batteries

Article

Comparative Analysis of Computational Times of Lithium-Ion Battery Management Solvers and Battery Models Under Different Programming Languages and Computing Architectures

Moin Ahmed [1], Zhiyu Mao [2,3,*], Yunpeng Liu [2,3], Aiping Yu [1], Michael Fowler [1] and Zhongwei Chen [2,3,*]

[1] Department of Chemical Engineering, University of Waterloo, Waterloo, ON N2L 3W3, Canada; m63ahmed@uwaterloo.ca (M.A.); aipingyu@uwaterloo.ca (A.Y.); mfowler@uwaterloo.ca (M.F.)
[2] Power Battery & System Research Center, Dalian Institute of Chemical Physics, Chinese Academy of Sciences, Dalian 116023, China; liuyunpeng1994@dicp.ac.cn
[3] State Key Laboratory of Catalysis, Dalian Institute of Chemical Physics, Chinese Academy of Sciences, Dalian 116023, China
* Correspondence: zhymao@dicp.ac.cn (Z.M.); zwchen@dicp.ac.cn (Z.C.)

Abstract: With the global rise in consumer electronics, electric vehicles, and renewable energy, the demand for lithium-ion batteries (LIBs) is expected to grow. LIBs present a significant challenge for state estimations due to their complex non-linear electrochemical behavior. Currently, commercial battery management systems (BMSs) commonly use easier-to-implement and faster equivalent circuit models (ECMs) than their counterpart continuum-scale physics-based models (PBMs). However, despite processing more mathematical and computational complexity, PBMs are attractive due to their higher accuracy, higher fidelity, and ease of integration with thermal and degradation models. Various reduced-order PBM battery models and their computationally efficient numerical schemes have been proposed in the literature. However, there is limited data on the performance and feasibility of these models in practical embedded and cloud systems using standard programming languages. This study compares the computational performance of a single particle model (SPM), an enhanced single particle model (ESPM), and a reduced-order pseudo-two-dimensional (ROM-P2D) model under various battery cycles on embedded and cloud systems using Python and C++. The results show that reduced-order solvers can achieve a 100-fold reduction in solution times compared to full-order models, while ESPM with electrolyte dynamics is about 1.5 times slower than SPM. Adding thermal models and Kalman filters increases solution times by approximately 20% and 100%, respectively. C++ provides at least a 10-fold speed increase over Python, varying by cycle steps. Although embedded systems take longer than cloud and personal computers, they can still run reduced-order models effectively in Python, making them suitable for embedded applications.

Keywords: battery management systems; lithium-ion battery continuum-scale models; battery packs

Citation: Ahmed, M.; Mao, Z.; Liu, Y.; Yu, A.; Fowler, M.; Chen, Z. Comparative Analysis of Computational Times of Lithium-Ion Battery Management Solvers and Battery Models Under Different Programming Languages and Computing Architectures. *Batteries* **2024**, *10*, 439. https://doi.org/10.3390/batteries10120439

Academic Editor: Pascal Venet

Received: 21 September 2024
Revised: 15 November 2024
Accepted: 26 November 2024
Published: 11 December 2024

1. Introduction

At present, battery cells comprising lithium-ion batteries (LIBs) are primarily used in the battery packs of consumer electronics, electrified vehicles, and renewable energy generation plants [1–3]. LIBs chemistries, containing a lithium transition metal oxide positive electrode and graphite negative electrode, offer excellent cycling life, a high specific energy density, low memory effects, and a low self-discharge rate [4–6]. Inside the LIB-based battery pack, the cells are electrically connected in series and parallel to give the desired pack voltage and capacity (Figure 1A). The battery pack is connected to a battery management system (BMS), which serves various functions (Figure 1A). These functions include, but are not limited to (1) sensing for voltage, current, and temperature, (2) protecting against extreme conditions such as excessive currents, under and high voltage limits, etc., (3) interfacing with the user on helpful information such as charge control

and range estimation and (4) battery state estimation for performance management and diagnostics [7,8]. Meanwhile, the battery cell within the battery pack can come in various form factors (Figure 1B). Various battery models, including the cell-scale equivalent-circuit model (ECM) and the continuum-scale physics-based model (PBM), are suitable for the battery cell state and health estimations of BMS applications [9–16]. From the mentioned models, ECMs are still standard in industrial BMS applications, majorly due to their ease of implementation and mathematical simplicity [17,18].

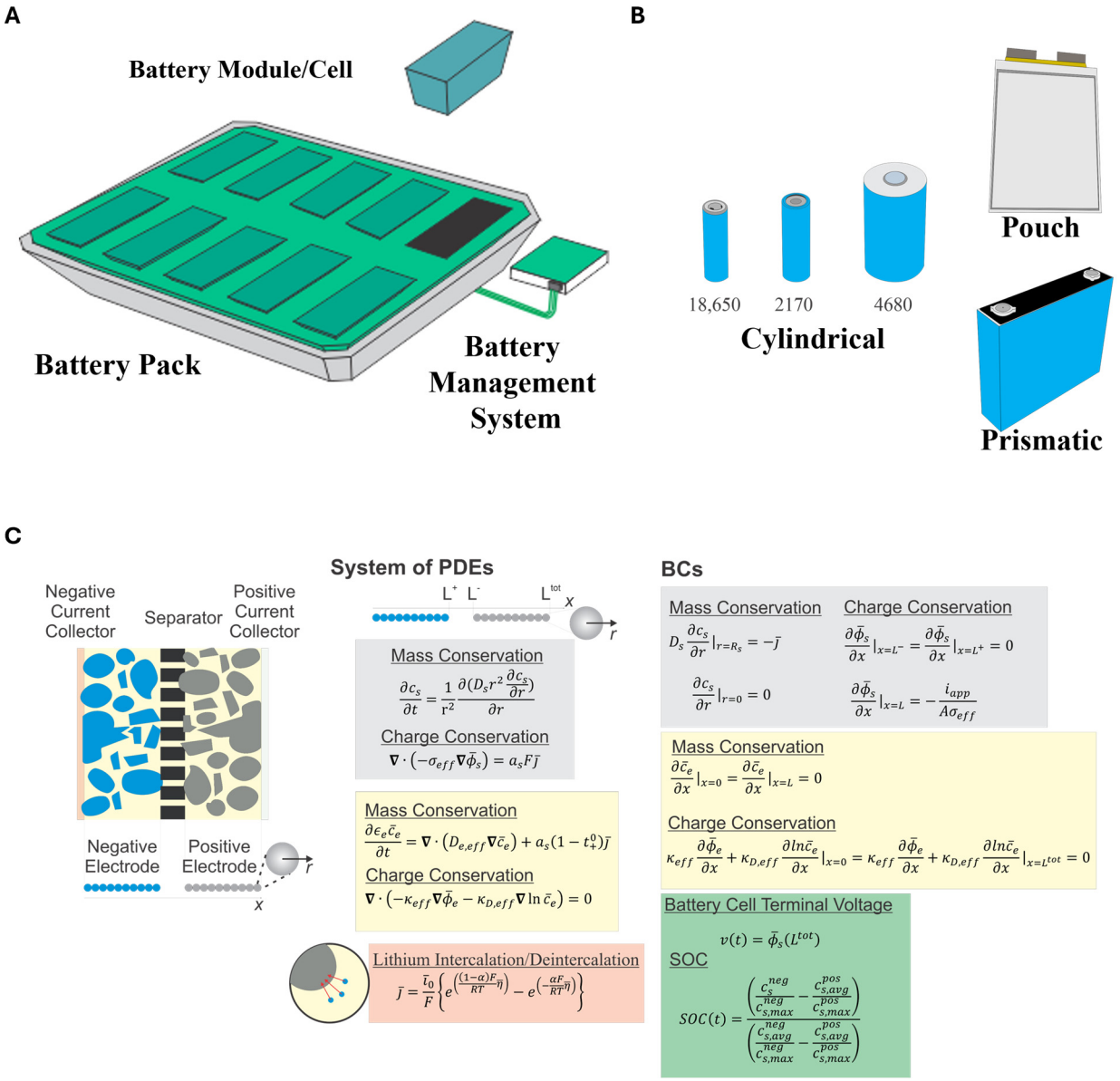

Figure 1. (**A**) Battery pack, comprising various battery cells, connected to a battery management system. (**B**) Various form factors of commercial lithium-ion batteries used in battery packs. (**C**) Coordinate systems, a system of PDEs, and their respective boundary conditions.

Both ECM and continuum-scale PBM estimate the extent of various battery cell polarizations during its operations. ECMs are considered phenomenological models since they use electric circuits to model the observable battery cell polarizations, such as polarizations that stem from the battery cell internal resistance, diffusional voltages, hysteresis effects, etc., and the model parameters are chosen to fit the experimental data [19]. PBMs, in particular pseudo-two-dimensional (P2D), on the other hand, use the concepts of mass transport

(in the solid electrode region and the liquid electrolyte region), charge conservation (in the solid electrode region and the liquid electrolyte region), and charge–transfer reactions (electrode–electrolyte interface) to model various battery cell polarizations [20,21]. The model equations in the P2D model were introduced by researchers Doyle, Newman, and Fuller in 1994 and are referred to as the DFN model [21]. Hence, the PBM parameters are based on the battery material's electrochemical, physical, and mechanical properties. PBMs are considered more accurate in predicting battery states than the ECM, and battery cell degradation models are more accessible to integrate with PBMs than ECMs [13,17,22]. Moreover, the solution of relevant PBM equations leads to the spatial and temporal distribution of lithium-ion concentrations and potentials across the battery cell as a lithium-ion flux at the electrode–electrolyte interface that cannot be obtained from traditional ECMs. This added information allows for the estimation of thermal evolution from various sources across the battery cell (e.g., thermal evolution from Joule heating due to lithium diffusivity in the electrolyte and electrode regions, etc.) and a thermal model across the battery cell. An accurate thermal evolution estimation is an important safety and performance feature for the thermal management of battery packs [23–25]. However, PBMs are mathematically more involved since they generally include a set of coupled partial differential equations (PDEs) (Figure 1C) [17,20,21,26,27]. These sets of coupled PDEs contain a more significant number of model parameters and require elaborate numerical solution schemes (compared to ECMs). An extensive set of equations are formed due to numerical solution schemes that must be solved at each time step. Consequently, compared to ECMs, PBMs usually have a higher computational demand and require a longer solution time. Moreover, implementing the Kalman filter for practical BMS applications is more cumbersome than that needed for ECMs due to the PBM's increased mathematical complexity.

In the literature, two main strategies have been employed to simplify the P2D's implementation and/or reduce its solution time for BMS applications, namely the (1) simplification of model equations [26,28] and the (2) improvement of the solution schemes [17,27]. Guo et al. introduced the single particle model (SPM) to calculate the lithium-ion battery cell's terminal voltage. The researchers assumed a constant lithium-ion flux throughout the positive and negative electrode regions during the battery cell operations, negligible battery cell polarization contributions from the electrolyte dynamics, and continuous electrolyte solution resistance. Also, the researchers assumed that the lithium concentration profile in each of the electrodes could be approximated from that of a single spherical particle for that electrode [26]. SPM is in good agreement with the experimental data for low charge/discharge rates (approximately < 0.5C) when the polarizations from the electrolyte dynamics are very low [29]. Later, Moura et al. extended the SPM with electrolyte dynamics (ESPM) for higher charge/discharge rates without resorting to the P2D model [28]. The ESPM demonstrated a stronger rapport with the experimental data for up to 2C [28]. On another front, a set of resultant equations from the numerical solution scheme of a coupled set of PDEs can be solved iteratively until convergence or simultaneously [17,27]. Han et al. proposed a finite volume method (FVM)-based numerically efficient algorithm for solving the P2D equation, whereby the authors iteratively solved a few equations (specifically the equations about the electrode and electrolyte potentials) and checked for their convergence from the results of another electrochemical model. The remaining equations were solved once the first converged, ensuring a faster solution time [17].

