

LIONSIMBA - a Matlab framework based on a finite volume model suitable for Li-ion battery design, simulation, and control

Marcello Torchio*, Lalo Magni*, R. Bhushan Gopaluni[†],
Richard D. Braatz[‡], and Davide M. Raimondo*

January 25, 2016

Abstract

Consumer electronics, wearable and personal health devices, power networks, microgrids, and hybrid electric vehicles (HEVs) are some of the many applications of lithium-ion batteries. Their optimal design and management are important for safe and profitable operations. The use of accurate mathematical models can help in achieving the best performance. This article provides a detailed description of a finite volume method (FVM) for a pseudo-two-dimensional (P2D) Li-ion battery model suitable for the development of model-based advanced battery management systems. **The objectives of this work are to provide: (i) a detailed description of the model formulation, (ii) a compiled and parametrizable Matlab framework for battery design, simulation, and control of Li-ion cells or battery packs, (iii) a validation of the proposed numerical implementation with respect to the commercial software COMSOL MultiPhysics and the Newman's DUALFOIL code, and (iv) some demonstrative simulations involving thermal dynamics, a hybrid charge-discharge cycle emulating the throttle of a HEV, a model predictive control of state of charge, and a battery pack simulation.**

1 Introduction

The increasing demand for portable devices (e.g., smartphones) and hybrid electric vehicles (HEVs) calls for the design and management of storage devices of high power density and reduced size and weight. During the many decades of research, different chemistries of batteries have been developed, such as Nickel Cadmium (NiCd), Nickel Metal Hydride (NiMH), Lead Acid and Lithium ion (Li-ion) and Lithium ion Polymer (Li-Poly) (e.g., see [1, 2, 3, 4]). Among electrochemical accumulators, Li-ion batteries provide one of the best tradeoff in terms of power density, low weight, cell voltage, and low self-discharge [5]. Mathematical models can support the design of new batteries as well as the development of new advanced battery management systems (ABMS) [6, 7, 8]. According to the literature, mathematical models for Li-ion battery dynamics fall within two main categories: Equivalent Circuit Models (ECMs) and Electrochemical Models (EMs). ECMs use only electrical components to model the dynamic behaviour of the battery. ECMs include (i) the R_{int} model where only a resistance and a voltage source are used to model the battery, (ii) the RC model (introduced by the company SAFT [9]) where capacitor dynamics have been added to the R_{int} model [10], and (iii) the Thevenin model, which is an extension of the RC model (e.g., see [11, 12] and references therein). In contrast, EMs explicitly represent the chemical processes that take place in the battery. While ECMs have the advantage of simplicity, EMs are more accurate due to their ability to describe

*M. Torchio, L. Magni, and D.M. Raimondo are with the University of Pavia, 27100 Pavia, Italy marcello.torchio01@ateneopv.it, {lalo.magni,davide.raimondo}@unipv.it

[†]R.B. Gopaluni is with the University of British Columbia, 2360 East Mall, Vancouver, BC, Canada V6T 1Z3 bhushan.gopaluni@ubc.ca

[‡]R.D. Braatz is with the Massachusetts Institute of Technology, Cambridge, MA, United States of America 02142 braatz@mit.edu

detailed physical phenomena [13]. The most widely used EM in the literature is the porous electrode theory-based pseudo-two-dimensional (P2D) model [14], which is described by a set of tightly coupled and highly nonlinear partial differential-algebraic equations (PDAEs). In order to exploit the model for simulation and design purposes, the set of PDAEs are reformulated as a set of ordinary differential-algebraic equations (DAEs). The model reformulation is very challenging to carry out in a way that is simultaneously computationally efficient and numerically stable for a wide range of battery parameters and operating conditions. To the authors’ best knowledge, no publication is available in the literature that provides a detailed step-by-step description of the numerical implementation of the P2D model or a freely available Matlab framework suitable for simulation, design, and development of ABMS for Li-ion batteries. In this article, starting from the P2D model, a computationally efficient and numerically stable finite volume DAE formulation is described in detail in order to facilitate implementation by the reader, while also addressing potential pitfalls and relative loopholes. Boundary conditions used to enforce physical meaningfulness of the system are thoroughly discussed and their numerical implementation clarified. **Particular attention is addressed to the handling of interface boundary conditions across different sections of the battery. Due to possible discontinuities between adjacent sections, a mishandling of such conditions may lead to unmeaningful physical solutions. Due to its intrinsic properties, the finite volume method has been chosen to easily deal with these particular interface conditions. Finally, based on the proposed finite volume discretization,** we provide the Li-ION SIMulation BAttery Toolbox (LIONSIMBA), a fully customizable Matlab-compiled framework suitable for simulating the dynamic behavior of Li-ion batteries. The framework is freely downloadable at <http://sisdin.unipv.it/labsisdin/lionsimba.php>. This article describes the features of our software. The user can implement his/her own custom-defined control algorithm to test different ABMS strategies, **simulate cell behavior, optimize manufacturing parameters or test battery packs composed of series-connected cells.** The package also allows the ready implementation of algorithms to estimate indices such as the State of Charge (SOC) and the State of Health (SOH). The SOC is an important property of batteries that quantifies the amount of remaining charge (e.g., [15]) and can be used to prevent damage, ensure safety, and minimize charging time [16]. The SOH index measures the ability of the battery to store and deliver electrical energy; similar to the SOC, estimation-based approaches are used to predict the value of the SOH (e.g., see [17, 18, 19]). The SOH tracks the long-term changes in a battery and its knowledge can help ABMS to anticipate problems through online fault diagnosis while providing charging profiles to slow down the battery aging. The package comes with the experimental parameters of the battery reported in [20]. An initialization file allows changes in battery and simulator parameters. The simulator works under Matlab using IDA [21] to solve the set of resulting DAEs with a good trade-off between accuracy and computational time.

In the following a description of the battery model and its numerical implementation is given. The proposed framework is then validated with respect to the results obtained using the commercial software COMSOL MultiPhysics [22] and the Newman’s Fortran code named DUALFOIL [23]. Finally, to demonstrate the effectiveness of the proposed software, simulations of thermal dynamics, model predictive control of state of charge, hybrid charge-discharge cycles and battery pack of series-connected cells are provided. The toolbox is equipped with all the Matlab source files able to reproduce the simulations presented in this work.

2 Battery Model

The P2D model consists of coupled nonlinear PDAEs for the conservation of mass and charge in the three sections of the battery – cathode, separator, and anode – denoted respectively by the indices p , s , and n . The positive and negative current collectors are denoted by a and z . The index $i \in \mathcal{S}$ is used to refer to a particular section of the battery, where $\mathcal{S} := \{a, p, s, n, z\}$. All model equations are reported in Tables 1 and 2. Variables $c_e(x, t)$, $c_s^{\text{avg}}(x, t)$, and $c_s^*(x, t) \in \mathbb{R}^+$ denote the electrolyte concentration, the average concentration in the solid particles, and the surface concentration in the solid particles of Li-ions respectively, where time $t \in \mathbb{R}^+$ and $x \in \mathbb{R}$ is the spatial direction along which

the ions are transported. Assuming spherical coordinates, the diffusion inside the solid particles with radius R_p is described using the Fick's law

$$\frac{\partial c_s(r, t)}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left[r^2 D_p^s \frac{\partial c_s(r, t)}{\partial r} \right], \quad (1)$$

with boundary conditions

$$\left. \frac{\partial c_s(r, t)}{\partial r} \right|_{r=0} = 0, \quad \left. \frac{\partial c_s(r, t)}{\partial r} \right|_{r=R_p} = -\frac{j(x, t)}{D_{\text{eff},i}^s},$$

where r represents the radial direction along which the ions intercalate within the particles. This model introduces a pseudo-second dimension (r). In order to reduce complexity and computational burden, the authors in [24] and [25] proposed different efficient reformulations for the solid-phase diffusion equation. As discussed by the authors in [26], according to the particular application, different model reformulations can be employed while maintaining good accuracy. For low to medium C rates, the diffusion length method [27] or the two-terms polynomial approximation method can be used without lack of precision. By increasing the C rates higher-order polynomial approximations or Pseudo Steady State (PSS) [28] approximation can be employed. For more details, refer to [26] and the references therein.

In the following, the model is presented assuming that the diffusion inside solid particles is described using the two-term polynomial approximation. Concentration profiles inside the particle are assumed to be a parabola in r and eq. (1) is approximated by means of average and surface solid particles concentration

$$\begin{aligned} \frac{\partial c_s^{\text{avg}}(x, t)}{\partial t} &= -3 \frac{j(x, t)}{R_p}, \\ c_s^*(x, t) - c_s^{\text{avg}}(x, t) &= -\frac{R_p}{D_p^s} \frac{j(x, t)}{5}. \end{aligned}$$

This reformulation leads to a one-dimensional problem in x by removing the pseudo-second dimension r . Despite the reduced computational burden, such approximation could lead to a decrease of the prediction accuracy for high rates, short time responses or pulse currents [26]. For these applications, higher-order polynomials or Fick's law of diffusion are recommended, as later discussed in Section 4.2.

The electrolyte and solid potential are represented by $\Phi_e(x, t)$ and $\Phi_s(x, t) \in \mathbb{R}$, while $T(x, t)$ and $j(x, t)$ represent the temperature and the ionic flux. Note that the ionic flux is present only in the positive and negative electrode, and not in the separator. The open circuit voltage (OCV) is denoted by U while the entropic variation of the OCV is denoted by $\frac{\partial U}{\partial T}$. The cathode, anode, and separator are composed of different materials; for a given section i , different electrolyte diffusion coefficients D_i , solid-phase diffusion coefficients D_i^s , electrolyte conductivities κ_i , porosities ϵ_i , thermal capacities $C_{p,i}$, thermal conductivities λ_i , densities ρ_i , solid-phase conductivities σ_i , particle surface area to volume a_i , maximum solid phase concentration $c_{s,i}^{\text{max}}$, overpotentials η_i , and particle radius $R_{p,i}$ can be defined. The terms R and F are the universal gas constant and the Faraday constant, respectively, with t_+ representing the transference number. The applied current density is $I_{\text{app}}(t)$, and T_{ref} denotes the environment temperature. In order to take into account for the properties of the different materials used in the battery, effective diffusion and conductivity coefficients are evaluated according to the Bruggeman's theory, with "eff" suffixes representing effective values of such coefficients. The thickness of the overall battery is L , where $L = \sum_k l_k$ and l_k , $k \in \{a, p, s, n, z\}$ represents the length of each battery section. Due to physical constraints, it is necessary to impose (i) zero-flux boundary conditions for the c_e diffusion equation at the two ends of the battery, (ii) Newton's cooling law for the dissipation of heat in the system, and (iii) null-flux conditions for Φ_s at the interface between electrodes and the separator as well as the enforcement of Ohm's law at the end of the electrodes. Given that only potential differences are measurable, without loss of generality, Φ_e can be set to zero at the end of the

anode. Similarly, on the cathode side, zero-flux conditions are imposed. Within the battery, interface conditions are imposed across the different materials. In order to get a more detailed description of the conductivity and diffusion phenomena inside the electrolyte, all the related coefficients are determined as a function of c_e and T , as discussed in [29].

Excessive heat generation may lead to performance degradation or, in extreme cases, thermal runaway of the cell [30, 31]. In order to address these possible safety issues, thermal dynamics are coupled to the set of conservation equations describing the system. Three different source terms are present, which are the ohmic, reversible, and reaction generation rates Q_{ohm} , Q_{rev} , and Q_{rxn} , respectively [32]. The ohmic generation rate takes into account heat generated as a consequence of the motion of Li-ions in the solid/liquid phase. The reaction generation rate accounts for heat generated due to ionic flux and over-potentials, and the reversible generation rate takes into account the heat rise due to the entropy change in the electrodes' structure. The next section uses the notation $\hat{x}_0 = l_a$, $\hat{x}_p = l_a + l_p$, $\hat{x}_s = l_a + l_p + l_s$, and $\hat{x}_n = l_a + l_p + l_s + l_n$. For a clearer comprehension, **bold** is used in tables for coefficients whose dependence on other variables is made explicit in other equations. The nomenclature of the variables is reported in Table 9. **Model equations have been taken from the work in [14], where for convenience the electrolyte potential is related to the ionic flux $j(x, t)$ rather than to the applied current density [33, 34]. The thermal model is taken from [32], while all the parameters describing the particular chemistry have been taken from [20].**

3 Numerical Implementation

Most numerical methods for model-based estimation and control algorithms require the model to be formulated in terms of AEs or DAEs rather than PDAEs. Different numerical methods can be used to achieve this objective. The reformulation process from PDAEs to AEs or DAEs is carried out by discretizing the domains of the independent variables (e.g., the time domain t and the n -dimensional spatial domain $x \in \mathbb{R}^n$). The discretization can involve both time and space, to produce AEs, or only space, to produce DAEs. An example of discretization in time and space is given by the FTCS (Forward-Time Central-Space) approach [35]. Other techniques, like the method of lines (MOL) [36], discretize only the space domain and leave the time as a continuous variable. When this latter approach is used, finite volume, finite difference, or finite element methods can be employed to obtain the set of DAEs. **Alternatively other approaches can be used. For instance the authors in [20] provides a numerical implementation where an orthogonal collocation approach together with an efficient coordinate transformation is proposed to solve the set of resulting DAEs. In this paper, in order to exploit the properties of variable-step solvers, MOL is used to reformulate the original set of PDAEs.** In particular, the finite volume method (FVM) is employed. Due to its ability to conserve properties with high accuracy (within numerical roundoff), the FVM has been used in literature to discretize models in a wide range of applications, such as heat transfer problems [37], flow and transport in porous media [38], or more general applications for hyperbolic problems as discussed in [39]. **In particular, the FVM together with the *harmonic mean* (HM) have been used to deal with possible discontinuities across different sections of the cell. To the best of authors knowledge, no work exists in literature which addresses in detail the numerical issues related to the implementation of the Li-ion cell model and, in particular, the handling of boundary conditions that ensure physical meaningfulness of the obtained solutions. For this reason, in the following, all the numerical details are addressed.**

3.1 Finite Volume formulation

Consider a general diffusion-convection equation defined on a domain in \mathbb{R}^N of the form

$$\frac{\partial \phi}{\partial t} + \nabla(\eta\phi) = \nabla(\Gamma\nabla\phi) + s \quad (2)$$

where ϕ is the unknown variable, η is the velocity, Γ is a diffusion coefficient and s a source term. Both the unknown ϕ and the source term s depend on time t and space $x \in \mathbb{R}^N$. For convenience

define $f(\phi) := \eta\phi - \Gamma\nabla\phi$. Integrating (2) over a spatial domain $\Omega \subset \mathbb{R}^N$ and applying the divergence theorem produces the integral form of the conservation law:

$$\int_{\Omega} \frac{\partial\phi}{\partial t} dV + \oint_{d\Omega} (f(\phi) \cdot n) dS = \int_{\Omega} s dV \quad (3)$$

where $d\Omega$ is the boundary of the domain Ω , n is the outward pointing unit normal on the boundary of the domain, and dV and dS represent the infinitesimal volume of Ω and the infinitesimal surface of the boundary $d\Omega$ respectively. Alternatively, this integral equation could be written directly as an exact conservation equation over any prescribed spatial domain.

