolant flow rates and the battery thermal capacitance is obtained as the average of all the experimental C.

The last experiment is a cooling curve test with the cooling system switched off, so that it is possible to identify the thermal resistance between the battery and the ambient. The temperature cooling profile provides information about τ_{amb}, so that R_{amb} can be derived with eq. (26).

Lastly, R_{cool} can be calculated for each coolant flow rate by using the electrical analogy of two resistors in parallel.

$$\frac{1}{R_{tot}} = \frac{1}{R_{amb}} + \frac{1}{R_{cool}} \tag{29}$$

By curve fitting, it is possible to identify the function $R_{cool}(\dot{m})$, that can be implemented in the simulation to solve the thermal problem.

APPENDIX B

Model implementation

This section will explain the implementation process of the multiphysics model of the battery system on the four software. We will give insights not only on the procedural steps to build the model, but also on the different approaches to simulation and how to manage parameters and values in lifetime simulation.

Amesim

Amesim library contains multiphysics components that enable user to build the battery system simulation in a very straightforward way. In "sketch" mode, we can import components from the library and connect them to develop the electric circuit and the cooling channels.

The battery pack is modelled via the existing block "*Battery pack*", which contains all the information regarding electric and thermal phenomena. In the "submodel" editor, we can select various electric models of the battery. As described in section 2.3.1, our electric model is based on an equivalent circuit model with two RC branches, therefore the corresponding submodel in Amesim is "*Advanced ECM of battery pack*" .In the block parameter, we can specify the location of the parametric maps, containing the values of the ECM resistances and capacitances, the OCV(SOC) curve and the entropic coefficient. These parametric maps are multidimensional lookup tables, saved in the .data format. To generate these files, the "Table editor" tool of Amesim enables users to import, edit and generate tables in the .data format, so that they can be interpreted by Amesim. The ECM Parameters are dependent on SOC, Temperature and Current, while OCV and Entropic coefficient are dependent only on SOC. In the battery pack parameter tab, it is possible to specify the number of cells in series and in parallel inside the pack. Also, the component "battery pack" contains information about heat loss for both reversible and irreversible phenomena. The icon includes a thermal port which can be connected to a thermal circuit, which models the cooling effect towards the ambient and the liquid cooling circuit.

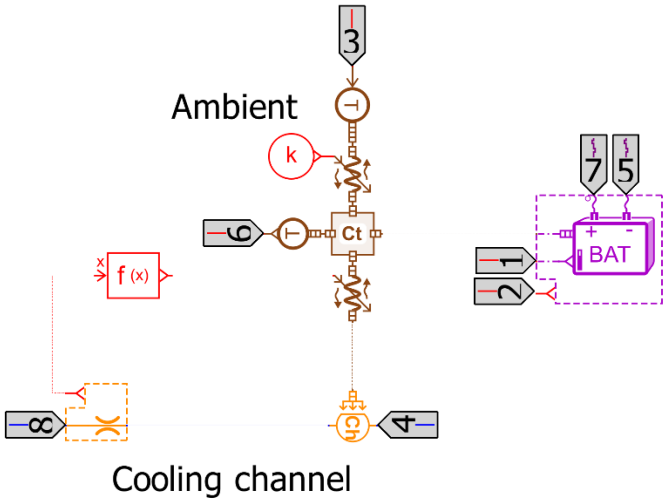

Figure 20 - Amesim Electric and thermal model

The battery is connected to a "*thermal capacitance*" block, which mimics the bulk thermal capacitance of the battery pack. This block sees the three thermal flows: the heat losses from the battery pack, the dissipation towards the ambient and the cooling effect of the heat exchanger. The ambient loss is modelled through a thermal resistance between the thermal capacitance and a fixed temperature source, representing the ambient. The liquid cooling heat flow happens through a variable thermal resistance between the battery and a fixed volume chamber, which sees the liquid coolant entering at a fixed temperature and volume flowrate. The fluid properties of the coolant are selected from the existing Amesim fluid library, which contains information about various cooling liquid properties for different temperatures and pressures. Figure 20 shows a snippet of the battery pack connected to the thermal model. The electric circuit is controlled by a current generator, which receives as input the charge/discharge profile of that specific case study.

For the accuracy test, the current input comes from a timeseries table which contains the current profile measured during the experiment. For lifetime simulations the current input is modelled with the "*statechart environment*". This Amesim tool is useful to model different states of the system, like charge, discharge, or rest. The 24-hours profiles described in section 2.2 can be implemented in a state chart, as shown in figure 21.