Specific to the BMS applications, ECM and PBM equations were utilized for the battery cell state-of-charge (SOC) and cell terminal voltage. It is worthwhile to note that typically, in the PBM schemes mentioned above, a 1D cartesian coordinate system is used for the spatial distribution of the lithium-ion concentrations in the liquid electrolyte region and the potential profile across the liquid electrolyte and solid electrode regions across the thickness of a battery cell. Meanwhile, the spatial distribution of the lithium-ion across a solid electrode region uses a 1D spherical coordinate system. Hence, for large format battery cells (e.g., the battery cell form factors include cylindrical, pouch, and prismatic battery cell formats), the PBMing approach alone does not include the cell-level effects of

the cell-level parameters (such as the electrode dimensions, dimensions of the electrical contact tabs protruding from the electrode, 3D placements of the cooling system to the electrodes, etc.) on the macroscopic processes (such as the electric conductivity and thermal evolution) [30]. The multi-scale multi-domain (MSMD) modeling framework, as introduced by National Renewable Energy Labs, tries to address this limitation in a computationally efficient manner [31]. In MSMD, different length scales from the cell level to macroscopic levels are separated into separate domains with separate relevant models and coordinate systems. For instance, there can be domains for an electrode particle, the entire electrode, and the battery cell. Relevant variables are solved in their respective domain and these solved variables (after relevant area and volume averaging) are passed onto the next domain. As an example, the lithium-ion flux during the charge transfer at the electrode particle–electrolyte interface is calculated at the particle domain and then transferred to the electrode domain after averaging over the electrode–electrolyte interfacial area. This areal flux is transferred to the cell domain after volume averaging over the specific electrode plate area. Furthermore, MSMD allows flexibility in the selection of the model and solver schemes for each of the domains [31]. For instance, the versatility of the model selection in MSMD was leveraged by Margi et al. where computational fluid dynamics were used for the determination of the spatial thermal distribution across a prismatic battery cell [32]. Due to its ability to combine different length scales, MSDS has been studied in the literature to understand the effects of cell-level parameters on the inhomogeneous thermal evolution and electrode state-of-charge across the electrode [24,30,31,33]. Kim et al. utilized MSDS to study the effects of the different stacked cell designs on the current density, SOC, and thermal evolution across the electrodes. Interestingly, insignificant variations in the cell terminal voltage were observed, though there were inhomogeneities in the cell kinetics, charging, and thermal evolution across the electrodes [31]. Moreover, by leveraging the thermal models at different scales, researchers have evaluated different battery thermal management systems using MSDS [24,30,33]. In their approach, Schmidt et al. extended the P2D model by incorporating the model that includes the particle size distribution across the electrode [30]. Kim et al. added a 3D model in the cell domain and demonstrated a 3D profile of the current density, and thermal evolution across the electrodes [31]. While the MSDS framework adds model accuracy to cell terminal voltage and SOC, it requires higher computation resources than the PBM, particularly from the inclusion of the model equations at the cell domain. The solution of this model's equation requires the creation of a two- or three-dimensional mesh and solution at each spatial node for every time step [30,34]. Various reduced-order methodologies have been implemented with MSDS which have resulted in reduced solution times [31].

This work compared various solvers and battery models based on their solution times and computational requirements on different architectures and programming languages. This research novelty is outlined below. The practical industrial application of PBMs in BMSs (including PBM application for the battery cell/pack SOC estimations in the BMS) requires the quantification of the PBM computational load in the current BMS hardware. The performance and computing speed of various numerical solvers for solving the PDE for the lithium-ion concentration in the solid electrode and liquid electrolyte region are analyzed. These solvers have been programmed in the C++ and Python programming languages and tested on various computing hardware whose computing specifications represent personal computing, cloud, and embedded systems. For instance, the embedded system, used in this work, uses a 1.2 GHz Broadcom BCM2837 processor with 1 GB RAM. C++ and Python are the programming languages that are used for programming microcontrollers and microprocessors (microcontrollers and microprocessors are used for the embedded systems including BMS) [35–37]. Furthermore, the performance, in terms of the solution times, of various reduced-order PBMs for the battery cells was compared using different programming languages and computing hardware. These battery models include the SPM, ESPM, and reduced-order P2D models. As mentioned above, the single particle model is suitable for battery cell applications where the charge/discharge currents

are ideally below 0.5 C. At the same time, the ESPM can be used for applications that require currents up to 2C. The electrode solvers and battery models are simulated under various cycling steps, including charge, discharge, discharge–rest, charge–rest, and hybrid pulse power characterization (HPPC). The physical and electrochemical model parameters used in this work have been borrowed from other literature works [26,27]. The physical and electrochemical model parameters are representative of an instance of a prismatic lithium-ion battery cell that comprises lithium cobalt oxide (LCO) chemistry.

In terms of the structure of this article, the subsequent Section 2 overviews the various LIB battery solver and model equations, the architecture of the applied codebase, and the specifications of the utilized hardware. The following Section 3 presents the results. Finally, this work's summary and its significant findings are summarized in Section 4.

2. Methodology

2.1. Electrode Concentration Solver

Lithium diffusion in the solid electrode region is assumed to be driven by the concentration gradient. It is given by the following PDE and boundary conditions.

$$\frac{\partial c_{s,j}}{\partial t} = \frac{1}{r^2}\frac{\partial\left(D_{s,j}r^2\left(\frac{\partial c_{s,j}}{\partial r}\right)\right)}{\partial r} \tag{1}$$

$$\frac{\partial c_s}{\partial r}\,|_{r=0} = 0 \tag{2}$$

$$D_s\frac{\partial c_s}{\partial r}\,|_{r=R_s} = -j_s \tag{3}$$

In the above equations, $c_{s,j}$ [mol/s^2] refers to the lithium-ion concentration in the solid electrode particle for the electrode k (k refers to either the positive or negative electrodes). Likewise, $D_{s,k}$ [m^2/s] represents the diffusivity of lithium-ions in the solid electrode particles.

The initial condition, defined below, is known by the user.

$$c_{s,j}(r,\ t=0) = c_{s,j}^{init} \tag{4}$$

$c_{s,j}^{init}$ represents the initial lithium-ion concentration in the solid electrode particles.

Moreover, the scaled version of the above PDE and the scale boundary conditions are also presented below for convenience. The scaled concentration, x_j, and the radius variables, R_j, are obtained by scaling them concerning the maximum concentration, $c_{s,j}^{max}$, and the electrode radius, R_j, respectively (i.e., $x_j = \frac{c_{s,j}}{c_{s,j}^{max}}$ and $\overline{r}_j = r_j/R_j$).

$$\frac{\partial x_j}{\partial t} = \frac{D_{s,j}}{R_j^2}\frac{1}{\overline{r}_j^2}\frac{\partial}{\partial \overline{r}_j}\left(\overline{r}_j^2\frac{\partial x_j}{\partial \overline{r}_j}\right)$$

$$x_j(\overline{r},\ t=0) = x_j^{init}$$

$$\frac{\partial x_j}{\partial r}\,|_{r=0} = 0$$

$$\frac{\partial x_j}{\partial r}\,|_{r=1} = \frac{J_j R_j}{c_{s,max}D_{s,j}(t)} = \delta_j(t)$$

The above PDE (Equation (1)) was solved numerically using the implicit Crank–Nicolson method, eigen-expansion method, and reduced-order polynomial formulation (both two-order polynomials and higher order polynomials), and relevant solver equations are summarized in the following subsections.

2.1.1. Crank–Nicolson Approximation

This section mentions the relevant equations in the use of an implicit numerical scheme for lithium concentration PDE (Equation (1)), the Crank–Nicolson scheme, which has a second-order accuracy in time [38]. Here, the backward time difference is applied to the

time derivative of the diffusion PDE, while the average of the centered–finite difference is applied to the spatial derivative. The following scheme is obtained after temporal and spatial discretization. Note that the spatial and temporal points are denoted by i and k, respectively, where $i = \{1, 2, \cdots, N\}$ and $k = \{1, 2, \cdots, K\}$.

The above equations for each spatial nodal point can be described in a matrix form as follows [39]. For the derivation of the equation below, refer to Section S1 in the Supporting Information.

$$\begin{bmatrix} 1+3A & -3A & 0 & \cdots & 0 \\ & \ddots & & & \\ & -\frac{A}{2}+\frac{B}{r_i} & 1+A & \frac{A}{2}+\frac{B}{r_i} & \\ & & & \ddots & \\ 0 & 0 & \cdots & -A & 1+A \end{bmatrix} \begin{bmatrix} c_1^{k+1} \\ \vdots \\ c_N^{k+1} \end{bmatrix} = \begin{bmatrix} 1+3A & -3A & & \cdots & 0 \\ & \ddots & & & \\ & \frac{A}{2}-\frac{B}{r_i} & 1+A & \frac{A}{2}+\frac{B}{r_i} & \\ & & & \ddots & \\ 0 & & \cdots & -A & 1+A \end{bmatrix} \begin{bmatrix} c_1^{k} \\ \vdots \\ c_N^{k} \end{bmatrix} - \begin{bmatrix} 0 \\ \vdots \\ \left(A+\frac{B}{R}\right)\left(\frac{2\Delta r J_j}{D_{s,j}}\right) \end{bmatrix} \tag{5}$$

The above expression can be solved by matrix multiplication with the inverse of the matrix in the RHS. Also, this matrix contains a tridiagonal matrix, which can be solved using Thomas, also known as the tridiagonal matrix, in the case of the diagonally dominant matrix [40].

2.1.2. Eigen-Expansion Method

Guo et al. utilized the eigen-expansion method to solve the PDE about lithium concentration in the solid electrode region profile (Equation (1)), which resulted in a set of algebraic and ordinary differential (ODE) equations [26]. The authors obtained the following expressions when the boundary conditions were made homogeneous, and the resultant variables were solved.

$$\frac{du_{k,j}(t)}{dt} = -\frac{\lambda_k^2 D_{s,p}(t)}{R_p^2} u_{k,p}(t) + \frac{2D_{s,p}(t)}{R_p^2}\delta_p(t)$$

$$u(0) = 0$$

$$x_j^{surf} = x_{ini,j} + \frac{1}{5}\delta_j(t) + 3\int_0^t \frac{D_{s,j}(\tau)}{R_j^2}\delta_j(\tau)dt + \sum_{k=1}^{N}\left[u_{j,k} - \frac{2\delta_j(t)}{\lambda_k^2}\right]$$

where δ_j represents the scaled lithium-ion flux.

2.1.3. Reduced-Order Polynomial Approximation

A volume-averaged quantity after the application of volume averaging over a spherical coordinate was given by [41]

$$\overline{f} = \frac{3}{r^3}\int_{r=0}^{r_k} r^2 f(r)dr$$

After applying the volume averaging quantity to the LHS and the RHS of Equation (1) above, the following ordinary ODE was obtained:

$$\frac{\partial \overline{c}_{s,k}}{\partial t} = -3\frac{j_k}{R_k} \tag{6}$$

with the initial condition

$$c_s(r, t = 0) = c_s^0$$

Furthermore, the lithium-ion concentration profile across the radial coordinates was approximated using a polynomial expression [41,42]:

- Two-order polynomial

$$\frac{\partial \bar{c}_{s,k}}{\partial t} = -3\frac{j_k}{R_k}$$
$$c_s^{surf} = \bar{c}_{s,k} - \frac{R_k}{D_s}\frac{j_k}{5}$$

- Higher-order polynomial

$$\frac{\partial \bar{c}_{s,k}}{\partial t} = -3\frac{j_k}{R_k}$$
$$\frac{d\bar{c}_{skr}}{dt} + \frac{30D_s}{R_s^2}\bar{c}_{skr} = -\frac{45}{2R_k^2}j_k$$
$$c_s^{surf} = \bar{c}_{s,k} - \frac{R_k}{35D_s}j_k + \frac{8R_k}{35}\bar{c}_{skr}$$

2.2. Electrolyte Concentration Solver

The mass transport of the lithium-ion within the electrolyte region across the battery cell is assumed to be driven by the concentration gradient and migration: the continuum-scale PDE and boundary and initial conditions are given below [17,20,21].

$$\frac{\partial \epsilon_e c_e}{\partial t} = \nabla\cdot(\mathrm{D_e}\nabla \mathrm{c_e}) + \mathrm{a_s}\left(1 - t_+^0\right)j \tag{7}$$

$$\frac{\partial c_e}{\partial x}\,|_{x=0} = \frac{\partial c_e}{\partial x}\,|_{x=L^{tot}} = 0 \tag{8}$$

$$c\,|_{t=0} = c_{e,init} \tag{9}$$

In the above equations, c_e and $c_{e,\,init}$ refers to the lithium-ion concentration and the initial lithium-ion concentration, respectively (with the units of mol/m^3).

