Electric Vehicle Enhanced Range, Lifetime And Safety Through INGenious battery management

D1.2 – Report on model order reduction

August 2019

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 713771
<table>
<thead>
<tr>
<th><strong>PROJECT SHEET</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Acronym</strong></td>
</tr>
<tr>
<td><strong>Project Full Title</strong></td>
</tr>
<tr>
<td><strong>Grant Agreement</strong></td>
</tr>
<tr>
<td><strong>Call Identifier</strong></td>
</tr>
<tr>
<td><strong>Topic</strong></td>
</tr>
<tr>
<td><strong>Type of Action</strong></td>
</tr>
<tr>
<td><strong>Project Duration</strong></td>
</tr>
<tr>
<td><strong>Coordinator</strong></td>
</tr>
<tr>
<td><strong>Consortium Partners</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Website</strong></td>
</tr>
<tr>
<td><strong>Title</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Related WP</strong></td>
</tr>
<tr>
<td><strong>Lead Beneficiary</strong></td>
</tr>
<tr>
<td><strong>Author</strong></td>
</tr>
</tbody>
</table>
| **Reviewers**   | Matthieu Ponchant (Siemens PLM)  
An Li (Siemens PLM)  
Tijs Donkers (TUe) |
<p>| <strong>Type</strong>        | Report |
| <strong>Dissemination level</strong> | PUBLIC |
| <strong>Due Date</strong>    | August 30, 2019 |
| <strong>Submission date</strong> | September 30, 2019 |
| <strong>Status and Version</strong> | Final, version 1.0 |</p>
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author/Reviewer</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>08/08/2019</td>
<td>Sturm Johannes <em>(TUM)</em> Lead Beneficiary</td>
<td>Draft version 1.1</td>
</tr>
<tr>
<td>V0.2</td>
<td>23/08/2019</td>
<td>Tijs Donkers <em>(TUe)</em> Matthieu Ponchant <em>(Siemens PLM)</em></td>
<td>Draft version 1.2</td>
</tr>
<tr>
<td>V0.3</td>
<td>28/08/2019</td>
<td>Sturm Johannes <em>(TUM)</em> Tijs Donkers <em>(TUe)</em></td>
<td>Draft version 1.3</td>
</tr>
<tr>
<td>V0.4</td>
<td>12/09/2019</td>
<td>Matthieu Ponchant <em>(Siemens PLM)</em></td>
<td>Final review</td>
</tr>
<tr>
<td>V1.0</td>
<td>30/09/2019</td>
<td>Carlo Mol <em>(VITO)</em> Coordinator</td>
<td>Submission to the EC</td>
</tr>
</tbody>
</table>
DISCLAIMER
The opinion stated in this report reflects the opinion of the authors and not the opinion of the European Commission.

All intellectual property rights are owned by the EVERLASTING consortium members and are protected by the applicable laws. Except where otherwise specified, all document contents are: © EVERLASTING Project - All rights reserved”. Reproduction is not authorised without prior written agreement.

The commercial use of any information contained in this document may require a license from the owner of that information.

All EVERLASTING consortium members are committed to publish accurate information and take the greatest care to do so. However, the EVERLASTING consortium members cannot accept liability for any inaccuracies or omissions nor do they accept liability for any direct, indirect, special, consequential or other losses or damages of any kind arising out of the use of this information.

ACKNOWLEDGEMENT
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 713771
**EXECUTIVE SUMMARY**

The report summarizes the activities and achievements of work package (WP) 1 regarding the “Model reduction, integration and pack model scaling” task (T1.2). From month 6 to 36 of the EVERLASTING project, the work of all participants in T1.2 focussed on the **Model Order Reduction** to find the most suitable implementation of physicochemical models in embedded systems such as microcontrollers used for Battery Management Systems. Numerical efficiency, memory allocation and modelling/parameterization error was investigated towards real experimental data of constant current charge and discharge as well as driving cycle scenarios regarding the EVERLASTING INR18650MJ1 (LGChem) cell. **State-Estimation** models using a physicochemical model together with a non-linear Kalman Filter were developed to evaluate the most suitable observer model configuration to monitor a lithium-ion battery. **Off-Line Simulation Models** have been profoundly investigated to gain minimum calculation time albeit numerically intensive physicochemical model approaches are considered to simulate the electrochemical-thermal behaviour of a lithium-ion battery. A simple method was also proposed to get a **reduced order electro-thermal model** from the physicochemical model, which can be easily integrated into a simulation platform such as Simcenter Amesim.

The report is mainly based on the related publications dealing with the **Model Order Reduction** (“Suitability of physicochemical models for embedded systems regarding a nickel-rich, silicon-graphite lithium-ion battery”), the **State-Estimation** (“State estimation of lithium-ion cells using a physicochemical model based extended Kalman filter”) for lithium-ion batteries and the **Off-Line Simulation Models** (“A computationally efficient implementation of a full and reduced-order electrochemistry-based model for Li-ion batteries” and “On trade-offs between computational complexity and accuracy of electrochemistry-based battery models”).

Data repositories at the TU Delft (i.e. Research Data Center) were published to the related publications and the publications are published as follows:

**Suitability of physicochemical models for embedded systems regarding a nickel-rich, silicon-graphite lithium-ion battery**
In Journal of Power Sources, Volume 436, 1 October 2019, 226834
- [https://mediatum.ub.tum.de/1514862](https://mediatum.ub.tum.de/1514862) (Preprint, Green Open Access)
- [https://doi.org/10.4121/uuid:6590260b-244c-4a17-b134-f7a7cda5b8b5](https://doi.org/10.4121/uuid:6590260b-244c-4a17-b134-f7a7cda5b8b5) (Data-Open Access)

**State estimation of lithium-ion cells using a physicochemical model based extended Kalman filter**
In Journal Applied Energy, Volume 223, 1 August 2018, Pages 103-123
- [https://doi.org/10.1016/j.apenergy.2018.04.011](https://doi.org/10.1016/j.apenergy.2018.04.011) (Gold Open Access)
- [https://doi.org/10.4121/uuid:a588c059-0d82-42fb-931d-8e805da79388](https://doi.org/10.4121/uuid:a588c059-0d82-42fb-931d-8e805da79388) (Data-Open Access)

**A computationally efficient implementation of a full and reduced-order electrochemistry-based model for Li-ion batteries**
In Journal Applied Energy, Volume 208, 1 December 2017, Pages 1285-1296
- [https://doi.org/10.1016/j.apenergy.2017.09.025](https://doi.org/10.1016/j.apenergy.2017.09.025) (Gold Open Access)

**On trade-offs between computational complexity and accuracy of electrochemistry-based battery models**
- Accepted, presentation at the IEEE Conference on Decision and Control (Nice, France) in December 2019 with following open-access manuscript and data repository.
## Table of Contents

### 1 SUITABILITY OF PHYSICOCHEMICAL MODELS FOR EMBEDDED SYSTEMS SUCH AS A MICROCONTROLLER REGARDING A NICKEL-RICH, SILICON-GRAPHITE LITHIUM-ION BATTERY ........................................... 13

1.1 Abstract of activities related to the implementation of physicochemical models in a microcontroller ......................................... 13
1.2 Introduction .............................................................................................................. 13
1.3 Model reduction of the P2D physicochemical model ............................................... 14
1.4 Modelling of a 18650 NMC-811/SiC lithium-ion battery ........................................ 15
1.4.1 Equivalent circuit model ........................................................................................ 16
1.4.2 PP- and EM-PCM using FDM and solid-diffusion approximation ................................. 17
1.4.3 OC-PCM using orthogonal collocation on Chebyshev nodes ................................ 18
1.4.4 Rigorous COMSOL-PCM ..................................................................................... 20
1.4.5 Spatial configuration and DAE size of the PCMs .................................................. 20
1.5 Microcontroller implementation .............................................................................. 20
1.5.1 PP- and EM-PCM stand-alone C-code ................................................................... 21
1.5.2 OC-PCM stand-alone C-Code ............................................................................... 23
1.6 Results and discussion ................................................................................................. 25
1.6.1 Validation and computational efficiency under constant load scenarios ............... 25
1.6.2 Reducing memory allocation via coarser spatial discretizations .............................. 27
1.6.3 Stand-alone C-code models on the microcontroller under constant load scenario ... 29
1.6.4 Validation and computational efficiency under driving cycle scenario .................. 35
1.7 Conclusions for model order reduction and implementation into microcontroller .... 38
1.8 References .................................................................................................................. 39
1.9 Appendix ................................................................................................................... 43

### 2 STATE ESTIMATION OF LITHIUM-ION CELLS USING A PHYSICOCHEMICAL MODEL BASED EXTENDED KALMAN FILTER .............................................. 47

2.1 Abstract of activities related to the state estimation of lithium-ion batteries using physicochemical models ..................................................... 47
2.2 Comparison of the state estimation performance for different simulated charging scenarios ................................................................. 49
2.3 Conclusion from state estimation process ................................................................. 51

### 3 OFF-LINE SIMULATION MODELS TO GAIN MINIMUM CALCULATION TIME CONSIDERING PHYSICOCHEMICAL MODEL APPROACHES .......................................................... 52

3.1 A computationally efficient implementation of a full and reduced-order electrochemistry-based model for Li-ion batteries ................................................................. 52
3.1.1 Introduction ............................................................................................................. 52
3.1.2 Model of lithium-ion batteries .............................................................................. 54
3.1.3 Model implementation ........................................................................................... 56
3.1.4 Model order reduction using POD and DEIM ......................................................... 58
3.1.5 Simulation study of proposed numerical scheme .................................................. 60
3.1.6 Simulation study of POD-DEIM-Based Reduced-Order Model ....... 64
3.1.7 Conclusions .......................................................................................................... 68
3.1.8 References .............................................................................................................. 69
3.2 On trade-offs between computational complexity and accuracy of electrochemistry-based battery models ................................................................................................................................................................................. 71
3.2.1 Introduction .................................................................................................................................................................................. 71
3.2.2 Battery Modelling .............................................................................................................................................................................. 72
3.2.3 Model Implementation ........................................................................................................................................................................... 75
3.2.4 Simulation Study ................................................................................................................................................................................ 77
3.2.5 Model Comparison ................................................................................................................................................................................. 80
3.2.6 Conclusions ....................................................................................................................................................................................... 82
3.2.7 References ....................................................................................................................................................................................... 83

4 Reduced order electro-thermal model ready for SIL and HIL BMS evaluation 85
4.1 Introduction ....................................................................................................................................................................................... 85
4.2 Electrochemical Model ......................................................................................................................................................................... 85
4.3 Reduced-order electro-thermal model ................................................................................................................................................ 86
4.3.1 Proposed model ................................................................................................................................................................................... 86
4.3.2 Model calibration ................................................................................................................................................................................. 89
4.3.3 Model validation ................................................................................................................................................................................ 92
4.4 Conclusions ....................................................................................................................................................................................... 96
4.5 References ....................................................................................................................................................................................... 97
# List of Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>ACRONYM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>18650</td>
<td>Cylindrical Cell Format with diameter of 18 mm and height of 65 mm</td>
</tr>
<tr>
<td>C</td>
<td>Graphite</td>
</tr>
<tr>
<td>CC</td>
<td>Constant Current</td>
</tr>
<tr>
<td>CV</td>
<td>Constant Voltage</td>
</tr>
<tr>
<td>DAE</td>
<td>Differential Algebraic Equation</td>
</tr>
<tr>
<td>DFN</td>
<td>Doyle-Fuller-Newman</td>
</tr>
<tr>
<td>DOD</td>
<td>Depth of Discharge</td>
</tr>
<tr>
<td>DVA</td>
<td>Differential Voltage Analysis</td>
</tr>
<tr>
<td>EIS</td>
<td>Electrochemical Impedance Spectroscopy</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method</td>
</tr>
<tr>
<td>MJ1</td>
<td>Everlasting Project Cell – INR18650-MJ1 from LGChem</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>NMC-811</td>
<td>Nickel Manganese Cobalt oxide in relation 8-1-1 (wt.-%)</td>
</tr>
<tr>
<td>OCP</td>
<td>Open Circuit Potential</td>
</tr>
<tr>
<td>OCV</td>
<td>Open Circuit Voltage</td>
</tr>
<tr>
<td>p2D</td>
<td>Pseudo Two-Dimensional</td>
</tr>
<tr>
<td>SiC</td>
<td>Silicon doped graphite</td>
</tr>
<tr>
<td>TUM</td>
<td>Technische Universität München</td>
</tr>
<tr>
<td>TUE</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>VITO</td>
<td>Unit of Energy Technology</td>
</tr>
<tr>
<td>WP</td>
<td>Work package</td>
</tr>
<tr>
<td>WPL</td>
<td>Work package leader</td>
</tr>
</tbody>
</table>
Figure 1 Overview of models .............................................................. 15
Figure 2 Measurement procedure to parameterize the ECM .......................... 16
Figure 3 Simplified flow chart of the stand-alone codes of PP- and EM-PCM implemented in MATLAB® 2017b. The approximation of the solid-diffusion PDE ('Mass balance (solid)') is either implemented via the polynomial profile (PP) [11] or an eigenfunction method (EM) [25] for the PP- and EM-PCM, respectively. ................................................................. 17
Figure 4 Simplified flow chart of the stand-alone-code of the OC-PCM implemented in MATLAB® 2017b. The spatial discretization is reformulated using Chebyshev orthogonal collocation [48] and the resulting DAE system is solved using an ODE solver (ode15s [54,64,65]) of MATLAB®. ...................... 19
Figure 5 Spatial discretization of the PCMs .................................................. 20
Figure 6 Flow chart of the stand-alone C-codes of the PP-/EM-PCM (left) and the OC-PCM (right) implemented in the microcontroller (STM32F407VG/T6, STMicroelectronics [9]). The PP-/EM-PCM and OC-PCM routines refer to the p2D-PCM models shown in Figure 3 and 4. The generated simulation results are sent per converged time step via an UART-to-USB converter to a desktop computer and evaluated in terms of computation performance and simulation accuracy. ........................................... 21
Figure 7 Usage of memory regarding the stand-alone C-code PP- and EM-PCM ........................................ 22
Figure 8 Usage of memory regarding the stand-alone C-code OC-PCM .......................... 24
Figure 9 Experimental validation with the INR18650-MJ1 NMC-811/SiC LIB of the simulation results including the ECM, the MATLAB®-code PCMs ('PP-PCM 10-5-10', 'OC-PCM 20-10-20-25' and 'EM-PCM 10-5-10') and the rigorous benchmark PCM ('COMSOL-PCM 53-8-40-20'). The mean cell voltage error (a, b) and temperature error (c, d) are shown together with the overall computation time (e, f) and averaged calculation time per converged time step (g, h) for the different CC charge (a, c, e and g) and discharge (b, d, f and h) tests at 25°C ambient temperature and convective cooling conditions [1]. .............................................................................................. 26
Figure 10 The relative (%) increase of modelling error for the cell voltage (a, b), the cell temperature (c, d), the surface concentration c_s (e, f) and the potential drop Φ_s (g and h) both located at the anode-separator interface x=L_{neg} is shown for gradually reducing the total number of spatial nodes in the MATLAB®-code PCMs (PP-, EM- and OC-PCM) in reference to the experimentally validated PCMs shown in Figure 9 ('PP-PCM 10-5-10', 'EM-PCM 10-5-10' and 'OC-PCM 20-10-20-25'). ........................ 28
Figure 11 Performance of C-code PCMs on the microcontroller at 1C-CC ..................................... 30
Figure 12 Evaluation of the simulation accuracy (a, b, c and d) and computational efficiency (e, f, g, and h) of the stand-alone C-code PP-, EM- and OC-PCM under constant current charge (0.2, 0.5, and 1C) and discharge (0.2, 0.5, 1, 1.5 and 2C) scenarios. The mean cell voltage (a, b) and temperature error (c, d) is shown in reference to the corresponding MATLAB®-code PCM for the coarse (2-1-2, 2-1-2 and 5-3-5-3 for PP-, EM- and OC-PCM, 'Coarse configurations') and maximum spatial configuration. The computational efficiency is shown similarly in form of the iteration time (e, f) and the mean number of iteration per 1 s step (g, h) .......................................................... 31
Figure 13 Computational performance of the C-code PP-, EM- and OC-PCM for simulating 1C CC charge (a and c) and discharge (b and d) on the STM32 at 50, 109 and 168 MHz. The horizontal line depicted in 'blue' marks the real-time threshold when the simulated time equals the total computation time (see 'Time factor = 1' in a and b) and the average iteration time per converged step equals the simulation step-size of 1 s (see '1000 ms' in c and d). .......................................................... 33
Figure 14 Computational performance of C-code PCMs at different processor frequencies ........................ 34
Figure 15 Computational performance of C-Code PCMs using FPU for 1C CC charge and discharge 34
Figure 16 Measurement results of the INR18650-MJ1 LIB ('Measurement MJ1') for the applied current (a), cell voltage (b) and temperature (f) under the adapted ARTEMIS [85] driving cycle. The subplot (c) shows a magnified part from 1200 to 1260 s for the measured and simulated (MATLAB®-code 'PP-PCM 5-3-5', 'EM-PCM 5-3-5' and 'OC-PCM 5-3-5-5') cell voltages in (b). The resulting error profiles for the PCMs are shown for the cell voltage (d) with a magnified part similar to (c). Subplot (g) shows the temperature error. ................................................................................................................. 36
Figure 17 Validation of the MATLAB®-code PCMs under the adapted ARTEMIS driving cycle. ........ 37
Figure 18 Computational performance of C-code PCMs simulating adapted ARTEMIS driving cycle. 37

D1.2 – Report on model order reduction
Author: Johannes Sturm (TUM) – August 2019
EVERLASTING - Grant Agreement 71377 (Call: H2020-GV8-2015)
Electric Vehicle Enhanced Range, Lifetime And Safety Through INGenious battery management
Figure 19 Summary of average cell voltage error for MATLAB and C-code PCMs. .................................................. 38
Figure 20 Physical properties of the electrolyte [87]. .............................................................................................. 43
Figure 21 Parameterization of p2D-PCM regarding the NMC-811/SiC LIB [88] (i.e. INR18650MJ1). 44
Figure 22 Differential algebraic equation system for the p2D-PCM. .................................................................. 44
Figure 23 Nomenclature I. ........................................................................................................................................ 45
Figure 24 Nomenclature II. ...................................................................................................................................... 46
Figure 25 Schematic representation of a MCMB/LiCoO₂ lithium-ion cell shown in the upper part with the subsequent pseudo two-dimensional reduction to the PCM [6]. The fundamental model equations are depicted for the solid (i.e. active material) and liquid (i.e. electrolyte) phase, including three different numerical approximation methods (i.e. DS1, PP and EM) for the solid-phase diffusion. The PP- and EM-approach are used for the development of the recursive state observer models using the EKF-algorithm. .......................................................... 48
Figure 26 Overview of the state-estimation process using the PP-PCM with an EKF. The dynamic state vector (x) differs for the EM-PCM based observer model in terms of neglecting the volume-averaged concentration flux (qs). ......................................................................................................................... 49
Figure 27 The regression of the SoC error based on the average lithium concentration within the cathode domain is shown for the CCCV (a), BCCV (b) and the PCCV (c) scenario for both observer models. ........................................................................................................................................ 50
Figure 28 Schematic of the Li-ion battery model during discharging................................................................. 54
Figure 29 Comparison of the battery voltages during the constant 5 C-rate discharge and starting from 100% SOC with (left) different size of grids for each phase, (right) different convergent tolerances and temporal discretisation steps. ........................................................................................................................ 60
Figure 30 Concentration distributions and reaction current density during the constant 5 C-rate discharge at various times, starting from 50% SOC. The electrode surface concentration, electrolyte concentration and reaction current density are shown, respectively. .................................................................................... 62
Figure 31 Voltage response for different model implementations, starting from 100% SOC during (a) constant 1 C-rate discharge, (b) dynamic (dis)charge profile accompanied by the current profile. . 63
Figure 32 Comparison of the full order model and the reduced order model with snapshots constructed at different maximum C-rates in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure). .................................................................................................................................. 64
Figure 33 Comparison of the full order model and the reduced order model with snapshots constructed at different numbers of reduced basis in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure)........................................................................... 65
Figure 34 Comparison of the full order model and the reduced order model implemented using different convergence tolerances in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure). .................................................................................................................................. 65
Figure 35 Comparison of the battery voltages between the full-order model and the POD-DEIM-based ROM, during a constant 5 C-rate discharge starting from 100% SOC. .............................................................................................. 66
Figure 36 : Comparison of the battery voltages between the full-order model and the POD-DEIM-based ROM, during a pulse charge/discharge starting from 100% SOC.......................................................... 67
Figure 37 Concentration distributions and reaction current density of the POD-DEIM-based ROM during the constant 5 C-rate discharge at various times, starting from 50% SOC. The electrode surface concentration, electrolyte concentration and reaction current density are shown, respectively...... 68
Figure 38 DFN modelling approach for a Li-ion cell. ................................................................................................ 72
Figure 39 Comparison of output voltage and root-mean-square error (RMSE) between the full DFN model and the simplified DFN model, with two different parameter sets which represent a high power (HP) battery [19] and a high energy (HE) [21] battery, where the initial state-of-charge is 20%.. 78
Figure 40 RMS error of the output voltage and computation time for varying grid parameters for the high-power (HP) [19] and the high-energy (HE) [21] parameter set. .................................................................................................................... 79
Figure 41 Output voltage and normalized electrolyte concentration for varying degrees of model order for the high-power (HP) [19] and the high-energy (HE) [21] parameter set. ...................................................................................... 80
Figure 42 Discharge curve and normalized electrolyte concentration for the DFN model (solid lines), SDFN model (dashed lines), and SPMe (dotted lines) at several C-rates, simulated for the high-power (HP) [19] and the high-energy (HE) [21] parameter set. ................................. 81
Figure 43 Output voltage using various models for the high-power (HP) [19] and the high-energy (HE) [21] parameter set. The applied current profile is an electric bike drive cycle from [22].

Figure 44 Reduced order circuit model

Figure 45 OCV transition between its high and low boundaries

Figure 46 Thermal model of the cell

Figure 47 Test 1: pulses test

Figure 48 Test 2: charge test

Figure 49 OCVd identified at different temperatures

Figure 50 Hysteresis between OCVd and OCVc at 25 °C

Figure 51 dU/dT used in the reduced order circuit model

Figure 52 Warburg impedance at 25 °C

Figure 53 Simulation sketch for model validation in Simcenter Amesim

Figure 54 Validation test profile

Figure 55 Validation test at 5 °C of ambient temperature (NRC = 1)

Figure 56 Validation test at 25 °C of ambient temperature (NRC = 1)

Figure 57 Validation test at 45 °C of ambient temperature (NRC = 1)

LIST OF TABLES

Table 1 Error regression of cell voltage for the observer models

Table 2 Comparison of computation time for different orders of subsystems solved during the constant 5 C-rate discharge, starting from 100% SOC

Table 3 Comparison between the computation time of the full-order model and the POD-DEIM-based model

Table 4 The effect of assumptions [A1]-[A4] introduced in Section 2 on RMS error (RMSE) for the high-power (HP) [19] and high-energy (HE) [21] parameter set

Table 5 Selected grid parameters of the varying degrees of model orders shown in Fig. 40

Table 6 The RMS errors (RMSE) of the output voltage V and computation times corresponding to the simulations shown in Figure 41

Table 7 Parameters for the reduced order circuit model

Table 8 Six combinations to calculate entropic coefficient

Table 9 Values of parameters for the thermal model

Table 10 RMS error for voltage estimation

Table 11 RMS error for temperature estimation
1 Suitability of physicochemical models for embedded systems such as a microcontroller regarding a nickel-rich, silicon-graphite lithium-ion battery

The basic methodology and results dealing with the model order reduction and implementation of the physicochemical model in a microcontroller was published in the Journal of Power Sources (Volume 436, 1 October 2019, 226834) [1] and subsequent sections are based on this publication referring to:

- https://doi.org/10.1016/j.jpowsour.2019.226834 (Under Subscription)
- https://mediatum.ub.tum.de/1514862 (Preprint, Green Open Access)

The accompanied repository of the published data is accessible as well under:
https://doi.org/10.4121/uuid:6590260b-244c-4a17-b134-f7a7cda5b8b5 (Open Access)

1.1 Abstract of Activities Related to the Implementation of Physicochemical Models in a Microcontroller

Local inhomogeneous electrode utilization in recent lithium-ion batteries tends to increase due to larger sizes and/or higher densification, which poses a challenge for accurate, model-based monitoring. Pseudo-two dimensional (p2D) physicochemical models (PCM) can offer such locality via calculating local potentials and concentrations through the thickness of the electrode stack and are numerically reduced for implementation in a microcontroller in this work. Finite difference method combined with solid-diffusion approximations and orthogonal collocation reformulation are applied to generate three MATLAB®- and three microcontroller-suitable C-code p2D-PCMs, which are experimentally validated towards constant current charge/discharge and driving cycle loads on a high-energy NMC-811/SiC-18650 lithium-ion battery. Benchmarking to an equivalent circuit model reveals similar mean cell voltage errors below 20 mV for the driving cycle. Reducing spatial elements reveals errors below 1 % for local (i.e. concentrations/potentials) and global states (i.e. cell voltage/temperature) and is applied to speed-up the C-code p2D-PCMs in the microcontroller (max. 168 MHz with 192 kB RAM) to calculate at least 37 % faster than real-time. Real-time computability is investigated via varying processor frequencies and using hardware acceleration schemes. The memory allocation to solve and store the p2D-PCMs on the microcontroller require 115 kB and 213 kB at a maximum, respectively.