According to the FVM, the spatial domain Ω is divided into a set of disjoint control volumes (CVs) Ω_k centered in $x_k \in \mathbb{R}^N$, such that $\Omega = \cup_k \Omega_k$ and $\Omega_i \cap \Omega_j = \emptyset, \forall i \neq j$. The average value of the unknown variables for each CV is

$$\bar{\phi}_k(t) \approx \frac{1}{G_k} \int_{\Omega_k} \phi(x, t) dV$$

where G_k represents the volume of Ω_k . Using this equation, the integrals in (3) can be reformulated as

$$\dot{\bar{\phi}}_k(t) + \sum_{j \in \mathcal{C}(k)} (F(\bar{\phi}) \cdot n)_{k,j} \approx \bar{s}_k(t) \quad (4)$$

where $\mathcal{C}(k)$ is the set of the neighbor cells to the k th CV and $(F(\bar{\phi}) \cdot n)_{k,j}$ is the normal component of the numerical approximation of $f(\phi) \cdot n$, directed toward x_j starting from x_k . An illustrative example of the set $\mathcal{C}(k)$ is given in Fig. 1. Suitable numerical approximations need to be employed for the term $F(\bar{\phi})$; given that the average values of the unknown variables $\bar{\phi}$ are computed in the FVM, interpolation techniques are employed to recover the value of such unknowns at the edges of the CVs [40]. The approximation of $F(\bar{\phi})$ is discussed in next section.

3.2 Discretization of the governing equations

The discretization method introduced in Section 3.1 is exploited to reformulate the set of governing equations summarized in Table 1. Given that all the unknowns of the Li-ion cell model are functions of the variables $t \in \mathbb{R}^+$ and $x \in \mathbb{R}$, the development of a 1D FVM model is addressed. In order to correctly carry out the discretization process, a mesh structure of the spatial domain is defined by subdividing the spatial domain into $N_a + N_p + N_s + N_n + N_z$ non-overlapping volumes with geometrically centered nodes (as depicted in Fig. 2). Every CV is associated with a center x_k and spans the interval $[x_{k-\frac{1}{2}}; x_{k+\frac{1}{2}}]$. To facilitate the treatment of boundary and interface conditions, the edges of each CV are aligned with the domain boundaries and internal interfaces. The width of every CV is defined as $\Delta x_i = l_i / N_i$, where i represents a particular section of the battery.

Once the discretization mesh is structured, the governing equations are discretized as summarized in Table 3. All the interface conditions used to enforce continuity between adjacent materials are discussed in Section 3.3.

Particular attention is required for the thermal dynamics. The reversible and reactive heat sources can be discretized as

$$\begin{aligned} \bar{Q}_{\text{rev},k} &= F a_i \bar{j}_{i,k}(t) \bar{T}_k(t) \frac{\partial U_{i,k}}{\partial T} \\ \bar{Q}_{\text{rxn},k} &= F a_i \bar{j}_{i,k}(t) \bar{\eta}_{i,k}(t) \end{aligned}$$

whereas the derivatives present in the ohmic source are numerically approximated as

$$\begin{aligned}\left.\frac{\partial\Phi_s(x,t)}{\partial x}\right|_{x_k} &\approx \frac{\bar{\Phi}_{s,k+1}(t) - \bar{\Phi}_{s,k-1}(t)}{2\Delta x_i} \\ \left.\frac{\partial\Phi_e(x,t)}{\partial x}\right|_{x_k} &\approx \frac{\bar{\Phi}_{e,k+1}(t) - \bar{\Phi}_{e,k-1}(t)}{2\Delta x_i} \\ \left.\frac{\partial\ln c_e(x,t)}{\partial x}\right|_{x_k} &\approx \frac{\bar{c}_{e,k+1}(t) - \bar{c}_{e,k-1}(t)}{2\Delta x_i \bar{c}_{e,k}(t)}\end{aligned}$$

using a central differencing scheme. Finally the term $\bar{Q}_{\text{source},k} := \bar{Q}_{\text{ohm},k} + \bar{Q}_{\text{rev},k} + \bar{Q}_{\text{rxn},k}$.

Equation (C2) in Table 3 requires the evaluation of $T(x,t)$, $c_e(x,t)$, and κ_{eff} at the edges of the CVs. For example, consider Fig. 3, where the value of the unknown \bar{T} has to be evaluated at the interface between two CVs. In order to recover such value, linear interpolation techniques are used. The same approach is also applied for c_e and κ_{eff} .

As discussed in Section 3.1, a suitable numerical approximation for $F(\bar{\phi})$ is needed. Given that no convective terms are present in the set of governing equations, numerical approximation is only required for the diffusive terms (e.g., $-\Gamma\nabla\phi$). In this work, all the diffusive terms are numerically approximated with a first-order scheme:

$$\begin{aligned}\left.\frac{\partial\phi(x,t)}{\partial x}\right|_{x_{k+\frac{1}{2}}} &\approx \frac{\bar{\phi}_{k+1}(t) - \bar{\phi}_k(t)}{\Delta x} \\ \left.\frac{\partial\phi(x,t)}{\partial x}\right|_{x_{k-\frac{1}{2}}} &\approx \frac{\bar{\phi}_k(t) - \bar{\phi}_{k-1}(t)}{\Delta x}\end{aligned}$$

All the values coming from the additional equations in Table 2 are obtained as a function of the average values of the unknowns. Equation (T) is used to obtain the values of T ; and equations (M1), (M2), and (M3) are used to obtain the values of c_e , c_s^{avg} , and c_* respectively. The values of Φ_s are obtained from (C1) while the values of Φ_e are calculated through (C2).

3.3 Implementation of Boundary and Interface conditions

Boundary conditions must be enforced to have a physically meaningful solution. As shown in Table 1, null-flux boundary conditions on the electrolyte diffusion equation c_e can be straightforwardly enforced by imposing $\frac{\partial c_e}{\partial x} = 0$ at $x = \hat{x}_0$ and $x = \hat{x}_n$. The same procedure can be used to enforce $\frac{\partial\Phi_e}{\partial x} = 0$ at $x = \hat{x}_0$, while $\Phi_e = 0$ at $x = \hat{x}_n$ is enforced by setting to zero the value of Φ_e at the last CV of the anode. Solid-phase potential boundaries are enforced by substituting $\frac{\partial\Phi_s}{\partial x}$ at $x = \hat{x}_0$ and $x = \hat{x}_n$ the value of $-I_{\text{app}}/\sigma_{\text{eff},i}$. Similarly, at $x = \hat{x}_p$ and $x = \hat{x}_s$, $\frac{\partial\Phi_s}{\partial x}$ is replaced by the value 0. To enforce heat exchange with the surrounding environment, the terms $\frac{\partial T}{\partial x}$ evaluated at $x = 0$ and $x = L$ are substituted with the terms $h(T_{\text{ref}} - \bar{T}_1)$ and $h(\bar{T}_{\text{end}} - T_{\text{ref}})$ respectively. The suffixes ₁ and _{end} refer to the first and last CV of the entire mesh. All these conditions have been formulated also for the FVM discretization as shown in Table 3.

Due to changes in material properties along the length of the battery, interface conditions are required to enforce continuity of the solution. For this reason, the values of different coefficients (e.g., $D_{\text{eff},i}$, $\kappa_{\text{eff},i}$, λ_i) need to be evaluated at the interface between two different materials. The easiest way would be to use an arithmetic mean; however, in some cases, this approach cannot accurately handle the abrupt changes of coefficients that may occur. Instead, the HM is employed to evaluate the value at the edges of the CVs. The HM of two generic coefficients (k_1 and k_2) can be expressed as

$$\frac{k_1 k_2}{\beta k_2 + (1 - \beta) k_1}$$

where β represents a weight to account for the difference between the different CV widths. A common value for β is $\beta = \frac{\Delta x_1}{\Delta x_2 + \Delta x_1}$, where Δx_1 and Δx_2 represent the CV widths. This formulation produces

results that are more robust in presence of the abrupt changes of the coefficients, without requiring a excessively fine grid in the vicinity of the interface [41].

Consider Fig. 4 where the interface across the last volume of the cathode and the first volume of the separator is depicted. Remember that, as discussed in Section 3.2, the mesh structure has been chosen in order to align the CV edges with the interfaces or physical boundaries of the battery. The value of $D_{\text{eff},k+\frac{1}{2}}$ can be obtained using the HM as

$$D_{\text{eff},k+\frac{1}{2}} = \frac{D_{\text{eff},k}D_{\text{eff},k+1}}{\beta D_{\text{eff},k+1} + (1 - \beta)D_{\text{eff},k}}$$

where $\beta = \frac{\Delta x_p}{\Delta x_p + \Delta x_s}$. The electrolyte diffusion in the last volume of the cathode is

$$\begin{aligned} \epsilon_p \frac{\partial \bar{c}_{e,k}(t)}{\partial t} &= D_{\text{eff}}^e \frac{(\bar{c}_{e,k+1}(t) - \bar{c}_{e,k}(t))}{\Delta x_p(\Delta \tilde{x})} \\ &\quad - D_{\text{eff},k-\frac{1}{2}} \frac{(\bar{c}_{e,k}(t) - \bar{c}_{e,k-1}(t))}{\Delta x_p^2} \\ &\quad + a_p (1 - t_+) \bar{j}_{p,k}(t) \end{aligned}$$

whereas

$$\begin{aligned} \epsilon_s \frac{\partial \bar{c}_{e,k+1}(t)}{\partial t} &= D_{\text{eff},k+\frac{3}{2}} \frac{(\bar{c}_{e,k+2}(t) - \bar{c}_{e,k+1}(t))}{\Delta x_s^2} \\ &\quad - D_{\text{eff}}^e \frac{(\bar{c}_{e,k+1}(t) - \bar{c}_{e,k}(t))}{\Delta x_s(\Delta \tilde{x})} \end{aligned}$$

in the first volume of the separator, with $\Delta \tilde{x} = \frac{\Delta x_s + \Delta x_p}{2}$. The same approach is used to enforce interface conditions where needed.

When dealing with battery packs, in particular with series-connected cells, all the aforementioned numerical scheme have to be replicated for each cell. Moreover, when temperature dynamics are considered, the numerical scheme has to be adapted in order to account for continuity fluxes across the cells. Indeed, if two cells are connected in series, at the current collectors interface across the two cells (e.g., at $x = x^*$) must hold that

$$-\lambda_{z,1} \frac{\partial T_1(x,t)}{\partial x} \Big|_{x=x^*} = -\lambda_{a,2} \frac{\partial T_2(x,t)}{\partial x} \Big|_{x=x^*},$$

where $T_i(x,t)$ refers to the temperature of the current collector of the i -th cell. Finally, Fourier's law for heat conductivity has to be enforced respectively at the cathode of the first cell and at the anode of the second cell.

4 Li-ion Simulation Battery Toolbox (LIONSIMBA)

Different implementations of Li-ion cell simulation can be found in literature which are written in languages such as Maple, Matlab, and Fortran (DUALFOIL [23]). Commercial software such as COMSOL Multiphysics [22] or Modelica [42] provide a variety of models to simulate the behavior of a Li-ion cell. The lack of freely available Matlab-based software able to simulate Li-ion cells makes it difficult to have a reference simulation environment for the design and evaluation of different ABMSs. For this reason, in this work, the Matlab-based Li-ION SIMulation BAttery Toolbox (LIONSIMBA) has been developed. Due to its native integration with the Matlab environment, the proposed software facilitates the development of other types of algorithms (e.g., identification of Li-ion cell parameters, control algorithms for optimal control etc...). LIONSIMBA is freely downloadable at:

<http://sisdin.unipv.it/labsisdin/lionsimba.php>

Based on the Li-ion cell model presented in the previous sections, LIONSIMBA is a Matlab package whose objective is to facilitate the development of different ABMSs, providing a reliable and ready-to-use simulation environment. The package comes with compiled Matlab files and editable *.m* scripts:¹

- **electrolyteDiffusionCoefficients.m**: computes the electrolyte diffusion coefficients.
- **electrolyteConductivity.m**: computes the electrolyte conductivity coefficients.
- **openCircuitPotential.m**: used to compute the Open Circuit Potential (OCP).
- **reactionRates.m**: computes the reaction rate coefficients for the ionic flux.
- **solidPhaseDiffusionCoefficients.m**: computes the solid phase diffusion coefficients.