2.2.1. Finite Element Formulation

Under a constant electrolyte volume fraction, the volume integral formulation for the above PDEs is given below [17,42].

$$\int \frac{\partial \epsilon_e c_e}{\partial t}dx = \int \frac{\partial}{\partial x}\left(D_e^{eff}\frac{\partial c_e}{\partial x}\right)dx + \int \frac{1 - t_+^0}{F}dx$$

The matrix representation after the discretization of the above integral form and the consideration of the boundary conditions is given below [17].

$$\begin{bmatrix} \epsilon_{e,1} + \frac{\Delta t}{2\Delta x_1}\left(\frac{D_{e,1}^{eff}+D_{e,2}^{eff}}{x_2-x_1}\right) & -\frac{\Delta t}{2\Delta x_1}\left(\frac{D_{e,1}^{eff}+D_{e,2}^{eff}}{x_2-x_1}\right) & 0 & \cdots & 0 \\ \frac{\Delta t}{2\Delta x_2}\left(\frac{D_{e,1}^{eff}+D_{e,2}^{eff}}{x_2-x_1}\right) & \epsilon_{e,2} + \frac{\Delta t}{2\Delta x_2}\left(\frac{D_{e,1}^{eff}+D_{e,2}^{eff}}{x_2-x_1} + \frac{D_{e,2}^{eff}+D_{e,3}^{eff}}{x_3-x_2}\right) & -\frac{\Delta t}{2\Delta x_2}\left(\frac{D_{e,2}^{eff}+D_{e,3}^{eff}}{x_3-x_2}\right) & & \\ & & & \ddots & \\ 0 & 0 & \cdots & & \end{bmatrix} \begin{bmatrix} c_1^{k+1} \\ \vdots \\ c_N^{k+1} \end{bmatrix}$$
$$= \begin{bmatrix} \epsilon_{e,1}c_1^k \\ \vdots \\ \epsilon_{e,N}c_N^k \end{bmatrix} + \frac{\Delta t\left(1-t_0^+\right)}{F}\boldsymbol{J_e} \tag{10}$$

2.2.2. Reduced-Order Polynomial Approximation

Similarly to the methodology employed for the polynomial approximation of the lithium concentrations in the solid electrode phase, approximations on the electrolyte spatial concentration distribution could be applied to the volume-averaged version of relevant PDE, boundary conditions, and initial conditions (Equations (7)–(10)) [41]. Once the resultant set of equations was rearranged and decoupled, we obtained the following equations for the electrolyte concentration distribution across the battery cell components [41].

$$c_{e,n}(x,t) = c_{lin} + \frac{q_{lin}(t)}{2L_n D_{e,n}}\left(L_n^2 - x^2\right) \tag{11}$$

$$c_{e,s}(x,t) = c_{lin} - \frac{q_{lin}(t)}{D_{e,s}}(x - L_n) + \left(\frac{q_{lin}(t) - q_{lip}(t)}{L_s D_{e,s}}\right)\frac{(x - L_n)^2}{2} \tag{12}$$

$$c_{e,p}(x,t) = c_{lip} + \frac{q_{lin}(t)}{2L_p D_{e,p}}\left(L_p^2 - \left(L_{cell} - x^2\right)\right) \tag{13}$$

The above equations relied on the time-variant quantities, c_{lip}, c_{lin}, q_{lip}, and q_{lin}. The algebraic expressions and ODEs representing these quantities are expressed below [41].

$$c_{lin} = c_{lip} + \frac{L_s\left(q_{lin} + q_{lip}\right)}{2D_{e,s}} \tag{14}$$

$$c_{lip} = c_{e,init} + \alpha_{in} q_{lin} + \alpha_{ip} q_{lip} \tag{15}$$

$$\frac{dq_{lin}}{dt} = \frac{1}{\mathcal{D}}\left(-B_2 q_{lin} - A_2 q_{lip} + A_3 B_2 j_n - A_2 B_3 j_p\right) \tag{16}$$

$$\frac{dq_{lip}}{dt} = \frac{1}{\mathcal{D}}\left(B_1 q_{lin} - A_1 q_{lip} + A_3 B_1 j_n - A_1 B_3 j_p\right) \tag{17}$$

Moreover, the intermediate variables and constants used in the above equations are expressed below [41].

$$\alpha_{in} = \frac{\frac{L_n L_s \epsilon_n}{2D_{e,s}} + \frac{L_s^2 \epsilon_s}{6D_{e,s}} + \frac{L_n^2 \epsilon_n}{3D_{e,n}}}{L_n\epsilon_n + L_s\epsilon_s + L_p\epsilon_p}$$

$$\alpha_{ip} = \frac{\frac{L_p L_s \epsilon_p}{2D_{e,s}} + \frac{L_s^2 \epsilon_s}{6D_{e,s}} + \frac{L_p^2 \epsilon_p}{3D_{e,n}}}{L_n\epsilon_n + L_s\epsilon_s + L_p\epsilon_p}$$

$$A_1 = L_n\epsilon_n\alpha_n + \frac{L_s L_n \epsilon_n}{2D_{e,s}} + \frac{L_n^2 \epsilon_{e,n}}{3D_{e,n}}$$

$$A_2 = L_P\epsilon_{e,p}\alpha_{in} - \frac{L_p^2 \epsilon_{e,p}}{3D_{e,p}}$$

$$A_3 = \left(1 - t_+^0\right) a_n L_n$$

$$B_1 = L_p\epsilon_{e,p}\alpha_{in}$$

$$B_2 = L_p\epsilon_{e,n}\alpha_{ip} - \frac{L_p^2 \epsilon_{e,p}}{3D_{e,p}}$$

$$B_3 = \left(1 - t_+^0\right) a_p L_p$$

$$\mathfrak{D} = A_1 B_1 - A_2 B_1$$

The above ordinary differential equations (Equations (16) and (17)), pertaining to the variables q_{lin} and q_{lip}, are solvers using the fourth-order Runge–Kutta method. Subsequently, c_{lin} and c_{lip} were solved, followed by the electrolyte concentrations at either ends of the battery cells as required by the enhanced single particle model (detailed in the following sub-section).

2.3. Battery Models

2.3.1. Single Particle Model

The single particle model (SPM) is the most widely used simplified model compared to the pseudo-two-dimensional (P2D) model due to its ease of implementation. It assumes that the lithium concentration profile across each electrode can be approximated to that

across a single spherical particle, and the electrochemical polarization from the electrolyte dynamics is negligible (Figure 2A). Consequently, the PDE that represents the diffusivity of the lithium in the solid electrode needs to be solved for both positive and negative electrodes. These PDEs for both the electrodes are presented below [22,26].

$$\frac{\partial c_{s,p}}{\partial t} = \frac{1}{r^2}\frac{\partial\left(D_{s,p}r^2\left(\frac{\partial c_{s,p}}{\partial r}\right)\right)}{\partial r} \tag{18}$$

$$\frac{\partial c_{s,n}}{\partial t} = \frac{1}{r^2}\frac{\partial\left(D_{s,n}r^2\left(\frac{\partial c_{s,n}}{\partial r}\right)\right)}{\partial r} \tag{19}$$

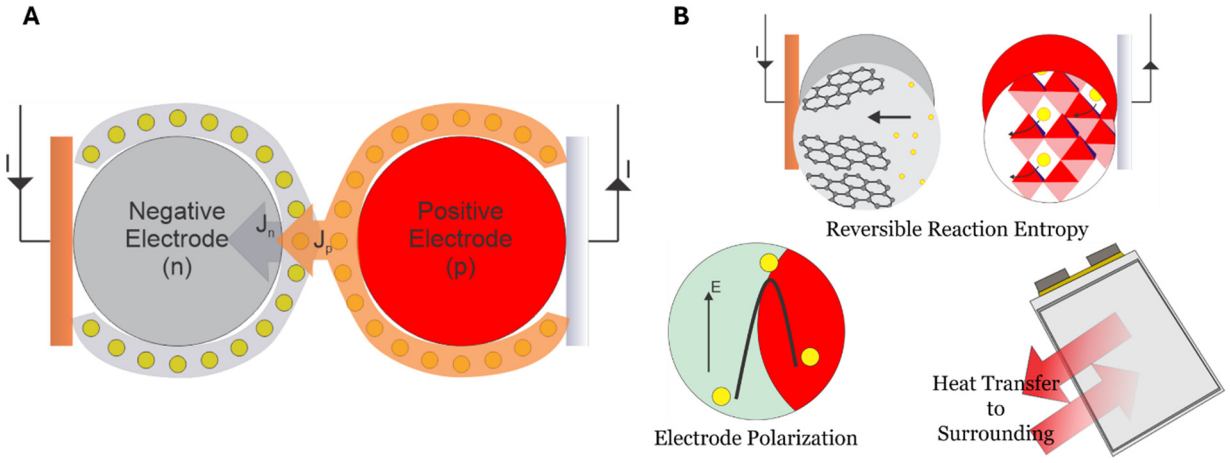

Figure 2. (**A**) Depiction of the single particle model (the direction of the arrows represents the lithium-ion flux during battery cell charge operation) and (**B**) various thermal sources considered in the lumped thermal model.

Both PDEs are subject to the boundary conditions (Equations (2) and (3)) and an initial condition (Equation (4)) as mentioned above. SPM assumes that the lithium flux across the electrode–electrolyte interface is constant and can be represented as given below [26].

$$J_p = \frac{I_{app}}{FS_p} = \frac{I_{app}R_p}{3F\epsilon_p Vol_p}$$
$$J_n = -\frac{I_{app}}{FS_n} = \frac{I_{app}R_n}{3F\epsilon_n Vol_n}$$

where i_{app} represents the applied current [A] across the battery cell terminal. By convention, the charge and discharge currents are represented by positive and negative values, respectively. Moreover, J_p and J_n refers to the lithium-ion flux in the positive and negative electrode regions, respectively (in mol m^{-2} s^{-1}). To be clear, negative flux represents the lithium flux out of the particle, while the positive flux represents the lithium flux out of the electrode particle. The sign in the above equation makes sense since the flux leaves a positive electrode and enters the negative electrode during charge, and the opposite is true for discharge. S_p and S_n represent the total electrochemically active area of the positive and negative electrode, respectively (in the units of m^2). Meanwhile, ϵ_j and Vol_j represent the volume fraction of the electrochemically active region of the solid electrode and the total volume of the electrode.

When the polarization from the electrolyte solution is considered known and given (i.e., $\phi_{e,n}\,|_{x=0} - \phi_{e.p}\,|_{x=L} = I_{app}R_{int}$), the Butler–Volmer equation can be rearranged to solve for the battery cell terminal voltage, and this leads to the following expression [26].

$$V_{cell} = OCP_p(SOC_p,\ T) - OCP_n(SOC_n,\ T) + \frac{2RT}{F}\text{asinh}\left(\frac{m_p}{2}\right) + \frac{2RT}{F}\text{asinh}\left(\frac{m_n}{2}\right) + I_{app}R_{int} \tag{20}$$

where OCP_p and OCP_n represent the open-circuit potential of the positive and negative electrode, respectively. V_{cell} represents the potential difference observed between the cell terminals. The results for the first and second terms represent the thermodynamic battery cell potential at a given electrode SOCs and temperature. In contrast, the fourth and fifth terms represent the polarization from the lithium diffusivity in the solid electrode region. Finally, as mentioned previously, the final term is the polarization from the electrolyte solution polarization. The functions for the electrode OCPs and the model parameters have been obtained from the literature [26].

2.3.2. Enhanced Single Particle Model

Moura et al. incorporated the electrolyte dynamics into the SPM to alleviate its major limitation, which is its applicability for low applied currents (<0.5C) [28]. This model is called the enhanced single particle model (ESPM) in this work. In this subsection, the model equations for the enhanced single particle model, as mentioned by Moura et al., are presented here with slight modifications [28]. These modifications stem from the fact that the definition of applied current, I_{app}, defined here is different than that described by Moura et al. The PDE representing the lithium-ion concentration in the electrolyte region was introduced above, and its formulation in 1D space is given below.

$$\frac{\partial c_e}{\partial t} = \frac{\partial}{\partial x}\left(\frac{D_e}{\epsilon_{e,j}}\frac{\partial c_e}{\partial x}\right) + a_{s.j}\left(1 - t_c^+\right)J_j$$

Specifically, the above expression for the negative electrode, separator, and positive electrode is as follows.

$$\frac{\partial c_{e,p}}{\partial t} = \frac{\partial}{\partial x}\left(\frac{D_{e,p}}{\epsilon_{e,j}}\frac{\partial c_{e,p}}{\partial x}\right) + a_{s.p}\left(1 - t_c^+\right)J_p \tag{21}$$

$$\frac{\partial c_{e,sep}}{\partial t} = \frac{\partial}{\partial x}\left(\frac{D_{e,sep}}{\epsilon_{e,sep}}\frac{\partial c_{e,sep}}{\partial x}\right) \tag{22}$$

$$\frac{\partial c_{e,n}}{\partial t} = \frac{\partial}{\partial x}\left(\frac{D_{e,n}}{\epsilon_{e,n}}\frac{\partial c_{e,n}}{\partial x}\right) + a_{s.n}\left(1 - t_c^+\right)J_n \tag{23}$$

Finally, the following expression gives the cell terminal voltage (V_{cell}) from the enhanced single particle model.

$$\begin{aligned} V_{cell} = OCP_p - \quad & OCP_n \\ & + \tfrac{2RT}{F}\arcsin\left(\tfrac{m_p}{2}\right) + \tfrac{2RT}{F}\arcsin\left(\tfrac{m_n}{2}\right) \\ & + I_{app}R_{cell} \\ & + \tfrac{L_p + 2L_{sep} + L_n}{2\kappa} I_{app}(t) \\ & + k_{conc}(t)\left[\ln c_{e,p}(L_{cell}, t) - \ln c_{e,n}(0,\ t)\right] \end{aligned} \tag{24}$$

In the above equation, additional terms stem from the resultant electrochemical polarization from the electrolyte dynamics compared to a similar equation from the SPM (Equation (21)). The relevant model parameters, pertaining to the electrolyte dynamics, have been obtained from the literature [27].