1.2 Introduction

Recent achievements in higher energy density of lithium-ion batteries (LIBs) promote inhomogeneous usage [2] either along the electrodes or through the thickness of the cell stack [1]. Therefore, monitoring and controlling of the battery’s states on local scale are necessary to guarantee efficient utilization and safety during both dynamic (i.e. driving cycle) or rather static loads (i.e. fast charging). Beside larger electrodes, thicker composite coatings, higher densification (i.e. porosity <20 %) and high capacitive active materials such as nickel-rich cathodes (e.g. NMC-811) and graphite-silicon composite anodes (SiC) are applied to increase the energy density. The resulting increase of capacity can lead to local current densities along the electrodes exemplarily up to 4.91 mA cm⁻² at 1C for a 3.35 Ah 18560 LIB (INR18650-MJ1, LGChem) incorporating low electrode porosities of 21.6 % and 17.1 % for the SiC anode and NMC-811 cathode [1]. As a result, inhomogeneous utilization through the cell stack and along the electrodes appears [2]. Beside the global states such as cell voltage, applied current and surface temperature, proper LIB monitoring should estimate also the local states such as potentials and concentrations in the...
electrolyte and the active material to ease harmful side-reactions such as lithium plating [3,4] or solid-electrolyte-interphase (SEI) growth and cracking [5] or to avoid critical hot spots [2]. State-of-the-art model-based monitoring incorporate equivalent circuit models (ECMs) as it offers fast calculation and easy parameterization. However, only global states can be simulated. Physicochemical models (PCMs) such as the pseudo-two dimensional (p2D) model [6] offer simulated local states based on porous electrode, concentrated solution theory and electrode kinetics through the thickness of the cell stack. However, this model comes with computational complexity due to solving its differential algebraic equation (DAE) system, which significantly slows down the calculation. Together with the complex parameterization, application in battery management system (BMS) outside the research field is hindered.

In this matter, we want to investigate the suitability of the p2D model in embedded systems (i.e. microcontroller) via evaluating the computational performance and simulation accuracy of p2D-PCMs using different spatial and time discretization’s, approximation schemes for the particle domain and solvers. Three different p2D-PCMs are parameterized for a 18650 NMC-811/SiC LIB (INR18650-MJ1 [1]) and implemented first in MATLAB® and second transferred into a stand-alone C-code for microcontroller implementation. Errors of parameterization, model reduction, transfer into the microcontroller and validation via measurements are outlined for constant current (CC) charge and discharge and a driving cycle scenario to evaluate the suitability of the p2D-PCMs for real-time simulation in embedded systems.

## 1.3 Model reduction of the P2D physicochemical model

To ease the computational inefficiency, model reduction can be applied to the p2D-PCM [6], which is summarized in fundamental reviews [7,8]. In this work, its actual implementation in the STM32F407VGT6 microcontroller (STM32, STMicroelectronics [9]) is evaluated towards computation speed and simulation accuracy, coming with crucial limitations in computation power (max. 168 MHz in a 32-bit ARM® Cortex®-M4 core) and memory resources offering only a maximum of 1024 kB flash memory to store and a maximum of 192 kB static random accessible memory (RAM) to solve the model. These limitations are often not considered in research as reductions are investigated on desktop computers.

In this matter, low spatial discretization’s with sufficient accuracy are favoured as the total number of spatial elements defines the size of the DAE, the related memory requirements and thus the computational effort. Spatial discretization of the particle domain (i.e. ‘pseudo’-domain, r-coordinate) can cause a large DAE system via discretizing the solid-diffusion partial differential equation (PDE). At every node in the electrolyte domain (x-coordinate), this PDE is solved for the concentration $c_s$ of lithium-ions, which tremendously raises the allocated memory. As only the particle-surface concentration $c_{ss}$ is needed for the kinetics, its numerical reduction is feasible. Approximation methods for the concentration profile in the particle were implemented in literature via volume averaging together with a parabolic profile (PP) [10–21] or diffusion length approaches [22–24], which suggests linearity between surface- and average-concentration $c_{s,ave}$. Sufficient accuracy and computation efficiency appeared via using an eigenfunction method (EM) [25–28], where the solution is derived from a truncated, analytical solution of an infinite series of eigenfunctions. Maintaining a spatial discretization of the PDE, reformulations to an ordinary differential equation (ODE) in time via spectral methods (e.g. orthogonal collocation (OC)) [29–35] showed an enormous calculation acceleration while guaranteeing sufficient accuracy. Also standard discretization schemes such as finite volume methods (FVM) (e.g. finite difference method (FDM) or finite element method (FEM)) [30,36–38] were investigated next to integral methods such as the duhamel superposition integral (DSI) [6,39–42]. Reducing the entire solid phase to a single particle (SPM, [16,43–45]) revealed promising computational efficiency, but is not regarded in the following as the original p2D-PCM [6] is focussed in this work. Referring to real-time computability and sufficient simulation accuracy as seen in our previous work [46], we focus on a FDM discretization for the electrolyte domain accompanied with two different approximation schemes (i.e. PP- [11] and EM [25]) for the particle domain. For comparison, the third p2D-PCM uses orthogonal collocation and thus maintains a spatial discretization of the particle domain. As the PP- [10] and EM-approximation are adopted from the corresponding
original work, which have already shown its validity, accuracy and computational efficiency, the reader is referred to these works [13–15,17,26,27,30,47] for more information. The OC-method was used as well in literature before [31,48] and a single work [34] investigated the performance on a microcontroller (ATMEL 32UC3A1512 at 16MHz and 512 kB RAM [34]) including 21 DAE which could be solved in at least 190 ms under 1C CC discharge. Unfortunately, no description of the actual implementation on this microcontroller is shown [34] and the work misses implementation recommendations, detailed computational performance analysis and application-near load scenarios. According to literature, the PP-, EM- and OC-PCM offer significant computation speed, reduction of DAE size and maintain sufficient accuracy to be implemented in the STM32 [9] used in this work. Far too little work [34] had focussed on an actual microcontroller implementation of the p2D-PCM in the past, which is one of the main objectives of this work together with evaluating the most suitable discretization/approximation scheme to gain real-time computation and low simulation error in embedded systems.

## 1.4 MODELLING OF A 18650 NMC-811/SiC LITHIUM-ION BATTERY

As shown in Figure 1, three different p2D-PCMs are investigated incorporating different spatial discretization's/approximations and two benchmark models (COMSOL-PCM and ECM) are used to simulate a 18650 NMC-811/SiC LIB [1].

<table>
<thead>
<tr>
<th>Model</th>
<th>Spatial discretization</th>
<th>Framework</th>
<th>Thermal model</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP-PCM</td>
<td>FDM</td>
<td>Parabolic</td>
<td>✓</td>
</tr>
<tr>
<td>EM-PCM</td>
<td>FDM</td>
<td>Eigenfunction</td>
<td>✓</td>
</tr>
<tr>
<td>OC-PCM</td>
<td>Orthogonal collocation</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>COMSOL-PCM</td>
<td>FDM</td>
<td>COMSOL</td>
<td>✓</td>
</tr>
<tr>
<td>ECM</td>
<td>n.a.</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

† Ref.[11] ‡ Ref.[25] †† Ref.[50] IV for STM32 microcontroller ††† Ref. [51]

**Figure 1 Overview of models**

The three p2D-PCMs are used as stand-alone MATLAB®- and C-code to simulate on a desktop computer and the STM32 microcontroller, respectively. The MATLAB®-codes are used for model parameterization and validation, determining the simulation error via reducing spatial discretization and evaluate the implementation error for the C-code equivalents in the microcontroller. As benchmarks, the COMSOL-PCM uses COMSOL Multiphysics® and the ECM is implemented in MATLAB/Simulink to simulate on a desktop computer. The PP- and EM-PCM revealed different suitability in terms of constant and dynamic loads [46] and are chosen in this work for evaluating a standard, equidistant spatial FDM discretization paired with different solid-approximations on a microcontroller instead of hardware and software oversized desktop PCs, which are not suitable to evaluate embedded system applicability. The OC-PCM uses no solid-approximation and no standard FDM-discretization, but a reformulation of the p2D-PCM equations to exclusively ODEs in time via Chebyshev orthogonal collocation, which revealed distinct speed-up on desktop PCs compared to models like the PP- and EM-PCM. However, this reformulation must be evaluated in a microcontroller to evaluate its suitability for embedded systems, which is investigated here. The DAE system of the p2D-PCM is shown in Figure 35 and the parameterization [1] is shown in Figure 33 and 34.
1.4.1 EQUIVALENT CIRCUIT MODEL

The ECM consists of a single capacitor/resistor network $R_1$ and $C_1$ ("RC"), an ohmic resistance $R_i$ and an open-circuit voltage $V_{OCCV}$. As the temperature has significant influence on the cell behaviour [49], the parameterization tests of the ECM were proceeded at 25 and 40°C beside the state of charge (SoC) dependency. This first-order model [50] offers the best compromise of accuracy and complexity [51] incorporating the fundamental equations as

\[
\begin{align*}
I_{cell} &= C_1 \frac{d V_1}{d t} + \frac{V_1}{R_1} \\
V_{cell} &= V_{OCCV} + V_1 + I_{cell} \cdot R_i \\
m \cdot c_p \frac{d T_{cell}}{d t} &= \left( V_{OCCV} - V_{cell} - \frac{d V_{OCCV}}{d t} \cdot T_{cell} \right) \cdot I_{cell} + I_{cell}^2 \cdot R_i - \alpha_\infty A_{surf} (T_{cell} - T_\infty)
\end{align*}
\]

With $I_{cell} > 0$ for charge and $I_{cell} < 0$ for discharge.

The resistance $R_i$ represents the ohmic resistance on the current collector foils, the RC network accounts for any transient dynamics referring to electrochemical processes [52] and the voltage source $V_{OCCV}$ represents the equilibrium state. To parameterize the ECM variables (i.e. $V_{OCCV}$, $R_i$, $R_1$, $C_1$, $\frac{d V_{OCCV}}{d t}$), three different INR18650-MJ1 cells were tested and the generated data was averaged and interpolated in 1 \% SoC steps. The tests included CC, constant voltage (CV), pulse current (PC) and electrochemical impedance spectroscopy (EIS) periods as summarized in Figure 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Measurement</th>
<th>Voltage (V)</th>
<th>Current (A)</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{OCCV}$</td>
<td>CC charge/discharge</td>
<td>2.5 - 4.2</td>
<td>0.033C</td>
<td>25 °C</td>
</tr>
<tr>
<td></td>
<td>CV</td>
<td>2.5 / 4.2</td>
<td>-0.01C/-0.01C</td>
<td></td>
</tr>
<tr>
<td>$\frac{d V_{OCCV}}{d t}$</td>
<td>CC charge/discharge</td>
<td>2.5 - 4.2</td>
<td>0.2C</td>
<td>ARCRef. [53]</td>
</tr>
<tr>
<td></td>
<td>Potentiometric method according to Zilberman et al. [55]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_1$, $C_1$</td>
<td>PC</td>
<td>2.5 - 4.2</td>
<td>±0.5/1C</td>
<td>25 °C</td>
</tr>
<tr>
<td></td>
<td>for 10/20 s</td>
<td></td>
<td>40 °C</td>
<td></td>
</tr>
<tr>
<td>$R_i$</td>
<td>PC/EIS III</td>
<td>2.5 - 4.2</td>
<td>±0.5/1C</td>
<td>25 °C</td>
</tr>
<tr>
<td></td>
<td>for 10/20 s</td>
<td></td>
<td>40 °C</td>
<td></td>
</tr>
</tbody>
</table>

1 Measurement equipment (BaSyTec CTS) defines charge > 0 and discharge < 0 ^1 1 h rest before PC; applied in 10 % SoC steps from 2.5 to 4.2 V and vice versa. ^II EIS at 0.042C before PC and zero-crossing at Re[Z] = 0 as initial point for the fitting algorithm of $R_i$

Figure 2 Measurement procedure to parameterize the ECM

In terms of $V_{OCCV}$, the charge and discharge measurement were averaged to compensate cell polarization effects. The entropic coefficient $\frac{d V_{OCCV}}{d t}$ was derived from accelerated rate calorimetry (ARC) [1] and validated via the potentiometric method according to Zilberman et al. [53]. The passive components $R_i$, $R_1$, and $C_1$ were parameterized by pulse fitting as depicted in Figure 2. 88 pulses for each temperature were used and a graphical illustration of the ECM and its parameters is shown in our supplementary part. The input variable is the applied current $I_{cell}$ from which the SoC variable is integrated over time. The ECM is implemented in MATLAB®/Simulink and solved via the ode14x [54] solver at a step-size of 1 s. The solving process and the necessary parameterization files are expected neither to overload the RAM and flash memory nor to exceed the computation power of the STM32 and other works [55–57] have already shown the actual implementation in microcontrollers. As this work focusses on the implementation and solving of the p2D-PCM, the ECM is not transferred into the microcontroller but used as a benchmark for state-of-the-art model-based monitoring of LIBs in real-time operating systems and is referenced for computation speed and simulation accuracy of the MATLAB®-code PCMs.
1.4.2 PP- AND EM-PCM USING FDM AND SOLID-DIFFUSION APPROXIMATION

The PP- and EM-PCM were already presented in our previous work [46] in terms of steady-state representation together with a non-linear Kalman Filter [58–60] for state estimation of a LiCoO2/LiC6 LIB. In this work, the MATLAB®-code PP- and EM-PCM are parameterized for a NMC-811/SiC LIB and transferred into stand-alone C-codes for the microcontroller. Figure 3 shows the flow chart of the PP- and EM-PCM, which differ in the approximation of the solid-diffusion PDE ('Mass Balance (solid)').

![Flow Chart of PP- and EM-PCM](image)

**Figure 3** Simplified flow chart of the stand-alone-codes of PP- and EM-PCM implemented in MATLAB® 2017b. The approximation of the solid-diffusion PDE ('Mass balance (solid)') is either implemented via the polynomial profile (PP) [11] or an eigenfunction method (EM) [25] for the PP- and EM-PCM, respectively.

The MATLAB® codes start with the parameterization (see Figure 33 and 34) and calculate the initial states by assuming an equilibrium state [42]. In the main part, a new time period $\Delta t$ is added until a stop condition as $t_{\text{max}}, V_{\text{min}}, V_{\text{max}}$ is met. The initial state vector for the iterative time step $k$ is set to the previous, consistent solution $k-1$ and the model is run to compute a new consistent solution of the state variables $x_k$. The iterative approximation $i$ of the model equations refers to every node $j$ in the electrolyte domain and calculates the model equations $\mathbf{g}$ (see Figure 24) and the corresponding jacobian $\mathbf{J}$. The Crank-Nicolson method [40] is used for first order time derivatives [46] and the time step is set to 1 s. Next, the matrix inversion [61] ($\mathbf{A}/\mathbf{b}$, MATLAB®) is used to generate the state update $\mathbf{dx}$. Note, that the temperature is calculated afterwards [46] according to heat generation $q$ and heat loss to the ambience $q_{\text{∞}}$. The heat calculation proposed from COMSOL Multiphysics® and other works [48] revealed similar results with small deviations up to 0.2 /0.7 % on average for 1C CC charge/discharge as the computational less expensive calculation [49,62,63] used in this work (see Figure 35).

To conclude, the PP- and EM-PCM use FDM together with solid-diffusion approximation and in this work we want to evaluate, if such standard techniques are sufficient to gain real-time computability of the p2D-PCM in the STM32 microcontroller.
1.4.3 OC-PCM using orthogonal collocation on Chebyshev nodes

The OC-PCM uses a spectral method to reformulate the spatial discretization on Chebyshev collocation nodes [48] of the DAEs in both the $x$ and $r$ domain. The resulting ODEs in time and algebraic equations (AEs) form a DAE system as

$$Mx' = f(t_k, x)$$

which is solved via the `ode15s` [54,64,65] solver of MATLAB®. In terms of the reformulation, an unknown continuous function is approximated by a polynomial, which is determined by its values at the so called Chebyshev nodes (i.e. $x_j = \cos\left(\frac{n_j \pi}{n}\right)$) for a given number of nodes $j$ after rescaling each domain into [-1,1] [48]. In contrast to other work [48], the electrolyte potential $\Phi_i$ accounts for activity formulation $f_\pm$ [6]

$$\frac{\partial \Phi_i(x,t)}{\partial x} = \frac{i_i(x,t)}{\kappa_{eff}} + \frac{2RT}{F}(1-t_0^\ddagger) \cdot \left(\frac{d \ln f_\pm}{d \ln c_i(x,t)}\right) \cdot \frac{\partial \ln c_i(x,t)}{\partial x}$$

and the temperature calculation is identical to the PP- and EM-PCM [49,62,63]. Figure 4 shows the flow chart of the OC-PCM. It starts with the calculation of the required Chebyshev differentiation matrix $D$, which is calculated once together with the mass matrix $M$ and the Clenshaw quadrature weights $\omega$ to evaluate the sum of finite integrals to determine the jacobian matrix $J$. Next, the initialization of the ODE solver (`ode15s` [54,64,65]) is defined via calculating an initial jacobian (`daeic12` [66]) and a first time step is estimated. The following Newton iterative solving uses the same thresholds for the tolerances $\epsilon_{abs}$ and $\epsilon_{rel}$ within the cell voltage range and the time span as given for the PP- and EM-PCM. If four iterations offer no convergence, the solver updates the jacobian and respectively the iteration matrix $M_i$ and the process starts again. If the jacobian is current and no convergence is expected, the step-size is decreased [64]. The solver itself uses a linearly implicit, one-step method based on numerical differentiation formulas (NDFs) implemented in backward differences [64,65,67], which uses an iteration matrix $M_i$ as

$$M_i = M - \frac{\Delta t}{(1-k)\gamma_k} \cdot J$$

to generate the state update $dx$. The term $\gamma_k$ represents the coefficients of the NDFs [64], $k$ the order of the NDF and $\kappa$ is a scalar factor [67].
D1.2 – Report on model order reduction
Author: Johannes Sturm (TUM) – August 2019

EVERLASTING - Grant Agreement 71377 (Call: H2020-GV8-2015)
Electric Vehicle Enhanced Range, Lifetime And Safety Through INGenious battery management

Figure 4 Simplified flow chart of the stand-alone-code of the OC-PCM implemented in MATLAB® 2017b. The spatial discretization is reformulated using Chebyshev orthogonal collocation [48] and the resulting DAE system is solved using an ODE solver (ode15s [54,64,65]) of MATLAB®.

At this point, the reader is referred to the original work [64,65] for more information about the solver itself and the iterative state update is outlined in more detail in our supplementary part. Note, the jacobian is calculated analytically at every spatial node $j$ for every derivative $\frac{\partial f}{\partial x}$ and passed directly to the solver instead of using the incorporated ode15s FDM. Thus, a discretization of 10-10-10-30 (i.e. 'anode-separator-cathode-particle’ domain) reveals approximately a 20 times faster calculation as seen in this work. Further improvement was achieved by using sparse structure (sparse [68]) of the jacobian and the mass matrix. Even if the used spectral method leads to full differentiation matrices while the jacobian for the DAE system is still sparse ($\approx 4\%$ are non-zero), using sparse linear algebra reduces the computational cost by a factor of 4 (referring to 10-10-10-30), which tends to increase for finer discretization’s.

In sum, the OC-PCM uses reformulation, which shows significant computational acceleration of the solving process on a desktop computer. In this work we evaluate the transfer of the OC-PCM into a stand-alone C-code including the ODE solver and the real-time ability of simulating a LIB on a microcontroller.
1.4.4 Rigorous COMSOL-PCM

As a benchmark, the liion-model [69] of COMSOL Multiphysics® is used and run via the LiveLink [44] application using MATLAB® 2017b. The approximation functions are set to linear and a total 53, 8 and 40 of spatial nodes in the anode, separator and cathode domain are used with 20 nodes in the particle domain. The temperature is calculated with a single ODE (see Figure 35). The DAE system is solved with the 'Multifrontal massively parallel sparse direct solver' (MUMPS) [70] at a fixed step-size of 1 s.

1.4.5 Spatial configuration and DAE size of the PCMs

The spatial discretization for the PP- and EM-PCM is denoted as $n_{neg}n_{sep}n_{pos}$, which corresponds to the respective number of nodes in the anode, separator and cathode domain. The total number of DAEs calculates as

$$n_{DAEs} = (n_{neg} + n_{pos} + 2) \cdot 6 + (n_{sep} - 1) \cdot 3$$

referring to the boundary interfaces ('+2') and electrode domains ('$n_{neg} + n_{pos}$') with six ($c_i$, $c_s$, $i_p$, $j_m$, $\Phi_c$, $\Phi_s$) and three DAEs ($c_p$, $i_p$, $\Phi_i$) at the internal nodes of the separator ('$n_{sep} - 1$'), respectively. In terms of the OC-PCM, the spatially discretized particle domain ('$n_p$') must be included as well as the calculation of the temperature ('+1'):

$$n_{DAEs} = (n_{neg} + n_{sep} + n_{pos} - 2) \cdot 2 + (n_p + 3) \cdot (n_{neg} + n_{pos}) + 1$$

Again, the term '-2' is referring to the boundary interfaces of the electrodes and the separator, which are implemented in a common node for the definition of $c_i$ and $i_p$. The term '$n_p + 3'$ refers to the solid-concentration $c_s$ and the molar flux $j_m$/ionic current density $i_i$ as well as the solid-potential $\Phi_s$. Figure 5 summarizes the spatial discretization's used in this work for the p2D-PCMs with the corresponding number of DAEs.

<table>
<thead>
<tr>
<th>Model</th>
<th>Indices</th>
<th>Number of spatial nodes</th>
<th>Particle</th>
<th>Number of DAEs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Anode</td>
<td>Separator</td>
<td>Cathode</td>
</tr>
<tr>
<td>PP-PCM</td>
<td>1-1-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2-1-2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5-3-5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>10-5-10</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>EM-PCM</td>
<td>1-1-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2-1-2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5-3-5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>10-5-10</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>OC-PCM</td>
<td>5-3-5-2</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5-3-5-3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5-3-5-5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20-10-20-25</td>
<td>20</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>COMSOL-PCM</td>
<td>53-8-40-20</td>
<td>53</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>


Figure 5 Spatial discretization of the PCMs

1.5 Microcontroller implementation

Primarily the small-sized RAM of microcontrollers and low processor frequencies imply challenges for solving the p2D-PCM, which poses no challenge for a standard desktop computer equipped exemplarily with 16 GB RAM at 3.2 GHz as used in this work for the MATLAB®-code PCMs. 192 kB of RAM and a maximum of 168 MHz are offered by the chosen microcontroller to solve the C-code p2D-PCMs in this work. Beside working without an operating system and with hardware modules like universal asynchronous receiver transmitter (UART) for data transmission, the transfer from the scripting language MATLAB® to the programming language C is a significant step as some framework related options such as matrix inversion ($A/b$ [61]), linear algebra operations (sparse [68]) or solvers (ode15s...
[54,64,65] are not available and must be transferred without overloading the memory. Note, that most of these specific functions cannot be exported via the MATLAB® to C export option [71] and even if, they would not be necessarily runnable on a microcontroller. Basically, the hardware abstraction layer (HAL) library (Cortex Microcontroller Software Interface Standard (CMSIS), ARM [72]) and the STM32 CubeMX software [73] were used to configure the system clock, peripherals and an initial code structure. The flow chart of both stand-alone C-codes (PP-/EM-PCM and OC-PCM) are shown in Figure 6.

![Flow chart of the stand-alone C-codes of the PP-/EM-PCM (left) and the OC-PCM (right) implemented in the microcontroller (STM32F407VGT6, STMicroelectronics [9]). The PP-/EM-PCM and OC-PCM routines refer to the p2D-PCM models shown in Figure 3 and 4. The generated simulation results are sent per converged time step via an UART-to-USB converter to a desktop computer and evaluated in terms of computation performance and simulation accuracy.](image)

Via running the C-code PCMs on the STM32 and sending the simulation results (UART-to-USB converter) to a desktop computer, computation efficiency and simulation accuracy can be analysed and compared to the corresponding MATLAB®-code PCMs. The analysis using the MATLAB®-code PCMs on a desktop computer is not useful to evaluate the performance in low-hardware/software environment, as multi-threading calculation, oversized memory capacities, the comprehensive operating system and the framework MATLAB® itself would distort the results.

1.5.1 PP- and EM-PCM stand-alone C-code

After power up (see Figure 6, left), global variables with fixed memory allocation are set, which are accessible in any case whilst the remaining variables are allocated and freed with every function call. Next, the processor calls the main function to configure the system clock, initialize the peripherals, load the parameterization including analytical functions, look-up tables and single parameters, set the stop conditions and call the initialization. Via UART the messages to be transmitted are initialized and the initial states are sent to the desktop computer using the UART-to-USB converter. The main loop is entered next and the time simulation is started. It ends as soon as a stop condition is met (see Figure 3). Note, that only the current state $k$ and the previous state $k - 1$ are stored on the STM32 -
otherwise the microcontroller’s memory would be exceeded after a short time period. The main loop runs the PCM as depicted in Figure 3 and transmits the current states to the desktop computer at every converged time step.