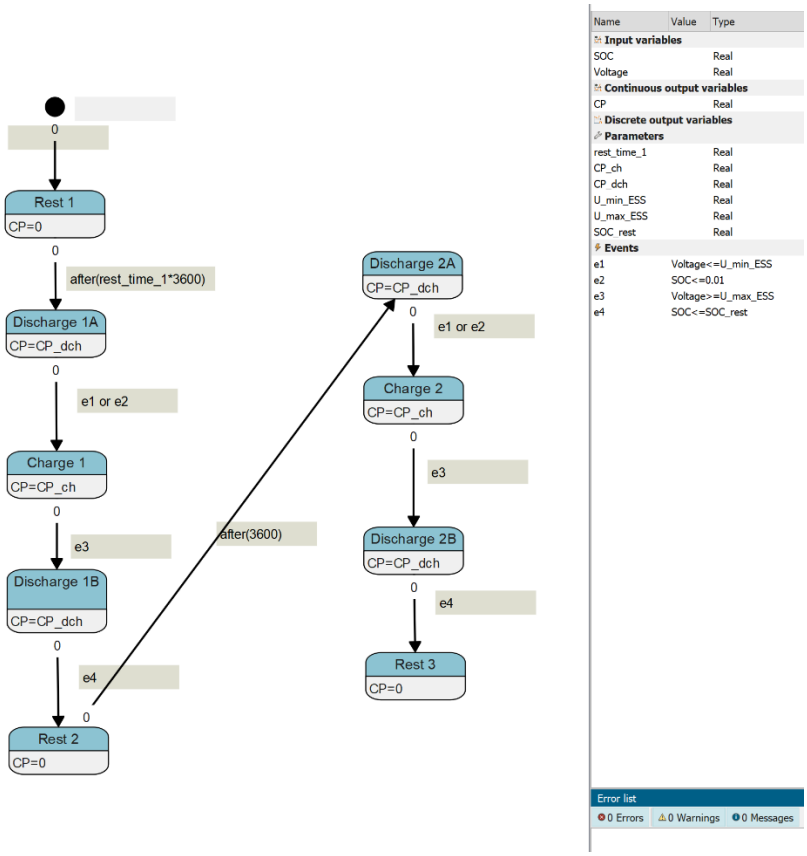

Figure 21 - Statechart environment in Simcenter Amesim

This scheme is composed of different states, whose output is an electric power value, and of transitions between subsequent states, which are triggered by events. These events can be related to physical quantities (e.g. when the terminal voltage overcomes the maximum cutoff voltage) or to time durations (e.g. resting time of one hour between two cycles). These logical events dictate the transition between states, and it is very convenient in the case the instant of transition is unknown a priori, but depends on other dynamic variables (Temperature, SOC, Voltage, …).

To extract the ageing stress factors in table 3, we used integrator blocks to compute the time average of physical quantities, and logic block to store maximum and minimum SOC. These simulation outputs are then fed in the ageing model to estimate capacity fade and resistance increase linked to the last 24 hours of operation; these updated capacity and resistance values become the input of the model to simulate the next 24 hours of operation. This process runs iteratively for the investigated lifetime of the battery pack. This loop is executed via an external Python script, whose duty is to run the lifetime simulation by calling the Amesim simulation inside a “for” loop. Every iteration of the loop executes one day of operation at a given state of health. Here follow the steps to conduce the lifetime simulation:

- At day zero, both Capacity SOH and Resistance SOH are 100%.
- "For" loop with N iteration, where N is the number of days in the lifetime simulation. At every iteration:
    - The SOH values are assigned to the Amesim model with the python command "`ameputgpar`", from the "`amesim`" python library.
    - Once the SOH is set, the python command "`amerunsingle`" runs the 24h Amesim simulation.
    - The stress factors are extracted with two python commands: "`amegetvarnamefromui`" and "`ameloadvarst`".
    - These stress factors are fed in the ageing model, which calculates the capacity fade and resistance increase related to the last 24h simulation.
    - New State of Health values are calculated and saved in an array. A new iteration starts.
- After N iteration, we can visualize the state of health profiles and measure the total runtime of the lifetime simulation.

Running the lifetime simulation from the python script does not require Amesim to be launched, since all the commands related to writing input, running the simulation, and extracting output are performed at text level.