2.3.3. Reduced-Order Pseudo-Two-Dimensonal Model

A reduced-order P2D was also used in this work due to its suitability as an embedded battery management system. The implementation of the reduced-order P2D model is similar to that found in the literature with some modifications [41,43,44]. Similarly to other works in the literature, it is assumed that the lithium-ion flux at the electrode–electrolyte interface was spatially consistent, and the lithium-ion flux was calculated using those utilized for the SPM and ESPM models. The PDEs for the lithium concentrations in

the solid electrode regions (Equations (19) and (20)) and the liquid electrolyte region (Equations (22)–(24)) were then solved using the polynomial approximation and FVM, respectively. Later, the cell terminal voltage was calculated using Equation (25), above. Then, the PDEs for the spatial potential distribution in the electrode and electrode regions were solved using the FVM. The PDEs for the spatial electrode distribution in the solid electrode and liquid electrolyte region are presented below, respectively.

$$\frac{\partial}{\partial x}\left(\sigma_{eff}\frac{\partial \phi}{\partial x}\right) = a_s F J_j \tag{25}$$

$$\frac{\partial}{\partial x}\left(\kappa_{\text{eff}}\frac{\partial \phi_e}{\partial x}\right) + \frac{\partial}{\partial x}\left(\kappa_{D,\text{eff}}\frac{\partial \ln c_e}{\partial x}\right) + a_s F J_j = 0 \tag{26}$$

where

$$\kappa_{D,eff} = \frac{2RT\kappa(t_+ - 1)}{F}\epsilon_e^{brugg}$$

In the above equations, the variables ϕ_s and ϕ_e represent the electrostatic potential in the solid electrode and liquid electrolyte regions, respectively. Furthermore, σ_{eff} and a_s represent the effective bulk ionic conductivity (calculated from the bulk ionic conductivity, $\sigma_{eff} = \sigma_s * \epsilon^{brugg}$) of the solid electrode particles and specific interfacial surface area, respectively. Similarly, κ_{eff} is the effective electrolyte conductivity (where $\kappa_{eff} = \kappa * \epsilon^{brugg}$). The specific implementation of Equations (26) and (27) using the FVM is similar to Han et al.'s and is briefly described in the Section S1 [17].

2.4. Energy Balance

In this work, the lumped thermal model was used where the spatial thermal distribution across the battery cell was neglected, and the battery cell thermal evolution was assumed to be consistent throughout the battery cell. The ODE representing the relevant energy balance is described below [26].

$$\begin{aligned}\rho C_p Vol_{cell}\frac{dT_{cell}}{dt} &= IT_{cell}\left[\frac{\partial U_p}{\partial t}\left(SOC_{p,\,surf}\right) - \frac{\partial U_n}{\partial t}\left(SOC_{n,\,surf}\right)\right] + I * (V_{cell} \\ &\quad - \left(U_p\left(SOC_{p,\,surf}\right) - U_n\left(SOC_{n,\,surf}\right)\right) + hA_{cell}(T_{cell} - T_{amb})\end{aligned} \tag{27}$$

In the above equation, T_{cell} represents the temperature of the battery cell. In this above equation, the three terms on the RHS represent the thermal contributions from the reversible electrode reaction entropy change, irreversible electrode polarizations, and the heat transfer from the surroundings (Figure 2B). Furthermore, the above equation was subjected to the following initial condition: the initial battery temperature at the start of any simulation was equal to the ambient temperature.

$$T\,|_{t=0} = T_{amb} \tag{28}$$

The parameter values used for the electrode, electrolyte, and the battery cell are tabulated below (Table 1).

Table 1. Parameter values for the battery cell models used in this research.

Symbol	Positive Electrode	Separator	Negative Electrode	Unit
L_j	70 *	20 *	73.5 *	µm
S_j	1.1167 *		0.7824 *	mm^2
$c_{s,j,max}$	51,410		31,833	mol m^{-3}
R_j	8.5 *		12.5 *	µm
$k_{j,ref}$	6.67×10^{-11}		1.76×10^{-11}	m^2 $mol^{-0.5}$ s^{-1}
$Ea_{r,j}$	58,000 *		28,000 *	m^2 s^{-1}
$D_{s,j,ref}$	1×10^{-14} *		3.9×10^{-14} *	m^2 s^{-1}

Table 1. *Cont.*

Symbol	Positive Electrode	Separator	Negative Electrode	Unit
$Ea_{D,j}$	2900 *		3500 *	$m^2\ s^{-1}$
κ_e	0.2875 **			$S\ m^{-1}$
t_c^+	0.38 **			
$c_{e,\ init}$	1000 *			$mol\ m^{-3}$
ρ	1626 *			$kg\ m^{-3}$
Vol_{cell}	2.425×10^{-8}			m^3
C_p	750 *			$J\ K^{-1}$
T_{ref}	298.15 *			K
hA_{cell}	0.085 *			$W\ m^{-2}\ K^{-1}$

* ref [26], ** ref [27].

2.5. Sigma-Point Non-Linear Kalman Filter

Research works have employed non-linear Kalman filters on the state–space representations of the continuum-scale battery models for battery cell state estimations in practical applications [45,46]. Kalman filters, which come under the domain of sequential probabilistic inference, find the efficient estimation of the system (for, e.g., the SOC of positive and negative electrodes) using the prior estimates while considering the randomness of the system's process and sensor noises. The sigma-point non-linear Kalman filter (SPKF) is suitable for non-linear models (as is the case of the battery models in this work) since (1) it does not require the calculation of the derivatives and (2) it samples the strategic points in the probability distribution function of the state and output equations [47]. The state–space representation contains the state and output equations as defined below [47].

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1},\ \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1})$$
$$\boldsymbol{y}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$$

In the above, $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ represent the random vector containing the estimates of the states and outputs, respectively. Meanwhile, $\boldsymbol{u}_k$, $\boldsymbol{w}_k$, and $\boldsymbol{v}_k$ represent the random vectors for the inputs, the process noise in the input signal, and the sensor noise in the output. Moreover, $f(\ldots)$ and $h(\ldots)$ represent the functions for the states and outputs, respectively.

For the practical state's estimations, algorithms pertaining to sequential probabilistic inference, such as SPKF, are critical and they can lead to increased computational resource utilization and solution times. Hence, in this work, SPKF was implemented on the isothermal single particle model to estimate its computational overload. The following sub-section provides the relevant SPKF equations.

Isothermal Single Particle Model

The random vectors for the states and outputs used in this work are defined as follows.

$$\boldsymbol{x}_k = \begin{bmatrix} soc_p \\ soc_n \end{bmatrix}$$
$$\boldsymbol{y}_k = [V_{cell}]$$

The numerical schemes for the determination of the lithium concentration in the electrode particles, as delineated in the above sections, are used for the electrode's soc_j determination. Meanwhile, the equation for the terminal voltage from the single particle model (Equation (21)) is used for the calculation of V_{cell}.

The SPKF is carried out in six steps, as outlined by Plett in his book, Battery Management Systems, and the relevant equations for these steps are as follows [47]. Even before using the SPKF equations, the random vector and their covariance matrices for the states and noises are augmented.

$$x_k^{aug} = \begin{bmatrix} x_k \\ w_k \\ v_k \end{bmatrix}$$

$$\Sigma_k^{aug} = \begin{bmatrix} \Sigma_{\tilde{x},k} & 0 & 0 \\ 0 & \Sigma_{\tilde{w}} & 0 \\ 0 & 0 & \Sigma_{\tilde{v}} \end{bmatrix}$$

Then, the sigma points, using the augmented states and covariance matrices are generated, which are stored in the following matrix. In the matrix below, the first column contains the state estimates. The subsequent L (where L denotes the dimension of the augmented matrix) columns contain the result of the addition of the augmented state vector with $\gamma\sqrt{\Sigma_k^{aug}}$. Similarly, the next L columns contain the result of the subtraction of the augmented state vector with $\gamma\sqrt{\Sigma_k^{aug}}$. The resultant matrix has the dimensions of $(L \times (2L + 1))$. Note that in the case of states being positive and negative SOC, L is equal to two ($L = 2$) and, hence, five sigma points are generated.

$$\mathcal{X}_{k-1}^{aug} = \left\{ x_{k-1}^{aug},\ x_k^{aug} + \gamma\sqrt{\Sigma_k^{aug}},\ x_k^{aug} + \gamma\sqrt{\Sigma_k^{aug}} \right\}$$

The rows of the above augmented matrix are then split to create the matrices, $\mathcal{X}_{k-1}^{x}$, $\mathcal{X}_{k-1}^{w}$, and $\mathcal{X}_{k-1}^{v}$ for the states, system noise, and sensor noise, respectively. These matrices are then used as follows.

Step 1: Preliminary states estimations, $\hat{x}_k^-$

$$\hat{x}_k^- = \mathbb{E}[f(x_{k-1},\ u_{k-1},\ w_{k-1})|\mathbb{Y}_{k-1}] \approx \sum_0^p \alpha_i^{(m)} f\left(\mathcal{X}_{k-1}^{x,+}, u_{k-1}, \mathcal{X}_{k-1}^{w,+}\right) = \sum_0^p \alpha_i^m \mathcal{X}_{k,i}^{x,-}$$

Since a centered difference Kalman filter (*CDKF*) (and further assuming Gaussian distribution) is used here, the value of $\alpha_i^{(m)}$ is as follows:

$$\alpha_0^{(m)} = \tfrac{3-L}{3}$$
$$\alpha_i^{(m)} = \tfrac{1}{6}$$

Step 2: Preliminary covariance estimates, $\Sigma_{\tilde{x},k}^-$

$$\Sigma_{\tilde{x},k}^- = \sum_0^p \alpha_i^{(c)} \left(\left(\tilde{\mathcal{X}}_{k,i}^{x,-} \right) \left(\tilde{\mathcal{X}}_{k,i}^{x,-} \right)^T \right)$$

Again, for the Gaussian distribution on the CDKF, the following is obtained for the $\alpha_i^{(c)}$.

$$\alpha_i^{(c)} = \tfrac{3-L}{3}$$
$$\alpha_i^{(c)} = \tfrac{1}{6}$$

Step 3: Calculation of the model output

In our case, since there are five sigma points, the output is calculated five times on each sigma point before the final output estimation is performed as represented below.

$$\mathcal{Y}_{k,i}^{x,-} = h\left(\mathcal{X}_{k-1}^{x,+}, u_{k-1}, \mathcal{X}_{k-1}^{v,+}\right)$$

$$\hat{y}_k = \mathbb{E}[h(x_k,\ u_k,\ v_k)|\mathbb{Y}_{k-1}] \approx \sum_0^p \alpha_i^{(m)} f\left(\mathcal{X}_{k-1}^{x,+}, u_{k-1}, \mathcal{X}_{k-1}^{v,+}\right) = \sum_0^p \alpha_i^m \mathcal{Y}_{k,i}^{x,-}$$

Step 4: Estimate of the estimator gain matrix, $\boldsymbol{L}_k$, using the output sensor reading

$$\Sigma_{\underset{\sim}{y,k}}^{-} = \sum_{0}^{p} \alpha_i^{(c)} (\boldsymbol{\mathcal{Y}}_{k,i} - \hat{y}_k)(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{y}_k)^T$$

$$\Sigma_{\underset{\sim}{xy,k}}^{-} = \sum_{0}^{p} \alpha_i^{(c)} (\boldsymbol{\mathcal{X}}_{k,i}^{x} - \hat{x}_k)(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{y}_k)^T$$

$$\boldsymbol{L}_k = \Sigma_{\underset{\sim}{xy,k}}^{-} \Sigma_{\underset{\sim}{y,k}}^{-}$$

Step 5: State Estimation

$$\hat{\boldsymbol{x}}_k^{+} = \hat{\boldsymbol{x}}_k^{-} + \boldsymbol{L}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)$$

Step 6: Covariance Estimation

$$\Sigma_{\underset{\sim}{x,k}}^{+} = \Sigma_{\underset{\sim}{x,k}}^{-} - \boldsymbol{L}_k \Sigma_{\underset{\sim}{y,k}}^{-} \boldsymbol{L}_k^T$$

2.6. Programming Language

The code for this work was written in-house in C++ and Python using the object-oriented programming approach [48,49]. Both programming languages have various inherent relative advantages and limitations. Amongst these pros and cons, C++ boasts a relatively higher computational performance, while Python programming language is relatively more straightforward to learn and implement [50,51]. This object-oriented programming approach was utilized to ensure the modularity of electrochemical, thermal, and degradation models. The codebase architecture is depicted by a class diagram specified by the unified modeling language (UML) (Figure 3) [52].

The conceptual class diagram depicts various classes' names and their relationships. The BatteryCell, NElectrode, Electrolyte, and PElectrode classes (note that the NElectrode and PElectrode class inherits from the Electrode class. The class diagram of the Electrode-Class is presented in) store the various parameters and functions of the battery cell, negative electrode, electrolyte, and positive electrode, respectively. Furthermore, the BatteryCell class contains an instance of the NElectrode, Electrolyte, and PElectrode. BaseCycler and other subsequent inherited classes store the information relevant to battery cycling and contain functions to offer the cycling current based on cycling time and other relevant cycling information. The electrochemical PDE, mentioned above, and the thermal and degradation model process their own solver Class that contains relevant methods to solve for the desired output as required. The instances of the BatteryCell, any relevant solver class(es), and the relevant cycler class are included as the instances in the battery cell models, including the one for the single particle model (SPM) and enhanced single particle model (ESPM). Furthermore, the SPKFSolver, the class that adds the functionality for Kalman filter calculations, uses the class for the single particle model.