In detail, the interpolation of the look-up tables $E_{eq}$ and $\frac{\partial E_{eq}}{\partial t}$ are defined on the STM32 via spline interpolation at the junction nodes, which offers differentiability compared to piecewise linear approximation. Based on a MATLAB® structure (spline [74]), the implementation includes the coefficients $c$, number of pieces $s$, order of polynomials $l$, range of the measured data and a pointer to the array of single intervals to define a spline as [75, 76]:

$$f(x) = c_{0,s}(x - x_s)^{l-1} + c_{1,s}(x - x_s)^{l-2} + \ldots + c_{l-1,s}(x - x_s) + c_{l,s}$$

Where the coefficients differ in each knot interval of $x \in [x_s, x_{s+1}]$

Not only the value of two adjacent intervals are matched but also their derivatives, which is a crucial point when calculating the jacobian. To reduce memory allocation, the coefficients are stored in an array and evaluated via pointer function. Using double precision [77], a total of 751 and 41 knots are used for $E_{eq}$ and $\frac{\partial E_{eq}}{\partial t}$ allocating 6008 and 328 Byte, respectively. The matrix inversion to calculate the inverse of the jacobian $J^{-1}$ uses a MATLAB® function (A/b [61]) in the MATLAB®-code PCMs. As standard C-algorithms (e.g. CMSIS) failed, the Gauss-Jordan (GJ) algorithm [78] was implemented, which is not a matrix-type specified algorithm like existing, tridiagonal-block-type algorithms (e.g. BAND(j), [79]). As floating-point numbers are used in the processor, the highest accuracy is gained at low absolute values. In order to minimize the error caused by performing floating point operations during the matrix inversion, the concept of pivoting [80] is applied, which leads to lower absolute values and thus higher accuracy. In addition, the STM32 provides a floating point unit (FPU) of single precision [77]. Before the inversion starts, the matrix-entries are converted into single precision, next the inversion takes place on the FPU and the results are converted into double precision in the end. The related loss of accuracy and computation speed up is discussed in this work. The 'Transmit states' action (see Figure 6) uses a virtual COM port between the STM32 and the desktop computer and a UART-to-USB converter (115200 bit s⁻¹), where the COM port is evaluated via a MATLAB® script. In terms of RAM, the size of $n_s$ (=6) state variables $x$, both the jacobian $J$ and its inverse $J^{-1}$ as well as the calculation of the solid-diffusion approximation $\chi$, determine the total size in Byte:

$$\text{size}(J, J^{-1}) = 2 \cdot (n_j \cdot n_s)^2 \cdot 8 \text{ Byte}$$

$$\text{size}(x) = 2 \cdot (n_j \cdot n_s) \cdot 8 \text{ Byte}$$

with a total number of nodes:

$$n_j = n_{neg} + n_{sep} + n_{pos} + 1$$

The multiplication with ‘2’ for $x$ and $x_s$ is necessary for the current and previous state. The additional states $n_s$ are two [11] for the PP- and six [25] for the EM-PCM. Figure 7 shows exemplarily the possible discretization’s and RAM/flash memory usage in the STM32 for the PP- and the EM-PCM.

<table>
<thead>
<tr>
<th>Model</th>
<th>PP-PCM</th>
<th>EM-PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretization</td>
<td>1-1-1</td>
<td>1-1-1</td>
</tr>
<tr>
<td>size($J$, $J^{-1}$) in Byte</td>
<td>9216</td>
<td>9216</td>
</tr>
<tr>
<td>size($x$) in Byte</td>
<td>384</td>
<td>384</td>
</tr>
<tr>
<td>size($x_s$) in Byte</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total size in Byte</td>
<td>9728</td>
<td>9728</td>
</tr>
<tr>
<td>Memory allocation in %</td>
<td>5.1</td>
<td>5.1</td>
</tr>
</tbody>
</table>

**Figure 7 Usage of memory regarding the stand-alone C-code PP- and EM-PCM**

1 Referring to the STM32 with 192 kB of RAM  
2 Referring to the STM32 with 1024 kB flash memory
The maximum runnable spatial configuration included 14 nodes in total as enough memory space for the variables and the solving process must be reserved. The minimum converging setup was found to be 2-1-2. The increase in memory by using the EM-approximation is negligible regarding the jacobian size increases as well and the overall RAM usage increases quadratically. In sum, the PP-/EM-PCM on the microcontroller need at least \( \approx 10 \text{ kB} \) up to a maximum of 115 kB during calculation and the maximum flash memory allocation consumed around 174 /178 kB of the maximum 1024 kB flash memory (\( \approx 17 /17.4 \% \)).

### 1.5.2 OC-PCM stand-alone C-Code

The structure of parameters, interpolation schemes and communication to the desktop computer of the C-code OC-PCM is similar to the C-code PP-/EM-PCM. A specified version of the ode15s solver [64–66] is developed in C offering main functionalities as:

- Calculation of initial jacobian (daeic12 [66])
- Initial step estimation \( k = 0 \)
- Calculation of iteration matrix \( M_i \)
- Iteration with simplified Newton method using GJ-inversion for \( M_i \)
- Calculating new jacobian \( J \)
- Adjusting step-size \( \Delta t \)

For calculating an initial jacobian, daeic12 [66] was extracted from the ode15s solver and transferred into C right after setting the initial values (see Figure 6). The sparse function [68] was adopted to gain a sparse jacobian via neglect of any zeros and the non-zero entries are stored in an array \( \frac{df}{dx} \) together with the respective index coordinate pair \( \frac{df}{dx} \text{index} \) to save memory. The estimation of the initial step-size is performed according to Curtis et al. [81]. The iteration matrix \( M_i \) is obtained at every iteration and uses the previously calculated jacobian (see Figure 4). The main loop integrates from the previous state \( k – 1 \) to the current state \( k \) and uses simplified Newton method [65,82] incorporating the GJ-inversion [78] for inverting \( M_i \) to generate the state update \( dx \). Step-size reduction and new jacobian calculation are implemented as described in section 1.4.3. Regarding the memory allocation, the iteration matrix \( M_i \) contributes as

\[
\text{size}(M_i) = 1 \cdot \left((n_{neg} + n_{pos})(m + 5) + 2n_{sep} - 3\right)^2 \cdot 8 \text{ Byte}
\]

Similar to the jacobian of the PP-/EM-PCM, the memory usage of the \( M_i \) increases quadratically and the spatial discretization in the particle domain \( m \) is here the main driver. The analytical calculation of the jacobian \( \frac{df}{dx} \) is stored as array of the non-zero entries as

\[
\text{size} \left( \frac{df}{dx} \right) = (n_{neg}(5n_{neg} + 3m + 6) + n_{sep}(3n_{sep} - 2) + n_{pos}(5n_{pos} + 3m + 6) - 5) \cdot 8 \text{ Byte}
\]

Therefore, the indices \( \text{index} \frac{df}{dx} \) of the position in the jacian (i.e. row and column) are stored as integers with 4 Byte each

\[
\text{size} \left( \frac{df}{dx} \text{index} \right) = 2 \cdot (n_{neg}(5n_{neg} + 3m + 6) + n_{sep}(3n_{sep} - 2) + n_{pos}(5n_{pos} + 3m + 6) - 5) \cdot 4 \text{ Byte}
\]

Improvement using unsigned 16 bit integer is optional but not considered in this work. The mass matrix \( M \) is stored as integer to

\[
\text{size}(M) = 1 \cdot \left((n_{neg} + n_{pos})(m + 5) + 2n_{sep} - 3\right) \cdot 4 \text{ Byte}
\]

and the backward differences for the NDFs [64,67] are crucial to find a consistent solution and their memory usage amounts to

\[
\text{size}(\nabla \text{m}) = 7 \cdot \left((n_{neg} + n_{pos})(m + 5) + 2n_{sep} - 3\right) \cdot 8 \text{ Byte}
\]

The factor '7' is required by the integration order and two additional states are saved for following iterations [64,67]. The variable \( x \) stores the states

\[
\text{size}(x) = 2 \cdot \left((n_{neg} + n_{pos})(m + 5) + 2n_{sep} - 3\right) \cdot 8 \text{ Byte}
\]

and the factor '2' accounts for the previous stored iteration. The maximum runnable spatial configuration 5-3-5-5 is in the same memory range as for the PP-/EM-PCM, whereas the coarsest configuration 5-3-5-2 offering sufficient accuracy needs at least five resp.
three nodes in the electrode resp. separator domain for the NMC-811/SiC parameterization. Note, that the minimum converging configuration appears to 3-2-3-2, but was neglected due to insufficient simulation accuracy. Figure 8 shows the range of runnable C-code OC-PCMs on the STM32.

In sum, the OC-PCM on the microcontroller needs at least 54 kB RAM during calculation and the maximum allocates around 100 kB. The overall model size uses around 212 kB of the maximum 1024 kB flash memory (≈ 20.7 %).

<table>
<thead>
<tr>
<th>Discretization</th>
<th>OC-PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5-3-5-2</td>
</tr>
<tr>
<td>size($M_i$) in Byte</td>
<td>42632</td>
</tr>
<tr>
<td>size($\frac{dF}{dx}$) in Byte</td>
<td>3088</td>
</tr>
<tr>
<td>size($\frac{dF}{dx}$)_{index} in Byte</td>
<td>3088</td>
</tr>
<tr>
<td>size($M$) in Byte</td>
<td>292</td>
</tr>
<tr>
<td>size($\nabla^w$) in Byte</td>
<td>4088</td>
</tr>
<tr>
<td>size($\mathbf{x}$) in Byte</td>
<td>1168</td>
</tr>
</tbody>
</table>

**RAM**

| Total size in Byte | 54356 | 68076 | 100316 |
| Memory allocation in % | 28.3 | 35.5 | 52.2 |

**Flash memory**

| Total size in kB | 212.08 | 212.27 | 212.95 |
| Memory allocation in % | ≈ 20.7 % |}

---

*Figure 8 Usage of memory regarding the stand-alone C-code OC-PCM*

1 Referring to the STM32 with 192 kB RAM  
2 Referring to the STM32 with 1024 kB flash memory
1.6 RESULTS AND DISCUSSION

1.6.1 VALIDATION AND COMPUTATIONAL EFFICIENCY UNDER CONSTANT LOAD SCENARIOS

The experimental validation of the three PCMs and the benchmark models is shown via thermographic measurements of the INR18650-MJ1 LIB at different CC charge (0.2/0.5/1C) and discharge (0.2/0.5/1.5/2C) rates at 25°C ambient temperature under convective cooling as presented in previous work [1]. A desktop computer equipped with an Intel(R) Core(TM) i5-6500 CPU at 3.20 GHz processor and 16 GB of RAM was used for calculation. The mean cell voltage error (a, b), temperature error (c, d), overall simulation time (e, f) and averaged calculation time per step (g, h) are shown in Figure 9. Regarding CC charging in Figure 9 a and c, the mean cell voltage error ranges from 10 mV for the ECM up to 20 mV for the OC-PCM until 0.5C and increases for the ECM at 1C up to 29 mV while a decreasing trend can be seen for all PCMs. The mean temperature error appears below 0.6 K for all models. Regarding discharge, the ECM matches quite well the measurements with errors on average below 21 mV and 0.1 K until 2C while all PCMs show increasing errors for higher C-rates (e.g. max. 68 mV for the PP-/EM-PCM and max. 1.5 K for the OC-PCM at 2C CC). The ECM shows increased modelling errors at 1C charging, which may be linked to limitations in the pore of the porous electrodes [1] and improvements can be achieved via using physically more meaningful, distributed-parameter ECMS [83] or adaptive, online-parameter estimation [84]. In general, all PCMs show increasing errors with higher applied C-rates (> 1C) as inhomogeneities along the electrodes (61.5 x 5.8 cm, width x height) increase as shown in our previous work [1], which cannot be modelled with a single-PCM as the electrode utilization is assumed to be homogeneous. Nevertheless, certain differences appear between the MATLAB®-code PCMs compared to the COMSOL-PCM. The temperature calculation is included in the jacobian of the OC-PCM, which is not implemented for the remaining PCMs and may have a significant influence on the cell voltage and temperature calculation beside the different spatial discretization and approximation schemes.
Figure 9 Experimental validation with the INR18650-MJ1 NMC-811/SiC LIB of the simulation results including the ECM, the MATLAB®-code PCMs ('PP-PCM 10-5-10', 'OC-PCM 20-10-20-25' and 'EM-PCM 10-5-10') and the rigorous benchmark PCM ('COMSOL-PCM 53-8-40-20'). The mean cell voltage error (a, b) and temperature error (c, d) are shown together with the overall computation time (e, f) and averaged calculation time per converged time step (g, h) for the different CC charge (a, c, e and g) and discharge (b, d, f and h) tests at 25°C ambient temperature and convective cooling conditions [1].

The reformulation and solving of the solid-diffusion PDE in the OC-PCM instead of using approximations contributes to the different cell voltages compared to the PP- and EM-PCM. The error for the OC-PCM remains at ≈ 22 mV for all scenarios, whereas the used approximations (PP [11] and EM [25]) lead to increased errors for 1.5 and 2C discharge (≈ 68 mV at 2C). As the liion-module [69] for the
COMSOL-PCM offers only the particle concentration at the centre, the surface and on average, an approximation may be used here as well, which may explain the appearing deviations.

In sum, all MATLAB®-code PCMs reveal sufficient accurate simulation of the electrochemical-thermal behaviour throughout the thickness of the NMC-811/SiC electrode stack under CC charge and discharge scenarios compared to the measured electrical-thermal behaviour of the INR18650-M1j LIB. Increasing errors appear, when inhomogeneities of the current density [1] along the electrodes are expected to increase for high applied C-rates (> 1C). The computation times in Figure 9 e and f of the COMSOL-PCM appear to be the slowest and the fastest appear for the ECM, as expected.

The same trend can be seen for the averaged calculation time per step (see Figure 9 g and h). Approximately 6.3 ms per step are needed at 1C CC discharge for the OC-PCM, while the ECM needs only 0.2 ms. This results in an approximately 32, 470, 865 and 1085 times faster computation for the ECM compared to the OC-, EM-, PP- and COMSOL-PCM, respectively. The benefit of using explicit functions of state and input variables to solve the ECM [83] instead of solving the DAE system of the PCM is not questioned here. Even if large frameworks are used, lean computational costs confirm that implementation in the microcontroller would lead to similar results. Therefore, the ECM is used as benchmark only and not implemented in the microcontroller. Around 16, 67 and 26 % of the desktop computer’s CPU and approximately 660, 670 and 793 MB RAM for the PP-/EM-, OC- and COMSOL-PCM are used at a full 1C CC discharge.

Considering also the computation time, the OC-PCM may be most suitable for fast computation but preferring low memory usage and CPU load, the PP- or the even faster calculating EM-PCM seem more suitable. The ECM allocates ≈ 1400 MB at ≈ 10 % CPU, which is mainly caused by using the MATLAB/Simulink framework. The simulation results are run on a desktop computer offering sufficient computation resources of 3.20 GHz and 16 GB RAM.

The STM32 offers maximum 168 MHz and 192 kB RAM and evaluating the most suitable PCM for embedded systems must be based on adequate conditions as proposed in control devices for a BMS, which are similar to the microcontroller of this work and offer similarly no commercial framework tool or a sophisticated operating system.

### 1.6.2 Reducing memory allocation via coarser spatial discretizations

For implementation in the microcontroller, the number of spatial elements in the PP-/EM-PCM and the OC-PCM is gradually reduced to save RAM and decrease computation time. The related increase of modelling error for the cell voltage $V_{cell}$, temperature $T_{cell}$, surface concentration $c_{s,s}$ and the potential drop $\Phi_s-\Phi_l$ both at the anode-separator interface is analysed in reference to the validated MATLAB®-code PCM configurations shown in Figure 9. Figure 10 shows the increase of error exemplarily for the 1C CC charge and discharge scenario.
Figure 10 The relative (%) increase of modelling error for the cell voltage (a, b), the cell temperature (c, d), the surface concentration $c_{s,s}$ (e, f) and the potential drop $\Phi_s-\Phi_l$ (g and h) both located at the anode-separator interface $x=L_{neg}$ is shown for gradually reducing the total number of spatial nodes in the MATLAB®-code PCMs (PP-, EM- and OC-PCM) in reference to the experimentally validated PCMs shown in Figure 9 ('PP-PCM 10-5-10', 'EM-PCM 10-5-10' and 'OC-PCM 20-10-20-25').

Regarding the global cell variables $V_{cell}$ and $T_{cell}$, similar error increase appears for charge and discharge for all PCMs. The PP- and EM-PCM show mean cell voltage error increase of $\approx 0.6\%$ for the minimum configuration (1-1-1), whereas the OC-PCM shows errors below 0.1 % in all cases. The deviance for the cell temperature is around 1.2 % for the PP- and EM-PCM, whereas nearly no deviance
could be seen for the reduced configurations of the OC-PCM. Note, a minimum of 5 and 3 nodes for the electrodes and separator domain in the OC-PCM appeared for guaranteeing convergence. Regarding the internal variables $c_{s, s}$ and $\Phi_s-\Phi_l$, higher deviations for the charge than for the discharge scenario appear with the chosen parameterization, which differs between charge and discharge to account for hysteresis effects of the open-circuit potentials [1]. Regarding the most coarse discretization, errors up to 7.3 /3 % compared to 2.7 /1.2 % for charge/discharge of the PP-/EM-PCM and the OC-PCM respectively appeared for $c_{s, s}$. In terms of $\Phi_s-\Phi_l$, the errors increase up to 16 % for charging regarding PP-/EM-PCM, whereas lower errors appear for the OC-PCM (≈ 1.4 %). The results indicate no distinct difference between the PP- and EM-PCM, when lean spatial discretization is chosen even at the lowest configuration of 1-1-1. The OC-PCM shows less error with decreasing number of nodes when mainly the spatial discretization in the particle is reduced - which reduces the overall size of the DAE enormously.

In general, the error on global variables such as cell voltage and temperature seems acceptable but when internal variables are used such as the potential drop at the anode-separator interface to indicate the onset of lithium plating~\cite{Luders.2019,Chu.2017}, distinct errors due to a lean spatial discretization must be considered for interpreting the results correctly. In terms of the microcontroller, spatial configurations of 2-1-2 and 5-3-5 are used for the PP- and EM-PCM as similar mean errors regarding 2-1-2 of 4 mV (0.12 %), 7E-2 K (0.25 %), 84 mol m$^{-3}$ (0.94 %) and 0.8 mV (0.3 %) are expected due to modelling error at 1C CC charge and discharge, which offer still sufficient accuracy to describe accurately the NMC-811/SiC INR18650-MJ1 LIB. In terms of the OC-PCM, the 5-3-5-3 and 5-3-5-5 are used which show maximum averaged errors regarding 5-3-5-3 of 0.25 mV (5E-3 %), 2E-3 K (2E-3 %), 6.7 mol m$^{-3}$ (8E-2 %) and 6E-2 mV (2E-2 %).

### 1.6.3 Stand-alone C-code models on the microcontroller under constant load scenario

The CC charge and discharge loads as shown in section 1.6.1 are simulated on the STM32 using the stand-alone C-codes of the PP-, EM- and OC-PCM. The simulations incorporate the coarse 2-1-2, 2-1-2 and 5-3-5-3 and the maximum 5-3-5, 5-3-5 and 5-3-5-5 spatial discretizations, respectively. Figure 12 shows the mean cell voltage and temperature error in reference to the corresponding MATLAB®-code. The computational performance is analysed via the mean iteration time and total number of iterations per 1 s time step on the STM32 processor at 168 MHz Figure 11 summarizes the analysis at 1C CC charge and discharge. The lowest mean cell voltage error for charge and discharge appears for the PP-PCM (< 0.4 mV) and the highest appears up to 3.8 mV for the EM-PCM. Reducing the number of spatial nodes (see 'Coarse configurations' in Figure 12), leads to higher deviations regarding the PP-PCM whereas the EM- and OC-PCM reveal less deviations.
Figure 11 Performance of C-code PCMs on the microcontroller at 1C-CC

In terms of the cell temperature for charge and discharge, similar trends can be seen except for the lowest temperature error, which is seen for the OC-PCM.

In sum, all C-code PCMs on the STM32 show sufficient accuracy below 4 mV of cell voltage and 0.4 K of cell temperature error, which is mainly caused by approximations and rounding errors. The analysis of the internal states such as concentrations and potentials are neglected here as similar low error ranges appear.

<table>
<thead>
<tr>
<th>Model</th>
<th>PP-PCM</th>
<th>EM-PCM</th>
<th>OC-PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretization</td>
<td>2-1-2</td>
<td>5-3-5</td>
<td>2-1-2</td>
</tr>
<tr>
<td></td>
<td>5-3-5</td>
<td></td>
<td>5-3-5-3</td>
</tr>
<tr>
<td>1C CC charging</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean $\Delta V_{cell}$ / mV $^1$</td>
<td>0.255</td>
<td>0.019</td>
<td>0.3956</td>
</tr>
<tr>
<td>Mean $\Delta T_{cell}$ / K $^1$</td>
<td>0.103</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>Mean number of iterations per step / -</td>
<td>4.02</td>
<td>4.31</td>
<td>2.17</td>
</tr>
<tr>
<td>Mean iteration time per step / ms</td>
<td>219</td>
<td>1052</td>
<td>124</td>
</tr>
<tr>
<td>1C CC discharging</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean $\Delta V_{cell}$ / mV $^1$</td>
<td>0.388</td>
<td>0.069</td>
<td>2.126</td>
</tr>
<tr>
<td>Mean $\Delta T_{cell}$ / K $^1$</td>
<td>0.08</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td>Mean number of iterations per step / -</td>
<td>3.72</td>
<td>4.17</td>
<td>1.56</td>
</tr>
<tr>
<td>Mean iteration time per step / ms</td>
<td>205</td>
<td>1021</td>
<td>90</td>
</tr>
</tbody>
</table>

$^1$ in reference to the corresponding MATLAB®-code PCM with the same discretization.
Figure 12: Evaluation of the simulation accuracy (a, b, c and d) and computational efficiency (e, f, g, and h) of the stand-alone C-code PP-, EM- and OC-PCM under constant current charge (0.2, 0.5, and 1C) and discharge (0.2, 0.5, 1, 1.5 and 2C) scenarios. The mean cell voltage (a, b) and temperature error (c, d) is shown in reference to the corresponding MATLAB®-code PCM for the coarse (2-1-2, 2-1-2 and 5-3-5-3 for PP-, EM- and OC-PCM, 'Coarse configurations') and maximum spatial configuration. The computational efficiency is shown similarly in form of the iteration time (e, f) and the mean number of iteration per 1 s step (g, h).

Regarding the computational performance in Figure 12 e and f, the fastest computation appears for the EM-PCM (max. 560 ms per step with min. 2 iterations) whereas the slowest is seen for the PP-PCM (max. 1052 ms per step with min. 4 iterations). When the coarse discretization is used, the OC-PCM reveals the slowest calculation (max. 563 ms per 1 s step). For the OC-PCM, the 1 s time step is
set as a maximum as the solver routine is implemented with the option, to reduce the step-size if no convergence appears (see Figure 4) and the OC-PCM is thus more susceptible to prolong the overall computation time. Thus, the minimal number of iterations appears for the OC-PCM and the PP-PCM needs the most iterations. The spatial discretization has less significant influence and only slightly reduces the number of iterations.

In sum, the EM-PCM is the fastest calculating model with 90 respectively 540 ms per 1 s time step for an entire 1C CC discharge using coarse respectively maximum spatial discretization in the STM32. The OC-PCM offers still calculation times below the real-time threshold even if the solver routine reduces the step-size during runtime. The PP-PCM is most likely to require the longest calculation time and number of iterations. Under CC load scenarios, the most appropriate choice for simulating the INR18650-MJ1 LIB in real-time on the microcontroller would be the EM-PCM.