All the parameters related to the simulator and to the battery are managed through the script **Parameters_init.m**. The customization of this script allows the user to disable features such as the thermal dynamics, change the number of CVs of the mesh, enable real-time display of results, and change the battery section lengths, thermal conductivities, porosities, and so on. The user can define the operating mode of the charge/discharge cycle by selecting between galvanostatic, potentiostatic, or variable current profile operations.

The script **getInputCurrent.m** contains an example for the definition of the variable current profile, and can be used to apply a customized current profile during the simulation of the Li-ion battery. A generic nonlinear function can be used for this purpose; extra parameters can be used inside this function: current time instant t , initial integration time t_0 , final integration time t_f , and a structure-containing extra user data. For example, a possible implementation is

$$I(t) = \alpha \frac{t - t_0}{t_f - t_0} + \xi, \quad [\alpha, \xi] \in \mathbb{R}$$

More sophisticated control strategies such as model predictive control (MPC) can also be implemented in this framework (see the next section for an example). An additional degree of freedom is set by the possibility of defining a custom algorithm for the estimation of SOC and SOH. Within the **Parameters_init.m** script, the user can set custom functions to be externally called after each integration step; these functions will receive all the integration data of the battery and an extra structure-containing user-defined data.

A simulation can be initiated by calling from the Matlab command line:

```
out = startSimulation(t0, tf, initialStates, I, param)
```

where

- t_0 : represents the initial integration time.
- t_f : represents the final integration time.
- $initialStates$: represents the structure of initial states.
- I : represents the value of the applied input current.
- $param$: represents the cell array of parameters structures to be used in simulation.

The structure $initialStates$ can be used as initial state from which to start a simulation. If left empty, LIONSIMBA will automatically compute a set of consistent initial conditions (CICs) starting from which the simulation will run. If $initialStates$ is used as a parameter, it has to be a set of CICs for the battery model in Table 3. In case it is not a set of CIC, the numerical integrator will fail to converge and no results will be provided. The $param$ array, if passed, is used as the set of parameters for the

¹This set of scripts refer to version 1.02 of the software; modifications or other scripts can be added in future releases of the software.

simulation. If empty, the software will use a set of parameters according to the settings defined by the user in the script `Parameters_init.m`. When designing ABMSs for battery packs with series-connected cells, a cell-wise balancing has to be guaranteed during charging [43, 44]. LIONSIMBA can support the user in this task by providing a full independent parametrization of each cell of the series. Indeed, if the param array contains multiple parameters structure, the software will perform a simulation of a battery pack composed of several cells connected in series as shown in Section 5.1. Each element of the pack can be parametrized individually, leading to independent simulations of each cell. Finally, the out structure will contain the values of all the dependent variables and parameters used in the simulations. The package requires the SUNDIALS [21] suite to be installed and correctly configured with Matlab; in particular, the solver IDA is used.

To obtain further help on any single script, the user can type

```
help <scriptname>
```

from the Matlab command line or refer to the software manual.

The numerical implementation of the LIONSIMBA has been carried out according to the rules outlined in Section 3 and the cell considered is a LiCoO_2 and LiC_6 system. All the parameter values have been taken from the real battery data in [20], and are summarized in Table 4.

4.1 LIONSIMBA Validation

While the experimental validation of the P2D model has been addressed by the authors in [14], in the following the numerical implementation of LIONSIMBA is validated by comparing the results coming from the proposed framework with the ones coming from the commercial software COMSOL MultiPhysics and the Fortran code DUALFOIL. While COMSOL has been supplied with the same model used in our framework, where a heat diffusion PDE is used to describe the thermal dynamics, DUALFOIL neglects the spatial distribution of the temperature and averages the heat generation rates over the cell [45]. For this reason, the comparison among the three different codes is carried out considering isothermal conditions. For completeness, the validation of the thermal enabled code is carried out only with respect to COMSOL. For isothermal and thermal enabled scenarios, a 1C discharge cycle is performed, while the same set of parameters are maintained across the different codes.

The comparison among the cell potentials $V(t)$ in the isothermal case is shown in Fig. 5(a): the results coming from the LIONSIMBA framework (circles) fits the results obtained using COMSOL (solid line) and DUALFOIL (diamonds), which are almost identical. Similarly, the electrolyte concentrations $c_e(x, t)$ and potentials $\phi_e(x, t)$, presented in Fig. 5(b) and 5(d) respectively, show good fitting performance. Finally the surface solid phase concentrations $c_s^*(x, t)$ results are presented in Fig. 5(c). For the thermal enabled scenario, in Fig. 6(a) the $V(t)$ profiles are shown, where the solid line represents COMSOL results while the circles LIONSIMBA data. As in the previous case, the fit shows good performance of LIONSIMBA. Internal states are depicted in Fig. 6(b), Fig. 6(c) and 6(d) where $c_e(x, t)$, $c_s^*(x, t)$ and $\Phi_e(x, t)$ are represented respectively. Finally, in Fig. 6(e) the comparison between temperature profiles is shown.

4.2 Solid-phase diffusion models

As introduced in Section 2, according to the P2D model developed in [14], the diffusion process inside the solid particles is described using Fick's law, where the presence of a second-pseudo dimension (r) can significantly increase the computational burden. However, according to the particular application under study, different approximations of eq. (1) can be employed without an important loss of accuracy. The choice of the solid-phase diffusion model has to be cautious: indeed when dealing with scenarios comprising high rate of charge/discharge, short time simulations or pulse currents, the usage of approximate models may have a negative impact on the accuracy of the results [26]. For this reason, LIONSIMBA allows the user to chose among three different models for the solid-phase diffusion:

- Fick's law diffusion equation (including the pseudo-second dimension r):

$$\frac{\partial c_s(r, t)}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left[r^2 D_{\text{eff}}^s \frac{\partial c_s(r, t)}{\partial r} \right]$$

with boundary conditions

$$\left. \frac{\partial c_s(r, t)}{\partial r} \right|_{r=0} = 0 \quad \left. \frac{\partial c_s(r, t)}{\partial r} \right|_{r=R_p} = -\frac{j(x, t)}{D_{\text{eff},i}^s}$$

- two-parameters polynomial approximation [25]:

$$\begin{aligned} \frac{\partial c_s^{\text{avg}}(x, t)}{\partial t} &= -3 \frac{j(x, t)}{R_p} \\ c_s^*(x, t) - c_s^{\text{avg}}(x, t) &= -\frac{R_p}{D_{\text{eff},i}^s} \frac{j(x, t)}{5} \end{aligned}$$

- higher-order polynomial approximation [25]:

$$\begin{aligned} \frac{\partial c_s^{\text{avg}}(x, t)}{\partial t} &= -3 \frac{j(x, t)}{R_p} \\ \frac{\partial q(x, t)}{\partial t} &= -30 \frac{D_{\text{eff},i}^s}{R_p^2} q(x, t) - \frac{45}{2} \frac{j(x, t)}{R_p^2} \\ c_s^*(x, t) - c_s^{\text{avg}}(x, t) &= -\frac{j(x, t) R_p}{35 D_{\text{eff},i}^s} + 8 R_p q(x, t) \end{aligned}$$

To evaluate the prediction accuracy of each approximate model, a comparison among the different $V(t)$ profiles is here addressed. In particular the influence of different C rates is considered. Let define the model summarized in Table 1 using the Fick's law of diffusion as the "full model". In Fig. 7(a) and Fig. 7(b) the comparison among the full model and approximate ones is depicted. When dealing with low to medium C rates (1C-2C) the adoption of two-parameters polynomial approximation is able to accurately represent the cell behavior. Despite this, while working at higher rates (5C), higher-order polynomial approximations give better results as shown in Fig. 7(c). Finally, when working with high C rates (i.e. HEV applications which run at 10C-20C), two-terms approximations are not able to represent accurately the cell behavior, while higher-order approximation still produces reasonable results as depicted in Fig. 7(d). To quantify the performance of each approximate model, root mean square error (RMSE) indices and normalized time indices (NTI) are summarized in Table 6. In each case the RMSEs are evaluated comparing approximate models solutions with respect to the full model solutions, while the normalized times are obtained as the ratio between the computational time required by approximate models and the time required by the full one to simulate the different scenarios. As it is possible to see, a two-terms polynomial approximation has good results for low C rates. Performance are lowered when higher C rates are considered. In all the scenarios, this approximation takes $\approx 80\%$ less time than the full model to simulate the cell. As expected, by using a higher-order polynomial approximations, the RMSEs for each scenario are lowered. Compared to the two-terms model, due to the addition of a set of ODEs, a significant increase in the computational time is recorded. Nevertheless, it is worth highlighting that at 10C the higher-order approximation has a RMSE of 1.8701%. This aspect, together with a computational time which is $\approx 37\%$ less than the full model one, makes this approximation a reasonable candidate for simulation up to 10C for this particular chemistry.

5 Simulations

Simulation results were obtained using Matlab R2014b on a Windows 7@3.2GHz PC with 8 GB of RAM for the experimental battery parameters in [20] with a cutoff voltage of 2.5 V and environmental temperature of 298.15 K. For the proposed chemistry, the 1C value is $\approx 30 \text{ A/m}^2$. The effectiveness and ease of use of the proposed framework is shown.

In the first scenario (Fig. 8), -1C discharge simulations are compared for a very wide range of heat exchange coefficient h , with high h being the most challenging for retaining numerical stability in dynamic simulations. As expected, decreasing the value of the parameter h leads to a faster increase of the cell temperature. Moreover, due to the coupling of all the governing equations, it is possible to note the influence of different temperatures on the cell voltage. In the second scenario (Fig. 9), for a fixed value of $h = 1 \text{ W/(m}^2 \text{ K)}$, different discharge cycles are compared at -0.5C , -1C , and -2C . According to the different applied currents, the temperature rises in different ways; it is interesting to note the high slope of the temperature during a -2C discharge, mainly due to the electrolyte concentration c_e being driven to zero in the positive electrode by the high discharge rate. In the third scenario, the framework is used to simulate a hybrid charge-discharge cycle, emulating the throttle of a HEV. During breaking, the battery gets charged. Table 5 resumes the configuration of the car throttle during simulations. In Fig. 10 it is possible to analyze the response of a single cell inside an HEV pack under a hybrid charge-discharge cycle. In this case effects of temperature among the different cells have been neglected. The solid potential behavior is primarily due to the different applied C rates, with discontinuous changes producing voltage drops. Different slopes of the voltage curve are related to the different C rates applied. Temperature rise is recorded in the first 50 seconds of simulations, which are followed by a slight decrease of the temperature mainly due to the exchange of heat with the surrounding environment ($h = 1 \text{ W/(m}^2 \text{ K)}$) and due to the lower current density applied. At around 250 s, temperature starts to increase due to the -1C rate applied during moderate speed; high slope of increase at around 410 s is due to the higher value of the discharge current which during an overtake reaches the value of -2C . Returning to moderate speed makes the temperature slope more gentle. During the last 10 seconds, temperature decreases due to the significant change in applied current and due to dissipation of heat with surrounding ambient. A sketch of the code used for this simulation is presented in Algorithm 1.

Algorithm 1 Car cycling example code

```

Input setup:
1: I = {−29.5, 14.75, −14.75, −29.5, −58, −29.5, 14.75}           ▷ Simulation current densities
2: time = {50, 10, 150, 200, 5, 200, 10}                         ▷ Duration of each element of  $I_{applied}$  (in seconds).
3: t0 = 0;                                                         ▷ Init all the useful variables
4: tf = 0;
5: initialStates.Y = [ ];
6: initialStates.YP = [ ];
7: Phis_tot = [ ];
8: t_tot = [ ];
9: T_tot = [ ];
Core script:
10: for i = 1:length(I) do
11:     tf = tf + time(i);
12:     results = startSimulation(t0,tf,initialStates,I(i),[ ]);
13:     Phis_tot = [Phis_tot;results.original.Phis];                ▷ Concatenate results
14:     T_tot=[T_tot;results.original.Temperature];
15:     t0 = time(i);
16:     initialStates = results.initialStates;                    ▷ Update initial states for the next simulation
17: end for

```

In Fig. 11, the application of an ABMS is addressed. In this particular simulation, a model predictive control algorithm [8] is adopted to drive the SOC of the battery to a given value, while accounting for input and output constraints. The initial SOC was around 20% and its reference value was set to 85%. According to LIONSIMBA, the estimation of the SOC can be easily carried out by defining a custom function. In this particular scenario, the SOC has been computed as

$$\text{SOC}(t) = \frac{1}{l_n c_s^{\max}} \int_0^{l_n} c_s^{\text{avg}}(x, t) dx$$

The temperature maximum bound was set to $313.5K$, with the voltage set to 4.2 V . The BMS applies a current density which is almost fixed at $1C$ value for the entire simulation, while starting to drop as the SOC approaches its final value. The behavior of the SOC is almost linear during the first 2500 s , while starting to change according to the current drop, in order to approach smoothly the final stage of charge. In Fig. 12 it is possible to see that, according to the different charging stages, the electrolyte concentration diffuses in different ways. Starting from $c_e^{\text{init}} = 1000\text{ mol/m}^3$, the input current induces a drop of concentration within the battery sections due to the diffusion of ions from the cathode to the anode. Approaching the final stage of charging, the concentration starts to converge back to the initial value of 1000 mol/m^3 and, around 5500 s , reaches the steady value. This behavior emphasizes the property of the FVM to conserve properties within numerical roundoff. Algorithm 2 provides a high-level description of how to implement a closed-loop controller in LIONSIMBA. In Fig.