2.7. Hardware Architecture

The hardware specifications, specifically the central processing unit (CPU) and the random-access memory (RAM) specifications, are listed in the table below (Table 2). This work uses Raspberry Pi 3 model B as the embedded system, and its hardware specifications are listed accordingly.

Table 2. CPU and memory hardware specifications used in this work.

	Embedded	**Personal Computer**	**Cloud**
CPU	Broadcom BCM2837 (1.2 GHz)	AMD Ryzen 7 6800H (3.2 GHz)	Intel(R) Xeon(R) CPU E5-2680 v3 (2.50 GHz)
RAM	1 GB LPDDR2	16.0 GB	1 GB DDR3

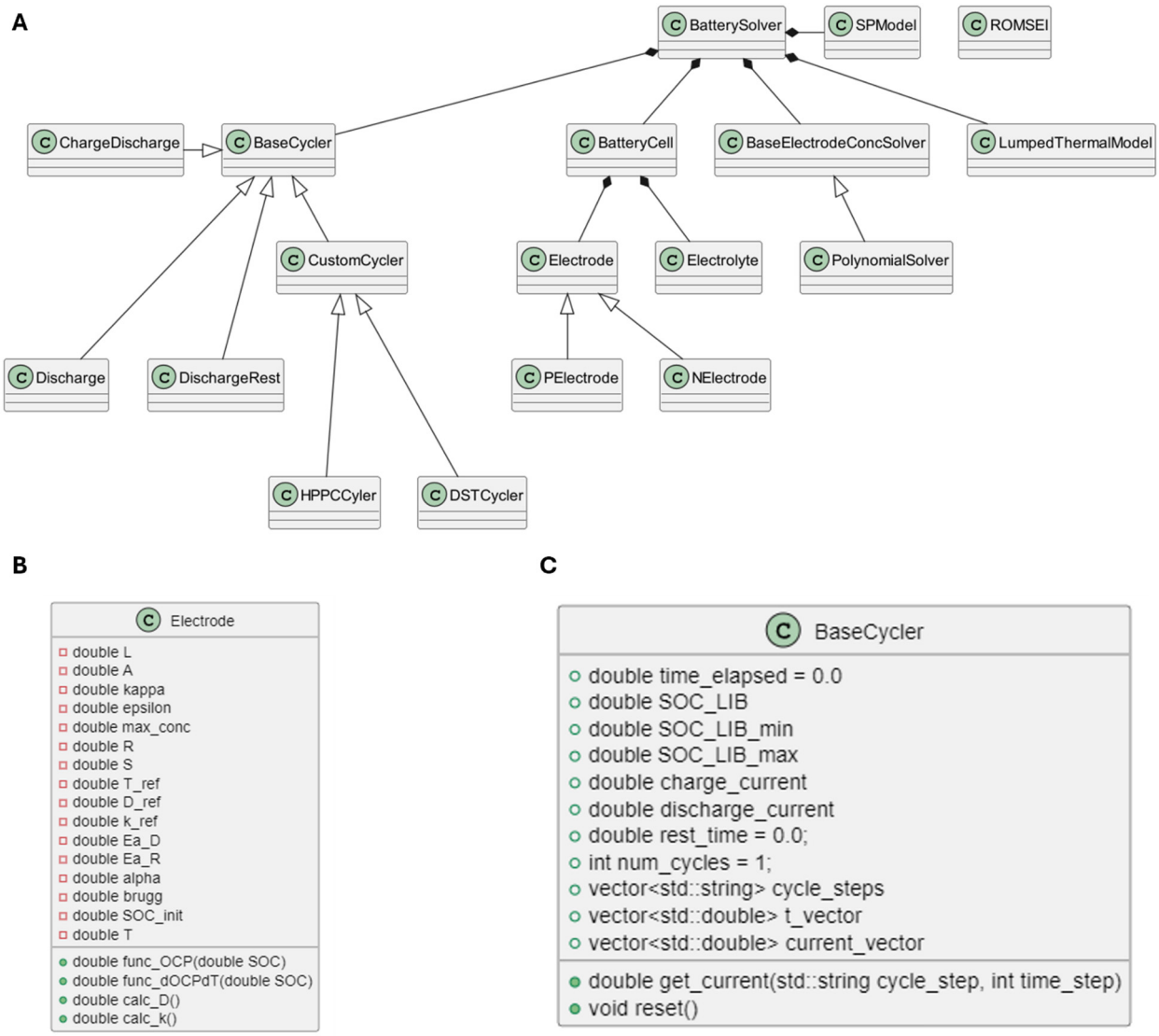

Figure 3. (**A**) Conceptual class diagram representing the codebase object-oriented architecture of the inhouse codebase used in this research. Class diagrams of the (**B**) Electrode, (**C**) BaseCycler. Arrows represent the inheritance of the class objects.

3. Results and Discussions

The lithium-ion concentration in the solid electrode region was solved using various solvers, namely, Crank–Nicolson, eigen-expansion method, and reduced-order polynomial approximations (second- and higher-order) on volume averaged quantities (refer to the Section 2.1). The electrode state-of-charge (SOC) for the negative and the positive electrode was solved using the solvers mentioned above during various battery cycling steps, as depicted below (Figure 4). First, the solvers for the electrode SOC were optimized for the Crank–Nicolson Scheme, eigen-expansion method, and the volume-averaging based on polynomial approximation. The solution time reduction of 46% was observed when the Thomas algorithm was used instead of performing the matrix inversion for the Crank–Nicolson scheme in the Python programming language on the personal computer (PC) system. Also, the memory usage was at least 10 times lower for the Thomas algorithm than for the matrix inversion (refer to Table S1). Therefore, the Thomas algorithm was applied to the Crank–Nicolson scheme (given the matrix system is diagonally dominant) after that. For the eigen-expansion method, the number of roots (variable N mentioned in eigen-expansion method in Section 2) required for the process was set to 5. When compared to the other number of roots for the eigen-expansion method

($N = 2, 5, 10, 20, 50,$ and 100), $N = 5$ seemed to give a good compromise between accuracy and solution speed (refer to Figure S2). Interestingly, the solution time scaled linearly with the number of roots (Figure S2A,D) when solved for the SOCs of the positive and negative electrodes. For instance, the solution times were 8.16, 20.14, 39.95, 79.89, 199.96, and 399.51 s for $N = 2, 5, 10, 20, 50,$ and 100, respectively. As for the volume averaging using polynomial approximation, the second-order and higher-order polynomials were compared. The second-order polynomial approximation was dropped since it could not capture the polarization after the change in the cycling step (Figure S3). Figures S4 and 4 show the electrode SOC as calculated from different solvers under static and dynamic cycling loads, respectively. Overall, the solvers agreed regarding the different cycling steps. When only charge or discharge cycling steps are considered, solvers for the expansion and volume averaging methods with higher polynomial approximation agreed with the Crank–Nicolson method. The disagreement in the electrode's SOCs during discharge, as calculated from the Crank–Nicolson method, and the other two solvers were during the initial periods with the absolute error magnitude of 0.008 (Figure S5A,B). As for the HPPC cycling step, errors of a similar magnitude were observed between these methods, with the highest errors occurring at the beginning of each pulse discharge step (Figure S5C,D).

The solver solution times of these solvers were compared in three computing hardwares with different hardware specifications and two different programming languages (Figure 5). Generally, compared to the Crank–Nicolson method, eigen-expansion and polynomial approximation on volume averaging methods are faster with an average factor (calculated by $\frac{avg.\ solution\ time_{solver}}{avg.\ solution\ time_{CN}}$) of about 2.1×10^{-1} and 1.36×10^{-3} for most cycling steps (except for HPPC) on the personal computing (PC) system. However, these factors were only 6.62×10^{-1} and 0.56×10^{-1} for the HPPC drive cycle for the respective solvers (Figure 5A) on the PC system. This reduction in solution time for the HPPC drive cycle is most probably from the high number of the set rest intervals, and consequently, all the solvers have to undergo these iterations. As expected, cloud and cloud-embedded hardware, which have lower performance specifications (in central processing units and memory) than the PC system, took longer than the PC system's solution times (Figure 5B,C). Interestingly, while the Crank–Nicolson and eigen-expansion method took an average of 1.37 and 1.36 times longer on the cloud, the polynomial approximation method took 2.78 times longer than the exact solvers' solution times on the PC system. Finally, when comparing programming languages on the same hardware (PC), Crank–Nicolson, eigen-expansion, and polynomial expansion methods proved to be, on average, 320.76, 158.07, and 18.23 times faster on C++, respectively, than on Python (Figure 5D). On the embedded systems, the Crank–Nicolson, the eigen, and the polynomial solver were over 130, 110, and 12 times faster on the C++ compared to the Python programming language, respectively.

Since the enhanced single particle model (ESPM) requires the solution for the PDE about the lithium-ion concentrations in the electrolyte regions (Equations (22)–(24)) to be solved in each of its iterations, its solvers were tested in different computing hardware and programming languages. For testing purposes, the PDE, pertaining to the lithium-ion diffusivity in the liquid electrolyte region, was solved for each time step during the discharge cycle. For the solver based on the finite volume method (FVM), the number of nodes was set to 10 for each of the negative electrode, separator, and positive electrode regions. Furthermore, the PDE after the FVM discretization could be solved by using the Thomas algorithm or regular matrix inversion. There was a 20.5% reduction in the solution time when the Thomas algorithm was used instead of regular matrix inversion (refer to Section S7). Hence, the Thomas algorithm was used throughout this section for the electrolyte solver discretized using the FVM. However, it is to be noted that, given our simulation parameters, there was not a significant usage in the memory usage during the solution determination using matrix inversion or Thomas algorithm. The electrolyte concentration profile across the battery cell at various times, as measured by the FVM solver, is depicted in Figure 6A. In this work, the C++ programming language demonstrated at least a ten-fold (10X) decrease in the solution times across all the studied computing systems

(Figure 6B). When the polynomial approximation on the average volume quantity was used, the solution time sped up 115 times, with the actual solution time of 0.854 s compared to 98.152 s from the FMV method.

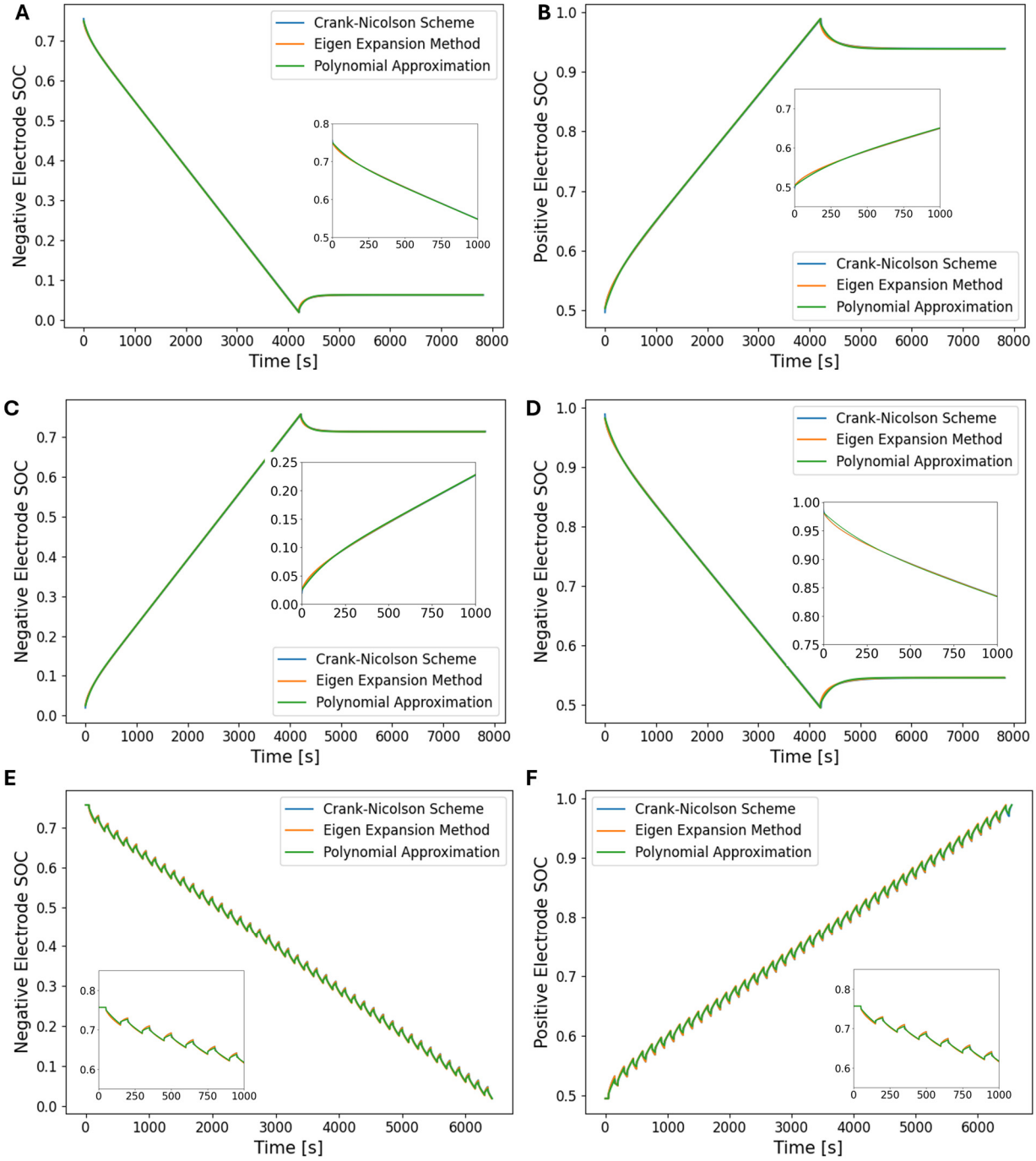

Figure 4. The solution of various solvers for the PDE about the lithium diffusivity in the solid electrode region for (**A**) a negative electrode and (**B**) a positive electrode under discharge–rest, (**C**) a negative electrode and (**D**) a positive electrode under charge–rest, and (**E**) a negative electrode and (**F**) a positive electrode under HPPC cycling steps.