### 1.6.3.1 Influence of the processor frequency

In application, lower processor frequency results in lower energy consumption and reducing the computation power can thus pose a challenge for the C-code p2D-PCMs to hold real-time computability. The 32-bit ARM Cortex M4 processor [9] of the STM32 offers a frequency range up to 168 MHz and three different configurations at 50, 109 and 168 MHz are used to simulate 1C CC charge and discharge with all C-code PCMs to evaluate the influence on the computation speed. The UART transfer time reveals negligible influence herein. Figure 13 a and b show the total computation time versus the simulated time and for all frequencies, the EM-PCM shows the fastest computation, whereas the PP-PCM the slowest. A more detailed analysis of the computation performance is shown in Figure 14. As the simulation accuracy and iteration number (see Figure 14) are not influenced by the processor frequency, only the mean iteration and total calculation time versus the simulated time (see 'Time reduction') are shown in Figure 14. Similarly to Figure 13 a and b, Figure 13 c and d illustrate the average iteration time per 1 s step. The blue horizontals in Figure 13 mark the real-time suitability, when the calculation time equals the simulated time (see 'Time factor' in Figure 14). At 168 MHz, the PP-PCM is slightly atop the threshold with +8 % /+6 %, whereas the OC- and the fastest calculating EM-PCM show time reductions of 6 % /10 % and 41 % /43 % during 1C CC charge /discharge. Only the EM-PCM reveals sufficient computation speed (max. 872 ms for 1C CC charge) at 109 MHz, whereas at 50 MHz none of the PCMs can simulate in real-time. A minimum increase of 89 % (1890 ms for 1 s at 1C CC discharge) is seen for the EM-PCM and over 3 times longer computation times than the actual simulated time appear for the PP- and OC-PCM.
Figure 13 Computational performance of the C-code PP-, EM- and OC-PCM for simulating 1C CC charge (a and c) and discharge (b and d) on the STM32 at 50, 109 and 168 MHz. The horizontal line depicted in ‘blue’ marks the real-time threshold when the simulated time equals the total computation time (see ‘Time factor = 1’ in a and b) and the average iteration time per converged step equals the simulation step-size of 1 s (see ‘1000 ms’ in c and d).
Figure 14 Computational performance of C-code PCMs at different processor frequencies

In sum, real-time suitability poses a challenge for the C-code PCMs on the microcontroller and under CC loads, the necessary speed-up can be achieved via using coarser spatial discretizations (e.g. 2-1-2 resp. 5-3-5-3 for the PP-/EM-PCM and OC-PCM). As seen in Figure 14, reducing the spatial configuration reveals the EM-PCM as fastest-calculating C-code PCM. As the accuracy is not distorted by the frequency, the most appropriate choice at low frequencies is the EM-PCM.

1.6.3.2 Influence of the microcontroller's accuracy

To reduce computation time, the jacobian/iteration matrix inversion is calculated using the FPU of single precision. The transfer needs two extra arrays of single precision for a current copy of the matrix and for storing the inverse. The coarse configurations 2-1-2 and 5-3-5-3 for the PP-/EM- and OC-PCM are used to avoid RAM overloads. 1C CC charge and discharge at 168 MHz are simulated and the results are shown in Figure 15. The maximum mean cell voltage error accounts to 1.862 mV for 1C CC discharge using the EM-PCM in reference to the corresponding MATLAB®-code PCM. The lowest error is seen for the OC-PCM below 0.12 mV. Negligible errors for the cell temperature (see Figure 15) appear. Comparing the results between single- (see Figure 15) and double-precision (see Figure 11), a trend of decreasing iterations appears (≈ 19 % less) for the EM-PCM at 1C CC charge.

Figure 15 Computational performance of C-Code PCMs using FPU for 1C CC charge and discharge

The benefit can be seen in the average iteration times per step in Figure 15. The EM- and OC-PCM show calculation times around 80 ms and the PP-PCM up to 156 ms. For constant loads, minimum reduction of 29 %, 23 % and 86 % for the PP-, EM- and OC-PCM appears when the FPU is used. The mean error of the PCMs may slightly increase while using FPU, but still sufficient accuracy is offered. Regarding the maximum calculation times, single time steps are simulated in approximately 600 ms.
for a 1 s time step at a total iteration number up to 9 for the OC-PCM. This may be critical, when real-time computation must be guaranteed. The PP- and EM-PCM show lower maximum iteration numbers and computation times, where the PP-PCM appears as the overall slowest converging model. In sum, the most benefit in using the FPU is gained with the OC-PCM but certain overshoots in calculation time and iteration number may be a problem for real-time suitability. The EM- and PP-PCM show a more stable calculation with slightly increased errors for the EM-PCM and slower computation speed for the PP-PCM. Comparing the total calculation time to the simulated times for 1C charge and discharge, a minimum reduction of 84 %, 92 % and 92 % appears at constant load simulations for the PP-, EM- and OC-PCM, respectively.

### 1.6.4 Validation and Computational Efficiency Under Driving Cycle Scenario

Referring to application of LIBs in EVs, the ARTEMIS [85] drive cycle was adapted to the INR18650-MJ1 LIB current range as seen in Figure 16 a. For the experiments, a cycler (CTS, BaSyTec) paired with a climate chamber (VT 4021, Vötsch Industrietechnik GmbH) at 25°C was used at initial 100 % SoC of the LIB. Figure 16 b illustrates the measured cell voltage together with a magnified region (Figure 16 c) showing the simulation results. The temperature profile (see Figure 16 f) and both voltage and temperature error (Figure 16 d, e and g) of the MATLAB®-code PP-, EM- and OC-PCM are shown. Figure 17 summarizes the related simulation accuracy for the MATLAB®-code PCMs.
Figure 16 Measurement results of the INR18650-MJ1 LIB ('Measurement MJ1') for the applied current (a), cell voltage (b) and temperature (f) under the adapted ARTEMIS [85] driving cycle. The subplot (c) shows a magnified part from 1200 to 1260 s for the measured and simulated (MATLAB®-code 'PP-PCM 5-3-5', 'EM-PCM 5-3-5' and 'OC-PCM 5-3-5-5') cell voltages in (b). The resulting error profiles for the PCMs are shown for the cell voltage (d) with a magnified part similar to (c). Subplot (g) shows the temperature error.

In this case, averaged open-circuit potentials \(E_{eq}\), see supplementary part) of lithiation and delithiation [1] were used to ease the effect of measurement-related polarization and improve the simulation of dynamic loads. The mean cell voltage error for the PP- and EM-PCM appear to be around 18.7 mV on average with a root mean squared error (RMSE) of 23.1 mV. Reducing the spatial nodes,
the error slightly increases but remain below 20 mV. The OC-PCM shows no distinct difference between the coarse and the maximum configuration and reveals the most accurate simulation with a mean error of 12.4 mV at a RMSE of 16.6 mV. The temperature error for all MATLAB®-code PCMs are below 0.15 K (RMSE < 0.2), which is in the range of the measurement accuracy (i.e. Pt100 sensor with ±0.15 K at 0°C, DIN/IEC Class A). For comparison, the ECM simulation results calculated on the same desktop computer revealed a mean cell voltage error of 13.7 mV at a RMSE of 17.5 mV and a mean temperature error of 0.22 K. The PCMs are in the same range of accuracy as the ECM and choosing the appropriate discretization or reformulation to gain real-time operability, a competitive alternative appears for model-based monitoring of LIBs.

![Figure 17 Validation of the MATLAB®-code PCMs under the adapted ARTEMIS driving cycle.](image)

The benefit of using a PCM lies in the simulated local concentrations and potentials, which can be used to develop sophisticated control algorithms such as avoiding lithium plating during fast charging [86]. Finally, the performance of simulating the ARTEMIS profile with the C-code PCMs on the STM32 is evaluated. Figure 18 summarizes the simulation accuracy in reference to the MATLAB®-code PCM (see Figure 17) together with the computation speed.

![Figure 18 Computational performance of C-code PCMs simulating adapted ARTEMIS driving cycle.](image)

The mean cell voltage deviation is below 0.4 mV for all PCMs, where the OC-PCM shows the most accurate implementation and the EM-PCM the maximum deviation to the corresponding MATLAB®-code PCM.

Again, the temperature deviations are in the range of measurement accuracy. Regarding the computational performance, the OC-PCM is not able to simulate in real-time due to the step-size reduction option in the solver. The EM-PCM shows the fastest calculation and for a step-size of 250 ms, only the coarse configuration is able to simulate with a time reduction of 37 % under real-time requirements. The PP-PCM required at least 190 ms on average, which corresponds to a time reduction of 24 %.

To conclude, the EM-PCM shows the most promising results for simulating an application-near scenario with fulfilling real-time requirements.
1.7 CONCLUSIONS FOR MODEL ORDER REDUCTION AND IMPLEMENTATION INTO MICROCONTROLLER

Trending towards high-energy LIBs, physicochemical model-based monitoring can help to account for inhomogeneities on local scales and improve the state-estimation process. Efficiently reduced p2D-PCMs are evaluated on a microcontroller using either FDM together with solid-diffusion approximations or Chebyshev orthogonal collocation to reformulate particle and electrolyte domain. Experimental validation with CC charge and discharge, ARTEMIS [85] driving cycle and benchmarking to ECM and rigorous COMSOL p2D-PCM showed accurate simulation for a NMC-811/SiC LIB.

In sum, the average cell voltage error of the p2D-PCMs can be summarized as modelling and parameter uncertainties, errors from spatial reduction and errors from implementation in the STM32 as shown in Figure 19.

<table>
<thead>
<tr>
<th>Model</th>
<th>PP-PCM</th>
<th>EM-PCM</th>
<th>OC-PCM</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error from modeling and parameters / mV</td>
<td>3.35 Ah NMC-811/SiC INR18650-MJ1 LIB</td>
<td>Compared to experimental data from</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1C CC CH/DCH</td>
<td>&lt;1/9.7</td>
<td>&lt;1/9.3</td>
<td>17/&lt;1</td>
<td>MATLAB</td>
</tr>
<tr>
<td>Driving cycle</td>
<td>18.6</td>
<td>18.6</td>
<td>12.4</td>
<td></td>
</tr>
<tr>
<td>Error from spatial reduction / mV</td>
<td>4.1/4</td>
<td>4.1/4</td>
<td>&lt;1/&lt;1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>STM32 with max. 168 MHz 032-bit ARM® Cortex®-M4</th>
<th>max. 1024 kB flash memory and max. 192 kB RAM [9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C CC CH/DCH</td>
<td>&lt;0.5/&lt;0.5</td>
<td>&lt;0.5/2.1</td>
</tr>
<tr>
<td>Driving cycle</td>
<td>&lt;0.5</td>
<td></td>
</tr>
</tbody>
</table>

Figure 19 Summary of average cell voltage error for MATLAB and C-code PCMs.

At low processor frequencies down to 50 MHz, crucial limitations appear for the p2D-PCMs to calculate in real-time. Using hardware acceleration such as the FPU, computation acceleration up to 86 % appeared, which can be recommended at low processor frequencies to gain real-time computability again.

Computation analysis under CC and driving cycle loads revealed the EM-PCM as best choice for simulating a single LIB at least 37 % faster than real-time, which consumes 21.9 kB RAM for solving and 175 kB flash memory for storing the model on the microcontroller.

Future work can investigate more robust solver in terms of the OC-PCM to improve the performance on a microcontroller and test the C-code EM-PCM model-based monitoring for estimating a LIB online and develop local-anode potential based fast charging profiles to avoid lithium plating.
1.8 References


1.9 Appendix

<table>
<thead>
<tr>
<th>Electrolyte</th>
<th>1 M LiPF₆ in PC/EC/DMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt diffusivity ( D_{11} ) / m² s⁻¹</td>
<td>( 10^-10^{-4} \times \left( \frac{15.4}{c_1} - 0.22c_1 \right) )</td>
</tr>
<tr>
<td>Ionic conductivity ( \sigma_1 ) / S m⁻¹</td>
<td>( 0.1c_1 \left( -10.5 + 0.068c_1 + 0.494c_1^2 + 0.674T - 0.0178c_1T \right) )</td>
</tr>
<tr>
<td>Activity ( \frac{d\ln f_{Li}}{d\ln c_1} ) / -</td>
<td>(-8.8610^{-4}c_1^2T - 6.9610^{-5}c_1^2T^2 + 2.510^{-5}c_1T^2 )</td>
</tr>
<tr>
<td>Transference ( \epsilon_2 ) / -</td>
<td>( 0.001 - 0.24c_1 + 0.988\left( 1 - 0.0052(T - 294) \right)c_1^2 )</td>
</tr>
<tr>
<td>Ref. concentration ( c_{ref} ) / mol m⁻³</td>
<td>0.38</td>
</tr>
</tbody>
</table>

\( ^1 \) Ref. [87]

**Figure 20 Physical properties of the electrolyte [87].**
**D1.2 – Report on model order reduction**

Author: Johannes Sturm (TUM) – August 2019

EVERLASTING - Grant Agreement 71377 (Call: H2020-GV8-2015)
Electric Vehicle Enhanced Range, Lifetime And Safety Through INGenious battery management

---

**Thermodynamics**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stoichiometry</td>
<td>0.852</td>
<td>0.222</td>
</tr>
<tr>
<td>Max. theoretical loading $b_p$</td>
<td>415 mAh g$^{-1}$</td>
<td>275.5 mAh g$^{-1}$</td>
</tr>
<tr>
<td>Density $\rho$</td>
<td>2.42 g cm$^{-3}$</td>
<td>4.87 g cm$^{-3}$</td>
</tr>
<tr>
<td>Concentration $c_{s,max}$</td>
<td>34684 mol m$^{-3}$</td>
<td>50060 mol m$^{-3}$</td>
</tr>
</tbody>
</table>

**Transport**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid diffusivity $D_s$</td>
<td>$5 \times 10^{-14}$ m$^2$ s$^{-1}$</td>
<td>$5 \times 10^{-14}$ m$^2$ s$^{-1}$</td>
</tr>
<tr>
<td>Specific activation $E_a$</td>
<td>1200 K</td>
<td>1200 K</td>
</tr>
<tr>
<td>Solid conductivity $\sigma_s$</td>
<td>100 S m$^{-1}$</td>
<td>0.17 S m$^{-1}$</td>
</tr>
</tbody>
</table>

**Kinetics**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reaction rate constant $k$</td>
<td>$3 \times 10^{-11}$ m$^{-1}$</td>
<td>$1 \times 10^{-11}$ m$^{-1}$</td>
</tr>
<tr>
<td>Specific activation $E_a$</td>
<td>3600 K</td>
<td>3600 K</td>
</tr>
<tr>
<td>Transfer coefficient $\alpha_{\text{tr}}$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

---

**Figure 21** Parameterization of p2D-PCM regarding the NMC-811/SiC LIB [88] (i.e. INR18650MJ1).

---

**Figure 22** Differential algebraic equation system for the p2D-PCM.
### Greek symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>Transfer coefficient</td>
</tr>
<tr>
<td>$\alpha_\infty$</td>
<td>Ambient heat transfer coefficient</td>
</tr>
<tr>
<td>β</td>
<td>Bruggeman coefficient</td>
</tr>
<tr>
<td>ε</td>
<td>Volume fraction</td>
</tr>
<tr>
<td>ε</td>
<td>Numerical tolerance</td>
</tr>
<tr>
<td>η</td>
<td>Overpotential</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Ionic conductivity</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Mass density</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Electrical conductivity</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>Electrical potential</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Variable</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Clenshaw-Curtis weights</td>
</tr>
</tbody>
</table>

### Indices

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Anodic reaction (oxidation)</td>
</tr>
<tr>
<td>act</td>
<td>Active area</td>
</tr>
<tr>
<td>app</td>
<td>Applied (<em>i.e.</em> current density)</td>
</tr>
<tr>
<td>c</td>
<td>Cathodic reaction (reduction)</td>
</tr>
<tr>
<td>eff</td>
<td>Transport corrected (Bruggeman correlation [95])</td>
</tr>
<tr>
<td>ext</td>
<td>External heat (<em>i.e.</em> from grid resistance)</td>
</tr>
<tr>
<td>l</td>
<td>Liquid phase (<em>i.e.</em> electrolyte)</td>
</tr>
<tr>
<td>max</td>
<td>Maximum</td>
</tr>
<tr>
<td>neg</td>
<td>Negative electrode (<em>i.e.</em> SiC)</td>
</tr>
<tr>
<td>pos</td>
<td>Positive electrode (<em>i.e.</em> NMC-811)</td>
</tr>
<tr>
<td>s</td>
<td>Solid phase (<em>i.e.</em> active particle)</td>
</tr>
<tr>
<td>sep</td>
<td>Separator</td>
</tr>
<tr>
<td>s,s</td>
<td>Solid phase (<em>i.e.</em> active particle surface)</td>
</tr>
<tr>
<td>surf</td>
<td>Surface</td>
</tr>
</tbody>
</table>

**Figure 23 Nomenclature I.**
Latin symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>m$^{-1}$</td>
<td>Specific surface</td>
</tr>
<tr>
<td>$b_g$</td>
<td>mAh g$^{-1}$</td>
<td>Maximum theoretical loading</td>
</tr>
<tr>
<td>$c$</td>
<td>mol m$^{-3}$</td>
<td>Concentration of lithium cations (Li$^+$)</td>
</tr>
<tr>
<td>$c_{s,max}$</td>
<td>mol m$^{-3}$</td>
<td>Maximum theoretical concentration of Li$^+$</td>
</tr>
<tr>
<td>$c_p$</td>
<td>J kg$^{-1}$ K$^{-1}$</td>
<td>Heat capacity</td>
</tr>
<tr>
<td>$D$</td>
<td>m$^2$ s$^{-1}$</td>
<td>Diffusion coefficient</td>
</tr>
<tr>
<td>$D$</td>
<td></td>
<td>Differentiation matrix</td>
</tr>
<tr>
<td>$E_{eq}$</td>
<td>V</td>
<td>Equilibrium potential vs. Li/Li$^+$</td>
</tr>
<tr>
<td>$\frac{\partial E_{eq}}{\partial T}$</td>
<td>V/K</td>
<td>Entropic coefficient</td>
</tr>
<tr>
<td>$f_{w}$</td>
<td></td>
<td>Mean molar activity coefficient of electrolyte</td>
</tr>
<tr>
<td>$F$</td>
<td>96 485 A s mol$^{-1}$</td>
<td>Faraday’s constant</td>
</tr>
<tr>
<td>$g$</td>
<td></td>
<td>Non-linear equations of p2D-PCM</td>
</tr>
<tr>
<td>$i$</td>
<td>A m$^{-2}$</td>
<td>Current density</td>
</tr>
<tr>
<td>$i_{app}$</td>
<td>A m$^{-2}$</td>
<td>Applied current density</td>
</tr>
<tr>
<td>$i_n$</td>
<td>A m$^{-2}$</td>
<td>Current density perpendicular to particle surface</td>
</tr>
<tr>
<td>$i_0$</td>
<td>A m$^{-2}$</td>
<td>Exchange current density</td>
</tr>
<tr>
<td>$j_n$</td>
<td>mol m$^{-2}$ s$^{-1}$</td>
<td>Pure-wall flux</td>
</tr>
<tr>
<td>$J$</td>
<td>Jacobian matrix</td>
<td></td>
</tr>
<tr>
<td>$k$</td>
<td>m s$^{-1}$</td>
<td>Reaction rate constant</td>
</tr>
<tr>
<td>$L$</td>
<td>m</td>
<td>Thickness</td>
</tr>
<tr>
<td>$m$</td>
<td>kg</td>
<td>Mass of cell</td>
</tr>
<tr>
<td>$M$</td>
<td>Mass matrix</td>
<td></td>
</tr>
<tr>
<td>$M_i$</td>
<td>Iteration matrix</td>
<td></td>
</tr>
<tr>
<td>$r$</td>
<td>m</td>
<td>r-coordinate particle domain of p2D-PCM</td>
</tr>
<tr>
<td>$R$</td>
<td>8.314 J mol$^{-1}$ K$^{-1}$</td>
<td>Gas constant</td>
</tr>
<tr>
<td>$R_{ext}$</td>
<td>$\Omega$ m$^2$</td>
<td>Grid resistance</td>
</tr>
<tr>
<td>$R_p$</td>
<td>m</td>
<td>Particle radius</td>
</tr>
<tr>
<td>$q$</td>
<td>W m$^{-2}$</td>
<td>Heat generation rate per area</td>
</tr>
<tr>
<td>$t$</td>
<td>s</td>
<td>Time</td>
</tr>
<tr>
<td>$T$</td>
<td>K</td>
<td>Temperature</td>
</tr>
<tr>
<td>$t_\text{Li}^+$</td>
<td>Transport number of Li$^+$</td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>m</td>
<td>x-coordinate in electrolyte domain of p2D-PCM</td>
</tr>
</tbody>
</table>

Figure 24 Nomenclature II.
2 STATE ESTIMATION OF LITHIUM-ION CELLS USING A PHYSICOCHEMICAL MODEL BASED EXTENDED KALMAN FILTER

The basic methodology and the results dealing with the state-estimation of LIBs using a physicochemical model together with a non-linear Kalman filter was published in the Journal Applied Energy (Volume 223, 1 August 2018, Pages 103-123) and the subsequent short summary is based on this publication referring to: https://doi.org/10.1016/j.apenergy.2018.04.011 (Gold Open Access)

The accompanied repository of the published data is accessible as well under: https://doi.org/10.4121/uuid:a588c059-0d82-42fb-931d-8e805da79388 (Open Access)

2.1 ABSTRACT OF ACTIVITIES RELATED TO THE STATE ESTIMATION OF LITHIUM-ION BATTERIES USING PHYSICOCHEMICAL MODELS

Two time-varying linear state-space representations of the generally accepted physicochemical model (PCM) of a lithium-ion cell are used to estimate local and global states during different charging scenarios. In terms of computational speed and suitability towards recursive state observer models, the solid-phase diffusion in the PCM of an exemplary MCMB/LiCoO$_2$ lithium-ion cell is derived with the aid of two different numerical reduction methods in the form of a polynomial profile (PP) and an eigenfunction method (EM).

As a benchmark, the PCM using the original duhamel superposition integral (DSI) approximation serves for the comparison of accuracy and computational speed. Figure 20 depicts the three models together with their simplifications of the solid-phase diffusion.
Figure 25 Schematic representation of a MCMB/LiCoO$_2$ lithium-ion cell shown in the upper part with the subsequent pseudo two-dimensional reduction to the PCM [6]. The fundamental model equations are depicted for the solid (i.e. active material) and liquid (i.e. electrolyte) phase, including three different numerical approximation methods (i.e. DSI, PP and EM) for the solid-phase diffusion. The PP- and EM-approach are used for the development of the recursive state observer models using the EKF-algorithm.

A modified spatial discretization via the finite volume method improves handling of boundary conditions and guarantees accurate simulation results of the PCM even at a low level of spatial discretization. The polynomial profile allows for a significant speed-up in computational time whilst showing a poor prediction accuracy during dynamic load profiles. The eigenfunction method shows a comparable accuracy as the benchmark for all load profiles whilst resulting in an even higher computational effort. The two derived observer models incorporate the state-space representation of the reduced PCM applying both the polynomial and eigenfunction approach combined with an Extended Kalman Filter (EKF) algorithm based on a novel initialization algorithm and conservation of lithium mass. The state-estimation process is summarized in Figure 21.
The estimation results of both models show robust and quick reduction of the residual errors for both local and global states when considering the applied current and the resulting cell voltage of the benchmark model, as the underlying measurement signal. The carried-out state estimation for a 4C constant charge current showed a regression of the cell voltage error to 1 mV within 30 s with an initial SOC error of 42.4% under a standard deviation of 10 mV and including process noise.

### 2.2 Comparison of the State Estimation Performance for Different Simulated Charging Scenarios

For comparing the two state-estimation processes using either the PP- or the EM-based observer model, three different charging scenarios including a constant (CCCV), multi-stage (BCCV-high and low current period) and a pulsed current scenario (PCCV) were pre-simulated with the DSI-PCM and altered by gaussian-white noise. In general, the observer models (PP- and EM-EKF) gain comparable results in terms of accuracy and computational speed for all charging scenarios regarded in this work. The robustness against varying boundary conditions is shown and the quick regression of the residual error is seen in all charging scenarios. The regression speed is summarized for the three charging scenarios referring to the initial SOC error, the remaining cell voltage error of 1 mV and the related estimation time in Table 1.
Table 1 Error regression of cell voltage for the observer models

<table>
<thead>
<tr>
<th>Regression Times</th>
<th>CCCV</th>
<th>BCCV-high</th>
<th>BCCV-low</th>
<th>PCCV</th>
<th>Observer model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial SoC Error / %</td>
<td>36.1</td>
<td>42.4</td>
<td>n.a.</td>
<td>35.2</td>
<td></td>
</tr>
<tr>
<td>Current rate / -</td>
<td>2.5 C</td>
<td>4C</td>
<td>0.1C</td>
<td>10C-pulse</td>
<td></td>
</tr>
<tr>
<td>Error of $V_{cell} = 1 mV \text{ or } 4 mV (I)$</td>
<td>650 s</td>
<td>67 s</td>
<td>210 s</td>
<td>130 s</td>
<td>PP-EKF</td>
</tr>
<tr>
<td></td>
<td>630 s</td>
<td>30 s</td>
<td>200 s</td>
<td>90 s</td>
<td>EM-EKF</td>
</tr>
</tbody>
</table>

Regarding only the cell voltage, the EM-EKF shows slightly faster regression compared to the PP-EKF. Particular interest is laid on the potential drop at the anode-separator interface and its spatial distribution within both electrodes. The observer models proofed the accurate recovery of this local indicator state for lithium plating from the measured current and cell voltage. As the observer models gain an accurate and robust estimation for local states, these could be used for control strategies in fast charging algorithms for controlling the applied current to the cell based on this indicator. In Figure 22, the regression of the SoC error of the limiting cathode based on the average lithium concentration is shown for all charging scenarios.

Figure 27 The regression of the SoC error based on the average lithium concentration within the cathode domain is shown for the CCCV (a), BCCV (b) and the PCCV (c) scenario for both observer models.