Algorithm 2 High level control code

Init script:

```

1: t0 = 0;
2: tf = dt;                                ▷ Simulations are run over a sampling time periods
3: initialStates.Y = [ ];
4: initialStates.YP = [ ];
5: Condition = 1;

```

Core script:

```

6: while Condition do
7:   I = ComputeControlLaw(initialStates);
8:   results = startSimulation(t0,tf,initialStates,I,[ ]);
9:   [...]                                ▷ Elaborate and concatenate the results and update the time indices
10:  initialStates = results.initialStates;    ▷ Update initial states for the next simulation
11:  if SOC reached reference value then
12:    Condition = 0
13:  end if
14: end while

```

13, simulations have been run disabling the thermal dynamics leading to an isothermal environment. This particular configuration can be exploited in order to assess the influence of different constant temperatures at which the battery can operate.

All the results of the proposed simulations can be reproduced by running the example scripts available with LIONSIMBA. Finally, Table 7 shows the times required by the simulator to perform the different scenarios, which are all under 100 s .

5.1 Battery pack of series-connected cells

In Figures 14 and 15 the results of a battery pack simulation are proposed. To emphasize the ability to independently parametrize each cell, in this scenario the SOC of cell #1 is set to the 95% of its initial value while the thickness of the cathode of cell #2 is doubled with respect to its initial value.

All the other parameters are the same for the three cells. In Fig. 14 the behavior of the output voltage of the overall pack versus each cell is depicted. It is possible to see that the starting voltage of the pack is around 12.1 V and decreases subjected to a 1C discharge current. In 3346 s the pack get completely discharged due to cell #1 which approaches earlier the cutoff voltage (set to 2.5 V). The lowered starting SOC has determined this behavior. In Fig. 15 the comparison among the electrolyte and solid-phase surface concentrations as well as the electrolyte potentials are shown. The cell #2 has a significant different behavior mainly due to the presence of a cathode with a thickness doubled with respect to the other two cells. Note that this variation has effects over the output voltages, as shown in Fig. 14. Besides cell # 1 which is starting from a different SOC value, the different behaviors of $V(t)$ between cell #2 and cell #3 are driven by the thickness variation.

6 Conclusions

This work describes a detailed procedure for the numerical implementation of the P2D model developed by the authors in [14]. By considering an approximate model for the solid-phase diffusion, the pseudo-second dimension is removed to reduce the computational complexity. The treatment of boundary conditions is addressed with particular attention to the interface conditions across the different sections of the battery. Following the procedures and rules outlined in Section 3, the reader can implement his/her own version of the model in different programming languages. Moreover, a freely available Matlab framework LIONSIMBA is provided that is suitable for battery design, simulation, and control. The framework is extended to account for different solid-phase diffusion models to meet required accuracy. The simulations demonstrate high numerical stability for different operating scenarios. The effectiveness of LIONSIMBA is verified considering a heterogeneous sequence of applied current coming from an HEV and through the assessment of an ABMS strategy, in particular, the model predictive control of state of charge. A battery pack composed of series-connected cells can be simulated by considering several independent cells with their own parameters. Due to its integration with the Matlab environment, the framework facilitates the development and test of different algorithms such as control algorithms, identification procedures or optimization of manufacturing parameters and so on. A timing benchmark of the proposed framework has been performed and the results are summarized in Table 8. The comparison among DUALFOIL, COMSOL and LIONSIMBA is presented. For each code, a 1C discharging cycle in isothermal conditions is simulated repeatedly; average simulation times are considered. At a first glance, timings appear to be all comparable among the different implementations. In particular, the average time of DUALFOIL results to be less or equal than the other two: this is mainly due to the fully compiled nature of the code. Due to its pseudo-compiled implementation, the LIONSIMBA framework has to be completely interpreted by MATLAB at runtime; this explains the slight increase in simulation time with respect to DUALFOIL. However, in all the scenarios, both LIONSIMBA and DUALFOIL outperform the performance of COMSOL. These preliminary results highlight the promising nature of the proposed framework as a reliable, efficient and freely available Matlab-based software for the P2D model simulation. Further developments (e.g. code optimization and distribution of compiled versions) can only concur to improve the current performance. Moreover, as the proposed simulations were written in standard serial mode, the computation time could be reduced by at least a factor of ten by using a multicore CPU using parallel DAE solvers. Modern versions of Matlab have easy-to-implement built-in options for distributing calculations among multiple cores on a single CPU, and among multiple CPUs.

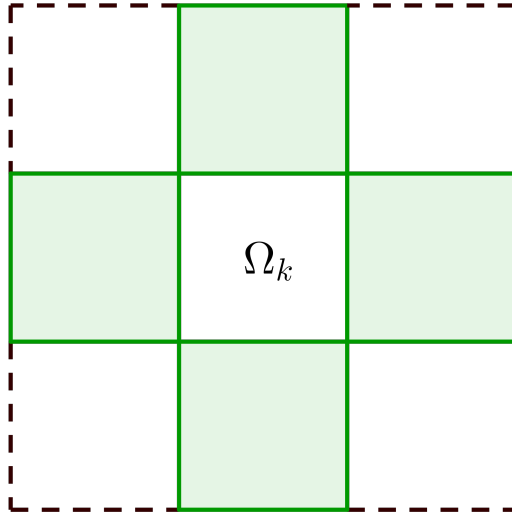


Figure 1: Example of a 2D FVM mesh where the set of neighbor cells $\mathcal{C}(k)$ is represented by the green cells.

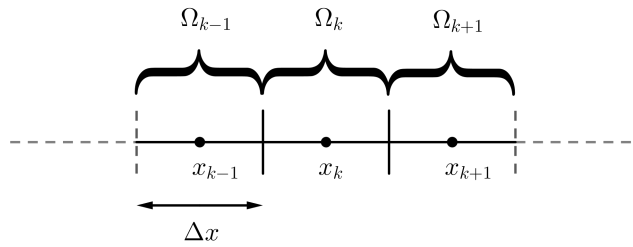


Figure 2: One-dimensional finite volume mesh

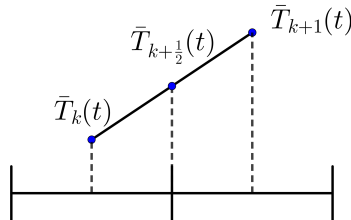


Figure 3: Interpolation technique to recover edge values of the unknowns.

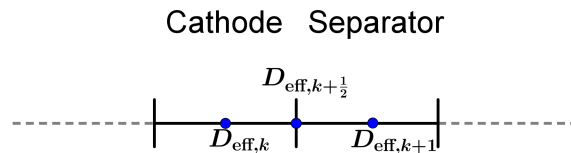
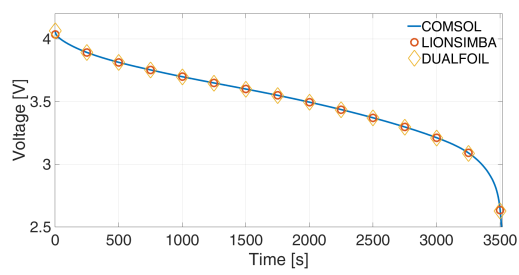
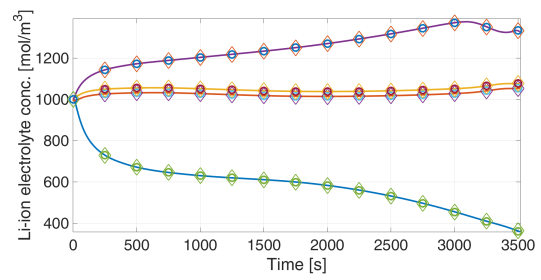


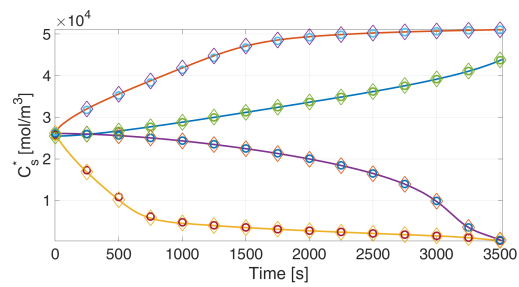
Figure 4: Electrolyte diffusion process: interface across the cathode and separator.



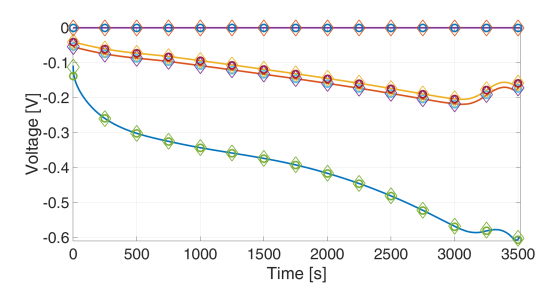
(a) Cell potential



(b) Electrolyte Li-ions concentration

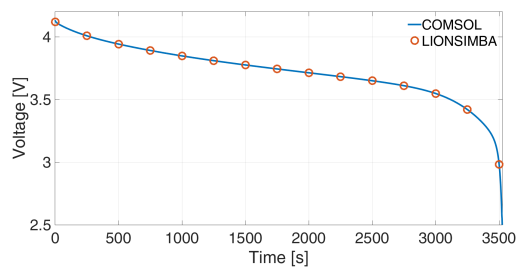


(c) Solid phase Li-ions concentration

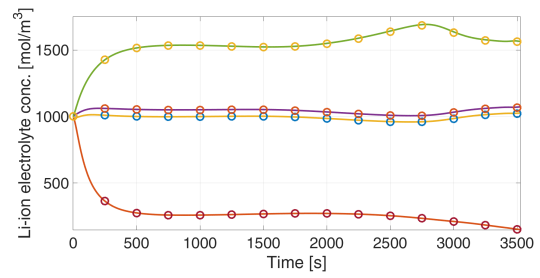


(d) Electrolyte potential

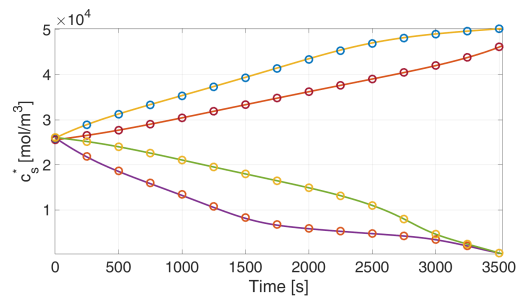
Figure 5: Validation of the proposed numerical implementation in isothermal conditions



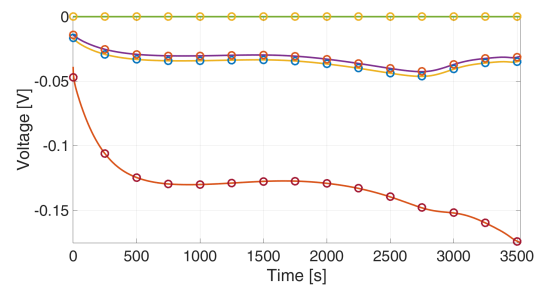
(a) Cell potential



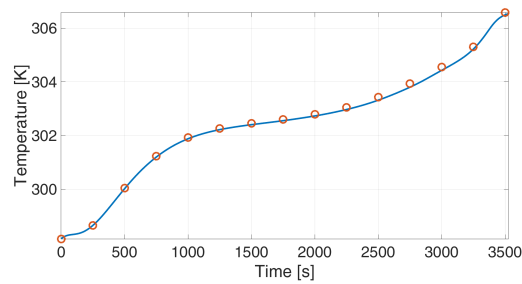
(b) Electrolyte Li-ions concentration



(c) Solid phase Li-ions concentration

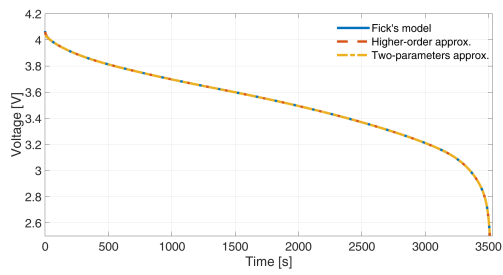


(d) Electrolyte potential

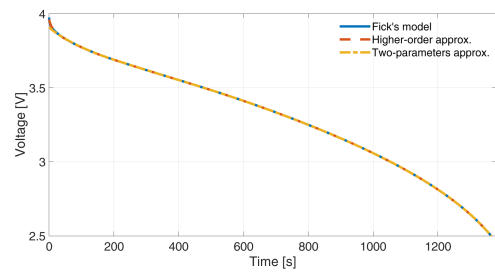


(e) Temperature

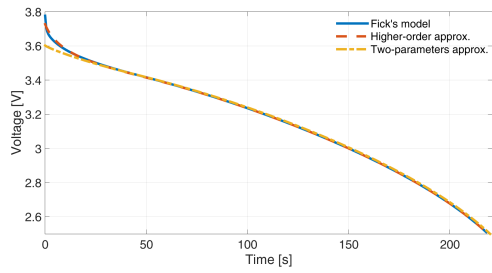
Figure 6: Validation of the proposed numerical implementation with thermal dynamics



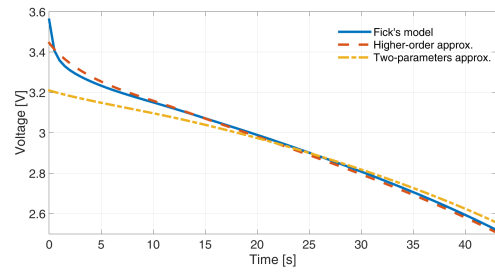
(a) 1C rate comparison



(b) 2C rate comparison



(c) 5C rate comparison



(d) 10C rate comparison

Figure 7: Comparison of the three different solid-phase diffusion equations implemented in LION-SIMBA.