For the empirical validation of the model with respect to test data, there is no need to automate the simulation. The simulation is run directly inside the Amesim environment, where it is also possible to visualize the required outputs, i.e., the voltage and the temperature profiles. It is sufficient to click on the component and select the parameter to extract, so that it can be saved in .csv format.

Simscape

The same battery system model has been implemented on the MATLAB/Simulink environment, taking advantage of the add-on libraries "Simscape" and "Simscape Battery". The Simscape tool gives the possibility to create multiphysics models in the Simulink environment by drag-and-dropping physical components from a library. Similarly to Amesim, these components contain the main physical equation that model various physical phenomena from different domains, such as electric, thermal or fluid. In Simscape, batteries can be modelled though the "*Battery (Table-Bases)*" block since it allows to define the ECM parameters though dynamic lookup tables. This component models an electrochemical cell through an ECM with up to four RC branches. All the ECM parameters are set through a 2D lookup table, depending on SOC and Temperature. There is no current dependency

of the ECM parameters in this Simscape block. The OCV is also given by a 2D lookup table, depending on SOC and Temperature. It is possible to model the thermal aspect of the battery, providing the thermal mass of the cell in J/K. The battery pack can be modelled via the creation of a new library block with the Simscape battery builder commands, in the MATLAB environment.

- In MATLAB, import the simscape.battery.builder library.
- Create a "cell" element via the command `Cell`. It is possible to specify the model options, such as geometry, thermal model, electric model.
- Once the cell is created, generate a parallel assembly with the command `ParallelAssembly`.
- Based on the newly created parallel assembly, generate the module using the command `Module`.
- Use the command `buildBattery` to convert the module into a new library component, which can then be drag-and-dropped in the Simulink/Simscape environment.

The thermal model of the battery in Simscape is only considering irreversible losses due to ohmic effects in the internal resistances. It does not model the reversible losses due to entropy change; however, this type of losses can be computed with Simulink blocks and be added in the thermal circuit with the block "*Controlled Heat Flow Rate Source*". For a fair comparison, we decided not to include reversible loss in the Simscape model, since the original battery block does not consider this effect. Unlike Amesim, the thermal capacitance is already inside the battery pack block, hence it can be directly connected to the ambient (*Temperature source*) through a Thermal Resistance. The dissipated heat through the liquid coolant is modelled via *a Variable Thermal Resistance*, connected to a *Constant Volume Chamber (TL)*, which sees as inlet a constant volume flow rate at a fixed temperature.

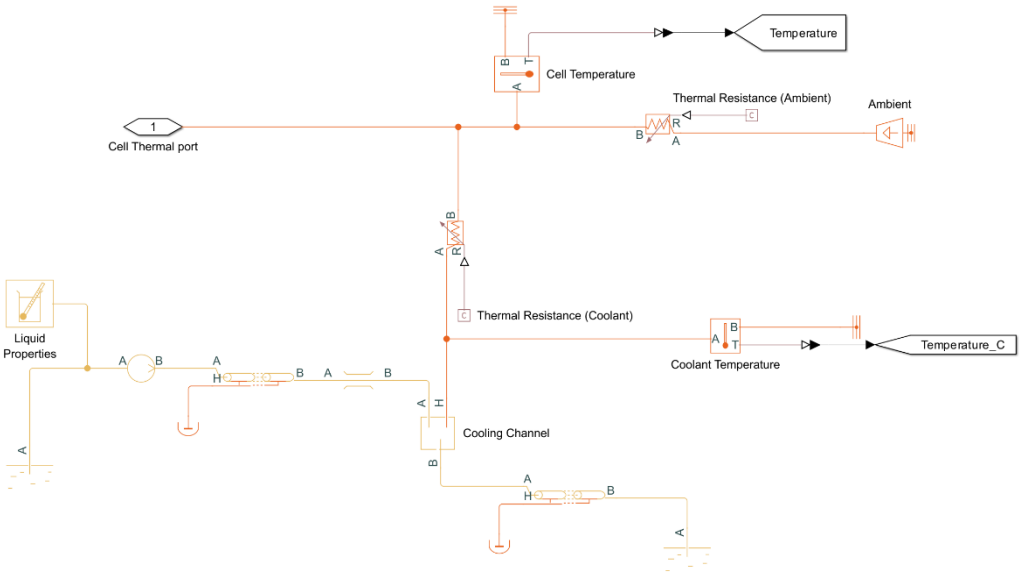

Figure 22 - Thermal model in Simscape

Liquid properties are given by the block Thermal Liquid Properties, where it is possible to select the fluid from a list and specify the mixture percentage.