Marginal differences are seen for the CCCV and BCCV scenario, whereas the pulsed current charging revealed a slightly quicker regression for the EM-EKF than for the PP-EKF, which is well in line with
the suitability for dynamic and constant discharge load scenarios shown in the simulation results. Looking at the overall charging times in Figure 22, a reduction of 24.3 %, 13.5 % and 42.3 % could be reached for the CCCV, BCCV and PCCV scenario, compared to a constant charge process comprising a 1C charging rate whilst avoiding possible lithium plating, which was monitored via the indicator state (i.e. $\Phi_s - \Phi_c$ at $x = l_{neg}$).

### 2.3 CONCLUSION FROM STATE ESTIMATION PROCESS

The generally accepted pseudo two-dimensional physicochemical model (PCM) for lithium-ion batteries is used for the simulation of constant and dynamic load scenarios. The non-linear differential algebraic equations were discretized in time via Crank-Nicolson formulation and the finite difference method with finite volume formulation was used for the fully-spatially-resolved PCM. Particular interest lied on the handling of boundary conditions with a low number of spatial discretization elements and the validity of the model was checked via comparison to reference models implemented in FORTRAN and COMSOL Multiphysics®. Further reducing the computational effort and enabling for a recursive formulation, the solid-phase diffusion equation was numerically approximated via a Polynomial Profile and an eigenfunction method. The simulation results confirmed the computational efficiency of the Polynomial Profile under constant current load and the eigenfunction method under dynamic load scenarios. The two computationally efficient PCMs were further used for implementation of two different linear state-space representations of the PCM using an Extended Kalman Filter algorithm which conserves lithium mass. To the authors’ best knowledge, this is the first approach using the PCM solved by finite volume method together with a non-linear Kalman Filter which accounts for lithium mass conservation and incorporates a robust Newton-Raphson initialization routine to ease the initial value problem. The state-estimation results showed a quick recovery of the cell’s state for the measured cell voltage and the applied current together with robustness against changing boundary conditions and pulsed current signals. The models were able to reduce the residual cell voltage error to 2 mV with an initial SoC error of 35.2 % in less than 90 s for a 10C pulsed charging scenario. Current and future work of the authors deals with the implementation of these two models on microcontrollers, as first implementations proved the suitability of the models presented in this work in low computational hardware environment. With the aid of this hardware-based implementation and a further development of the presented observer models towards constraints on the local battery states, novel fast charging strategies will be investigated to reduce charging time whilst avoiding lithium plating. The state estimation process will also be extended in terms of integrating temperature effects.

Further development of the presented observer models can focus on the implementation of side-reactions such as lithium-plating and solid-electrolyte interphase to account for aging phenomena and also extend the approach to estimate the state-of-health (SoH).
3 Off-Line Simulation Models to Gain Minimum Calculation Time Considering Physicochemical Model Approaches

The basic methodology and results for section 3.1 “A Computationally Efficient Implementation of a Full and Reduced-Order Electrochemistry-Based Model for Li-Ion Batteries” refer to the related publication in the Journal Applied Energy (Volume 208, 1 December 2017, Pages 1285-1296), available under: https://doi.org/10.1016/j.apenergy.2017.09.025 (Gold Open Access)

The basic methodology and results for section 3.2 “On Trade-Offs Between Computational Complexity and Accuracy of Electrochemistry-Based Battery Models” is currently under publication and will be presented at the IEEE Conference on Decision and Control (Nice, France) in December 2019.

3.1 A Computationally Efficient Implementation of a Full and Reduced-Order Electrochemistry-Based Model for Li-Ion Batteries

Lithium-ion batteries are commonly employed in various applications owing to high energy density and long service life. Lithium-ion battery models are used for analysing batteries and enabling power control in applications. The Doyle-Fuller-Newman (DFN) model is a popular electrochemistry-based lithium-ion battery model which represents solid-state and electrolyte diffusion dynamics and accurately predicts the current/voltage response using a set of nonlinear partial differential equations. However, implementation of the full DFN model requires significant computation time. This chapter proposes a computationally efficient implementation of the full DFN battery model, which is convenient for real-time applications. The proposed implementation is based on applying model order reduction to a spatial and temporal discretisation of the governing model equations. For model order reduction, we apply proper orthogonal decomposition and discrete empirical interpolation method, which leads to a set of reduced order nonlinear algebraic equations. These equations are solved using a particular numerical scheme, based on a damped Newton's method. In a simulation study, the computational efficiency of the proposed implementation is shown and the resulting accuracy is presented.

3.1.1 Introduction

Featured by high energy density and long service life, lithium-ion (Li-ion) batteries are used in various applications, such as consumer electronics, portable devices and (hybrid) electric vehicles. In order to properly analyse, design and control the batteries in the aforementioned applications, models of Li-ion batteries are needed. Among available battery models, the Doyle-Fuller-Newman (DFN) model is a popular one, see, e.g., [1,2], which describes solid-state and electrolyte diffusion dynamics and electric potentials (over space and time) and is governed by a set of nonlinear partial differential equations (PDEs). Solving the complete DFN model requires significant computation time, which precludes the DFN model to be used in real-time control and monitoring applications. These real-time applications include state-of-charge or state-of-energy estimation [3-7], which requires estimates of the Li-ion concentration in solid and electrolyte phases, as well as optimal and fast charging of the
Several techniques have been proposed in the literature to address the highly complex DFN model. For instance, a single particle model has been employed to model the internal states of the Li-ion battery \cite{11} and a simplified multi-particle model via linearization has been proposed in \cite{12}. These papers simplify the battery model by focussing on the solid phase diffusion, which is an important phenomenon in battery model, see, e.g., \cite{13,14}. These solid-state diffusion dynamics can be simplified analytically, as was done in, e.g., \cite{15,16}, which considerably reduces the computational complexity.

Besides the simplifications, model order reduction techniques have been proposed for simplifying battery models. However, most of the existing papers in the literature use model order reduction techniques for (infinite-dimensional) linear systems to obtain reduced order battery models, such as Padé approximations \cite{15,17}, balanced truncations \cite{18}, or residue grouping \cite{16}, see \cite{19} for an overview. Others use a Galerkin projection based on several Legendre polynomials \cite{20}, or a proper orthogonal decomposition (POD) \cite{21}. Still, these latter two methods have only been applied to a linearized DFN model in \cite{20,21}. Even though linearized models are valid for low (dis)charge currents, they become less accurate at higher currents. To summarize, the aforecited papers demonstrate model order reduction techniques for batteries on a priori simplified models (i.e., ones that only consider part of the dynamic behaviour, as discussed above) or even start from equivalent circuit models, see, e.g., \cite{22}.

In addition to the complexity of the modelling equations, the simulation time and required memory are affected by the particular numerical method for solving the DFN model equations \cite{27}. Numerical methods have been studied in \cite{16,24}, while the computational performance of the suggested models has not been discussed. Furthermore, a computationally efficient implementation of the full DFN model has been proposed in \cite{25}. However, \cite{25} uses a simplification that was proposed in \cite{16}, which makes the model less accurate in comparison to the full DFN model. Moreover, \cite{26} presents another computationally efficient implementation of the nonlinear Li-ion battery model, however it uses a simplification on the solid phase diffusion equations.

In this chapter, we propose a computationally efficient implementation of the full DFN model. The proposed implementation is based on applying model order reduction to a discretisation of the infinite dimensional DFN model. The discretisation follows our previous work \cite{27}, where we have applied a spatial discretisation to convert the PDEs into a set of differential-algebraic equations (DAEs) and a temporal discretisation, based on a backward Euler method \cite{28}, to convert the entire model into a set of coupled nonlinear algebraic equations. To solve the discretised modelling equations, a computationally efficient numerical method is presented, based on a damped Newton's method, see, e.g., \cite{29}. While it has been shown in \cite{27} that the discretised DFN model already allows simulating the DFN model faster than real-time, an accurate model implementation requires small step sizes in the discretisation, which still yields a very high-order model.

To reduce the order of the discretised DFN model, this chapter proposes to apply model order reduction, which was not done in our previous work \cite{27}. In particular, we apply a combination of POD \cite{30} and discrete empirical interpolation method (DEIM) \cite{31} to the full nonlinear DFN model, instead of applying it to a linearized or a priori simplified battery model, as was done in many of the aforecited papers. We employ the efficient implementation of \cite{27} in the model order reduction and extend the simulation results of \cite{27} to develop a suitable model, used for model order reduction. An extensive simulation study demonstrates that applying model order reduction leads to a lower-dimensional set of modelling equations, which requires less memory on an embedded real-time platform, have lower computational complexity and only have a minor loss of accuracy, when compared to the original DFN model implementation. Since the model order reduction has been applied to an electrochemistry-based model, which properly models the physical behaviour and internal states of the battery, these models are potentially interesting for real-time battery monitoring and control applications.

The rest of the chapter is organized as follows. Section 3.1.2 outlines the DFN model. Then, a computational procedure is presented in Section 3.1.3 that implements the full DFN model. Section 3.1.4 introduces a model order reduction technique that simplifies the model and reduces the computation time and required memory. The simulation results of the full-order model implementation...
are given in Section 3.1.5, where computational performance analysis and model validation are discussed. Section 3.1.6 presents the simulation results of the reduced-order model implementation based on POD and DEIM that includes the snapshot and computational efficiency analysis. Finally, conclusions are drawn in Section 3.1.7.

### 3.1.2 Model of lithium-ion batteries

The DFN model [1,2], considered in this chapter, is a one-dimensional physics-based electrochemical model of a Li-ion battery. Figure 28 shows the three different regions between the current collectors that determine the battery behaviour: the negative electrode, the separator and the positive electrode. During discharging, Li-ion de-intercalate from a solid-phase in the (porous) negative electrode, represented by spherical particles, and diffuse in a dissolved electrolyte phase through the separator towards the (porous) positive electrode, again represented by spherical particles, where they intercalate in the electrode. The reverse process occurs during charging.

![Figure 28 Schematic of the Li-ion battery model during discharging.](image)

The DFN model is governed by a set of nonlinear PDEs that describe conservation of mass and charge in both the solid and electrolyte phases across the battery. A brief summary of the model is given below and the interested reader is referred to [1,2], or to [16] for a more control-oriented exposition of the model.

- The concentration of Li-ion in the solid-phases \( c_s \) in both electrodes (i.e., \( x \in [0, \delta_\text{neg}] \) for the negative electrode and \( x \in [L - \delta_\text{pos}, L] \) for the positive electrode, where \( \delta_\text{neg} \) and \( \delta_\text{pos} \) are the thicknesses of negative electrode and positive electrode, respectively), derived from Fick’s law of diffusion as a function of the radial coordinate inside the spherical particle \( r \in [0, R_s] \) (where \( R_s \) is the radius of the spherical particles) and time \( t \in \mathbb{R}^+ \), is given by
  \[
  \frac{\partial c_s}{\partial t} = \frac{D_s}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial c_s}{\partial r} \right),
  \tag{1a}
  \]
  with boundary conditions
  \[
  \frac{\partial c_s}{\partial r} \bigg|_{r=0} = 0, \quad -D_s \frac{\partial c_s}{\partial r} \bigg|_{r=R_s} = \frac{j_{\text{Li}}}{\alpha_s F}.
  \tag{1b}
  \]
  In these expressions, \( D_s \) is the diffusion coefficient in the solid-phase, \( j_{\text{Li}} \) is the volume-specific rate of the electrochemical reaction, which will be defined in the following, \( \alpha_s = 3 \varepsilon_s / R_s \) is the specific interfacial surface area, where \( \varepsilon_s \) denotes the active material volume fraction, and \( F \) is Faraday’s constant.

- The potential of the solid-phase \( \phi_s \) is described by Ohm’s law as
  \[
  \frac{\partial}{\partial x} \left( \sigma_{\text{eff}} \frac{\partial \phi_s}{\partial x} \right) = j_{\text{Li}}
  \tag{2a}
  \]
  with boundary conditions
  \[
  \sigma_{\text{eff}} \frac{\partial \phi_s}{\partial x} \bigg|_{x=0} = \frac{i_{\text{app}}}{\Lambda_{\text{surf}}}, \quad \sigma_{\text{eff}} \frac{\partial \phi_s}{\partial x} \bigg|_{x=\delta_\text{neg}} = 0,
  \tag{2b}
  \]
  for the negative electrode and
The concentration of Li-ion in the electrolyte phase $c_e$ as a function of $x \in [0, L]$ and $t \in R^+$ is given by
\[
\varepsilon_e \frac{\partial c_e}{\partial t} = \frac{\partial}{\partial x} \left( D_e^\text{eff} \frac{\partial c_e}{\partial x} \right) + \frac{1 - t^*_e}{F} j_e, \tag{3a}
\]
with boundary conditions
\[
D_e^\text{eff} \frac{\partial c_e}{\partial x} \bigg|_{x=0} = D_e^\text{eff} \frac{\partial c_e}{\partial x} \bigg|_{x=L} = 0, \tag{3b}
\]
where $\varepsilon_e$ denotes the porosity (i.e., the electrolyte phase volume fraction), $D_e^\text{eff}$ denotes the effective diffusion coefficient of Li-ion in the electrolyte phase, such that $D_e^\text{eff} = D_e \varepsilon_e^p$, where $D_e$ is the Li-ion diffusion coefficient in the electrolyte phase and $p$ is the Bruggeman porosity exponent, and $t^*_e$ is the transference number of Li-ion.

The potential of the electrolyte phase $\phi_e$ is given by
\[
\frac{\partial}{\partial x} \left( \kappa^\text{eff} \frac{\partial \phi_e}{\partial x} + \kappa_B^\text{eff} \frac{\partial \ln c_e}{\partial x} \right) = -j_e, \tag{4a}
\]
with boundary conditions
\[
\kappa^\text{eff} \frac{\partial \phi_e}{\partial x} \bigg|_{x=0} = \kappa^\text{eff} \frac{\partial \phi_e}{\partial x} \bigg|_{x=L} = 0 \tag{4b}
\]
where $\kappa^\text{eff} = \kappa c_e^p$ denotes the effective ionic conductivity and $\kappa$ is given by $\kappa = 15.8 c_e \exp(0.85(1000 c_e)^{1.4})$. Moreover, the electrolyte ionic diffusional conductivity $\kappa_B^\text{eff}$ is described as
\[
\kappa_B^\text{eff} = \frac{2RT\kappa^\text{eff}}{F} \left( t^*_e - 1 \right) \left( 1 + \frac{\partial \ln f_e}{\partial \ln c_e} \right), \tag{5}
\]
where $R$ is the universal gas constant, $T$ is the absolute temperature, and $f_e$ denotes the activity coefficient which is assumed to be constant in this chapter.

The PDEs (1)-(4) are coupled through the volume-specific rate of reaction occurring at the solid/electrolyte interface $j_{Li}$. This reaction rate satisfies a Butler-Volmer kinetic equation [16] described as
\[
j_{Li} = a_s i_0 \left( \exp \left( \frac{a_a F}{RT} \eta \right) - \exp \left( - \frac{a_c F}{RT} \eta \right) \right), \tag{6}
\]
where $i_0$ denotes the exchange current density given by
\[
i_0 = k_0 c_s (c_s^\text{max} - c_{e,s}) \frac{a_s c_e}{c_s^\text{max}}, \tag{7}
\]
in which $k_0$ is the kinetic rate constant, $a_a$ is the anodic transfer coefficient, $a_c$ is the cathodic transfer coefficient, $c_s^\text{max}$ the maximum concentration in the solid-phase, $c_{e,s}(x,t) = c_s(R_s, x, t)$ is the concentration at the surface of the spherical particle. Moreover, the variable $\eta$ in (6) denotes the overpotential and is defined as
\[
\eta = \phi_e - \phi_e - U, \tag{8}
\]
where $U$ denotes the open-circuit potential (equilibrium potential), which is a predefined nonlinear function of the Li-ion concentration at the surface of the spherical particle $c_{e,s}$.

Finally, the (measurable) terminal battery voltage is given by
\[
V(t) = \phi_e(L, t) - \phi_e(0, t) - \frac{R_s}{A_{surf}} i_{app}(t), \tag{9}
\]
where $R_s$ denotes the surface-specific resistance of the current collector. Since it is very challenging to solve this set of nonlinear PDEs, spatial and temporal discretisation methods are used to obtain (approximate) solutions to this set of PDEs.
3.1.3 Model Implementation

In this section, an efficient implementation of the full model (1)-(9) is proposed. The implementation involves two steps: the first step is a spatial and temporal discretisation that leads to a set of nonlinear algebraic equations; the second step is solving the nonlinear algebraic equations, for which a numerical method based on a damped Newton’s method is proposed [29]. While this chapter uses a similar discretisation scheme as in [25], the numerical method for solving the algebraic equations is different.

3.1.3.1 Spatial and Temporal Discretisation

First, a spatial discretisation is applied to convert the PDEs (1)-(4) into a set of DAEs. The governing equation of particle concentration (1) is discretised using a finite difference method (FDM), see, e.g., [32] and (2)-(4) are discretised applying a finite volume method (FVM), see, e.g., [33]. Both FDM and FVM lead to sparse linear systems of equations, which considerably simplify the computation. FDM has been used to discretise (1) because only the surface concentration is coupled with other PDEs. Moreover, using FDM avoids extrapolation of the volume averages towards the surface, as was done in [21]. FVM is chosen to discretise (2)-(4) due to the fact that FVM can be easily formulated with unstructured meshes. However, when the number of grid points is large enough, both FDM and FVM are equally good approximations and one can make a very accurate approximation of the governing PDEs using either FDM or FVM.

In the spatial discretisation, the radial coordinate of the spherical particles \( r \in [0, R]\) is discretised at radius \( r_i = (i - 1)\delta_r, i \in \{1, ..., n_r\}\), with interval size \( \delta_r = R/(n_r - 1) > 0\), where \( n_r\) denotes the number of grid points along the radial coordinate. The position along the cross-section of the battery \( x \in [0, L]\) is discretised at \( x^{(j)} = \sum_{k=1}^{j} \delta_x^{(k)}\), with interval size \( \delta_x^{(k)} > 0\) that can vary with \( x\), where \( j, k \in \{1, ..., n_x\}\) and \( n_x\) denotes the number of intervals along the cross-section coordinate. The spatial intervals are chosen such that \( \sum_{k=1}^{n_x} \delta_x^{(k)} = L\), \( \sum_{k=1}^{n_r} \delta_x^{(k)} = \delta_+\), \( \sum_{k=n_x+n_r+1}^{n} \delta_x^{(k)} = \delta_-\), where \( n_r\), \( n_x\), \( n_p\) denote the number of intervals (control volumes) along the \( x\)-dimension in the negative electrode, separator, and positive electrode, respectively and satisfy \( n_x = n_n + n_s + n_p\).

The spatial discretisation of (1) using FDM is given by

\[
\frac{d}{dt} c_s^{(i,j)} = \begin{cases} 
\frac{2D_s^{(j)} (c_s^{(2,j)} - c_s^{(1,j)})}{\delta_T^2} & \text{for } i = 1 \\
\frac{D_s^{(j)} (r_{i-1} c_s^{(i-1,j)} - 2r_i c_s^{(i,j)} + r_{i+1} c_s^{(i+1,j)})}{r_i \delta_T^2} & \text{for } i \in \{2, ..., n_r-1\} \\
\frac{2D_s^{(j)} (c_s^{(n_r-1,j)} - c_s^{(n_r,j)})}{\delta_T^2} - \frac{\delta_T + R_s z_u^{(1)}}{\delta_x R_s a_s^{(j,F)}} & \text{for } i = n_r
\end{cases}
\]

where the time variable \( t\) is omitted from this expression for compactness of the notation. Moreover, \( c_s^{(i,j)}(t) = c_s(r_i, x_j, t)\), with \( x_j \in \{\sum_{k=1}^{i-1} \delta_x^{(k)}, \sum_{k=1}^{i} \delta_x^{(k)}\}\).

To obtain a convenient discrete-space approximation of (2)-(4), it should be observed that all these expressions have the form

\[
a_0 \frac{\partial \omega}{\partial t} = \frac{\partial}{\partial x} \left( a_1 \frac{\partial \omega}{\partial x} + a_2 \frac{\partial \ln c_e}{\partial x} \right) + a_3 \frac{\partial \omega}{\partial t}
\]

with boundary conditions

\[
-a_1 \frac{\partial \omega}{\partial x} \bigg|_{x=\beta_1} = a_4, \quad a_1 \frac{\partial \omega}{\partial x} \bigg|_{x=\beta_2} = a_5.
\]

Namely, (2) is retrieved by choosing \( \omega = \phi_{ce}, a_0 = a_2 = 0, a_1 = a_{eff}\) and \( a_3 = -1\); (2b) by \( a_4 = i_{app}/A_{surf}\), \( a_5 = 0\), \( \beta_1 = 0\) and \( \beta_2 = \delta_-\); (2c) by \( a_4 = 0\), \( a_5 = i_{app}/A_{surf}\), \( \beta_1 = L - \delta_+\) and \( \beta_2 = L\). Similarly, (3) is formalised by choosing \( \omega = c_e, a_0 = \varepsilon_e, a_1 = \delta_{eff}, a_2 = a_3 = a_5 = 0, a_3 = (1 - t_0^2)/F\), \( \beta_1 = 0\) and \( \beta_2 = L\).

Finally, (4) corresponds to \( \omega = \phi_{er}, a_0 = a_4 = a_5 = 0, a_1 = \kappa_{eff}, a_2 = \kappa_{eff}, a_3 = 1, \beta_1 = 0\) and \( \beta_2 = L\).

The spatial discretisation using the FVM yields...
where \( J^0 = \{1\}, J^c = \{2, \ldots, n_n - 1\} \) and \( J^f = \{n_n\} \) for the combination of (2a) and (2b); \( J^0 = \{n_n + n_s + 1\}, J^c = \{n_n + n_s + 2, \ldots, n_x - 1\} \) and \( J^f = \{n_x\} \) for the combination of (2a) and (2C); \( J^0 = \{1\}, J^c = \{2, \ldots, n_x - 1\} \) and \( J^f = \{n_x\} \) for both (3) and (4). The discrete-space approximations (10) and (12), derived from (1)-(4), can be written in a compact form given by

\[
M \frac{d}{dt} x(t) = A(x(t)) x(t) + f(x(t)) + B I_{app}(t),
\]

where

\[
x = \begin{bmatrix} x_1^T & x_2^T & x_3^T & x_4^T \end{bmatrix}^T \in R^{(2n_x+n_s+n_n+n_p+n_s)},
\]

\[
x_1 = \begin{bmatrix} c_z^{(1,1)} & \cdots & c_z^{(n_z,1)} & \cdots & c_z^{(1,n_p)} & \cdots & c_z^{(n_z,n_p)} \end{bmatrix}^T \in R^{n_z},
\]

\[
x_2 = \begin{bmatrix} c_e^{(1)} & \cdots & c_e^{(n_e)} \end{bmatrix}^T \in R^{n_s},
\]

\[
x_3 = \begin{bmatrix} \phi_a^{(1)} & \cdots & \phi_a^{(n_a)} \end{bmatrix}^T \in R^{n_s},
\]

\[
x_4 = \begin{bmatrix} \phi_s^{(1)} & \cdots & \phi_s^{(n_s)} & \phi_s^{(n_s+n_s+1)} & \cdots & \phi_s^{(n_p)} \end{bmatrix}^T \in R^{n_p},
\]

and

\[
M = diag(I_{n_z,n_n}, I_{n_s}, 0_{n_s}, 0_{n_p}),
\]

\[
A(x_2(t)) = diag(A_1, A_2, A_3(x_2(t)), A_4),
\]

\[
f(x(t)) = \begin{bmatrix} f_1^T(x(t)) & f_2^T(x(t)) & f_3^T(x(t)) & f_4^T(x(t)) \end{bmatrix}^T,
\]

\[
B = \begin{bmatrix} 0_{1 \times (n_z+n_n)} & 0_{1 \times n_s} & 0_{1 \times n_p} & b_a \end{bmatrix}^T,
\]

where \( n_{np} = n_n + n_p \) and diag\((Q_1, \ldots, Q_N)\) denotes a block diagonal matrix with matrices \( Q_1, \ldots, Q_N \) on its diagonal and zero matrices on its off-diagonal blocks. The matrices \( A_i \) for \( i \in \{1, \ldots, 4\} \), obtained from (10) and (12), are sparse matrices, and \( f_i(x(t)) \) for \( i \in \{1, \ldots, 4\} \) capture the second and third terms of (10) and (12). The nonlinear terms of (13) is due to the governing equation of electrolyte-phase potential \( \phi_s \), given by (4), and the Butler-Volmer equation, given by (6). After spatial discretisation, a temporal discretisation using backward Euler method, see, e.g., [28], is performed to convert the resulting DAEs (13) into a set of coupled nonlinear algebraic equations. By choosing a constant step size \( \delta_t > 0 \), the temporal discretisation of (13) results in

\[
F(x(t_j)) = \begin{bmatrix} F_1(x(t_j)) \\ F_2(x(t_j)) \\ F_3(x(t_j)) \\ F_4(x(t_j)) \end{bmatrix} = 0_{N \times 1},
\]

\[
F_1(x(t_j)) = \begin{bmatrix} A_1 \delta_t - I_{n_z,n_n} & x_1(t_j) & f_1(x(t_j)) \end{bmatrix} \delta_t + x_1(t_{j-1}),
\]

\[
F_2(x(t_j)) = \begin{bmatrix} A_2 \delta_t - I_{n_s} & x_2(t_j) & f_2(x(t_j)) \end{bmatrix} \delta_t + x_2(t_{j-1}),
\]

\[
F_3(x(t_j)) = \begin{bmatrix} A_3 (x_2(t_j)) & x_3(t_j) & f_3(x(t_j)) \end{bmatrix},
\]

\[
F_4(x(t_j)) = A_4 x_4(t_j) + f_4(x(t_j)) + b_a I_{app}(t_j),
\]

with \( N = 2n_x + n_s + n_p + n_{np} \) at sample times \( t_j = j\delta_t, j \in N \).
3.1.3.2 Numerical Scheme

The strategy for solving the nonlinear algebraic equations (14) at sample times $t_j$ that we use in this chapter, is based on a Newton's method. The Newton's method originates from the Taylor's series approximation [28] and is a general method for finding the results of a nonlinear algebraic equation. The nonlinear algebraic equations (14) can be solved by solving

$$x^{(k+1)}(t_j) = x^{(k)}(t_j) - J(x^{(k)}(t_j))^{-1} F(x^{(k)}(t_j)),$$

(15)

where $k \in \{0, 1, ..., K\}$ represents the iterations, $x \in \mathbb{R}^N$, $F(x^{(k)}(t_j))$ is the vector function and $J(x^{(k)}(t_j))$ is the Jacobian matrix of (14a). To solve (15), the Jacobian $J(x^{(k)}(t_j))$ of $F(x^{(k)}(t_j))$ as in (14a) has to be computed at every iteration $k$, which is computationally demanding.