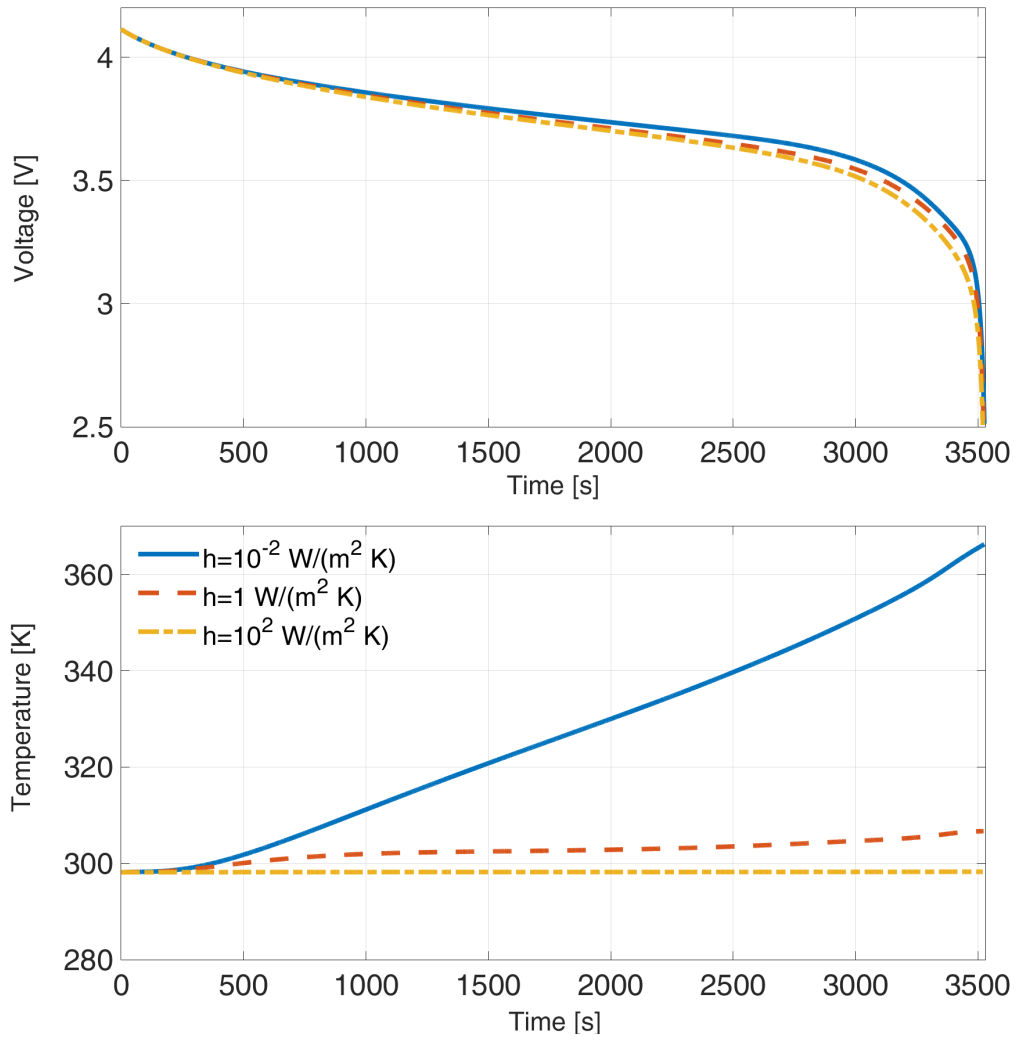


Figure 8: -1C discharge cycle run under different heat exchange parameters: blue line $h = 0.01\text{W}/(\text{m}^2 \text{ K})$, dashed orange line $h = 1 \text{ W}/(\text{m}^2 \text{ K})$ and dot-dashed yellow line $h = 100 \text{ W}/(\text{m}^2 \text{ K})$.

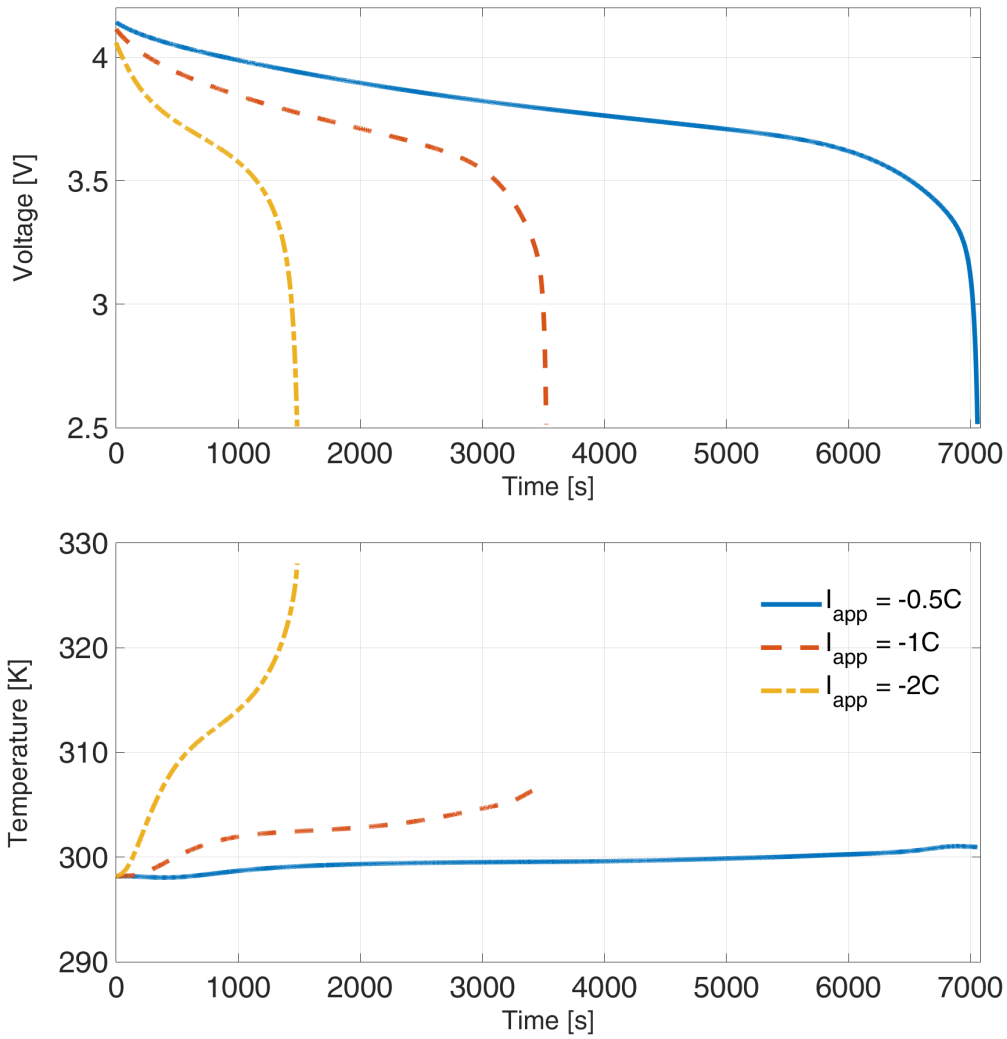


Figure 9: Full discharge cycle run under different C rates: $-2C$ (dot-dashed yellow), $-1C$ (dashed orange line), and $-0.5C$ (blue line).

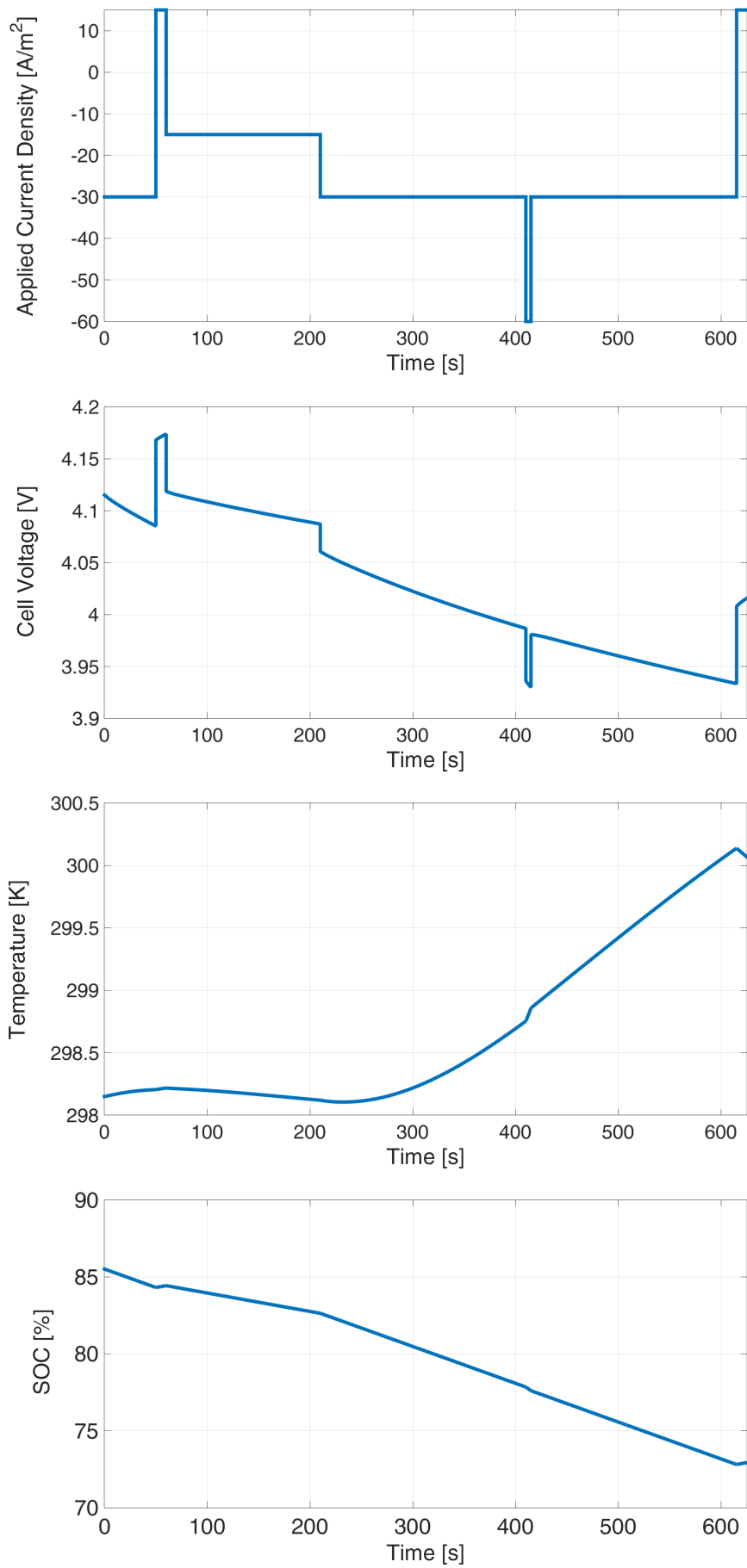


Figure 10: Hybrid charging-discharging cycle.

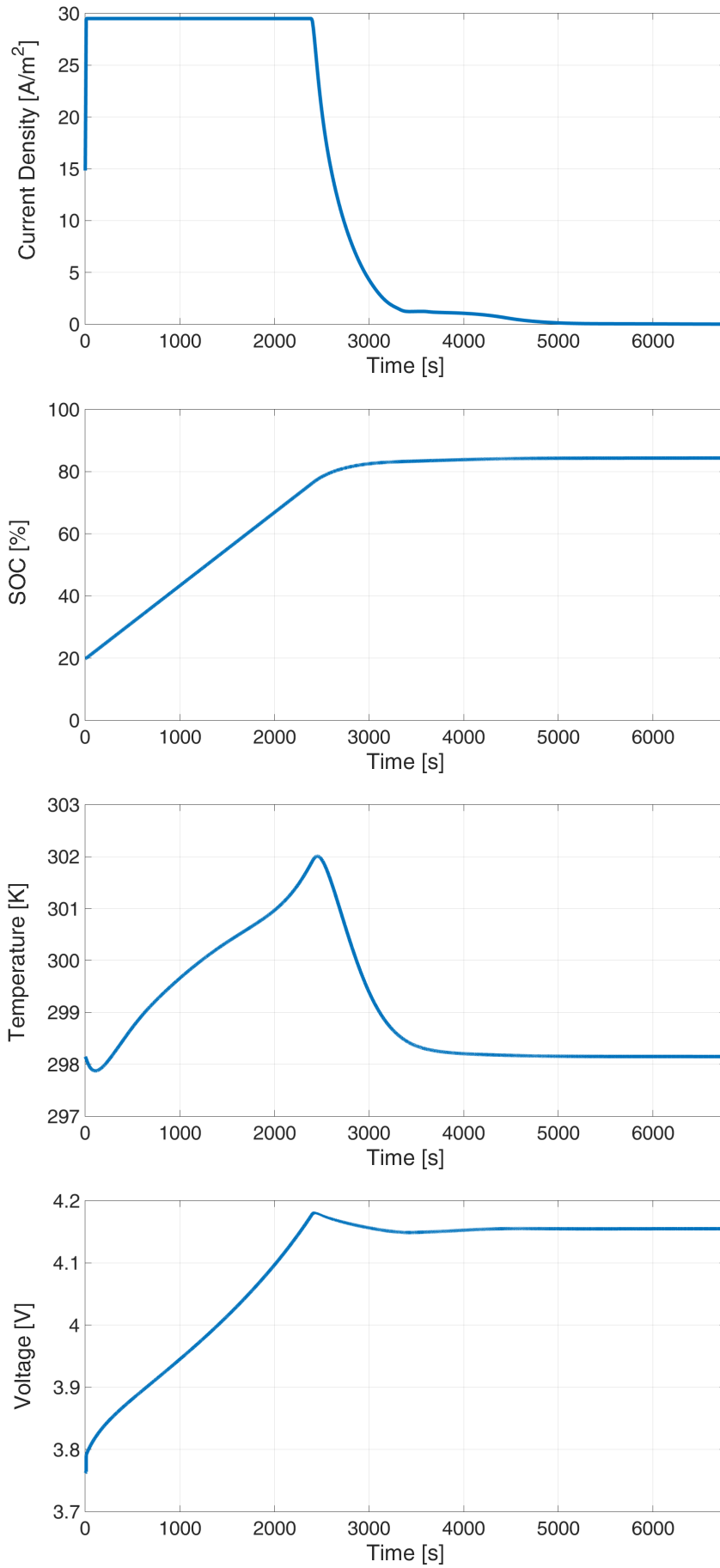


Figure 11: *ABMS control*: an MPC algorithm [8] is used to drive the charge of the battery from 20% to 85% while considering voltage, temperature, and current constraints.