The electric circuit is controlled by the block Controlled Current Source, which takes as an input the current profile of the simulation. Similarly to the implementation on Amesim, the current profile is given as a timeseries in the empirical validation simulation, while is given by a state chart in lifetime simulations. The logic behind the state chart is the very same as the one used in Amesim. In the Simulink environment, the state chart is part of an add-on library called "*Stateflow*", which contains the block Chart.

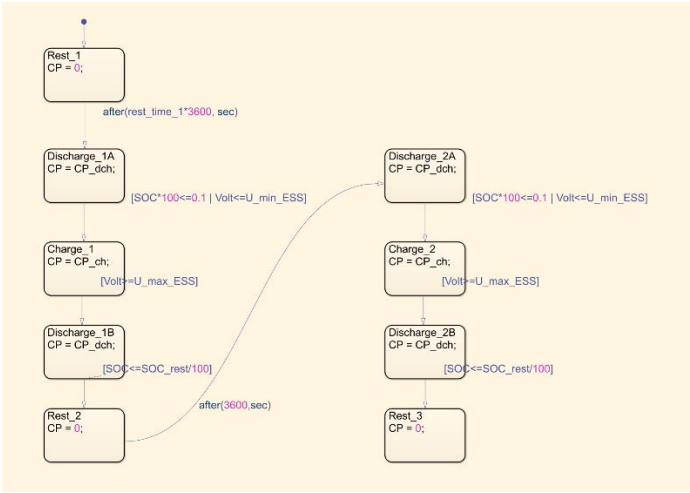

Figure 23 - Stateflow in Simulink/Simscape

Also in this case, the Simulink/Simscape simulation is run from an external script for lifetime simulation. However, there is no need of using python, since the Simulink/Simscape environment is conveniently interconnected with the MATLAB environment, hence this language will be used. The script-simulation interaction logic is the same as the one used for the Python-Amesim interaction. In this case, the MATLAB-Simulink/Simscape interaction is preferable since these environments share the same workspace, where variables and parameters are stored. This means that all the parameters and lookup tables can be imported via MATLAB script, so that they are automatically interfaced with Simulink/Simscape simulation. The lifetime simulation is run with a MATLAB script with the following logic:

- Import all parameters and lookup tables in MATLAB, from external .txt files, with the command `readmatrix`.
- Set the initial SOH for capacity fade and resistance increase to 100%.
- "`For`" loop with N iteration, where N is the number of days in the lifetime simulation. At every iteration:
    - The SOH related to Capacity and Resistance is assigned to the simulation as a parameter.
    - The Simulink/Simscape 24h simulation is run via the MATLAB command `sim`, specifying as options the Fast Restart mode to "`on`" and the simulation mode to "`Accelerator`".
    - The output of the command sim contains all the information about the simulation, including the stress factors.
    - These stress factors are fed to the ageing model and the capacity fade and resistance increase linked to the last 24h simulation are obtained.
    - State of health is updated based on the newly computed capacity fade and resistance increase. New iteration begins.
- After N iteration, we can visualize the state of health profiles and measure the total runtime of the lifetime simulation.

While in this case the simulations are run from MATLAB command, in the empirical validation the simulation is run directly from the Simulink/Simscape environment. The investigated output variables can be exported to the MATLAB workspace with the block ToWorkspace. In this way it is possible to visualize the voltage and temperature profiles and compute the relative error with respect to test data.

Simulink

Simulink offers a signal-flow environment that can be used to implement dynamic models based on differential equations. Unlike all the other simulation tools, Simulink does not provide Multiphysics components, such as batteries, pipes, voltage sensors and others. This means that the user needs to model the relevant governing equation that represents the system.