Instead of solving the Jacobian matrix $J(x^{(k)}(t_j))$ of the whole system $F(x^{(k)}(t_j))$, we solve (14) by sequentially applying Newton's method to the four subsystems (14b)-(14e), while following a Gauss-Seidel approach [28]. Because we propose to employ Newton's method in a sequential manner, we only need the Jacobian matrices of the subsystems (14b)-(14e), which are much smaller and are sparse. This makes them easier to compute and leads to a considerable reduction of computational complexity. Finally, a damping coefficient $\gamma$ has been applied to warrant convergence of the results. This leads to

$$x_i^{(k+1)}(t_j) = x_i^{(k)}(t_j) - \gamma J_i(x_i^{(k)}(t_j))^{-1} F_i(x_i^{(k)}(t_j)),$$

(16a)

for $i \in \{1, ..., 4\}$ and $\gamma \in (0, 1]$, where

$$\hat{x}^{(k)} = \begin{bmatrix} (x_1^{k+(l,1)})^T & (x_2^{k+(l,2)})^T & (x_3^{k+(l,3)})^T & (x_4^{k+(l,4)})^T \end{bmatrix}^T,$$

(16b)

and

$$J(i, j) = \begin{cases} 1 & \text{if } i > j \\ 0 & \text{elsewhere} \end{cases}$$

(16c)

The algorithm is considered converged at sample times $t_j$ with a given tolerance tol when

$$|x_i^{(k+1)}(t_j) - x_i^{(k)}(t_j)| < \text{tol} \quad \text{for } i \in \{1, ..., 4\}$$

and considered not converged if a certain maximum number of iterations $K$ is exceeded. Note that in the numerical scheme (16), the order of the subsystems $i \in \{1, ..., 4\}$ in which (26) is solved can be changed. This might influence the computational efficiency, which will be discussed in detail later.

3.1.4 Model Order Reduction Using POD and DEIM

In the previous section, a numerical scheme is given to simulate the DFN model with full spatial and temporal discretisation. Although the implementation is already faster than real-time, as will be shown in Section 3.1.5, a further decrease in computation time can be obtained by applying model order reduction (MOR) techniques. In addition to a reduction in the computation time, MOR results in smaller matrices that describe the model. This leads to lower memory requirements for the implementation of the model, which is valuable for real-time embedded platforms.

3.1.4.1 Proper Orthogonal Decomposition

Proper orthogonal decomposition [30] is a data-based MOR technique used to find a low-dimensional approximation of large-scale systems. The key to the POD method is to extract the basic functions that contain the most relevant features from the system of interest, in the sense that $x_i(t_j) \approx \hat{U}_i \hat{x}_i(t_j)$ for every $x_i(t_j), i \in \{1, ..., 4\}$, as in (13b), by a suitable choice of the matrix $\hat{U}_i \in \mathbb{R}^{n_i \times q_i}$, where $q_i \ll n_i$. The orthonormal matrix $\hat{U}_i$ is obtained by solving

$$\min_{\hat{U}_i \in \mathbb{R}^{n_i \times q_i}} \sum_{j=1}^{m} \|x_j(t_j) - \hat{U}_i \hat{x}_i(t_j)\|_2^2,$$

subject to $\hat{U}_i^T \hat{U}_i = I_{q_i}$, using a snapshot of state trajectories $x_i(t_j)$ at sample times $t_j = j \delta_t$, with $i \in \{1, ..., 4\}$ and $j \in \{1, ..., m\}$.
The POD basis \( \mathcal{U}_i \) in (18) can be constructed from the left singular vector of the snapshot matrix \( x_i = [x_i(t_1), ..., x_i(t_m)] \in \mathbb{R}^{n_i \times m} \) obtained by a singular value decomposition (SVD). Here, the SVD of \( x_i \) is given by

\[
x_i = U_i \Sigma_i V_i^T
\]

where \( U_i \in \mathbb{R}^{n_i \times n_i} \) and \( V_i \in \mathbb{R}^{m \times m} \) are the left and right singular vectors, respectively. Moreover

\[
\Sigma_i = \begin{bmatrix}
\Sigma_{i,1} & 0_{n_i \times (m-n_i)} \\
0_{(m-n_i) \times n_i} & 0_{(m-n_i) \times (m-n_i)}
\end{bmatrix} \in \mathbb{R}^{n_i \times m}
\]

where \( \Sigma_{i,1} = \text{diag}(\sigma_{i,1}^{(1)} \ldots \sigma_{i,1}^{(n_i)}) \in \mathbb{R}^{n_i \times n_i} \) is a diagonal matrix containing the singular values, where \( l_i = \text{rank}(x_i) \). The optimal solution of (18), denoted by \( \mathcal{U}_i \in \mathbb{R}^{n_i \times q_i} \), is given by the matrix formed by the first \( q_i \ll l_i \) columns of \( U_i \). The choice of the matrix \( \mathcal{U}_i \) provides a bound on the approximation error, because

\[
\sum_{j=1}^m ||x_i(t_j) - \mathcal{U}_i \mathcal{U}_i^T x_i(t_j)||_2^2 = \sum_{k=q_i+1}^{n_i} (\sigma_{i,k})^2.
\]

This means that the accuracy of the approximation is directly determined by the number of columns of \( U_i \in \mathbb{R}^{n_i \times n_i} \) used to construct \( \mathcal{U}_i \in \mathbb{R}^{n_i \times q_i} \). It should be noted that the choice of snapshot influences the accuracy of the reduced-order model (ROM), which will be discussed in Section 6.

The model described by (14) can now be reduced by applying the POD-Galerkin projection to every function \( F(x(t_j)) \), i.e., \( \mathcal{U}_i^T F(\mathcal{U}_i \mathcal{U}_i^T x_i(t_j)) = 0 \). For (14b), the ROM equation becomes

\[
\mathcal{U}_i^T \left( A_1 \delta_t - l_{n_y n_p} \right) \mathcal{U}_i \mathcal{U}_i^T x_i(t_j) + \mathcal{U}_i^T \mathcal{B}_i \left( \mathcal{U}_i \mathcal{U}_i^T x_i(t_j) \right) \delta_t + \mathcal{U}_i^T \mathcal{D}_i \mathcal{U}_i \mathcal{U}_i^T x_i(t_{j-1}) = 0_{q_i \times 1}
\]

The ROM equations for (14c)-(14e) are obtained similarly. The linear terms of (22) and the other reduced basis models of (14b)-(14e) have a computational complexity that depends on the number of columns \( q_i \) of \( U_i \) used to construct \( \mathcal{U}_i \) for each \( i \in \{1, \ldots, 4\} \) and allows for a considerable reduction of computation time and storage of the model equations. However, the nonlinear terms \( f_i(x(t_j)) \) in (14b)-(14e) still require evaluations on the original full-order model dimension. As a result, solving this set of equations is still as costly as or even more costly than solving the original model. A solution to this problem is presented in the following.

### 3.1.4.2 Discrete empirical interpolation

The discrete empirical interpolation method is proposed by [31] as an effective way to approximate the nonlinear parts of ROM by projecting the nonlinearities onto a subspace spanned by a basis determined by some selected interpolation indices.

In particular, the nonlinear function \( f_i(t) \) extracted from (14b)-(14e) is also projected onto a lower dimensional subspace given by

\[
f_i(t) \approx \bar{W}_i c_i(t),
\]

for some matrix \( \bar{W}_i \in \mathbb{R}^{n_i \times k_i} \) and coefficient vector \( c_i(t) \). The matrix \( \bar{W}_i \in \mathbb{R}^{n_i \times k_i} \) is obtained by applying an SVD on a snapshots matrix of the nonlinear functions \( \{f_i(x(t_1)) \ldots f_i(x(t_m))\} \), where \( \bar{W}_i \) is constructed from the first \( k_i \) columns of the left singular vector \( W_i = [w_i^1 \ldots w_i^{n_i}] \in \mathbb{R}^{n_i \times n_i} \) of the snapshot matrix. The coefficient vector \( c_i(t) \) can be calculated by some selected interpolation indices, given by

\[
P_i^T f_i(t) \approx P_i^T \bar{W}_i c_i(t),
\]

where \( P_i = \left[ e_i^{(p_1)} \ldots e_i^{(p_{k_i})} \right] \in \mathbb{R}^{n_i \times k_i} \) is a matrix with \( e_i^{(p)} = [0 \ldots 0 1 0 \ldots 0]^T \in \mathbb{R}^{n_i} \) and \( p_i \) is the interpolation index, which can be calculated using the algorithm presented in [31]. This leads to an approximation of (23), described as

\[
f_i(t) \approx \bar{W}_i c_i(t) = \bar{W}_i (P_i^T \bar{W}_i)^{-1} P_i^T f_i(t).
\]

The DEIM allows the nonlinear parts in (14), \( f_i(x_i(t_j)) \) for \( i \in \{1, \ldots, 4\} \), to be approximated by

\[
f_i(x_i(t_j)) \approx \bar{W}_i (P_i^T \bar{W}_i)^{-1} P_i^T f_i(x_i(t_j)) = \bar{W}_i (P_i^T \bar{W}_i)^{-1} f_i(P_i^T x_i(t_j))
\]

and the approximation for the Jacobian of the nonlinear term in (15) becomes

\[
J_{f_i}(x_i(t_j)) \approx \bar{W}_i (P_i^T \bar{W}_i)^{-1} J_{f_i}(P_i^T \mathcal{U}_i \mathcal{U}_i^T x_i(t_j)) \mathcal{U}_i^T 0_i,
\]
\[ J_{f_i}(P_i^i x_i) = J_{f_i}(\bar{x}_i) = \text{diag}\left(f'(\bar{x}_i^{(1)}), \ldots, f'(\bar{x}_i^{(k)})\right), \]  

and \( \bar{x}_i(t_i) = P_i^T x_i(t_i) \), which can be computed with a small selected number of interpolation indices. It should be noted that the equality in (26) is due to a one-to-one correspondence between the state vector \( x_i \) and the nonlinear vector function given in (14), i.e., \( f_i(x_i) = f_i^{(k)} \). This is because of the FVM and FDM structures. Furthermore, \( U_i \frac{dW_i}{P_i^i} \) does not depend on time, meaning that it can be precomputed and the complexity in evaluating the nonlinearities of (22) and the other POD-based equations becomes proportional to the small selected number of interpolation indices \( k_i \).

### 3.1.5 Simulation study of proposed numerical scheme

In this section, computational performance and accuracy of the numerical scheme of Section 3.1.3 are studied for a 6 (Ah) Li-ion hybrid electric vehicle (HEV) battery with the parameters taken from [16] and [25]. It should be noted that all simulation results presented in the following and the conclusions drawn for these simulation results are specific for the battery (and its parameters) under consideration. The methodology proposed in this chapter and the analysis techniques presented below are general and can be repeated for cells with different model parameters.

#### 3.1.5.1 Computational Performance Analysis

The computational performance and accuracy of the proposed implementation depend on the spatial and temporal discretisation, the damping coefficient \( \gamma \), the tolerance used to determine convergence of (16), as well as the order of equations in which (16) is solved. The study is done for a constant current discharge of 5 C-rate (i.e., 30(A)), starting from 100% state-of-charge (SOC), i.e., a fully charged battery.

**Figure 29 Comparison of the battery voltages during the constant 5 C-rate discharge and starting from 100% SOC with (left) different size of grids for each phase, (right) different convergent tolerances and temporal discretisation steps.**

First, the influence of the number of spatial discretisation steps (i.e., the number of finite differences or control volumes) of the negative electrode \( n_n \), separator \( n_s \), positive electrode \( n_p \) and the solid particles \( n_r \) on the accuracy is analysed, while keeping the damping coefficient \( \gamma \), the tolerance tol and the temporal discretisation step \( \delta_t \) constant. We assume that increasing the number of discretisation steps leads to a more accurate approximation of the PDEs. The left figure in Figure 29 illustrates a comparison of the battery voltages with different size of grids for each phase during a constant 5C-rate discharge, starting from 100% SOC. Note that the depth-of-discharge (DOD) is defined as DOD = 1 – SOC. It can be observed from the left figure in Figure 2 that the number of spatial discretisation steps through the \( x \)-dimension of the model (i.e., \( n_n, n_s \) and \( n_p \)) influences the accuracy negligibly. Namely, increasing \( n_n, n_s \) and \( n_p \) from \( n_n = 12, n_s = 6 \) and \( n_p = 8 \) to \( n_n = 50, n_s = 25 \) and \( n_p = 36 \) hardly changes the solutions in the left-hand side of Figure 2. However, as can be seen from the left figure...
of Figure 2, the number of spatial discretisation steps of the solid particle (i.e., \(n_s\)) does play a significant role in accuracy. Namely, increasing \(n_r\) from \(n_r = 44\) to \(n_r = 110\), the solution changes, where we assume that \(n_s = 110\) is the most accurate result in this case. According to the left figure of Figure 2, increasing \(n_r\) from \(n_r = 44\) to \(n_r = 88\) leads to a significant change in voltage, while increasing \(n_r\) from \(n_r = 88\) to \(n_r = 110\) only leads to a small change in voltage. This conclusion is, at least for the battery type used in this chapter, in agreement with the simplifications made to obtain the so-called single particle model [14], where the focus is also on preserving the solid-state diffusion dynamics. To strike a proper balance between accuracy and speed, all the simulations in the remainder of this section are done with \(n_s = 12, n_s = 6, n_p = 8\) and \(n_r = 88\), resulting a 1832nd-order model.

In addition, the influence of the convergence tolerance \(\tau\) and temporal discretisation step \(\delta_t\) is studied, while both affecting the accuracy and computation time. The right figure of Figure 2 shows the comparison of the battery voltages with different convergent tolerances and temporal step sizes for, again, a 5 C-rate discharge, starting from 100 % SOC. It can be observed that the case where \(\tau = 10^{-2}\) and \(\delta_t = 0.7\) (s), indicated by the red dashed line, and the case in which \(\tau = 10^{-4}\) and \(\delta_t = 0.2\) (s), indicated by a black dotted line, coincide with each other throughout most of the discharge, with some small difference at the beginning. Still, simulating the model with \(\tau = 10^{-4}\) and \(\delta_t = 0.2\) (s) takes 1095 (s), while with \(\tau = 10^{-2}\) and \(\delta_t = 0.7\) (s) takes only 613 (s). It should also be noted from the right figure in Figure 2 that increasing the temporal discretisation step to \(\delta_t = 1\) (s) causes the model to fail to converge at the end of simulation (i.e., around DOD = 0.85), albeit at an improved computational performance.

As was observed in Section 3.1.3, the numerical scheme (16) allows for changing the order of subsystems \(i \in \{1, \ldots, 4\}\) in which they are solved. This might affect computational performance and convergence of the method. In particular, it affects the maximum allowable damping coefficient \(\gamma\) for which the algorithm converges, in the sense that convergence is achieved for \(k < K\), with \(K = 1000\). In general, a large damping coefficient \(\gamma\) means that convergence is achieved for smaller number of iterations \(k\), thereby improving computation time.

In Table 2, a comparison of the computation time with different arrangements of the subsystem sequences is shown for a 600 (s) long simulation with convergence tolerance \(\tau = 10^{-2}\) and temporal discretisation step \(\delta_t = 0.7\) (s) of, again, a 5 C-rate discharge, starting from 100 % SOC. It can be observed from this table that the maximum allowable damping coefficient \(\gamma\) and computation time differ when (16) is solved in a different order. Since typically the convergence speed decreases when \(\gamma\) decreases, we aimed at finding the largest \(\gamma\) that would lead to successful convergence of the numerical scheme. The fastest computation time is achieved by taking the sequence \(c_s-c_e-\phi_e-\phi_s\) and damping coefficient \(\gamma = 1\), which leads to a computation time almost half of the simulated real-time. While this result might vary in different simulations, this table is used as a reference to choose the order in which the subsystems in (16) are solved.

<table>
<thead>
<tr>
<th>Order of subsystems</th>
<th>Maximum coefficient (\gamma)</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_s-c_e-\phi_e-\phi_s)</td>
<td>1</td>
<td>271.868</td>
</tr>
<tr>
<td>(c_s-\phi_e-\phi_s-c_e)</td>
<td>1</td>
<td>281.948</td>
</tr>
<tr>
<td>(c_e-c_s-\phi_e-\phi_s)</td>
<td>1</td>
<td>292.480</td>
</tr>
<tr>
<td>(\phi_e-\phi_s-c_e-c_s)</td>
<td>1</td>
<td>326.850</td>
</tr>
<tr>
<td>(\phi_e-\phi_s-c_e-\phi_e)</td>
<td>1</td>
<td>357.362</td>
</tr>
<tr>
<td>(\phi_s-\phi_e-c_e-c_s)</td>
<td>1</td>
<td>359.004</td>
</tr>
<tr>
<td>(c_s-\phi_s-\phi_e)</td>
<td>1</td>
<td>400.177</td>
</tr>
<tr>
<td>(\phi_s-\phi_e-c_s-\phi_e)</td>
<td>1</td>
<td>493.219</td>
</tr>
<tr>
<td>(c_s-\phi_e-\phi_s)</td>
<td>1</td>
<td>513.315</td>
</tr>
<tr>
<td>(\phi_s-\phi_e-c_s-\phi_e)</td>
<td>1</td>
<td>516.376</td>
</tr>
<tr>
<td>(c_e-\phi_e-c_s-\phi_s)</td>
<td>1</td>
<td>534.828</td>
</tr>
<tr>
<td>(c_e-\phi_e-\phi_s)</td>
<td>1</td>
<td>545.520</td>
</tr>
</tbody>
</table>
3.1.5.2 Accuracy and Performance of analysis of the Numerical Model

Let us now compare the model implementation presented in Section 3.1.3 to the ones used in [16,23]. As a first step, a simulation of a constant 5 C-rate discharge, starting from 50 % SOC, has been done. Figure 30 depicts the simulation result of the electrode surface concentration $c_s$, the electrolyte concentration $c_a$, and the reaction current density $j_{Li}$, across the battery at various times, with convergence tolerance tol = $10^{-2}$ and temporal discretisation step $\delta t = 0.7$ (s). Then comparing this simulation results with the results presented in [16], it is concluded that the results are quite similar, with very slight difference that may be caused by the different numerical methods used to solve the DFN model.

To compare the computational performance, a constant 1 C-rate (i.e., 6 (A)) discharge, starting from 100 % SOC, has been simulated and the result is compared with the following models

- An iterative method with a 1832nd-order model described in [25];

**Figure 30 Concentration distributions and reaction current density during the constant 5 C-rate discharge at various times, starting from 50 % SOC. The electrode surface concentration, electrolyte concentration and reaction current density are shown, respectively.**

```plaintext
<table>
<thead>
<tr>
<th>Model</th>
<th>$C_s/c_{s,max}$</th>
<th>$j/L_i$ (A/cm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_e-cs-\phi_s-cs$</td>
<td>0.85</td>
<td>546.984</td>
</tr>
<tr>
<td>$\phi_e-cs-cs-e$</td>
<td>0.8</td>
<td>585.164</td>
</tr>
<tr>
<td>$\phi_e-cs-cs-cs$</td>
<td>0.8</td>
<td>408.385</td>
</tr>
<tr>
<td>$\phi_e-cs-cs-cs$</td>
<td>0.8</td>
<td>572.978</td>
</tr>
<tr>
<td>$\phi_e-e-cs-cs$</td>
<td>0.75</td>
<td>600.031</td>
</tr>
<tr>
<td>$\phi_e-e-cs-cs$</td>
<td>0.75</td>
<td>663.374</td>
</tr>
<tr>
<td>$\phi_e-e-cs-cs$</td>
<td>0.75</td>
<td>675.431</td>
</tr>
<tr>
<td>$\phi_e-e-cs-cs$</td>
<td>0.75</td>
<td>728.444</td>
</tr>
</tbody>
</table>
```
• An implementation using (16) for the 1832nd-order model with the same simplification used in [25] that we call simplified implementation;
• An implementation using Matlab solver `fsolve' to solve (14) directly, where a 952nd-order model is used due to limitations of `fsolve'.

It should be noted that the computationally efficient implementation proposed in [25] uses a different iterative numerical scheme and that the terminal voltage of the model implementation is validated with the experimental data from [24]. However, the implementation in [25] uses a simplification of the open-circuit potential equation. To be precise, the open-circuit equation is computed using an averaged concentration referring to equations (56)-(58) in [16], instead of using the surface concentration $c_{x, e}$. This leads to an open-circuit potential that is independent of the spatial variable $x$. Furthermore, (14) can be solved by the Matlab solver `fsolve'. However, `fsolve' is very slow in solving stiff problems, hence a lower-order model has been implemented by `fsolve', compared to the other numerical implementations.

The simulation results of the proposed 1832nd-order model implementation is given in a blue solid line in Figure 31, while the red dashed line indicates the battery voltage from [25], implemented by the Matlab code presented in the appendix of [34]. To demonstrate that the difference between these two models can be attributed to the averaging done in the computation of the open-circuit voltage, this simplification is also implemented and solved using the numerical scheme proposed in this chapter. The simulated voltage using this simplification is given by the green dash-dotted line in Figure 31. In addition, the result of implementation done by Matlab solver `fsolve' is indicated by the black dotted line.

![Graph showing voltage response for different model implementations](image)

**Figure 31 Voltage response for different model implementations, starting from 100% SOC during (a) constant 1 C-rate discharge, (b) dynamic (dis)charge profile accompanied by the current profile.**

It can be observed from Figure 31 that the prediction of terminal voltage from the implementation in [25] and the simplified model implementation (in which the open-circuit potential does not depend on $x$) fit quite well, indicating the same accuracy for both implementations. However, the proposed (un simplified) model implementation (in which the open-circuit potential is computed using $c_{x, e}/c_{e, max}$) is slightly different from that in [25]. Nevertheless, the difference is very small and it does not influence the trends of the prediction. With a lower-order model, the implementation by `fsolve' also has a lower accuracy than the other three implementations at the end of the voltage prediction. For a 600 (s) duration, with the same temporal discretisation step $\delta t = 1$ (s) as [25], the implementation proposed in [25] takes around 144 (s) for 1 C-rate discharge using Matlab 2016a on a laptop with a 2.9 (Ghz) Intel Core i7 processor, whereas 207 (s) by the implementation proposed in this chapter and 3653 (s) by Matlab solver `fsolve'.

To further investigate the difference between the model presented in [25] and the model proposed in this chapter, another simulation is done using a dynamic (dis)charge profile, shown in right side of Figure 31. From this figure, it can be observed that the terminal voltage obtained by the model implementation described in [25] and the simplified implementation fit very well, but are quite
different from the model implementation proposed in Section 3.1.3 and that done by Matlab solver `fsolve'. The main difference is caused by simplification on the open-circuit potentials. During the 1200 (s) dynamic (dis)charge profile, the implementation proposed in this chapter takes around 279 (s), whereas the implementation from [25] takes 878 (s) and the implementation done by `fsolve' takes 3911 (s). As such, the computational efficiency varies with respect to different (dis)charge profile and discretisation step sizes. A remarkable result is that the model implementation of 3.1.3 can provide a computation time that is 3-5 times faster than real-time and 14-17 times faster than Matlab solver `fsolve'.

### 3.1.6 Simulation Study of POD-DEIM-Based Reduced-Order Model

In this section, computational performance and accuracy of the POD-DEIM-based ROM presented in Section 3.1.4, implemented by the numerical scheme of Section 3.1.3, are studied using the same HEV battery from Section 3.1.5. The simulation results of the ROM are compared to the full-order model with state dimension 1832, presented in the previous section.