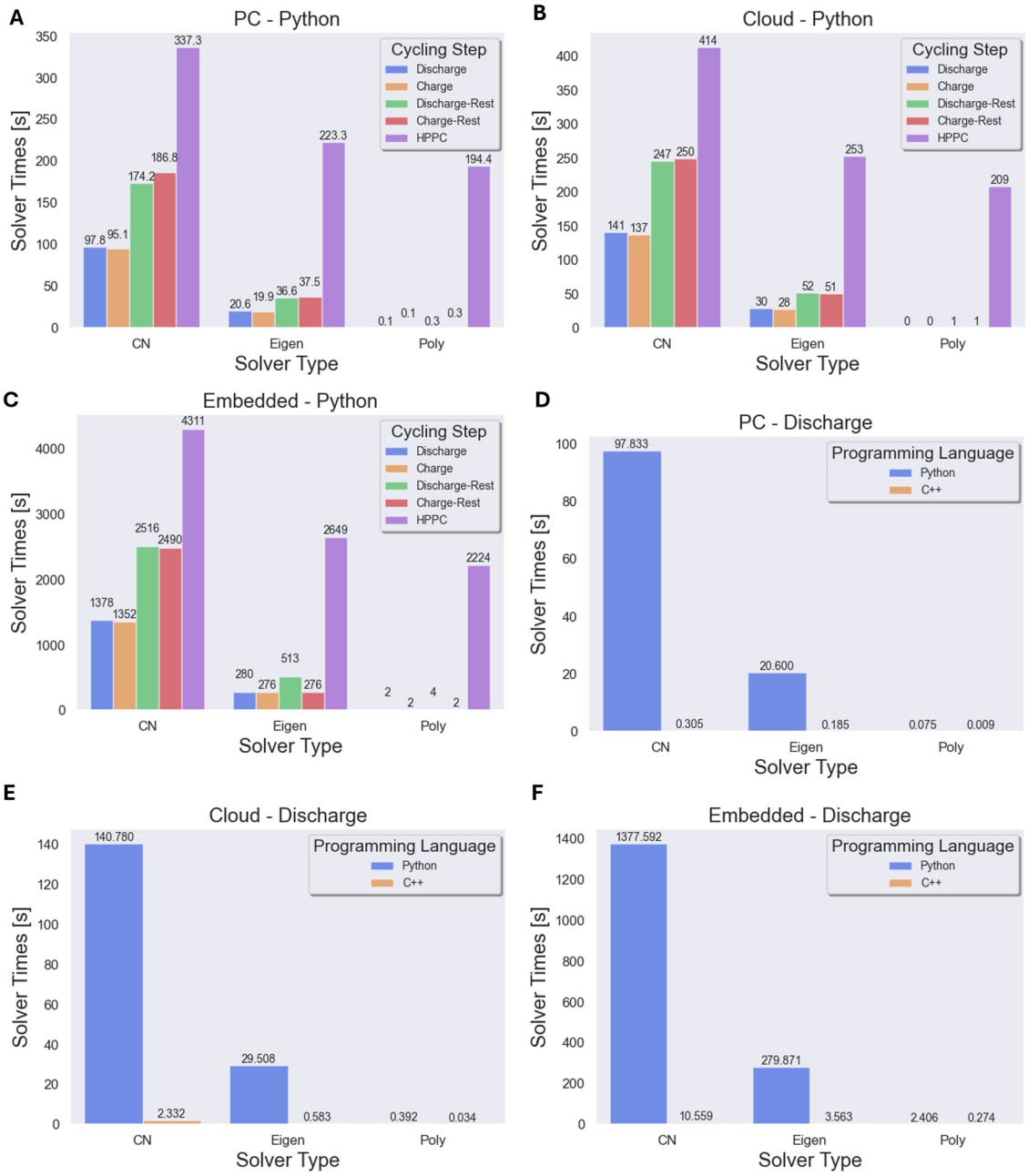

Figure 5. The solution times of various cycling steps in the (**A**) PC, (**B**) cloud, and (**C**) embedded systems for the negative electrode. The solution times of the cycling steps performed in the Python and C++ programming language on the PC using (**D**) eigen and (**E**) polynomial solvers. (**F**) The solution times of the polynomial solver on the embedded system are presented.

Figure 7 shows the battery models under static and dynamic battery cycling loads. Before proceeding, it is noteworthy that the solution times of the isothermal single particle model (SPM) were comparable to that of the enhanced self-correcting equivalent circuit model (ESC). These solution times were 2.256 s and 3.39 s for the SPM and ESC, respectively, when the simulations were performed using the Python programming language on the PC system. For the mentioned comparison, the open-circuit potential (OCV) of the battery cell was estimated from the electrode's open-circuit potential (OCP) and later fitted via polynomial approximation (Figure S8). Furthermore, the readers should refer to Section S9 for further details on the ESC circuit, model equations, and model parameters [19]. Under the static cycling load, the SPM and ESPM models agree, although they deviate during

dynamic HPPC cycling (Figure 7A–C). For the ESPM and ROM-P2D, a higher-order polynomial approximation on the volume-averaged lithium concentration quantity was used as the electrode concentration solver, while the finite volume method (FVM) was used for the electrolyte solver. Similarly for the ROM-P2D, FVM was used for the electrode and electrolyte spatial potential distribution solver. Generally, a two-fold increase in the solution time was observed when the simulations between the ESPM and ROM-P2D were conducted (an instance for the discharge step under isothermal conditions on a PC using Python programming language is depicted in Figure 7D). Again, the HPPC cycling load required a higher computation time, possibly because of the fixed set time for the rest and discharge periods (Figure 7E). When the thermal model is added, the solution time increases by about 10% (Figure 7F). Furthermore, a ten-fold reduction was achieved when C++ was used instead of the Python programming language (Figure 7G). It should be noted that the simulation time was almost equal to the real time when ROM-P2D was conducted on an embedded system under Python programming language (Figure 7H).

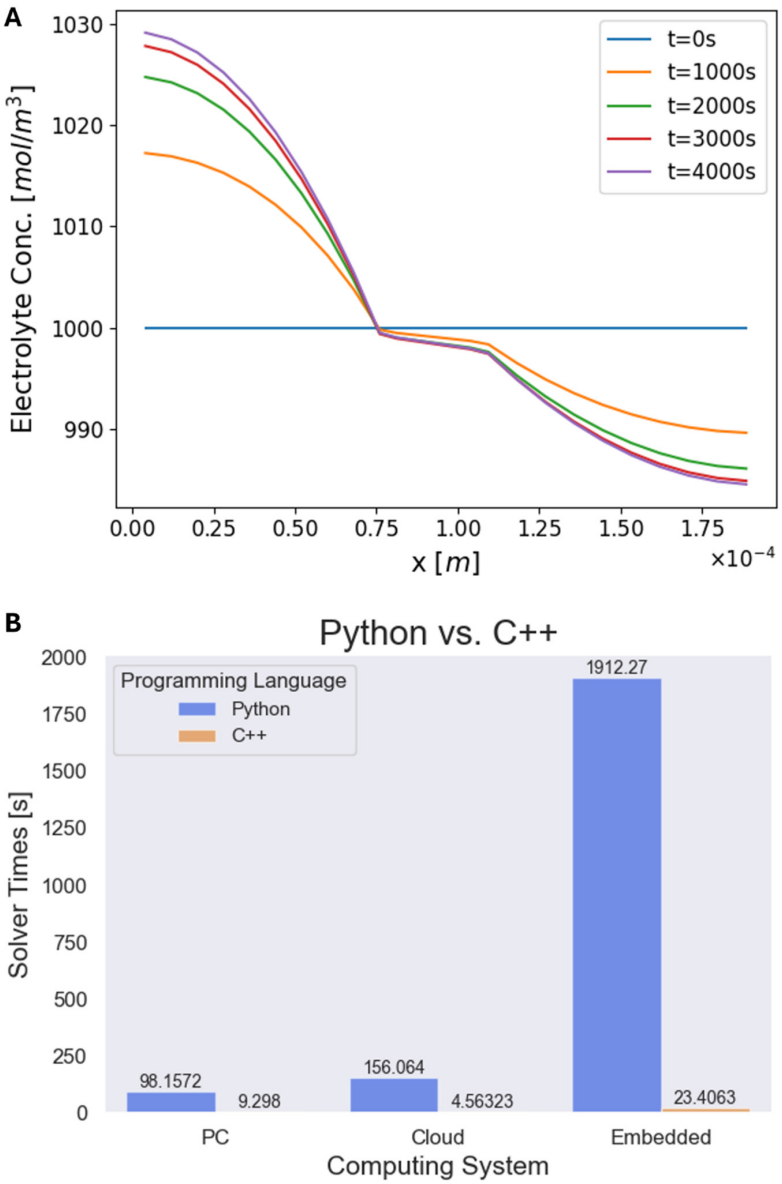

Figure 6. (**A**) Electrolyte concentration at different times across the battery cell during discharge. (**B**) The solution times of the FVM solver in Python and C++ programming languages across various computing systems.

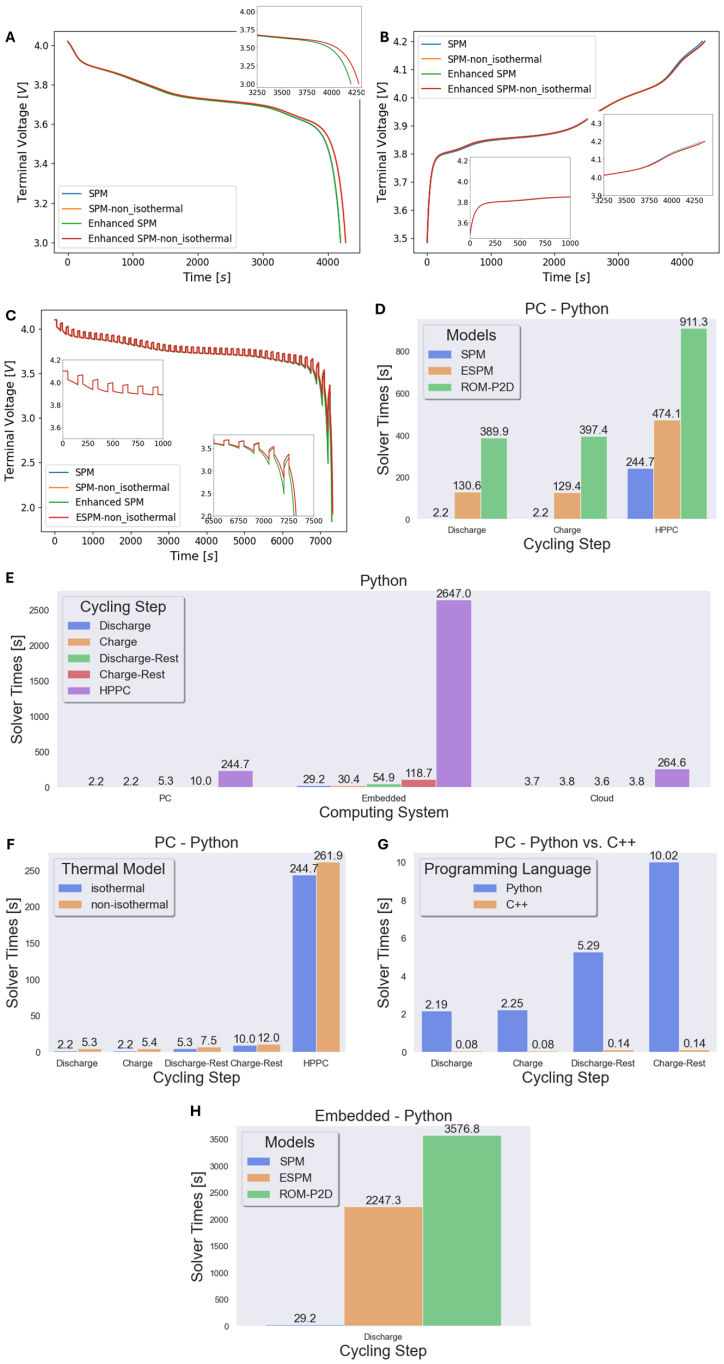

Figure 7. Time series of the battery cell terminal voltage as calculated from various battery models under (**A**) constant current discharge, (**B**) constant current charge, and (**C**) dynamic HPPC cycling. (**D**) Solution times of various charging steps as conducted using different battery models in Python in the PC. (**E**) Solution times of the SPM under various cycling steps under a PC system. (**F**) Solution times of isothermal and non-isothermal SPM. (**G**) Comparison of the computation time between Python and C++ programming languages. (**H**) The solutions times of the isothermal discharge step for different battery models using Python in embedded system.