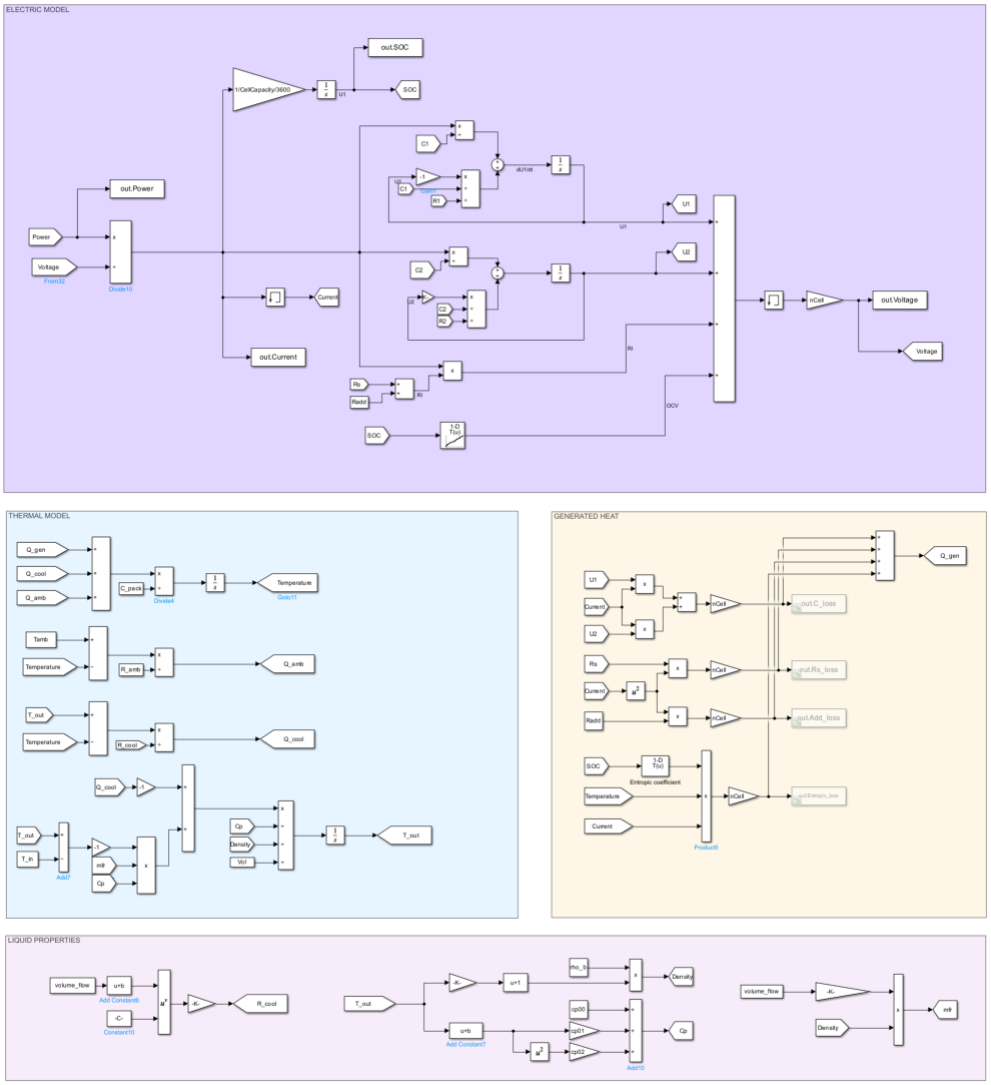

Figure 24 - Simulink multiphysics model

These equations were explained in section 2.3, an they describe electric and thermal effect and can have been implemented in the Simulink model mainly with logic and algebraic blocks, such as Sum, Product or Integrator. The state chart environment is the same as the one described in the previous section, Simulink/Simscape.

Also in this case, it is convenient to run the lifetime simulation from a MATLAB script, since the Simulink-MATLAB environments share the same workspace. The logic behind the script is the very same as the one used for the Simulink/Simscape lifetime simulation and described in paragraph 3.4.2. The only difference is that the command sim inside the for loop, launches the Simulink simulation instead of the Simulink/Simscape one.

Dymola

The same battery system Multiphysics model has been implemented in Dymola. This tool contains many libraries that include not only blocks and components, but also examples of already built systems. Examples have been used as a starting point for building the cell model and the cooling circuit.

The cell model used in our case is imported from the *Battery* library of Dymola, which contains different cell geometries. Every cell model is composed of three sub models, namely the *thermal*, *electric* and *ageing* model, as shown in figure 25.