#### 3.1.6.1 Study of Snapshot and Number of Reduced Basis

As mentioned in Section 3.1.4, the choice of snapshot is an important factor in constructing the POD basis as it influences the accuracy of the ROM considerably. The main idea of building up the snapshot is to collect the data set reflecting the most relevant system dynamics. As such, the dynamic (dis)charge profile in Figure 31 is chosen to construct the snapshot in the simulation study. Moreover, some scaling of the current amplitude and temporal discretisation are applied to Figure 3.1.4 while generating the snapshot, in order to study the influence of the C-rate and the temporal step sizes of the snapshot on the accuracy of the ROM implementation.

First, the influence of the maximum C-rate defined for the snapshot on the accuracy of the ROM is analysed, while keeping the temporal discretisation step $\delta_t$ constant. Figure 32 shows a comparison of the full order model and the reduced order model with snapshots constructed at different maximum C-rates in terms of battery voltage (cf. the left figure in Figure 32) and absolute error between the full order model and the reduced order models (cf. the right figure in Figure 32). From this figure, it can be seen that the accuracy of the battery voltages increases as C-rate decreases, and the maximum C-rate influences the simulation result negligibly when it is less than 0.5 C-rate (i.e., 3 (A)). Moreover, the C-rates generating the snapshots do not influence the computation times as all are in the range of 155-160 (s). The maximum C-rate can be chosen from any of these four C-rates as long as the absolute error is less than a certain small number (e.g., $5 \times 10^{-3}$ (V) as the criterion in this section). Here, 0.25 C-rate (i.e., 1.5 (A)) is chosen to generate the snapshot.

![Figure 32](image-url) Comparison of the full order model and the reduced order model with snapshots constructed at different maximum C-rates in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure).

After analysis of the C-rates, the influence of the temporal discretisation step $\delta_t$ on the accuracy and computation time is studied. With the same time length, the size of the snapshot matrix increases as $\delta_t$ decreases. This consequently increases the computation time of the ROM basis. According to the obtained results, the temporal discretisation step does not influence the accuracy of the ROM. The computation time changes slightly as long as the temporal discretisation step is in a moderate range that does not cause a huge snapshot matrix. As such, the temporal discretisation step in the snapshot is chosen as $\delta_t = 0.1$ (s).
According to (21), the number of reduced basis can be determined by the summation of corresponding eigenvalues that captures most of the system energy, given by

$$\text{Energy} = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.99999$$

where $q$ is the number of reduced basis and $n$ is the total number of basis. With respect to (29), the reduced basis of each system of equations for $c_s$, $c_s$, $\phi_e$ and $\phi_s$ becomes $q_1 = 156$, $q_2 = 26$, $q_3 = 26$ and $q_4 = 20$, respectively.

Figure 33 illustrates a comparison of the full order model and the reduced order model with snapshots constructed at different numbers of reduced basis in terms of battery voltage (cf. the left figure in Figure 33) and absolute error between the full order model and the reduced order models (cf. the right figure in Figure 33). It can be observed from the blue solid line and the red dash-dotted line in Figure 33 that the accuracy of the ROM does not change by decreasing the number of reduced basis of $c_s$, $\phi_e$, and $\phi_s$ from $q_2 = 26$, $q_3 = 26$ and $q_4 = 20$ to $q_2 = 4$, $q_3 = 4$ and $q_4 = 4$, respectively. However, changing the number of basis of $c_s$ does play a significant role in accuracy. The accuracy and computation time decrease as the number of reduced basis of $c_s$ decreases from $q_1 = 156$ to $q_1 = 44$. Moreover, the accuracy becomes poor at the end of the battery voltage prediction, once the number of reduced basis of $c_s$ is lower than 66. It even becomes out of the criterion when $q_1 = 44$. To strike a balance between the accuracy and computation time, the number of reduced basis of each system of equations are suggested to be chosen higher than $q_1 = 66$, $q_2 = 4$, $q_3 = 4$ and $q_4 = 4$.

As discussed in Section 3.1.5, the convergence tolerance $tol$ in the proposed implementation affects the accuracy and computation time. However, the value of tolerance for the ROM implementation cannot be easily compared to that in the previous section, since the ROM has a much smaller state dimension than the full-order model that makes it more difficult to converge. Figure 34 represents a comparison of the full order model and the reduced order model implemented using different convergence tolerances in terms of battery voltage (cf. the left figure in Figure 34) and absolute error between the full order model and the reduced order models (cf. the right figure in Figure 34). It can be observed that the accuracy gets poor with the growth of $tol$, and the computation time decreases (e.g., implementation with $tol = 0.1$ takes 264 (s), $tol = 0.5$ takes 154 (s) and $tol = 1$ takes 126 (s)). While with large tolerances the computation time decreases, the absolute errors exceed $5 \times 10^{-3}$ (V). As such, the rest of simulations are done with $tol = 0.5$.

Figure 33 Comparison of the full order model and the reduced order model with snapshots constructed at different numbers of reduced basis in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure).

Figure 34 Comparison of the full order model and the reduced order model implemented using different convergence tolerances in terms of battery voltage (top figure) and absolute error between the full order model and the reduced order models (bottom figure).
3.1.6.2 Accuracy and Performance Analysis of the POD-DEIM-based ROM

Let us compare the simulation results of the POD-DEIM-based ROM to the results of our proposed implementation with full spatial and temporal discretisation described in Section 3.1.3. It should be noted that the simulation results of the full-order model are obtained by a 1832nd-order model with the number of states $n_{cs} = 1760$, $n_{ce} = 26$, $n_{ph_e} = 26$ and $n_{ph_s} = 20$. According to the simulation study in Section 3.1.6.1, the POD-DEIM-based ROM is chosen as a 100th-order model with number of states $n_{cs} = 88$, $n_{ce} = 4$, $n_{ph_e} = 4$ and $n_{ph_s} = 4$.

Figure 35 illustrates a comparison of the battery voltages between the full-order model and the POD-DEIM-based ROM during a constant 5 C-rate discharge, starting from 100 % SOC. As can be observed during the constant 5 C-rate discharge, the terminal voltage of the POD-DEIM-based ROM fits the full-order model quite well. For a 600 (s) duration with the temporal discretisation step $\delta_t = 1$ (s), the full-order model implementation takes 168 (s), whereas the POD-DEIM-based ROM takes 138 (s). Furthermore, the POD-DEIM method reduces the model from 1832nd-order to 100th-order while maintaining the same accuracy during the 5C-rate discharge.

To further study the computational efficiency of the POD-DEIM-based ROM implementation, another simulation has been done applying a 3000 (s) pulse (dis)charge profile, shown in Figure 36. The POD-DEIM-based ROM has a very similar accuracy to the full-order model, with the absolute error smaller than $5 \times 10^{-3}$ (V). For this 3000 (s) pulse (dis)charge with the temporal discretisation step $\delta_t = 1$ (s), the full-order model implementation takes 1805 (s), whereas the POD-DEIM-based ROM implementation takes 85 (s).

Moreover, Table 3 states the computation time of the full-order model and the POD-DEIM-based ROM during a constant discharge, pulse (dis)charge and dynamic (dis)charge profile. A notable result is that the POD-DEIM-based ROM implementation done by our proposed numerical scheme can realise a computation time that is 4-35 times faster than real-time. Furthermore, the reduction in the model order leads to a model that requires less memory to store the model equations, which is useful in real-time embedded applications.
Figure 36: Comparison of the battery voltages between the full-order model and the POD-DEIM-based ROM, during a pulse charge/discharge starting from 100 % SOC.

Table 3 Comparison between the computation time of the full-order model and the POD-DEIM-based model

<table>
<thead>
<tr>
<th>(dis)charge profile</th>
<th>Full-order model computation time (s)</th>
<th>POD-DEIM-based ROM computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 C-rate discharge with 600 (s) duration</td>
<td>168</td>
<td>138</td>
</tr>
<tr>
<td>Pulse (dis)charge with 3000 (s) duration</td>
<td>1805</td>
<td>85</td>
</tr>
<tr>
<td>Hybrid (dis)charge profile with 1200 (s) duration</td>
<td>279</td>
<td>157</td>
</tr>
</tbody>
</table>

Figure 37 represents the concentration distributions and reaction current density of the POD-DEIM-based ROM during the constant 5 C-rate discharge across the battery at various times, starting from 50 % SOC. With the POD-DEIM-based ROM, it can still be observed that the internal states such as electrode surface concentration $c_{s,e}$, the electrolyte concentration $c_e$, electrode potential $\phi_s$, and electrolyte potential $\phi_a$. As shown in Figure 37, the electrode surface concentration $c_{s,e}$ has the same performance as Figure 30 except some small deviations on both side of the battery. The electrolyte concentration $c_e$ arrives at a steady-state distribution at the both sides of the battery a bit later than Figure 3, but this can be fixed by increasing the order of the ROM. However, the reaction current density is not accurate any more due to the way of description of the model reduction problem.
3.1.7 CONCLUSIONS

The Doyle-Fuller-Newman (DFN) model is a one-dimensional physics-based model that accurately describes the internal behaviour of Li-ion batteries. The model is governed by a set of coupled nonlinear partial differential equations (PDEs), which makes its implementation computationally costly. In this chapter, a computationally efficient implementation of the full nonlinear DFN model has been presented. The proposed implementation is based on applying model order reduction (MOR) to a spatial and temporal discretisation of the nonlinear PDEs. The proper orthogonal decomposition and discrete empirical interpolation method have been used for MOR, which lead to a set of reduced order nonlinear algebraic equations. Then, these equations are solved using a damped Newton's method. In a simulation study, the computational efficiency of the proposed implementation is shown, and the resulting accuracy is presented.

An extensive simulation study has been done to analyse and demonstrate the computational performance and accuracy of the implementation and shows the potential for real-time applications. For the battery used in the simulation study, the full-order model implementation can provide a computation time that is 3-5 times faster than real-time. Additionally, the POD-DEIM-based reduced-order model implementation can achieve a computation time that is 4-35 times faster than real-time and reduce the model order significantly (18 times), while maintaining most of the accuracy of the full-order model implementation. Furthermore, the less memory is needed to store the model equations, which is desirable for embedded real-time application.

Since the proposed method has been applied on an electrochemistry-based model, it properly models the physical behaviour and internal states of the battery. Therefore, future work will focus on using the developed reduced order model for real-time state-of-charge estimation, as well as optimal fast charging. Furthermore, the method presented in this chapter can be applied to more accurate battery models, leading to a more accurate reduced order model.
REFERENCES


ON TRADE-OFFS BETWEEN COMPUTATIONAL COMPLEXITY AND ACCURACY OF ELECTROCHEMISTRY-BASED BATTERY MODELS

In this chapter, we propose several simplifications to the so-called Doyle-Fuller-Newman (DFN) model, which is a popular electrochemistry-based battery model. This simplified DFN (SDFN) model allows for a computationally very efficient implementation. The simplifications are as a result of several assumptions, which will be justified for two different parameter sets. Furthermore, we will show how a trade-off can be made for model complexity, through the coarseness of the discretization, between model accuracy and computation time for this SDFN model. Finally, the SDFN model proposed is compared to the DFN model as well as an implementation of the single-particle model, for the two parameter sets. This will show that by making specific assumptions, simplifications can be made that have no significant impact on the model accuracy, while the computation time can be drastically decreased. This leads to a simulation time of over 3600 times faster than real-time.

3.2.1 INTRODUCTION

With the emergence of electric vehicles and widespread usage of mobile devices, batteries are taking an increasingly important role. To effectively manage and control batteries, battery models are needed, e.g., for state-of-charge/state-of-health estimation [1], active balancing [2], and fast charging [3,4]. Equivalent circuit models (ECM) are often used for these purposes, where the battery is modelled using (passive) circuit elements [5,6]. However, although ECMs are computationally fast, these types of models provide limited information on the internal states of the battery, such as concentrations and potentials. The information of these internal states is crucial for more precise control. Therefore, electrochemistry-based models have generated increasing interest in control applications. A widely-used electrochemistry-based model is the so-called Doyle-Fuller-Newman (DFN) model [7], which is described by a set of partial differential equations (PDEs). While the DFN model can describe internal states, due to its complexity, computation times to simulate the model have made it, so far, unsuitable for control-oriented applications.

There are several ways to reduce the computational burden of the DFN model. One way to reduce complexity is to apply model reduction to the DFN model. A common technique is to make a polynomial approximation of the solid-phase concentration diffusion [8,9]. Model reduction is also applied on the complete DFN model through the use of techniques such as Galerkin projections [10] and proper orthogonal decomposition [11,12]. An overview of this type of model reduction can be found in [13]. However, the reduction is mostly in the number of equations, while the reduction in computation time is only marginal, when applied to the DFN model [12]. Another way to reduce complexity is by simplifying the model equations, see, e.g., [14]. A popular example is the so-called single-particle model (SPM) [15,16], in which the equations for solid-phase concentration and potentials are simplified. Another approach is to linearize the governing equations of the DFN model, see e.g. [17]. However, generally such simplifications lead to a significant loss in accuracy, and, usually, the simplified models are validated for a single parameter set. While a simplified model could show good accordance with the DFN model for one parameter set, it may not be the case for another parameter set.

Besides addressing the complexity of the DFN model, another way to reduce computation time, is to develop algorithms that compute the solution to the DFN model more efficiently, see e.g. [12,18]. In these papers, the PDEs that describe the DFN model are spatially and temporally discretized, which results in a set of nonlinear algebraic equations (AEs). Numerical methods are then developed in an attempt to solve this set of AEs as efficiently as possible.

In this chapter, we propose to simplify the equations of the DFN model, to arrive at a simplified DFN (SDFN) model. This SDFN model aims to allow for a more computationally efficient implementation, thereby reducing computation time. After spatial and temporal discretization of the SDFN model, we will show that through substitution of equations, the number of equations describing the SDFN model can be considerably reduced to a small set of algebraic equations. We will study the impact of these assumptions on model accuracy for two different sets of parameters, to show the validity of these
assumptions. Furthermore, we will study the impact of varying spatial accuracy on the trade-off between model accuracy and computation time. Finally, we compare the model accuracy and computation time of the DFN model, the proposed SDFN model, and the SPM for two different parameter sets. We will show that based on the proposed assumptions, the simplifications made allow for a significant decrease in simulation time, resulting in a simulation speed of over 3600 times faster than real-time.

The remainder of this chapter is as follows. In Section 3.2.2, the DFN model is formulated, after which the assumptions are given that lead to the proposed SDFN model and the SPM. In Section 3.2.3, a computationally efficient implementation of the SDFN model is proposed. In Section 3.2.4, the assumptions made for the SDFN model are validated, after which a computational complexity analysis of the SDFN is performed, and the results of the comparison between the DFN model, SDFN model and SPM model are shown. Finally, conclusions are drawn in Section 3.2.5.

### 3.2.2 Battery Modelling

In this section, we briefly formulate the Doyle-Fuller-Newman (DFN) model, and introduce several simplifications of which we will show in Section 3.2.4 have no significant impact on the accuracy of the model, both in the input-output behaviour as well as the internal states. Further simplifications on the resulting model will be applied to arrive at the so-called single particle model (SPM), and the SPM with electrolyte dynamics (SPMe) [15].

#### 3.2.2.1 Doyle-Fuller Newman Model

The DFN model is a widely used electrochemistry-based model introduced in [7]. Figure 38 illustrates the modelling approach for a Li-ion cell. In the $x$ dimension, the cell is divided into three regions, namely the negative electrode, the separator, and the positive electrode. In the electrodes, Li-ions exist essentially in two phases. In the solid phase, Li-ions are intercalated into the solid-phase material, which is represented by spheres with radius $R_s$. In the electrolyte phase, Li-ions exist in a dissolved state in the electrolyte. In the separator, Li-ions exist only in the electrolyte phase. During charging, intercalated Li-ions exit the solid particles in the positive electrode and enter the solid particles in the negative electrode. During discharging, the opposite process happens. We will shortly summarize the governing equations of the DFN model, which will be mostly based on the formulation given in [19].

![Figure 38 DFN modelling approach for a Li-ion cell.](image)

The DFN model is governed by four coupled partial differential equations (PDEs):

- The Li-ion concentration in the solid phase $c_s$ for $x \in [0, \delta_-) \cup (L - \delta_+, L]$ is given by Fick’s law as
  \[
  \frac{\partial c_s}{\partial t} = \frac{D_s}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial c_s}{\partial r} \right),
  \]  
  with boundary conditions
  \[
  \left. \frac{\partial c_s}{\partial r} \right|_{r=0} = 0, \quad -D_s \left. \frac{\partial c_s}{\partial r} \right|_{r=R_s} = j_n
  \]
with $D_s$ the diffusion coefficient of lithium in the solid phase, and $j_n$ is the net molar flux of Li-ions exiting the particle. Furthermore, $\delta_-$ and $\delta_+$ are the thickness of the negative electrode and positive electrode, respectively, and $L$ is the thickness of the cell, see Figure 1.

- The Li-ion concentration in the electrolyte phase $c_e$ for $x \in [0,L]$ is given by
  \[
  \varepsilon_e \frac{\partial c_e}{\partial t} = \frac{\partial}{\partial x} \left( D_e \frac{\partial c_e}{\partial x} + a_s (1 - t_+^e) j_n \right) \tag{2a}
  \]
  with boundary conditions
  \[
  D_e^\text{eff} \frac{\partial c_e}{\partial x} |_{x=0} = D_e \frac{\partial c_e}{\partial x} |_{x=L} = 0 \tag{2b}
  \]
  where $D_e^\text{eff} = D_e \varepsilon_e^p$ is the effective Li-ion diffusion coefficient in the electrolyte phase, in which $\varepsilon_e$ is the electrolyte phase volume fraction, $\varepsilon_s$ is the Bruggeman porosity exponent, and $D_e$ is the diffusion constant of Li-ions in the electrolyte. Furthermore, in (2), $a_s = 3 \varepsilon_e / R_s$ is the specific interfacial surface area, in which $\varepsilon_+$ is the active material volume fraction, and $t_+^e$ is the transference number of Li ions.

- The potential in the solid phase $\phi_s$ for $x \in [0, \delta_-] \cup [L - \delta_+, L]$ is given by Ohm’s law, i.e.,
  \[
  \frac{\partial}{\partial x} (\sigma^\text{eff} \frac{\partial \phi_s}{\partial x}) = a_s F j_n \tag{3a}
  \]
  with boundary conditions
  \[
  \sigma^\text{eff} \frac{\partial \phi_s}{\partial x} |_{x=0} = \frac{j_{\text{app}}}{A_{\text{surf}}}, \quad \sigma^\text{eff} \frac{\partial \phi_s}{\partial x} |_{x=\delta_-} = 0 \tag{3b}
  \]
  \[
  \sigma^\text{eff} \frac{\partial \phi_s}{\partial x} |_{x=L-\delta_+} = \frac{j_{\text{app}}}{A_{\text{surf}}} \tag{3c}
  \]
  where $F$ is Faraday’s constant, $\sigma^\text{eff} = \varepsilon_e$ is the effective electronic conductivity of a porous electrode, in which $\sigma$ is the conductivity of the solid material, $A_{\text{surf}}$ is the area of the electrode plate, and $j_{\text{app}}$ is the applied current through the battery, with $j_{\text{app}} > 0$ indicating charging.

- The potential in the electrolyte phase $\phi_e$ for $x \in [0,L]$ is given by
  \[
  \frac{\partial}{\partial x} \left( \kappa^\text{eff} \frac{\partial \phi_e}{\partial x} + \kappa^\text{eff} \frac{2RT}{F} (t_+^0 - 1) \frac{\partial \ln c_e}{\partial x} \right) = -a_s F j_n \tag{4a}
  \]
  with boundary conditions
  \[
  \kappa^\text{eff} \frac{\partial \phi_e}{\partial x} |_{x=0} = \kappa^\text{eff} \frac{\partial \phi_e}{\partial x} |_{x=L} = 0, \tag{4b}
  \]
  in which $\kappa^\text{eff} = \kappa \varepsilon_e^p$ is the effective ionic conductivity, where $\kappa$ is given by $\kappa = 0.00158 c_e \exp(5.63 c_e^{1/4} / 10^3)$, $R$ is the universal gas constant, and $T$ is the absolute temperature.

The above PDEs (1)-(4) are coupled by a Butler-Volmer rate equation, which describes the chemical reaction rate at the solid/electrolyte interface. This rate equation is given by
  \[
  j_n = \frac{i_0}{F} \left( \exp \left( \frac{\alpha_a F}{RT} \eta \right) - \exp \left( - \frac{\alpha_c F}{RT} \eta \right) \right), \tag{5a}
  \]
  which is only defined for $x \in [0, \delta_-] \cup [L - \delta_+, L]$ and assumed zero for $x \in (\delta_-, L - \delta_+)$. In (5a), $\alpha_a$ is the anodic transfer coefficient, $\alpha_c$ is the cathodic transfer coefficient, and the overpotential at the electrodes $\eta$ is defined as
  \[
  \eta = \phi_x - \phi_e - U, \tag{5b}
  \]
  in which $U$ denotes the equilibrium potential of the electrode, which can be given by a pre-defined function typically of the solid-phase concentration at the solid-electrolyte interface $c_e(x,t) = c_s(R_s,x,t)$. Furthermore, the exchange current density $i_0$ in (5a) is given by
  \[
  i_0 = k_0 c_e^{\alpha_q} (c_{e,\text{max}} - c_e)^{\alpha_a - \alpha_q}, \tag{5c}
  \]
  where $k_0$ is the rate constant of the electrochemical reaction, and $c_{e,\text{max}}$ is the maximum concentration in the solid-phase. Note that since there are no particles in the separator, $j_n$ can be defined to be zero in this region.

Finally, the terminal battery voltage is computed with
  \[
  V(t) = \phi_x(L,t) - \phi_x(0,t) + \frac{R_f}{A_{\text{surf}}} i_{\text{app}}(t), \tag{6}
  \]
  in which $R_f$ is an empirical contact resistance.
3.2.2.2 Simplified Doyle-Fuller-Newman Model

In order to reduce computational complexity, some simplifications can be made on the DFN model. We will show using simulation results in Section 3.2.4 that the simplifications proposed below do not result in any significant sacrifice in accuracy for two different parameter sets. The assumptions made to arrive at this simplified model are as follows.

[A1] The transfer coefficients \(a_a\) and \(a_c\) are assumed to be equal. Since \(a_a + a_c = 1\) by definition, we have \(a_a = a_c = 0.5\). This assumption is common, see e.g. [1,3-16,18].

[A2] Under [A1], the rate equation can be linearized with respect to the overpotential \(\eta\) around the origin, due to the fact that \(\frac{a_c}{RT} = \frac{a_a}{RT} \gg \eta\).

[A3] The effective ionic conductivity \(\kappa_{eff}\) is constant over \(x\), i.e., \(\kappa_{eff} = \kappa(c_e) c_{p}^e\), where \(c_e\) is the average concentration in the electrolyte.

[A4] The time derivative of the electrolyte concentration \(\frac{\partial c_e}{\partial t}\) is sufficiently small so that the term \(\ln c_e\) in (4) can be linearized with respect to \(c_e\) around a linearization point \(c_{e_0}\). Thus, \(\ln c_e \approx \ln c_{e_0} + \frac{c_{e_0} - c_{e}}{c_{e_0}}\).

Using [A1] and [A2], the resulting linearized Butler-Volmer equation (of (5a)-(5b)) can be written as

\[
\dot{j}_n = \frac{i_0}{RT}(\phi_s - \phi_e - U) - \eta
\]

Furthermore using [A3] and [A4], (8) can be simplified to

\[
\frac{\partial}{\partial x} \left( \kappa_{eff} \frac{\partial \phi_e}{\partial x} + \frac{2RT}{F} (t_e^0 - 1) \frac{\partial}{\partial x} \left( \ln c_e^* + \frac{c_{e_0} - c_e}{c_{e_0}} \right) \right) = -a_s F j_n
\]

with the boundary conditions given in (4b). Thus, the simplified DFN model is given by (1)-(3), (8), (4b), (7), (5c) and (6). Note that through the simplifications we have eliminated much of the non-linearity of the DFN model. The only non-linearities with respect to the state variables \(c_s, c_e, \phi_s, \phi_e\) remain in the equation for \(i_0(c_s, c_e)\) as well as the equilibrium potential \(U(c_e)\).

3.2.2.3 Single-Particle Model

The SPM is defined by the main assumption that the diffusion dynamics inside the solid particles is the slowest process, and therefore dominates over the other dynamics [20]. Under this assumption, we can formulate the following corresponding assumptions [15].

[A5] The concentration in the solid phase \(c_s\) is constant over \(x\).

[A6] The exchange current density \(i_0\) can be approximated by \(\tilde{i}_0 = i_0(c_e^*, c_s)\), where \(c_s^*\) is the mean electrolyte concentration over each of the electrodes. Note that as a consequence from [A5], \(\tilde{i}_0\) only changes over time.