Later model solution times using a single particle model with a non-linear Kalman filter were tested. In practical BMS applications, a Kalman filter (a category of sequential probabilistic inference) is used to estimate the SOC given the system and sensor noise inherent during application [47,53,54]. SPKF, specifically the centered difference Kalman filter (CDKF), was applied to the single particle model under the under discharge. Random noise

was added to the input discharge current and output voltage using the normal distribution (specifically, the mean of 0.0 and standard deviation of 0.01) (Figure 8A). This served as part of the system noise and will add noise to the simulated battery cell terminal voltage. Further random noise from the normal distribution was added to the terminal voltage and this added noise acted as the sensor noise. SPFK was applied in the single particle model and its output was displayed and compared with the noisy cell terminal voltage in Figure 8B. The polynomial approximation on the averaged concentration quantities was used as the numerical scheme for the electrode SOC determination. The resultant SOC estimations are also depicted below. It should that be noted that for SPKF simulations, the time difference between each iteration was set to 0.2 s. On Python programming language, it was observed that the inclusion of SPKF roughly increased the solution time twice-fold (Figure 8E). Interestingly, the simulation time for the single particle model with SPKF on the embedded system with Python programming language was much lower than the actual time. It can be estimated that the current hardware is sufficient for conducting reliable battery cell real-time states estimation. However, battery pack state estimation might require multiple battery cell simulations.

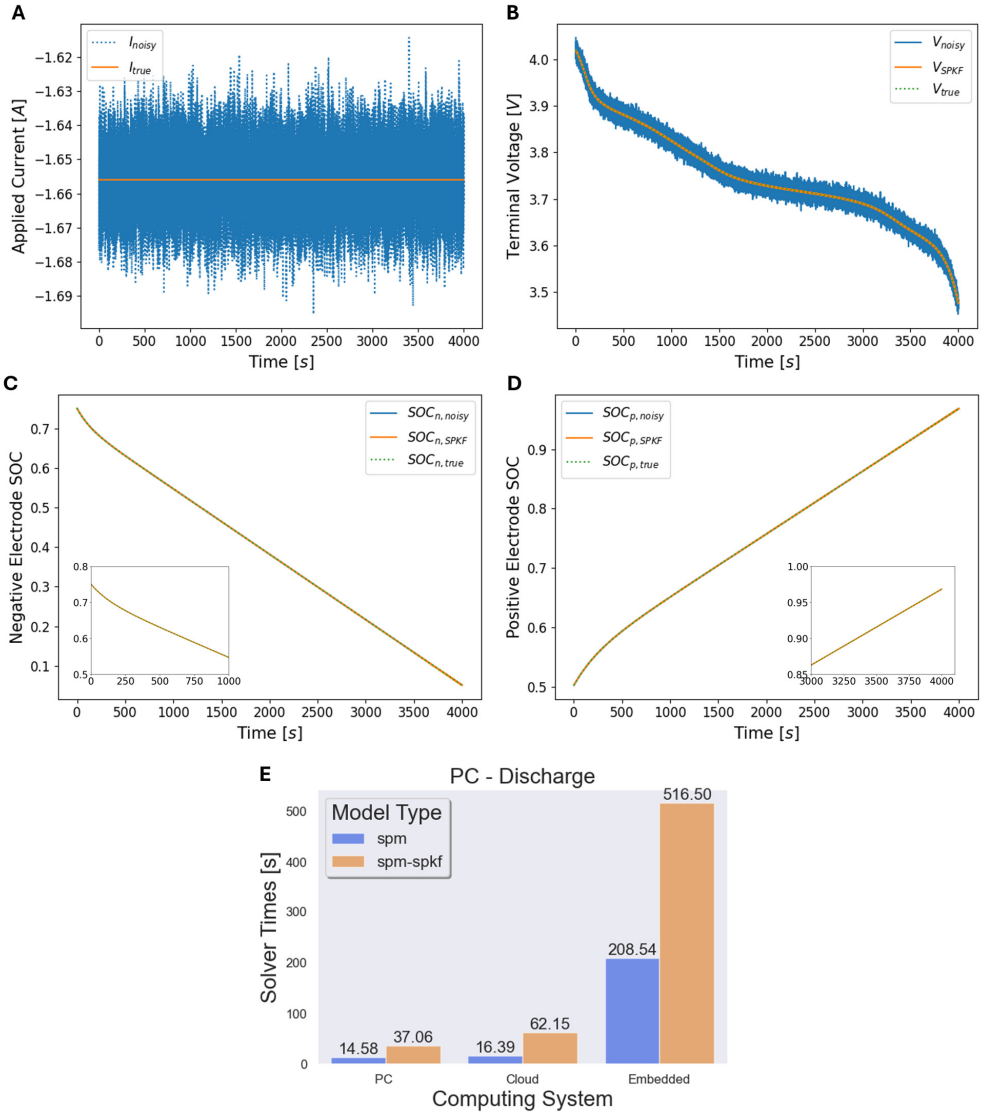

Figure 8. (**A**) The true (or intended) vs. the actual noisy applied current. (**B**) The actual and SPKF estimated outputs. The estimated SOC for the (**C**) negative and (**D**) positive electrodes. (**E**) The computational times of the SPM vs. SPM with SPKF across multiple computing systems.

4. Conclusions

The demand for lithium-ion batteries (LIBs) and LIB-based battery packs has steadily increased over recent years and is projected to grow in the foreseeable future. In practical applications, LIB battery packs are connected to the battery management system (BMS) that is responsible for (1) user and LIB battery pack safety, (2) battery cell balancing, and (3) LIB pack state and health estimations. An accurate battery state estimation is challenging due to the complex and non-linear phenomena that occur during its charge/discharge operations. Phenomenological cell-level models, such as the equivalent circuit model (ECM), and continuum-scale physics-based models (PBMs) have been proposed in the literature for battery cell and pack state estimations. Compared to ECMs, PBM has been shown to have higher accuracy and model fidelity; however, they suffer from high computation times. This work compares the computational performance of PBMs, namely the single particle model (SPM) and enhanced single particle model (ESPM) under various battery cycles on embedded and cloud systems using Python and C++. In particular, representative hardware with relevant hardware specifications for embedded personal computers and cloud computing systems was used. A codebase in Python and C++ programming languages was utilized with similar object-oriented programming architecture. The PBM parameters, particularly for the lithium cobalt oxide (LCO), were obtained from the literature. The cycling steps considered were the discharge, charge, discharge–rest, charge–rest, and hybrid pulse power characterization (HPPC).

As per the PBM requirement, the lithium-ion concentrations in the solid positive and negative electrode particles were calculated and compared using the implicit Crank–Nicolson, eigen-expansion, and polynomial approximation on the volume average concentration. Similarly, the solvers for the lithium-ion concentration in the liquid electrolyte region were solved using the finite volume method and polynomial approximation on the volume average concentrations. The Thomas algorithm accelerated matrix operations by 46% and 21% for the Crank–Nicolson and finite volume methods, respectively, and reduced the memory use. Eigen-expansion and polynomial approximations displayed reasonable accuracy when the simulated electrode state-of-charge (SOC) was compared with the Crank–Nicolson scheme. Moreover, the eigen-expansion method and polynomial approximation demonstrated 5 and 735 times faster solution times for most cycling steps. The eigen-expansion method and polynomial approximation proved to be, on average, 158.07 and 18.23 times faster in C++, respectively, than in Python. A similar trend across the programming languages was observed when the battery models (namely the SPM, ESPM, ROM-P2D) were analyzed. Moreover, it was interesting to note that a type of ECM, i.e., an enhanced self-correcting model, displayed a longer solution time than the single particle model. While the embedded system took longer to solve for the full electrochemical battery model due to its lower processing and memory specifications, it could still perform the simulations faster than the real-time system, making the PBM suitable for real-time embedded applications. Furthermore, the inclusion of thermal models added about 20% to the solution times. When the non-linear sigma-point Kalman filter was applied to the single particle model, the simulation time increased roughly by two-fold across all the hardware. However, even on the embedded systems, the simulation times remained well within the real-time constraints, supporting the feasibility of real-time state estimations of battery cells using continuum-scale simplified EMs, such as the SPM, with reduced-order solvers.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/batteries10120439/s1, Figure S1: The (A) positive and (B) negative electrode's open-circuit potential; Figure S2: The (A) solution times, (B) plots, and (C) absolute errors from the use of a various number of roots for the calculation of the negative electrode's state-of-charge (SOC) during the battery cell discharge. Similarly, the (A) solution times, (B) plots, and (C) absolute errors of the positive electrode's state-of-charge (SOC) during the battery cell. These simulation times are from the simulations performed on the personal computer in the Python programming language. Figure S3: The polynomial approximation of various orders on the volume averaged equation of the partial differential equation pertaining to the lithium diffusivity in the solid electrode region.

Figure S4: The solution of various solvers for the PDE pertaining to the lithium diffusivity in the solid electrode region for (A) the negative electrode and (B) the positive electrode under discharge, (C) the negative electrode and (D) the positive electrode under charge cycling steps. Figure S5: Absolute error between the Crank–Nicolson method and the eigen-expansion and polynomial expansion methods for the (A) negative electrode and the (B) positive electrode under the discharge cycling step. Furthermore, the absolute error for these methods for the (C) negative and the (D) negative electrodes under the HPPC cycling step. Figure S6: The results from the inverse and the Thomas algorithm for the FVM solver on the electrolyte concentration PDE after 4000s of constant discharge. Figure S7: Battery models under constant current (A) discharge and (B) charge static battery cycle. Figure S8: (A) The electrical circuit for the ESC. (B) Actual and fitted battery cell open-circuit voltage (OCV). The dashed lines depict the electrode's open-circuit potentials (OCP). (C) The obtained time–terminal voltage comparison from the SPM and ESC. Table S1: Solution times of the different solution methods performed in Python programming language in the PC system; Table S2: Model parameters for enhanced self-correcting equivalent circuit model.

Author Contributions: Conceptualization, Z.M.; Methodology, M.A. and A.Y.; Software, M.A. and A.Y.; Validation, M.A., A.Y. and M.F.; Investigation, M.A.; Resources, Y.L. and M.F.; Data curation, Y.L. and M.F.; Writing—original draft, M.A.; Visualization, Z.M.; Supervision, Z.M. and Z.C.; Project administration, Z.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No: XDB0600400.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The authors of this work would like to acknowledge the Department of Chemical Engineering at the University of Waterloo.

Conflicts of Interest: The authors declare no competing interests.

Nomenclature

Abbreviations and Acronyms

BMS	Battery management system
CDKF	Centered difference kalman filter
ESC	Enhanced self-correcting equivalent circuit model
ECM	Equivalent circuit model
FVM	Finite volume method
HPPC	Hybrid pulse characterization
LCO	Lithium cobalt oxide
LHS	Left hand side
LIB	Lithium-ion battery
ODE	Ordinary differential equation
RHS	Right hand side
SPM	Single particle model
ESPM	Enhanced single particle model
P2D	Pseudo-two-dimensional
PBM	Physics-based model
PC	Personal computer
PDE	Partial differential equation
ROM-P2D	Reduced-order pseudo-two-dimensional
SPKF	Sigma-point Kalman filter

Symbols

$\epsilon_{e,j}$	Volume fraction of the electrolyte in the region j
$\epsilon_{s,j}$	Volume fraction of the electrode in the region j
κ	Electrolyte ionic conductivity [S m^{-1}]
ρ	Density [kg/m^3]
A_{cell}	Surface area of the battery cell
$a_{s,j}$	Specific interfacial area of the electrode [m^2/m^3]

$c_{e,init}$	Initial lithium-ion concentration across the battery cell [mol/m^3]
$c_{e,j}$	Lithium-ion concentration in the electrolyte region in the battery cell region j [mol/m^3]
$c_{s,j}$	Lithium concentration in the solid electrode region [mol/m^3]
$D_{e,j}^{eff}$	Effective lithium-ion diffusivity in the electrolyte region in the region j [m^{2}s^{-1}]
$D_{s,j}$	Lithium-ion diffusivity in the solid electrode region [mol/m^3]
F	Faraday's constant (96487 C/mol)
j	Region across the battery cell (p for positive electrode, sep for the separator, and n for negative electrode region)
J_e	Lithium-ion flux in the electrolyte region [mol/(m^{2}s)]
J_j	Lithium-ion flux across the electrode–electrolyte interface at the electrode j [mol/(m^{2}s)]
h	Heat transfer coefficient [W/(m^{2}K)]
I	Applied battery cell terminal current [A]
L_j	Thickness of region j [m]
R	Natural gas constant (8.314 J/(mol K))
R_j	Particle radius of electrode j [m]
R_{int}	Battery cell internal resistance from the electrolyte solution [Ω]
T_{amb}	Ambient temperature [K]
T_{cell}	Battery cell temperature [K]
soc_j	State-of-charge of the electrode j
t_+^0	Cationic transference number
S_j	Electrochemically active area for electrode j [m^2]
x	Coordinate across the battery cell [m]
U_j	Open-circuit potential of the electrode j [V]
V_{cell}	Battery cell terminal voltage [V]
Vol_j	Volume of the electrode j [m^3]
Vol_{cell}	Volume of the battery cell [m^3]