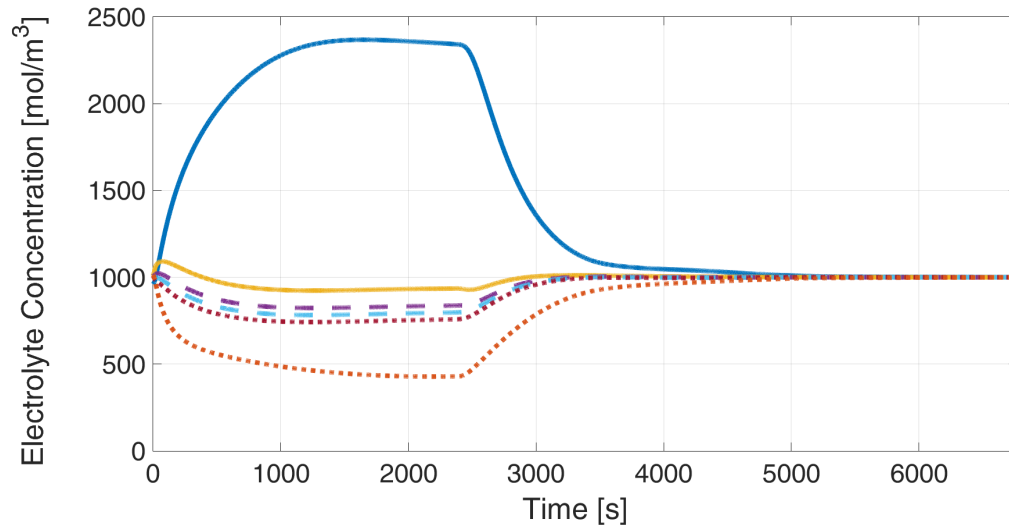


Figure 12: *ABMS control – Electrolyte concentration*: The behavior of the first and last volume of each section of the battery is depicted, where the continuous lines belong to the cathode, the dashed lines to the separator and the dotted lines to the anode.

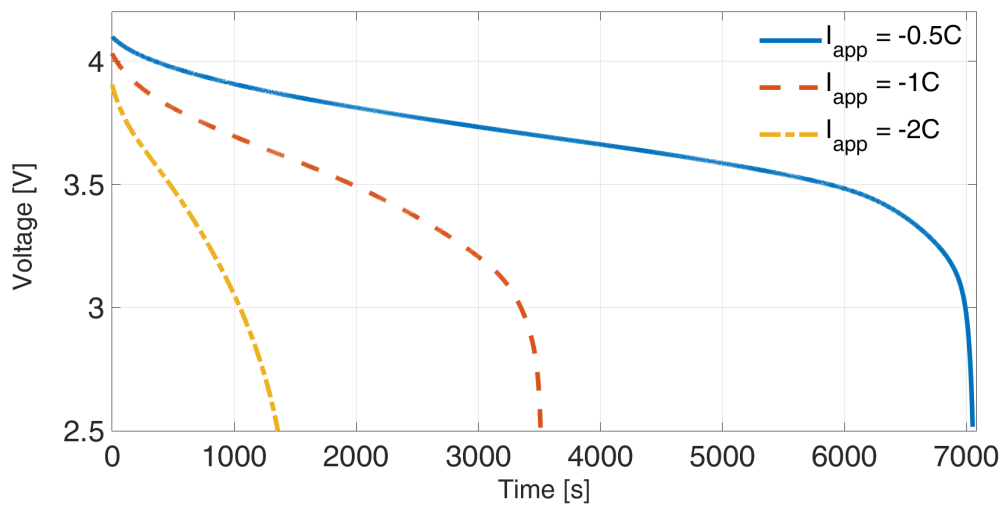


Figure 13: Full discharge cycle in an isothermal environment: blue line $-0.5C$, dashed orange line $-1C$, and dot-dashed yellow line $-2C$.

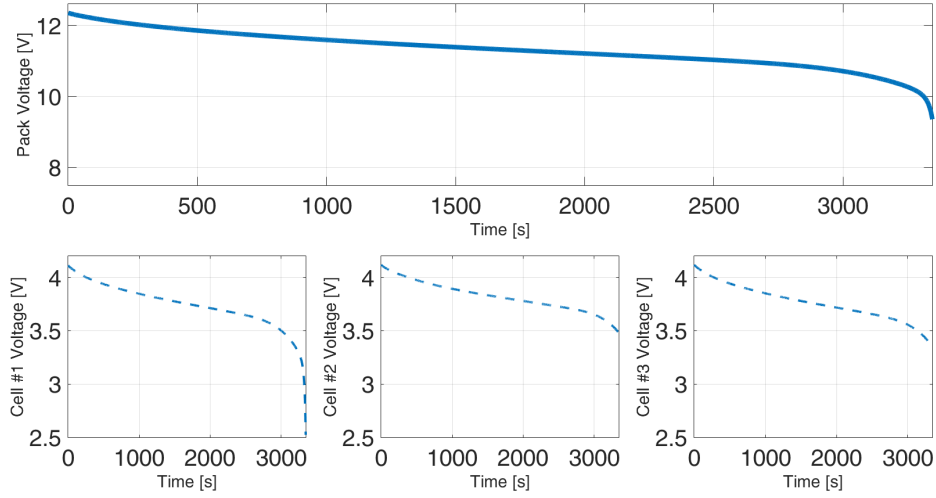


Figure 14: Simulation of a 3-cells pack. The upper curve represents the overall voltage of the 3 series connected Li-ion cells, while the lower plots depict the voltage of each cell in the pack. The different parametrization of each cell determines different behaviors.

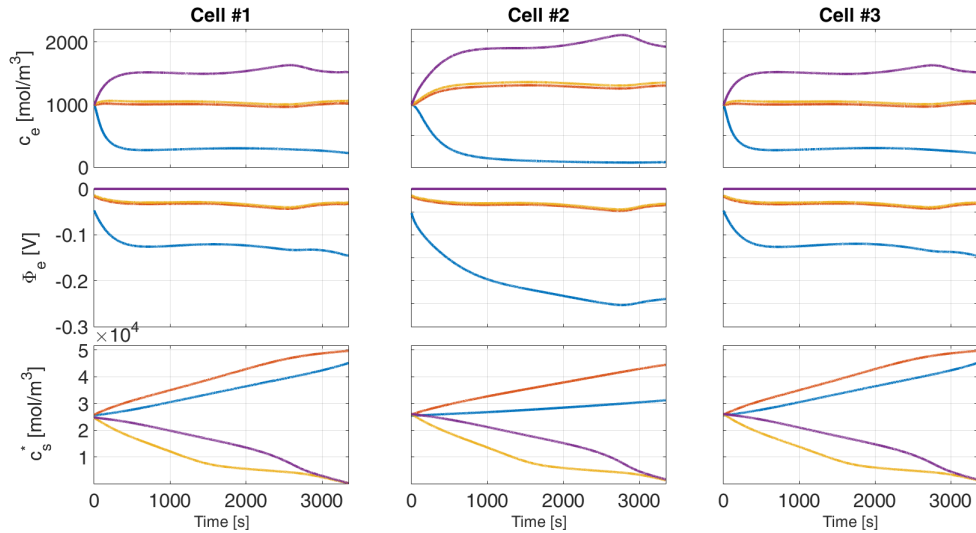


Figure 15: Simulation of a 3-cells pack. Different internal states profiles inside the three cells. Individual parametrizations leads to different behaviors.

$$\rho_i C_{p,i} \frac{\partial T(x,t)}{\partial t} = \frac{\partial}{\partial x} \left[\lambda_i \frac{\partial T(x,t)}{\partial x} \right] + \frac{I_{\text{app}}^2(t)}{\sigma_{\text{eff},i}}$$

Positive and Negative Electrodes, $i \in \{p, n\}$

$$\epsilon_i \frac{\partial c_e(x,t)}{\partial t} = \frac{\partial}{\partial x} \left[\mathbf{D}_{\text{eff},i} \frac{\partial c_e(x,t)}{\partial x} \right] + a_i (1 - t_+) j_i(x,t)$$

$$\frac{\partial c_s^{\text{avg}}(x,t)}{\partial t} = -3 \frac{j_i(x,t)}{R_{p,i}}$$

$$c_s^*(x,t) - c_s^{\text{avg}}(x,t) = -\frac{R_{p,i}}{\mathbf{D}_{\text{eff},i}^s} \frac{j_i(x,t)}{5}$$

$$\frac{\partial}{\partial x} \left[\sigma_{\text{eff},i} \frac{\partial \Phi_s(x,t)}{\partial x} \right] = a_i F j_i(x,t)$$

$$a_i F j_i(x,t) = -\frac{\partial}{\partial x} \left[\kappa_{\text{eff},i} \frac{\partial \Phi_e(x,t)}{\partial x} \right] + \frac{\partial}{\partial x} \left[\kappa_{\text{eff},i} \Upsilon T(x,t) \frac{\partial \ln c_e(x,t)}{\partial x} \right]$$

$$\rho_i C_{p,i} \frac{\partial T(x,t)}{\partial t} = \frac{\partial}{\partial x} \left[\lambda_i \frac{\partial T(x,t)}{\partial x} \right] + \mathbf{Q}_{\text{ohm},i} + \mathbf{Q}_{\text{rxn},i} + \mathbf{Q}_{\text{rev},i}$$

$$j_i(x,t) = 2 \mathbf{k}_{\text{eff},i} \sqrt{c_e(x,t) (c_{s,i}^{\text{max}} - c_{s,i}^*(x,t)) c_{s,i}^*(x,t)} \sinh \left[\frac{0.5R}{FT(x,t)} \eta_i(x,t) \right]$$

$$\eta_i(x,t) = \Phi_s(x,t) - \Phi_e(x,t) - \mathbf{U}_i$$

Separator, $i = s$

$$\epsilon_i \frac{\partial c_e(x,t)}{\partial t} = \frac{\partial}{\partial x} \left[\mathbf{D}_{\text{eff},i} \frac{\partial c_e(x,t)}{\partial x} \right]$$

$$0 = -\frac{\partial}{\partial x} \left[\kappa_{\text{eff},i} \frac{\partial \Phi_e(x,t)}{\partial x} \right] + \frac{\partial}{\partial x} \left[\kappa_{\text{eff},i} T(x,t) \Upsilon \frac{\partial \ln c_e(x,t)}{\partial x} \right]$$

$$\rho_i C_{p,i} \frac{\partial T(x,t)}{\partial t} = \frac{\partial}{\partial x} \left[\lambda_i \frac{\partial T(x,t)}{\partial x} \right] + \mathbf{Q}_{\text{ohm},i}$$

$$-\lambda_a \frac{\partial T(x,t)}{\partial x} \Big|_{x=0} = h(T_{\text{ref}} - T(x,t))$$

$$-\lambda_z \frac{\partial T(x,t)}{\partial x} \Big|_{x=L} = h(T(x,t) - T_{\text{ref}})$$

$$\frac{\partial c_e(x,t)}{\partial x} \Big|_{x=\hat{x}_0, \hat{x}_n} = 0$$

$$\sigma_{\text{eff},i} \frac{\partial \Phi_s(x,t)}{\partial x} \Big|_{x=\hat{x}_0, \hat{x}_n} = -I_{\text{app}}(t)$$

$$\sigma_{\text{eff},i} \frac{\partial \Phi_s(x,t)}{\partial x} \Big|_{x=\hat{x}_p, \hat{x}_s} = 0$$

$$\frac{\partial \Phi_e(x,t)}{\partial x} \Big|_{x=\hat{x}_0} = 0$$

$$\Phi_e(x,t) \Big|_{x=\hat{x}_n} = 0$$

$$-\lambda_z \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_0^-} = -\lambda_p \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_0^+}$$

$$-\lambda_n \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_n^-} = -\lambda_z \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_n^+}$$

$$-\mathbf{D}_{\text{eff},p} \frac{\partial c_e(x,t)}{\partial x} \Big|_{x=\hat{x}_p^-} = -\mathbf{D}_{\text{eff},s} \frac{\partial c_e(x,t)}{\partial x} \Big|_{x=\hat{x}_p^+}$$

$$-\mathbf{D}_{\text{eff},s} \frac{\partial c_e(x,t)}{\partial x} \Big|_{x=\hat{x}_s^-} = -\mathbf{D}_{\text{eff},n} \frac{\partial c_e(x,t)}{\partial x} \Big|_{x=\hat{x}_s^+}$$

$$-\kappa_{\text{eff},p} \frac{\partial \Phi_e(x,t)}{\partial x} \Big|_{x=\hat{x}_p^-} = -\kappa_{\text{eff},s} \frac{\partial \Phi_e(x,t)}{\partial x} \Big|_{x=\hat{x}_p^+}$$

$$-\kappa_{\text{eff},s} \frac{\partial \Phi_e(x,t)}{\partial x} \Big|_{x=\hat{x}_s^-} = -\kappa_{\text{eff},n} \frac{\partial \Phi_e(x,t)}{\partial x} \Big|_{x=\hat{x}_s^+}$$

$$-\lambda_p \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_p^-} = -\lambda_s \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_p^+}$$

$$-\lambda_s \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_s^-} = -\lambda_n \frac{\partial T(x,t)}{\partial x} \Big|_{x=\hat{x}_s^+}$$