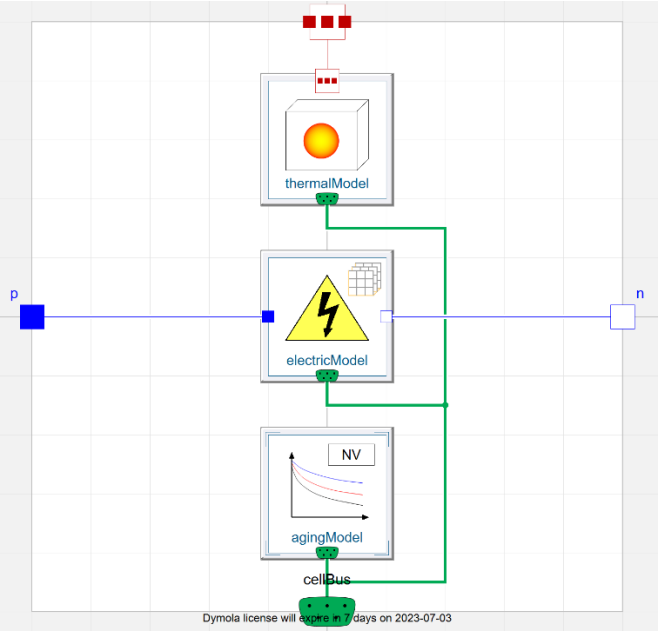

Figure 25 - Cell model in Dymola

Each of these three models can be selected among a list of already existing thermal, electric and ageing models, or can be customized based on what the user needs. In our case, we used an ECM electric model with two RC branches, whose parameters are based on three-dimensional lookup tables. This model is shown in Figure 26.

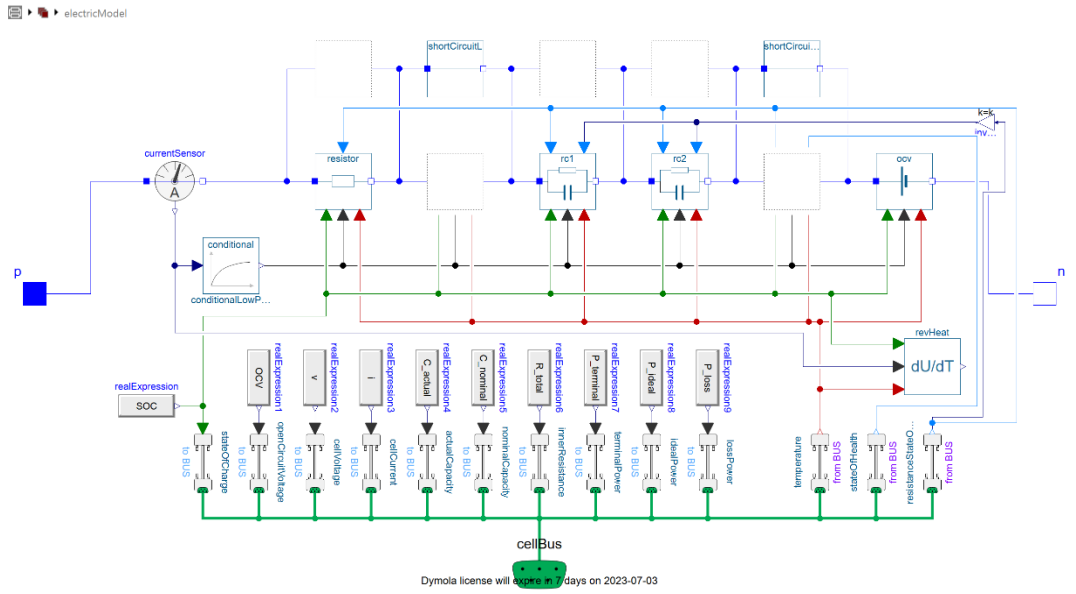

Figure 26 - Electric model in Dymola

All the parametric maps are saved in one file in the `.sdf` format, which is a convenient way to manage multi-dimensional tables. This file is managed with the "*SDF Editor*" tool, included in the Dymola installation bundle. The thermal model has been modified, so that the cell is represented by one thermal capacitance. The ageing model has been created from scratch, implementing the lookup tables and equation described in section 2.3.3. This ageing model has been developed leveraging both the graphical representation and the text layer behind it. All the signal routings, parameter sensing, and lookup tables are implemented via drag-and-drop graphical component from the Dymola library, while all the equations of the ageing model have been implemented as OpenModelica scripts in the text layer. Once the Cell model is completed, it can be scaled up to a pack by using the Battery Pack block in the Battery library.