References

1. Masias, A.; Marcicki, J.; Paxton, W.A. Opportunities and Challenges of Lithium Ion Batteries in Automotive Applications. *ACS Energy Lett.* **2021**, *6*, 621–630. [CrossRef]
2. Wu, Y. Applications of Lithium-Ion Batteries. In *Lithium-Ion Batteries*; CRC Press: Boca Raton, FL, USA, 2020; pp. 540–571. [CrossRef]
3. Deng, D. Li-ion batteries: Basics, progress, and challenges. *Energy Sci. Eng.* **2015**, *3*, 385–418. [CrossRef]
4. Rahn, C. Electrochemistry. In *Battery Systems Engineering*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2013; pp. 11–22. [CrossRef]
5. Choi, J.W.; Aurbach, D. Promise and reality of post-lithium-ion batteries with high energy densities. *Nat. Rev. Mater.* **2016**, *1*, 16013. [CrossRef]
6. Tarascon, J.-M.; Armand, M. Issues and challenges facing rechargeable lithium batteries. *Nature* **2001**, *414*, 359–367. [CrossRef] [PubMed]
7. Cheng, K.W.E.; Divakar, B.P.; Wu, H.; Ding, K.; Ho, H.F. Battery-management system (BMS) and SOC development for electrical vehicles. *IEEE Trans. Veh. Technol.* **2011**, *60*, 76–88. [CrossRef]
8. Lu, L.; Han, X.; Li, J.; Hua, J.; Ouyang, M. A review on the key issues for lithium-ion battery management in electric vehicles. *J. Power Sources* **2013**, *226*, 272–288. [CrossRef]
9. Hu, X.; Li, S.; Peng, H. A comparative study of equivalent circuit models for Li-ion batteries. *J. Power Sources* **2012**, *198*, 359–367. [CrossRef]
10. Ahmed, M.; Zheng, Y.; Amine, A.; Fathiannasab, H.; Chen, Z. The role of artificial intelligence in the mass adoption of electric vehicles. *Joule* **2021**, *5*, 2296–2322. [CrossRef]
11. He, H.; Xiong, R.; Fan, J. Evaluation of Lithium-Ion Battery Equivalent Circuit Models for State of Charge Estimation by an Experimental Approach. *Energies* **2011**, *4*, 582–598. [CrossRef]
12. Lai, X.; Zheng, Y.; Sun, T. A comparative study of different equivalent circuit models for estimating state-of-charge of lithium-ion batteries. *Electrochim. Acta* **2018**, *259*, 566–577. [CrossRef]
13. Elvira, D.G.; Machado, R.; Plett, G.L.; Trimboli, M.S.; Blavi, H.V.; Pastor, A.C.; Salamero, L.M. Simplified Li Ion Cell Model for BMS Coupling an Equivalent Circuit Dynamic Model with a Zero Dimensional Physics Based SEI Model. *J. Electrochem. Soc.* **2021**, *168*, 110526. [CrossRef]
14. Tran, M.-K.; Mathew, M.; Janhunen, S.; Panchal, S.; Raahemifar, K.; Fraser, R.; Fowler, M. A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters. *J. Energy Storage* **2021**, *43*, 103252. [CrossRef]

15. Tanim, T.R.; Rahn, C.D.; Wang, C.Y. State of charge estimation of a lithium ion cell based on a temperature dependent and electrolyte enhanced single particle model. *Energy* **2015**, *80*, 731–739. [CrossRef]
16. Li, J.; Adewuyi, K.; Lotfi, N.; Landers, R.; Park, J. A single particle model with chemical/mechanical degradation physics for lithium ion battery State of Health (SOH) estimation. *Appl. Energy* **2018**, *212*, 1178–1190. [CrossRef]
17. Han, S.; Tang, Y.; Rahimian, S.K. A numerically efficient method of solving the full-order pseudo-2-dimensional (P2D) Li-ion cell model. *J. Power Sources* **2021**, *490*, 229571. [CrossRef]
18. Plett, G. Battery Boot Camp. In *Battery Management System*, 1st ed.; Artech House: Norwood, MA, USA, 2015; pp. 1–28.
19. Plett, G. Equivalent Circuit Models. In *Battery Management Systems*; Artech House: Norwood, MA, USA, 2015; Volume 1, pp. 30–59.
20. Plett, G. Continuum-scale Cell Models. In *Battery Management Systems*; Artech House: Norwood, MA, USA, 2015; pp. 135–171.
21. Fuller, T.F.; Doyle, M.; Newman, J. Simulation and Optimization of the Dual Lithium Ion Insertion Cell. *J. Electrochem. Soc.* **1994**, *141*, 1–10. [CrossRef]
22. Plett, G.L. Microscale Cell Models. In *Battery Management Systems*; Artech House: Norwood, MA, USA, 2015; pp. 65–135.
23. Ojo, O.; Lang, H.; Kim, Y.; Hu, X.; Mu, B.; Lin, X. A Neural Network Based Method for Thermal Fault Detection in Lithium-Ion Batteries. *IEEE Trans. Ind. Electron.* **2021**, *68*, 4068–4078. [CrossRef]
24. Kumar, K.; Sarkar, J.; Mondal, S.S. Analysis of ternary hybrid nanofluid in microchannel-cooled cylindrical Li-ion battery pack using multi-scale multi-domain framework. *Appl. Energy* **2024**, *355*, 122241. [CrossRef]
25. Plett, G. Thermal Modelling. In *Battery Management Systems*, 1st ed.; Artech House: Norwood, MA, USA, 2015; pp. 276–301.
26. Guo, M.; Sikha, G.; White, R.E. Single-Particle Model for a Lithium-Ion Cell: Thermal Behavior. *J. Electrochem. Soc.* **2011**, *158*, A122–A132. [CrossRef]
27. Han, R.; Macdonald, C.; Wetton, B. A fast solver for the pseudo-two-dimensional model of lithium-ion batteries. *arXiv* **2021**, arXiv:2111.09251.
28. Moura, S.J.; Argomedo, F.B.; Klein, R.; Mirtabatabaei, A.; Krstic, M. Battery State Estimation for a Single Particle Model with Electrolyte Dynamics. *IEEE Trans. Control Syst. Technol.* **2017**, *25*, 453–468. [CrossRef]
29. Rahimian, S.K.; Rayman, S.; White, R.E. Extension of physics-based single particle model for higher charge–discharge rates. *J. Power Sources* **2013**, *224*, 180–194. [CrossRef]
30. Schmidt, A.; Oehler, D.; Weber, A.; Wetzel, T.; Ivers-Tiffée, E. A multi scale multi domain model for large format lithium-ion batteries. *Electrochim. Acta* **2021**, *393*, 139046. [CrossRef]
31. Kim, G.-H.; Smith, K.; Lee, K.-J.; Santhanagopalan, S.; Pesaran, A. Multi-Domain Modeling of Lithium-Ion Batteries Encompassing Multi-Physics in Varied Length Scales. *J. Electrochem. Soc.* **2011**, *158*, A955. [CrossRef]
32. Magri, L.; Sequino, L.; Ferrari, C. Simulating the Electrochemical-Thermal Behavior of a Prismatic Lithium-Ion Battery on the Market under Various Discharge Cycles. *Batteries* **2023**, *9*, 397. [CrossRef]
33. Gwak, G.; Ju, H. Multi-Scale and Multi-Dimensional Thermal Modeling of Lithium-Ion Batteries. *Energies* **2019**, *12*, 374. [CrossRef]
34. Rizvi, S.; Tahir, M.W.; Ramzan, N.; Merten, C. Multiscale-multidomain model order reduction of Lithium-ion batteries for automobile application: A review. *J. Energy Storage* **2024**, *99*, 113390. [CrossRef]
35. Plauska, I.; Liutkevičius, A.; Janavičiūtė, A. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics* **2023**, *12*, 143. [CrossRef]
36. Dokic, K.; Radisic, B.; Cobovic, M. MicroPython or Arduino C for ESP32—Efficiency for neural network edge devices. *Commun. Comput. Inf. Sci. (CCIS)* **2020**, *1187*, 33–43. [CrossRef]
37. Choroszucho, A.; Golonko, P.; Bednarek, J.; Sumorek, M.; Żukowski, J. Comparison of high-level programming languages efficiency in embedded systems. In Proceedings of the Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments, Wilga, Poland, 27 May–2 June 2019; Volume 11176, pp. 1800–1808. [CrossRef]
38. Thomas, J.W. *Numerical Partial Differential Equations: Finite Difference Methods*; Springer: New York, NY, USA, 1995.
39. Thibault, J.; Bergeron, S.; Bonin, H.W. On finite-difference solutions of the heat equation in spherical coordinates. *Numer. Heat Transf.* **1987**, *12*, 457–474. [CrossRef]
40. Lee, W. Tridiagonal Matrices: Thomas Algorithm—University of Limerick, n.d. Available online: https://www.yumpu.com/en/document/read/4721668/tridiagonal-matrices-thomas-algorithm-university-of-limerick (accessed on 19 May 2024).
41. Krishnan, H.; Piyush, T.; Ramachandran, S. Theoretical Framework of the Reduced Order Models. In *Mathematical Modeling in Lithium-Ion Batteries*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 33–42.
42. Torchio, M.; Magni, L.; Gopaluni, R.B.; Braatz, R.D.; Raimondo, D.M. LIONSIMBA: A Matlab Framework Based on a Finite Volume Model Suitable for Li-Ion Battery Design, Simulation, and Control. *J. Electrochem. Soc.* **2016**, *163*, A1192–A1205. [CrossRef]
43. Kumar, V.S. Reduced order model for a lithium ion cell with uniform reaction rate approximation. *J. Power Sources* **2013**, *222*, 426–441. [CrossRef]
44. Kumar, V.S.; Gambhire, P.; Hariharan, K.S.; Khandelwal, A.; Kolake, S.M.; Oh, D.; Doo, S. An explicit algebraic reduced order algorithm for lithium ion cell voltage prediction. *J. Power Sources* **2014**, *248*, 383–387. [CrossRef]
45. Marelli, S.; Corno, M. Model-Based Estimation of Lithium Concentrations and Temperature in Batteries Using Soft-Constrained Dual Unscented Kalman Filtering. *IEEE Trans. Control Syst. Technol.* **2021**, *29*, 926–933. [CrossRef]
46. Li, W.; Fan, Y.; Ringbeck, F.; Jöst, D.; Han, X.; Ouyang, M.; Sauer, D.U. Electrochemical model-based state estimation for lithium-ion batteries with adaptive unscented Kalman filter. *J. Power Sources* **2020**, *476*, 228534. [CrossRef]

47. Plett, G. Battery State Estimations. In *Battery Management Systems*; Artech House: Norwood, MA, USA, 2016; pp. 69–165.
48. Welcome to Python.org, (n.d.). Available online: https://www.python.org/ (accessed on 20 May 2024).
49. Cplusplus. Available online: https://cplusplus.com/ (accessed on 20 May 2024).
50. Ateeq, M.; Habib, H.; Umer, A.; Rehman, M.U. C++ or Python? Which one to begin with: A learner's perspective. In Proceedings of the 2014 International Conference on Teaching and Learning in Computing and Engineering (LaTiCE), Kuching, Malaysia, 11–13 April 2014; pp. 64–69. [CrossRef]
51. Ahmadi, M.; Abadi, M.Q.H. A review of using object-orientation properties of C++ for designing expert system in strategic planning. *Comput. Sci. Rev.* **2020**, *37*, 100282. [CrossRef]
52. About the Unified Modeling Language Specification Version 2.5.1 (n.d.). Available online: https://www.omg.org/spec/UML/2.5.1 (accessed on 21 May 2024).
53. Wang, L.; Ma, J.; Zhao, X.; Li, X.; Zhang, K.; Jiao, Z. Adaptive robust unscented Kalman filter-based state-of-charge estimation for lithium-ion batteries with multi-parameter updating. *Electrochim. Acta* **2022**, *426*, 140760. [CrossRef]
54. Zheng, Y.; Gao, W.; Ouyang, M.; Lu, L.; Zhou, L.; Han, X.; Zheng, Y.; Gao, W.; Ouyang, M.; Lu, L.; et al. State-of-charge inconsistency estimation of lithium-ion battery pack using mean-difference model and extended Kalman filter. *J. Power Sources* **2018**, *383*, 50–58. [CrossRef]