Table 1: Li-ion P2D model governing equations

Open Circuit Potential (Thermal dependence)

$$U_p = U_{p,\text{ref}} + (T(x, t) - T_{\text{ref}}) \left. \frac{\partial U_p}{\partial T} \right|_{T_{\text{ref}}}$$

$$U_n = U_{n,\text{ref}} + (T(x, t) - T_{\text{ref}}) \left. \frac{\partial U_n}{\partial T} \right|_{T_{\text{ref}}}$$

Entropy change

$$\left. \frac{\partial U_p}{\partial T} \right|_{T_{\text{ref}}} = -0.001 \left(\frac{0.199521039 - 0.928373822\theta_p + 1.364550689000003\theta_p^2 - 0.6115448939999998\theta_p^3}{1 - 5.661479886999997\theta_p + 11.47636191\theta_p^2 - 9.82431213599998\theta_p^3 + 3.048755063\theta_p^4} \right)$$

$$\left. \frac{\partial U_n}{\partial T} \right|_{T_{\text{ref}}} = \frac{0.001 \left(0.005269056 + 3.299265709\theta_n - 91.79325798\theta_n^2 + 1004.911008\theta_n^3 - 5812.278127\theta_n^4 + 19329.7549\theta_n^5 - 37147.8947\theta_n^6 + 38379.18127\theta_n^7 - 16515.05308\theta_n^8 \right)}{\left(1 - 48.09287227\theta_n + 1017.234804\theta_n^2 - 10481.80419\theta_n^3 + 59431.3\theta_n^4 - 195881.6488\theta_n^5 + 374577.3152\theta_n^6 - 385821.1607\theta_n^7 + 165705.8597\theta_n^8 \right)}$$

Open Circuit Potential (Reference value)

$$U_{p,\text{ref}} = \frac{-4.656 + 88.669\theta_p^2 - 401.119\theta_p^4 + 342.909\theta_p^6 - 462.471\theta_p^8 + 433.434\theta_p^{10}}{-1 + 18.933\theta_p^2 - 79.532\theta_p^4 + 37.311\theta_p^6 - 73.083\theta_p^8 + 95.96\theta_p^{10}}$$

$$U_{n,\text{ref}} = 0.7222 + 0.1387\theta_n + 0.029\theta_n^{0.5} - \frac{0.0172}{\theta_n} + \frac{0.0019}{\theta_n^{1.5}} + 0.2808e^{0.9-15\theta_n} - 0.7984e^{0.4465\theta_n-0.4108}$$

$$\theta_p = \frac{c_{s,p}^*(x, t)}{c_{s,p}^{\text{max}}}$$

$$\theta_n = \frac{c_{s,n}^*(x, t)}{c_{s,n}^{\text{max}}}$$

Heat source terms (Anode and Cathode)

$$Q_{\text{ohm},i} = \sigma_{\text{eff},i} \left(\frac{\partial \Phi_s(x, t)}{\partial x} \right)^2 + \kappa_{\text{eff},i} \left(\frac{\partial \Phi_e(x, t)}{\partial x} \right)^2 + \frac{2\kappa_{\text{eff},i}RT(x, t)}{F} (1 - t_+) \frac{\partial \ln c_e(x, t)}{\partial x} \frac{\partial \Phi_e(x, t)}{\partial x}, \quad i \in \{p, n\}$$

$$Q_{\text{rxn},i} = F a_i j_i(x, t) \eta_i(x, t), \quad i \in \{p, n\}$$

$$Q_{\text{rev},i} = F a_i j_i(x, t) T(x, t) \left. \frac{\partial U_i}{\partial T} \right|_{T_{\text{ref}}}, \quad i \in \{p, n\}$$

Heat Source terms (Separator)

$$Q_{\text{ohm},i} = \kappa_{\text{eff},i} \left(\frac{\partial \Phi_e(x, t)}{\partial x} \right)^2 + \frac{2\kappa_{\text{eff},i}RT(x, t)}{F} (1 - t_+) \frac{\partial \ln c_e(x, t)}{\partial x} \frac{\partial \Phi_e(x, t)}{\partial x}, \quad i = s$$

Various Coefficients

$$D_{\text{eff},i} = \epsilon_i^{\text{brugg}_i} \times 10^{-4} \times 10^{-4.43 - \frac{54}{T(x, t) - 229 - 5 \times 10^{-3} c_e(x, t)} - 0.22 \times 10^{-3} c_e(x, t)}$$

$$\kappa_{\text{eff},i} = \epsilon_i^{\text{brugg}_i} \times 10^{-4} \times c_e(x, t) \left(\begin{array}{l} -10.5 + 0.668 \cdot 10^{-3} \cdot c_e(x, t) + 0.494 \cdot 10^{-6} c_e^2(x, t) + \\ (0.074 - 1.78 \times 10^{-5} c_e(x, t) - 8.86 \times 10^{-10} c_e^2(x, t)) T(x, t) + \\ (-6.96 \times 10^{-5} + 2.8 \times 10^{-8} c_e(x, t)) T^2(x, t) \end{array} \right)$$

$$k_{\text{eff},i} = k_i e^{-\frac{E_a^{k_i}}{R} \left(\frac{1}{T(x, t)} - \frac{1}{T_{\text{ref}}} \right)} \quad \Upsilon := \frac{2(1 - t_+)R}{F}$$

$$D_{\text{eff},i}^s = D_i^s e^{-\frac{E_a^{D_i^s}}{R} \left(\frac{1}{T(x, t)} - \frac{1}{T_{\text{ref}}} \right)}$$

$$\sigma_{\text{eff},i} = \sigma_i (1 - \epsilon_i - \epsilon_{f,i})$$

$$(T) \rho_i C_{p,i} \frac{\partial \bar{T}_k(t)}{\partial t} = \frac{1}{\Delta x_i} \left[\lambda_i \frac{\partial T(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} + \frac{I_{\text{app}}^2(t)}{\sigma_{\text{eff},i}}$$

$$\left[\lambda_i \frac{\partial T(x, t)}{\partial x} \right] \Bigg|_0 = h(T_{\text{ref}} - \bar{T}_1(t))$$

$$\left[\lambda_i \frac{\partial T(x, t)}{\partial x} \right] \Bigg|_L = h(\bar{T}_{\text{end}}(t) - T_{\text{ref}})$$

Positive and Negative Electrodes, $i \in \{p, n\}$

$$(M1) \epsilon_i \frac{\partial \bar{c}_{e,k}(t)}{\partial t} = \frac{1}{\Delta x_i} \left[D_{\text{eff},i} \frac{\partial c_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} + a_i(1 - t_+) \bar{j}_{i,k}(t)$$

$$\frac{\partial c_e(x, t)}{\partial x} \Bigg|_{\hat{x}_0} = 0$$

$$\frac{\partial c_e(x, t)}{\partial x} \Bigg|_{\hat{x}_n} = 0$$

$$(M2) \frac{\partial \bar{c}_s^{\text{avg}}(t)}{\partial t} = -3 \frac{\bar{j}_k(t)}{R_{p,i}}$$

$$(M3) \bar{c}_s^*(t) - \bar{c}_s^{\text{avg}}(t) = -\frac{R_{p,i}}{D_{\text{eff},i}^s} \frac{\bar{j}_k(t)}{5}$$

$$(C1) \left[\sigma_{\text{eff},i} \frac{\partial \Phi_s(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} = a_i F \bar{j}_{i,k}(t) \Delta x_i$$

$$\left[\sigma_{\text{eff},i} \frac{\partial \Phi_s}{\partial x} \right] \Bigg|_{\hat{x}_0, \hat{x}_n} = -I_{\text{app}}$$

$$\frac{\partial \Phi_s(x, t)}{\partial x} \Bigg|_{\hat{x}_p, \hat{x}_s} = 0$$

$$(C2) \left[\kappa_{\text{eff},i} \frac{\partial \Phi_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \left[\kappa_{\text{eff},i} T(x, t) \Upsilon \frac{\partial \ln c_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} =$$

$$\frac{\partial \Phi_e(x, t)}{\partial x} \Bigg|_{\hat{x}_0} = 0$$

$$\bar{\Phi}_{e,\text{end}} = 0$$

$$\Delta x_i a_i F \bar{j}_{i,k}(t)$$

$$(T) \rho_i C_{p,i} \frac{\partial \bar{T}_k(t)}{\partial t} = \frac{1}{\Delta x_i} \left[\lambda_i \frac{\partial T(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} + \bar{Q}_{\text{source},k}$$

$$\bar{j}_{i,k}(t) = 2 \kappa_{\text{eff},i} \sqrt{\bar{c}_{e,k}(t) (c_{s,i}^{\text{max}} - \bar{c}_{s,k}^*(t)) \bar{c}_{s,k}^*(t)} \sinh \left[\frac{0.5R}{FT_k(t)} \bar{\eta}_{i,k}(t) \right]$$

$$\bar{\eta}_{i,k}(t) = \bar{\Phi}_{s,k}(t) - \bar{\Phi}_{e,k}(t) - \bar{U}_{i,k}$$

Separator, $i = s$

$$(M1) \epsilon_i \frac{\partial \bar{c}_{e,k}(t)}{\partial t} = \frac{1}{\Delta x_i} \left[D_{\text{eff},i} \frac{\partial c_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}}$$

$$(C2) \left[\kappa_{\text{eff},i} \frac{\partial \Phi_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \left[\kappa_{\text{eff},i} T(x, t) \Upsilon \frac{\partial \ln c_e(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} = 0$$

$$(T) \rho_i C_{p,i} \frac{\partial \bar{T}_k(t)}{\partial t} = \frac{1}{\Delta x_i} \left[\lambda_i \frac{\partial T(x, t)}{\partial x} \right] \Bigg|_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} + \bar{Q}_{\text{ohm},k}$$

Parameter	Description	Aluminium CC	Anode	Separator	Cathode	Carbon CC
c_c^{init}	Initial concentration in the electrolyte	-	1000	1000	1000	-
$c_s^{\text{avg,init}}$	Initial solid-phase concentration	-	25,751	-	26128	-
c_s^{max}	Maximum solid-phase concentration	-	51,554	-	30555	-
D_i	Electrolyte diffusivity	-	7.5×10^{-10}	7.5×10^{-10}	7.5×10^{-10}	-
D_i^s	Solid-phase diffusivity	-	10^{-14}	-	3.9×10^{-14}	-
k_i	Reaction rate constant	-	2.334×10^{-11}	-	5.031×10^{-11}	-
l_i	Thickness	10^{-5}	8×10^{-5}	2.5×10^{-5}	8.8×10^{-5}	10^{-5}
$R_{p,i}$	Particle radius	-	2×10^{-6}	-	2×10^{-6}	-
ρ_i	Density	2700	2500	1100	2500	8940
$C_{p,i}$	Specific heat	897	700	700	700	385
λ_i	Thermal conductivity	237	2.1	0.16	1.7	401
σ_i	Solid-phase conductivity	3.55×10^7	100	-	100	5.96×10^7
ϵ_i	Porosity	-	0.385	0.724	0.485	-
a_i	Particle surface area to volume	-	885,000	-	723,600	-
$E_{a_i}^{D_i^s}$	Solid-phase diffusion activation energy	-	5000	-	5000	-
$E_{a_i}^{k_i}$	Reaction constant activation energy	-	5000	-	5000	-
brugg	Bruggeman's coefficient	-	4	4	4	-
F	Faraday's constant	-	-	-	-	-
R	Universal Gas constant	-	-	-	-	-
t_+	Transference number	-	-	-	-	-
$\epsilon_{f,i}$	Filler fraction	-	0.025	-	0.0326	-

Table 4: List of parameters used in simulation [20]

Time (s)	C rate	Description
0–50	–1 C	Moderate speed
50–60	0.5 C	Charge
60–210	–0.5C	Normal speed
210–410	–1C	Moderate speed
410–415	–2C	Overtaking
415–615	–1C	Moderate speed
615–620	0.5C	Charge

Table 5: Throttle configuration for hybrid charging-discharging simulation

	1C		2C		5C		10C	
	RMSE	NTI	RMSE	NTI	RMSE	NTI	RMSE	NTI
two-parameters	0.0822%	20.0265%	0.2535%	18.3982%	1.5849%	22.4554%	6.5403%	23.6452%
higher-order	0.0165%	37.2567%	0.0532%	44.4125%	0.3575%	60.2540%	1.8701%	62.7276%

Table 6: Comparison of different approximation methods for the diffusion in the solid particles. Root Mean Square Error (RMSE) and the Normalized Time Index (NTI) are shown.

C rate	h value	Simulation Duration	Effective Simulation Time
1C	0.01	3523 s	72 s
1C	1	3523 s	81 s
1C	100	3523 s	77 s
0.5C	1	7050 s	56 s
2C	1	1522 s	85 s

Table 7: Timing comparisons of different simulation scenarios

	# of discrete nodes				
	10	20	30	40	50
COMSOL	96 s	114 s	143 s	189 s	244 s
DUALFOIL	28 s	57 s	97 s	137 s	185 s
LIONSIMBA	28 s	69 s	105 s	134 s	223 s

Table 8: Timing comparisons among different P2D model implementations. The number of discretized nodes has been set equal for each section of the cell.