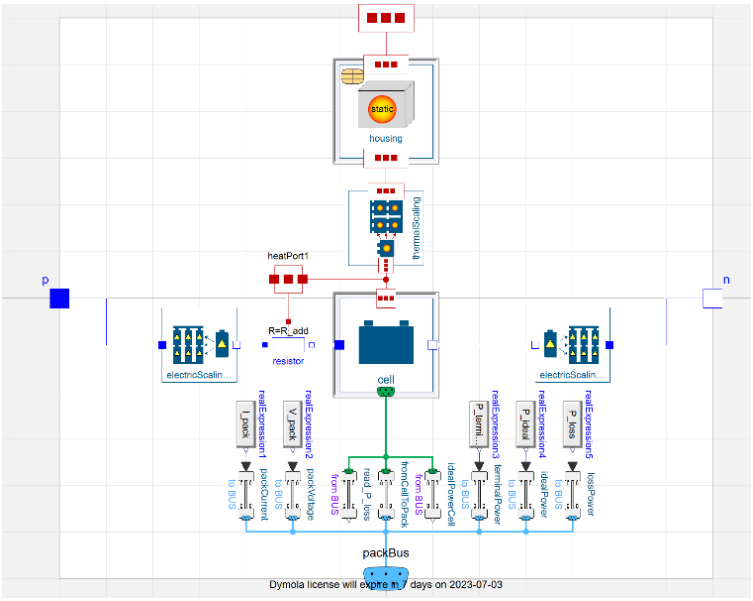

Figure 27 - Battery pack model in Dymola

This component lets users specify number of series and parallel cells in one pack, and the thermal distribution inside the pack, which, in this case, is homogeneous.

The battery pack dissipates the heat through a thermal resistance towards the ambient and through a thermal resistance towards the cooling channels. These channels are imported from the Cooling library of Dymola, where it is also possible to specify the cooling liquid out of a list of parametrized fluids.

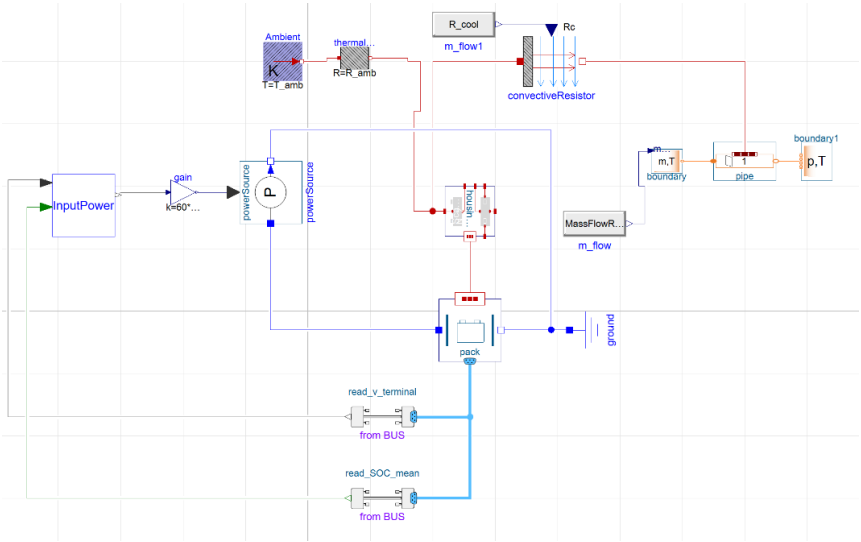

Figure 28 - Electric and cooling circuit on Dymola

The electric circuit is controlled by a current generator, whose input comes from a state chart (for lifetime simulation) or from a timeseries (for empirical validation). The state chart environment is composed of events and transition, similarly to the other simulation software. The main difference in this implementation is that the 24h profile is looped to restart once every 24 hours, while before the end time of the simulation was set to be exactly 24 hours. In this case, it is possible to perform lifetime simulation within the software environment, without the need for an external script that automates the simulation runs. The end time of the lifetime simulation in Dymola is set equal to the investigated battery lifespan (10 years), and degradation is automatically calculated at cell level while the battery is operating.

For the empirical validation, the inputs are imported as *TimeTable* into the battery system model. These input regards both the current profile and the coolant volumetric flow rate.

The simulation results can be visualized in plots or tables in the "*simulation*" tab. These values can then be exported in different data format, so that they can be compared with the other simulation software output.

END OF DOCUMENT