$I_{\text{app}}(t)$	Applied current density [A/m ²]
$c_e(x, t)$	Electrolyte salt concentration [mol/m ³]
$c_s^{\text{avg}}(x, t)$	Solid-phase average concentration [mol/m ³]
$c_s^*(x, t)$	Solid-phase surface concentration [mol/m ³]
$j(x, t)$	Ionic flux [mol/(m ² s)]
$\Phi_e(x, t)$	Electrolyte potential [V]
$\Phi_s(x, t)$	Solid potential [V]
$T(x, t)$	Temperature [K]
D_{eff}^s	Effective solid-phase diffusion coefficient [m ² /s]
D_{eff}	Effective electrolyte diffusion coefficient [m ² /s]
σ_{eff}	Effective solid-phase conductivity [S/m]
κ_{eff}	Effective electrolyte conductivity [S/m]
k_{eff}	Effective reaction rate
Q_{ohm}	Ohmic heat source term [W/m ³]
Q_{rev}	Reversible heat source term [W/m ³]
Q_{rxn}	Reaction heat source term [W/m ³]
U_{ref}	Open Circuit Voltage [V]
$\left. \frac{\partial U}{\partial T} \right _{T_{\text{ref}}}$	Open Circuit Potential Entropic Variation [V/K]

Table 9: Nomenclature

7 Figures captions

Figure 1: Example of a 2D FVM mesh where the set of neighbor cells $\mathcal{C}(k)$ is represented by the green cells.

Figure 2: One-dimensional finite volume mesh

Figure 3: Interpolation technique to recover edge values of the unknowns.

Figure 4: Electrolyte diffusion process: interface across the cathode and separator.

Figure 5: Validation of the proposed numerical implementation in isothermal conditions

Figure 5(a): *Cell potential*

Figure 5(b): *Electrolyte Li-ions concentration*

Figure 5(c): *Solid phase Li-ions concentration*

Figure 5(d): *Electrolyte potential*

Figure 6: Validation of the proposed numerical implementation with thermal dynamics

Figure 6(a): *Cell potential*

Figure 6(b): *Electrolyte Li-ions concentration*

Figure 6(c): *Solid phase Li-ions concentration*

Figure 6(d): *Electrolyte potential*

Figure 7: Comparison of the three different solid-phase diffusion equations implemented in LION-SIMBA.

Figure 7(a): 1C rate comparison

Figure 7(b): 2C rate comparison

Figure 7(c): 5C rate comparison

Figure 7(d): 10C rate comparison

Figure 8: $-1C$ discharge cycle run under different heat exchange parameters: blue line $h = 0.01 \text{ W}/(\text{m}^2 \text{ K})$, dashed orange line $h = 1 \text{ W}/(\text{m}^2 \text{ K})$ and dot-dashed yellow line $h = 100 \text{ W}/(\text{m}^2 \text{ K})$.

Figure 9: Full discharge cycle run under different C rates: $-2C$ (dot-dashed yellow), $-1C$ (dashed orange line), and $-0.5C$ (blue line).

Figure 10: Hybrid charging-discharging cycle.

Figure 11: *ABMS control*: an MPC algorithm [8] is used to drive the charge of the battery from 20% to 85% while considering voltage, temperature, and current constraints.

Figure 12: *ABMS control - Electrolyte concentration*: The behavior of the first and last volume

of each section of the battery is depicted, where the continuous lines belong to the cathode, the dashed lines to the separator and the dotted lines to the anode.

Figure 13: Full discharge cycle in an isothermal environment: blue line $-0.5C$, dashed orange line $-1C$, and dot-dashed yellow line $-2C$.

Figure 14: Simulation of a 3-cells pack. The upper curve represents the overall voltage of the 3 series connected Li-ion cells, while the lower plots depict the voltage of each cell in the pack. The different parametrization of each cell determines different behaviors.

Figure 15: Simulation of a 3-cells pack. Different internal states profiles inside the three cells. Individual parametrizations leads to different behaviors.

8 Tables captions

Table 1: Li-ion P2D model governing equations

Table 2: Additional equations

Table 3: FVM P2D equations

Table 4: List of parameters used in simulation [20]

Table 5: Throttle configuration for hybrid charging-discharging simulation

Table 6: Comparison of different approximation methods for the diffusion in the solid particles. Root Mean Square Error (RMSE) and the Normalized Time Index (NTI) are shown.

Table 7: Timing comparisons of different simulation scenarios

Table 8: Timing comparisons among different P2D model implementations. The number of discretized nodes has been set equal for each section of the cell.

Table 9: Nomenclature

References

- [1] S. K. Dhar, S. R. Ovshinsky, P. R. Gifford, D. A. Corrigan, M. A. Fetcenko, and S. Venkatesan, "Nickel/metal hydride technology for consumer and electric vehicle batteries – A review and update," *Journal of Power Sources*, vol. 65, no. 1, pp. 1–7, 1997.
- [2] D. Linden, *Handbook of Batteries and Fuel Cells*. McGraw-Hill Book Company, New York, NY, USA, 1984.
- [3] P. Ruetschi, "Review on the lead-acid battery science and technology," *Journal of Power Sources*, vol. 2, no. 1, pp. 3–120, 1977.
- [4] W. Zhang, "A review of the electrochemical performance of alloy anodes for lithium-ion batteries," *Journal of Power Sources*, vol. 196, no. 1, pp. 13–24, 2011.
- [5] P. Van den Bossche, F. Vergels, J. Van Mierlo, J. Matheys, and W. Van Autenboer, "SUBAT: An assessment of sustainable battery technology," *Journal of Power Sources*, vol. 162, no. 2, pp. 913–919, 2006.
- [6] S. Santhanagopalan, Q. Guo, P. Ramadass, and R. E. White, "Review of models for predicting the cycling performance of lithium ion batteries," *Journal of Power Sources*, vol. 156, no. 2, pp. 620–628, 2006.
- [7] V. Ramadesigan, P. W. C. Northrop, S. De, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian, "Modeling and simulation of lithium-ion batteries from a systems engineering perspective," *Journal of The Electrochemical Society*, vol. 159, no. 3, pp. R31–R45, 2012.
- [8] M. Torchio, N. A. Wolff, D. M. Raimondo, L. Magni, U. Kreuer, R. B. Gopaluni, J. A. Paulson, and R. D. Braatz, "Real-time model predictive control for the optimal charging of a lithium-ion battery," in *Proceedings of the American Control Conference*, 2015, pp. 4536–4541.
- [9] V. H. Johnson, A. A. Pesaran, T. Sack, and S. America, *Temperature-dependent battery models for high-power lithium-ion batteries*. National Renewable Energy Laboratory City of Golden, 2001.
- [10] B. Y. Liaw, G. Nagasubramanian, R. G. Jungst, and D. H. Doughty, "Modeling of lithium ion cells – A simple equivalent-circuit model approach," *Solid State Ionics*, vol. 175, no. 1, pp. 835–839, 2004.
- [11] H. He, R. Xiong, and J. Fan, "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach," *Energies*, vol. 4, no. 4, pp. 582–598, 2011.
- [12] X. Hu, S. Li, and H. Peng, "A comparative study of equivalent circuit models for Li-ion batteries," *Journal of Power Sources*, vol. 198, pp. 359–367, 2012.
- [13] S. K. Rahimian, S. Rayman, and R. E. White, "Comparison of single particle and equivalent circuit analog models for a lithium-ion cell," *Journal of Power Sources*, vol. 196, no. 20, pp. 8450–8462, 2011.
- [14] C. M. Doyle, "Design and Simulation of Lithium Rechargeable Batteries," Ph.D. dissertation, University of California, Berkeley, August 1995.
- [15] R. B. Gopaluni and R. D. Braatz, "State of charge estimation in Li-ion batteries using an isothermal pseudo two-dimensional model," in *Proceedings of the 10th IFAC International Symposium on Dynamics and Control of Process Systems*, 2013, pp. 135–140.

- [16] K. L. Man, K. Wan, T. O. Ting, C. Chen, T. Krilavicius, J. Chang, and S. H. Poon, "Towards a hybrid approach to SOC estimation for a smart battery management system (BMS) and battery supported cyber-physical systems (CPS)," in *Proceedings of the 2nd Baltic Congress on Future Internet Communications*, 2012, pp. 113–116.
- [17] J. Remmlinger, M. Buchholz, M. Meiler, P. Bernreuter, and K. Dietmayer, "State-of-health monitoring of lithium-ion batteries in electric vehicles by on-board internal resistance estimation," *Journal of Power Sources*, vol. 196, no. 12, pp. 5357–5363, 2011.
- [18] Z. Guo, X. Qiu, G. Hou, B. Y. Liaw, and C. Zhang, "State of health estimation for lithium ion batteries based on charging curves," *Journal of Power Sources*, vol. 249, pp. 457–462, 2014.
- [19] Y.-H. Chiang, W.-Y. Sean, and J.-C. Ke, "Online estimation of internal resistance and open-circuit voltage of lithium-ion batteries in electric vehicles," *Journal of Power Sources*, vol. 196, no. 8, pp. 3921–3932, 2011.
- [20] P. W. C. Northrop, V. Ramadesigan, S. De, and V. R. Subramanian, "Coordinate transformation, orthogonal collocation, model reformulation and simulation of electrochemical-thermal behavior of lithium-ion battery stacks," *Journal of The Electrochemical Society*, vol. 158, no. 12, pp. A1461–A1477, 2011.
- [21] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.
- [22] L. Cai and R. E. White, "Mathematical modeling of a lithium ion battery with thermal effects in COMSOL Inc. Multiphysics (MP) software," *Journal of Power Sources*, vol. 196, no. 14, pp. 5985–5989, 2011.
- [23] FORTRAN Programs for the Simulation of Electrochemical Systems. [Online]. Available: <http://www.cchem.berkeley.edu/jsngrp/fortran.html>
- [24] V. R. Subramanian, V. D. Diwakar, and D. Tapriyal, "Efficient macro-micro scale coupled modeling of batteries," *Journal of The Electrochemical Society*, vol. 152, no. 10, pp. A2002–A2008, 2005.
- [25] V. Ramadesigan, V. Boovaragavan, J. C. Pirkle, and V. R. Subramanian, "Efficient reformulation of solid-phase diffusion in physics-based lithium-ion battery models," *Journal of The Electrochemical Society*, vol. 157, no. 7, pp. A854–A860, 2010.
- [26] Q. Zhang and R. E. White, "Comparison of approximate solution methods for the solid phase diffusion equation in a porous electrode model," *Journal of Power Sources*, vol. 165, no. 2, pp. 880–886, 2007.
- [27] C. Wang, W. Gu, and B. Liaw, "Micro-macroscopic coupled modeling of batteries and fuel cells i. model development," *Journal of the Electrochemical Society*, vol. 145, no. 10, pp. 3407–3417, 1998.
- [28] S. Liu, "An analytical solution to Li/Li+ insertion into a porous electrode," *Solid State Ionics*, vol. 177, no. 1, pp. 53–58, 2006.
- [29] L. O. Valøen and J. N. Reimers, "Transport properties of LiPF₆-based li-ion battery electrolytes," *Journal of The Electrochemical Society*, vol. 152, no. 5, pp. A882–A891, 2005.
- [30] D. Bernardi, E. Pawlikowski, and J. Newman, "A general energy balance for battery systems," *Journal of the Electrochemical Society*, vol. 132, no. 1, pp. 5–12, 1985.

- [31] T. M. Bandhauer, S. Garimella, and T. F. Fuller, “A critical review of thermal issues in lithium-ion batteries,” *Journal of The Electrochemical Society*, vol. 158, no. 3, pp. R1–R25, 2011.
- [32] K. Kumaresan, G. Sikha, and R. E. White, “Thermal model for a li-ion cell,” *Journal of the Electrochemical Society*, vol. 155, no. 2, pp. A164–A171, 2008.
- [33] K. Smith and C.-Y. Wang, “Solid-state diffusion limitations on pulse operation of a lithium ion cell for hybrid electric vehicles,” *Journal of Power Sources*, vol. 161, no. 1, pp. 628–639, 2006.
- [34] D. M. Bernardi and J.-Y. Go, “Analysis of pulse and relaxation behavior in lithium-ion batteries,” *Journal of Power Sources*, vol. 196, no. 1, pp. 412–427, 2011.
- [35] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*. Springer-Verlag New York, 2013, vol. 12.
- [36] W. E. Schiesser, *The Numerical Method of Lines*. Academic Press, San Diego, 1991.
- [37] J. Chai and S. Patankar, “Finite-volume method for radiation heat transfer,” *Advances in Numerical Heat Transfer*, vol. 2, pp. 109–141, 2000.
- [38] P. Jenny, S. H. Lee, and H. A. Tchelepi, “Adaptive multiscale finite-volume method for multiphase flow and transport in porous media,” *Multiscale Modeling & Simulation*, vol. 3, no. 1, pp. 50–64, 2005.
- [39] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002, vol. 31.
- [40] J. ten Thije Boonkkamp and M. Anthonissen, “The finite volume-complete flux scheme for advection-diffusion-reaction equations,” *Journal of Scientific Computing*, vol. 46, no. 1, pp. 47–70, 2011.
- [41] S. Patankar, *Numerical Heat Transfer and Fluid Flow*. CRC Press, 1980.
- [42] M. Tiller, *Introduction to Physical Modeling with Modelica*. Springer Science & Business Media, 2012.
- [43] S. W. Moore and P. J. Schneider, “A review of cell equalization methods for lithium ion and lithium polymer battery systems,” SAE Technical Paper, Tech. Rep., 2001.
- [44] W. Bentley, “Cell balancing considerations for lithium-ion battery systems,” in *Battery Conference on Applications and Advances, 1997., Twelfth Annual*. IEEE, 1997, pp. 223–226.
- [45] L. Rao and J. Newman, “Heat-generation rate and general energy balance for insertion battery systems,” *Journal of the Electrochemical Society*, vol. 144, no. 8, pp. 2697–2704, 1997.