As a result of [A5], the flux of Li-ions at the solid-electrolyte interface \(j_n\) can be assumed to be equal for all particles in each of the electrodes, and hence \(j_n\) can be directly computed from (3), i.e.,

\[
\tilde{j}_n(x, t) = \begin{cases} \frac{i_{app}(t)}{A_{surf}(\delta + F)} & \text{for } x \in [0, L] \\ \frac{i_{app}(t)}{A_{surf}(\delta - F)} & \text{for } x \in [L - \delta, L] \end{cases}
\]

Note that, as a result of (9), we have that (1), (2), and (8) are no longer coupled, which means that these equations can be solved explicitly for a given \(i_{app}\). Moreover, to further increase computational efficiency, (8) can be analytically integrated twice, to obtain

\[
\phi_e(L, t) - \phi_e(0, t) - \frac{2L - \delta_e - \delta_s}{2\kappa_{eff} A_{surf}} i_{app}(t) = \frac{2RT}{F} (1 - \iota_s^0) \left( \ln c_e^*(L, t) - \ln c_e^*(0, t) + \frac{c_e(L, t)}{c_e^*(L, t)} - \frac{c_e(0, t)}{c_e^*(0, t)} \right)
\]

Then, with [A6], the solid-phase potentials can be obtained from (7) and (9), and substituted into (6), which gives

\[
\Psi(i) = \frac{RT}{i_0(L) A_{surf}(\delta - F)} i_{app}(t) - \frac{RT}{i_0(0) A_{surf}(\delta + F)} i_{app}(t) - \phi_e(L, t) + \phi_e(0, t) + U(c_e(L, t)) - \psi(c_e(0, t)) = \frac{RT}{A_{surf}} i_{app}(t).
\]

This model, given by (1), (2), (5c), (10) and (11), is usually referred to as the SPMs (Single Particle Model with electrolyte dynamics), while SPM is usually referred to the model that is obtained after removing the \(\phi_e(L, t) - \phi_e(0, t)\) term in (11) [15]. Furthermore, while the derivation of the SPMs and
SPM model given here is based on the work in [15], there are some differences between the models obtained. These differences lie in the fact that the above-formulated SPM model is derived from a simplified DFN model, using Assumptions [A1]-[A4], while in [15], the SPM model is derived from the DFN model given by (1)-(6), with [A3] applied. We will show the validity of these assumptions in Section 3.2.4.

### 3.2.3 Model Implementation

The objective of this chapter is to compare the computational complexity and the accuracy of several electrochemistry-based models, including the newly proposed SDFN model. To have low computational complexity of the proposed SDFN model, we propose a computationally efficient implementation of this model. This implementation is warranted by Assumptions [A1]-[A4] and involves several steps. Firstly, fairly standard spatial and temporal discretization is applied to arrive at a set of algebraic equations (AEs). Secondly, the set of AEs is reduced to a smaller set of AEs through substitution, after which the resulting set of AEs can be solved using Newton’s method. The discretization procedure described here is similar to the procedure presented in [12]. Therefore, in this section, we will shortly summarize this procedure to formulate the set of AEs that arise from the discretization. For further details on the discretization approach, the reader is referred to [12].

#### 3.2.3.1 Discretization

As a first step, spatial discretization is applied on the partial differential equations (PDEs) (1)-(3) and (8). The equation describing the diffusion of the solid-phase concentration (1) is discretized along the radial direction using a finite-difference-method (FDM), to arrive at a set of differential algebraic equations (DAEs). The other equations (2), (3), and (8) are discretized using a finite-volume-method (FVM), after which the resulting set of nonlinear DAEs can be written as

\[
\frac{d}{dt} c_s = A_{cs} c_s + B_{cs} j_n \quad (12a)
\]

\[
\frac{d}{dt} c_e = A_{ce} c_e + B_{ce} j_n \quad (12b)
\]

\[
0 = A_{eA} \phi_s + B_{eA} j_n + C_{eA} j \quad (12c)
\]

\[
0 = A_{e\phi} \phi_s + B_{e\phi} j_n + C_{e\phi} j \quad (12d)
\]

where the bold-faced characters refer to their respective vector variables, which are defined as

\[
c_s(t) = \left[ c_s(x_i, r_j, t), \ldots, c_s(x_i, r_{n,r,n}, t), \ldots, c_s(x_{n_n + n_p}, r_{n,r,n}, t) \right]^T \quad (13a)
\]

\[
c_e(t) = \left[ c_e(x_i, t), \ldots, c_e(x_{n_n + n_p}, t) \right]^T \quad (13b)
\]

\[
\phi_s(t) = \left[ \phi_s(x_1, t), \ldots, \phi_s(x_{n_n}, t), \ldots, \phi_s(x_{n_n + n_p}, t) \right]^T \quad (13c)
\]

where \( x_i \) and \( r_j \) are the grid points of the discretization, and \( \phi_s, j_n \) are defined similarly to \( c_s \) and \( \phi_s \), respectively. Furthermore, in (13) \( n_n, n_p, n_p \) are the number of elements of the FVM discretization, and \( n_r, n_r, n_p \) are the number of elements of the FDM discretization. How to construct matrices \( A_{ij}, B_{ij}, i \in \{c_s, c_e, \phi_s, \phi_e\}, C_{ij}, \) and \( D_{\phi_e} \) is explained in detail in [12]. Note that in contrast to the set of DAEs obtained after spatial discretization in [12], except for the coupling through \( j_n \), the four sets of equations (12) are linear in the state variables \( c_s, c_e, \phi_s, \phi_e \). The four sets of DAEs (12) are coupled by the linearized (with respect to \( \eta = \phi_s - \phi_e - \theta \)) Butler-Volmer rate equation, written as

\[
j_n = \text{diag} \left( \frac{\theta (\bar{c}_s, \bar{c}_e)}{RT} \right) (\bar{\phi}_s - \bar{\phi}_e - \bar{U}(\bar{c}_s)) \quad (14)
\]

in which the diag() denotes a diagonal matrix with the elements of vector \( \nu \) on the main diagonal. Furthermore, in (14), the barred variables \( \bar{c}_s, \bar{c}_e, \bar{\phi}_s, \) and \( \bar{\phi}_e \) refer to selected version of their boldfaced counterparts, where \( \bar{c}_s \) denotes the vector of solid-phase surface concentrations, and \( \bar{c}_e \) and \( \bar{\phi}_e \) denote the parts of \( c_e \) and \( \phi_e \) given in the electrodes, respectively. Mathematically, \( \bar{c}_s, \bar{c}_e, \bar{\phi}_s \) and \( \bar{c}_s, \bar{c}_e, \bar{\phi}_e \) are related, respectively, as follows

\[
\bar{c}_s = A_{cs} c_s, \bar{c}_e = A_{ce} c_e, \bar{\phi}_e = A_{\phi_e} \phi_e, \quad (15a)
\]

where
The differential equations (12) can further be discretized in time with sampling time \( \delta_t \) using a backward Euler scheme, to arrive at the following set of AEs

\[
\begin{align*}
0 &= \bar{A}_{c_k}(t_{k}) + \bar{B}_{c_k} \phi_{t_{k}}(t_{k}) + c_k(t_{k-1}) \\
0 &= \bar{A}_{s_k}(t_{k}) + \bar{B}_{s_k} \phi_{t_{k}}(t_{k}) + c_k(t_{k-1}) \\
0 &= A_{\phi_k, \Phi_k}(t_{k}) + B_{\phi_k, \Phi_k}(t_{k}) + c_k(t_{k}) \\
0 &= A_{\phi_k, \Phi_k}(t_{k}) + B_{\phi_k, \Phi_k}(t_{k}) + D_{\phi_k} \left( \Xi_{\Phi_k} \phi_{c_k}(t_{k}) + \Xi_{\Phi_k} \phi_{c_k}(t_{k}) \right)
\end{align*}
\]  

where \( t_k \in \{1, ..., t_f \} \) is the sample time, in which \( t_f \) is the final sample time. Furthermore, \( \bar{A}_{c_k} = \delta_t A_{c_k} - I \), \( \bar{B}_{c_k} = \delta_t B_{c_k} \), \( \bar{A}_{s_k} = \delta_t A_{s_k} - I \), \( \bar{B}_{s_k} = \delta_t B_{s_k} \), \( \Xi_{\Phi_k} = \text{diag}(c_k(t_{k-1}))^{-1} \), and \( \Xi_{\Phi_k} = \text{diag}(c_k(t_{k-1})) \). Note that we have chosen the linearization point \( c_k \) as \( c_k(t_{k-1}) \) in accordance with assumption [A4]. The approximation \( \ln(c_k) = \Xi_{\phi_k} c_k + \Xi_{\phi_k} \) is then valid as long as the difference between \( c_k(t_{k}) \) and \( c_k(t_{k-1}) \) is relatively small.

### 3.2.3.2 Solution Method

The set of nonlinear AEs (16) obtained after discretization can be solved using any root-finding algorithm, such as Newton's method. However, due to the relatively large number of state variables, the use of such algorithms can be computationally slow. A large part of the computational effort is in the computation of the inverse of the Jacobian of the AEs (16). Therefore, in [12] a method was proposed in which the Newton's method was applied sequentially to each set of equations of (16), thereby reducing the computation of the inverse of a single large Jacobian to the computation of the inverses of four smaller Jacobians. However, in doing so, some information of the large Jacobian is lost, which means that a quadratic convergence rate can no longer be achieved. Still, the computation time of the large Jacobian is sufficiently large, that the method proposed in [16] is faster than solving (16) directly using Newton's method.

Rather than sequentially solving (16), we propose a solution method, which retains the full information of the Jacobian of (16), but reduces the number of AEs. This can be done by substitution of equations, such that one state variable remains. In doing so, the full information of the Jacobian of (16) is contained in a smaller Jacobian related to equation of the remaining variable. The derivation involves solving (14) for the chosen state variable, and solving their associated equation given in (16) for \( j_n \).

State variable \( \psi_i \) is both linear in \( j_n \) and \( j_n \). Hence, the set of AEs (16) will be reduced to one set of AEs such that \( \phi_{c_k, \Phi_k} \) can be implicitly solved using the Newton's method, from which \( c_k, c_e, \Phi_k \) can be obtained.

The derivation of the reduced set of AEs is as follows. First, we solve (14) for \( \phi_{s, \Phi_s} \), which gives

\[
\phi_s(t_k) - \Phi_s(t_k) - F_{j_n}(\bar{c}_{s_k}(t_k), c_k(t_{k-1}))j_n(t_k) - U(\bar{c}_{s_k}(t_k)) = 0
\]

where

\[
F_{j_n}(\bar{c}_{s_k}, c_k) = \text{diag} \left( \frac{2a}{\delta t} a(c_k, \bar{c}_{s_k}) \right)
\]

and

\[
\bar{A}_{s_k}(t_{k}) - \bar{B}_{s_k} \phi_{t_{k}}(t_{k}) = 0
\]

These allow the state variables \( \bar{c}_{s_k}, c_k \) to be expressed as a function of \( \phi_s \) by first substituting \( j_n \) in (18) into their respective associated equations given in (16), i.e.,

\[
\begin{align*}
\bar{A}_{s_k}(t_{k}) &= \bar{B}_{s_k} \phi_{t_{k}}(t_{k}) + c_k(t_{k-1}) \\
\end{align*}
\]

for \( i \in \{s, e\} \), then solving this for their respective state variables and pre-multiplying by \( \bar{A}_{c_k}, \bar{A}_{c_k} \), resulting in

\[
\bar{c}_{i_k}(t_k) = \Gamma_{i_k} \phi_{t_{k}}(t_k) + \Phi_{i_k} \phi_{s_k}(t_k) + \Theta_{i_k}
\]

where

\[
\begin{align*}
\Gamma_{i_k} &= \bar{A}_{i_k} \bar{A}_{t_{k}}^{-1} \bar{B}_{s_k} \phi_{t_{k}}(t_{k}) \\
\Phi_{i_k} &= \bar{A}_{i_k} \bar{A}_{t_{k}}^{-1} \bar{B}_{s_k} \phi_{s_k}(t_{k}) \\
\Theta_{i_k} &= \bar{A}_{i_k} \bar{A}_{t_{k}}^{-1} c_k(t_{k-1})
\end{align*}
\]

for \( i \in \{s, e\} \). Note the presence of the full state vectors \( c_k, c_e \), which can be obtained by (19a), except without the pre-multiplication by \( \bar{A}_{c_k} \) and \( \bar{A}_{e_k} \), respectively. Similarly, \( \Phi_k \) can be expressed as a function of \( \phi_s \), and \( \phi_e \) by substituting (18) into (16d), i.e.,
Assumptions [A1]-[A4] have an impact on the accuracy of the model. The next section will quantify the impact of Assumptions [A1]-[A4] on model accuracy.

In order to validate Assumptions [A1]-[A4] described in Section 3.2.2, the full DFN model is compared to the simplified DFN (SDFN) model described in Section 3.2.2.2. Specifically, we will first validate the SDFN model by studying the computational performance and accuracy of the SDFN model described in Section 3.2.4.1 Validation of Assumptions [A1]-[A4]. The next section will quantify the impact of Assumptions [A1]-[A4] made in Section 3.2.2.2 on the accuracy compared to the full DFN model. Then, the computational complexity of the implementation of the SDFN model is studied. Finally, the SDFN model is compared to the SPMe given in Section 3.2.4.
In Table 4, the root-mean-square error (RMSE) of the output voltage \( V \) and normalized electrolyte concentration \( \frac{c_e}{c_{e,0}} \) between the full DFN model and the DFN model with varying simplifications is shown for the two different parameters sets. More specifically, each RMSE is computed over three separate simulations, where the initial SoC is selected as 20\%, 50\%, and 80\%. The RMS errors of the normalized electrolyte concentration shows how the assumptions affect the internal states. In both parameter sets, aside from the output voltage RMSE of the HE case, by far the largest RMS error is made with [A3]. It should be noted, however, that even this error is still small relative to the operating values of the output voltage, which is usually between 2.7 V – 4 V, and normalized electrolyte concentration. Note that the normalized electrolyte concentration is a ratio, and therefore \( 0.3 \times 10^{-3} \) is indeed a very small error. Furthermore, it can be noticed that the RMS errors shown for the HE parameter set are generally larger than the HP parameter set. This can be explained by the fact that the solid-phase diffusion is less limiting, while solid-phase conductivity is higher in the HP parameter set than in the HE parameter set. Therefore, over-potentials are generally higher in the HE case than in the HP case, which in turn leads to larger concentration gradients in the HE case. Higher over-potentials violate Assumption [A2], while larger deviations from the average electrolyte concentration violates Assumption [A3]. In Figure 39, the output voltage of the full and simplified DFN model, and its associated RMS error are shown. We observe that the full and simplified model have no visible difference in output voltage for both parameter sets, which further justifies Assumptions [A1]-[A4].

**Table 4 The effect of assumptions [A1]-[A4] introduced in Section 2 on RMS error (RMSE) for the high-power (HP) [19] and high-energy (HE) [21] parameter set.**

<table>
<thead>
<tr>
<th>RMSE</th>
<th>( V ) [mV]</th>
<th>( \frac{c_e}{c_{e,0}} ) [10^{-3}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A1] &amp; [A2]</td>
<td>0.00045 0.017</td>
<td>0.0013 0.0069</td>
</tr>
<tr>
<td>[A3]</td>
<td>0.0047 0.021</td>
<td>0.023 0.30</td>
</tr>
<tr>
<td>[A4]</td>
<td>0.00064 0.00040</td>
<td>0.0019 0.00016</td>
</tr>
<tr>
<td>SDFN</td>
<td>0.0047 0.021</td>
<td>0.024 0.30</td>
</tr>
</tbody>
</table>

**Figure 39 Comparison of output voltage and root-mean-square error (RMSE) between the full DFN model and the simplified DFN model, with two different parameter sets which represent a high power (HP) battery [19] and a high energy (HE) [21] battery, where the initial state-of-charge is 20 \%.**
### 3.2.4.2 Computational Complexity

In the discretization approach described in Section 3, there is freedom in selecting the number of volume elements \( n_n, n_s, n_p \) of the finite-volume-method discretization and the number of elements \( n_{r,n}, n_{r,p} \) of the finite-difference-method discretization. Choosing these parameters large enough leads to a better approximation of the diffusion dynamics in the battery. However, a finer discretization also leads to increased computation time. To analyse this trade-off, a study has been done where the SDFN is simulated with various sets of grid parameters \( n_n, n_s, n_p, n_{r,n}, n_{r,p} \). As a first case, a certain set of base grid parameters \( n_n, n_s, n_p, n_{r,n}, n_{r,p} = 50 \) is chosen. Then, each grid parameter will be varied between 1 and 50, for which the RMS error of the output voltage (with respect to the simulation with the base grid parameters) and simulation time is computed. The presented computation times are an average computed over 5 simulations.

![Figure 40 RMS error of the output voltage and computation time for varying grid parameters for the high-power (HP) [19] and the high-energy (HE) [21] parameter set.](image)

In Figure 40, the results of this first case study are shown. We see that, generally, in both parameter sets, as the value of the grid parameter increases, the RMS error decreases, while computation time increases. However, some grid parameters seem to be have a larger impact on the RMS error and computation time than others. With the HP parameter set, the grid parameters that relate to the solid-phase diffusion \( n_{r,n} \) and \( n_{r,p} \) have the largest impact on RMS error, while \( n_n, n_s, n_p \) hardly change the RMS value with increasing values. This indicates that diffusion dynamics in the solid phase largely dominates diffusion dynamics in the electrolyte phase, as, apparently, diffusion in the electrolyte phase does not need a high discretization. Thus, for the HP parameter set, to retain model accuracy and limit computation time, \( n_{r,n} \) and \( n_{r,p} \) need to be relatively large, while \( n_n, n_s, n_p \) can be small. For the HE parameter set, the grid parameters related to the positive electrode have the most impact on the RMS error. Here, to retain model accuracy and computation time, \( n_p \) and \( n_{r,p} \) need to be chosen relatively large. Furthermore, for both parameter sets, it can be seen that reducing \( n_n \) and \( n_p \) is the most favourable in reducing simulation time.

### Table 5 Selected grid parameters of the varying degrees of model orders shown in Fig. 40.

<table>
<thead>
<tr>
<th>Model order</th>
<th>( n_n )</th>
<th>( n_s )</th>
<th>( n_p )</th>
<th>( n_{r,n} )</th>
<th>( n_{r,p} )</th>
<th>Comp. time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP [19]</td>
<td>low</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>8.28</td>
</tr>
<tr>
<td>HE [21]</td>
<td>low</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>3</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>8.35</td>
</tr>
</tbody>
</table>
In Figure 41, the output voltage and normalized electrolyte concentration are shown, where the SDFN model is simulated with varying degrees of model order for the two parameter sets. The selected model orders are based on the analysis above, to show the trade-off that is made between model accuracy and computation time. The specific grid parameters of each model order, together with their respective computation time, are shown in Table 5. The low model orders have been selected such that there is a good representation of the output voltage, with the lowest model order possible, according to the analysis made above. The medium model orders have been selected such that both the output voltage and normalized electrolyte concentration are represented well, with the lowest model order possible. We observe that output voltage can be modelled well with a relatively coarse discretization over the cell length, especially in the HP case. However, in order to model the internal states well, both in the HP and HE case, \( n_m, n_s, n_p \) are important, as they relate to diffusion across the cell.

**Figure 41 Output voltage and normalized electrolyte concentration for varying degrees of model order for the high-power (HP) [19] and the high-energy (HE) [21] parameter set.**

### 3.2.5 Model Comparison

In Figure 42, discharge curves and normalized electrolyte concentration are shown for both parameter sets at various C-rates for three different models, i.e., DFN model, SDFN model, and SPMe. We can see that for both parameter sets, in all cases, there is no visible difference between the DFN and SDFN model, which further shows the validity of [A1]-[A4]. We can further see that for the HP parameter set, there are no visible differences between the three models up to 10 C. At 50 C, the SPMe shows some inaccuracy at the output voltage compared to the DFN model, even though the normalized electrolyte concentration is still close to the DFN model. Interestingly, the opposite is observed with the HE parameter set, where the electrolyte concentration with the SPMe corresponds to the DFN model rather well up to 0.5 C, while the difference in output voltage is substantially worse. Therefore, we observe that an accurate modelling of the output voltage does not necessarily mean an accurate representation of the internal states, and vice versa.
Figure 42 Discharge curve and normalized electrolyte concentration for the DFN model (solid lines), SDFN model (dashed lines), and SPMe (dotted lines) at several C-rates, simulated for the high-power (HP) [19] and the high-energy (HE) [21] parameter set.

In Figure 43, the output voltage is shown for the HP and HE battery simulated with a measured current profile of an electric-bike drive cycle, taken from [22]. This current profile is chosen as it represents a typical use of a battery, with a relatively fast-changing current, to show how the different battery models perform in such conditions. We see that consistent with the above analysis, for the HP case, all three models correspond well, while for the HE case the DFN and SDFN are visually identical, and the output voltage of the SPMe is significantly different.

Figure 43 Output voltage using various models for the high-power (HP) [19] and the high-energy (HE) [21] parameter set. The applied current profile is an electric bike drive cycle from [22].

In Table 6, the RMS errors and computation times corresponding to the simulations shown in Figure 6 are shown. Here, the computation times are an average over 5 simulations. Furthermore, the computation time is measured from the point where the main simulation loop starts to the point where the loop ends. The computation time of pre-defined matrices, outside this loop, is not included, since it is independent of simulation time. We observe that the SPMe is the fastest model, as would be expected due to the lower model complexity, although with the HE parameter set, the RMSE is unacceptable. The SDFN model, on the other hand, has very small RMS errors, while the computation time is at least an order of magnitude smaller than the computation time of the DFN model. These
results show that by making the Assumptions [A1]-[A4], simplifications can be made that have no significant impact on the model accuracy, while the computation time can be drastically decreased, to achieve a simulation time of over 3600 times faster than real-time.

Table 6 The RMS errors (RMSE) of the output voltage $V$ and computation times corresponding to the simulations shown in Figure 41.

<table>
<thead>
<tr>
<th></th>
<th>RMSE $V$ [mV]</th>
<th>Comp. time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP [19]</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>HE [21]</td>
<td>-</td>
<td>22.6</td>
</tr>
<tr>
<td>SDFN</td>
<td>0.021</td>
<td>0.71</td>
</tr>
<tr>
<td>SPMe</td>
<td>0.19</td>
<td>0.30</td>
</tr>
</tbody>
</table>

### 3.2.6 CONCLUSIONS

In this chapter, we have proposed several simplifications to the so-called Doyle-Fuller-Newman (DFN) model, which is a popular electrochemistry-based battery model. This simplified DFN (SDFN) model has allowed for a computationally very efficient implementation. In this approach, the SDFN model is temporally and spatially discretized, after which a solution method is used to solve the resulting set of algebraic equations (AEs). Due to the assumptions made, in this solution method, the resulting set of AEs can be reduced to a smaller set of AEs, which can be efficiently solved using Newton's method. The set of assumptions made have been justified by showing that the impact of the assumptions on the accuracy of the DFN model is negligible. This has been done for two different parameter sets, where one represents a high-power (HP) battery and the other a high-energy (HE) battery. Furthermore, we have made a computational analysis of the SDFN model to show how a trade-off can be made between model accuracy and computation time, through the coarseness of the discretization. Finally, the SDFN model has been compared to the DFN model as well as an implemented SPM, for the two parameter sets. We have shown that while the SPMe is accurate for the HP parameter set, its accuracy is unacceptable for the HE parameter set. The SDFN model, on the other hand, has very small RMS errors, while the computation time is at least an order of magnitude smaller than the computation time of the DFN model. This has shown that by making Assumptions [A1]-[A4], simplifications can be made that have no significant impact on the model accuracy, while the computation time can be drastically decreased, to achieve a simulation time of over 3600 times faster than real-time.
3.2.7 References


4 REDUCED ORDER ELECTRO-THERMAL MODEL READY FOR SIL AND HIL BMS EVALUATION

4.1 INTRODUCTION

Battery management system (BMS) is a critical component for electric vehicles. During the development of BMS, several types of test can be done in order to evaluate its performance: (1) Software-in-the-Loop (SiL) test to evaluate the BMS control algorithm; (2) Hardware-in-the-Loop (HiL) test to evaluate the real time performance of the BMS; (3) test with real battery pack to finally validate the BMS performance. For the SiL and HiL tests, a battery pack model is needed to represent the electro-thermal behaviour of the cells in the pack. The battery pack model is consisted of individual cell model which can be either characterized with experimental measurements or with simulated data from a complex model (e.g., electrochemical model). The battery pack model must have a good compromise of accuracy/simplicity to be real-time compatible for the SiL and HiL tests. In our study, a complex electrochemical model has been characterized from experimental measurements of a battery cell. The problem to solve is to get a reduced order electro-thermal cell model from the electrochemical model and build a battery pack model which is real-time compatible and ready for the SiL and HiL tests of BMS.

The objective of this chapter is to present a simple method to get a reduced order electro-thermal cell model from a complex electrochemical cell model and then build a battery pack model with the reduced order cell model. The remaining part of this chapter is organized as follows. Section 4.2 presents the electrochemical model. Section 4.3 describes the method to get the reduced order electro-thermal model. It also presents the validation of the method by comparing the simulation results between the two models. Section 4.4 concludes the chapter.

4.2 ELECTROCHEMICAL MODEL

In our study, the well-known pseudo-two dimensional (p2D) electrochemical-thermal model [1] has been used which is parameterized and validated at various temperatures and current loads for an 18650 nickel-rich, silicon-graphite lithium-ion cell of 3.35 Ah as presented in the work of Sturm et al. [2]. The p2D model used here simulates potentials and concentrations in the active materials and the electrolyte based on porous electrode theory and concentrated solution theory combined via electrode kinetics throughout the thickness of a single electrochemical cell unit containing anode, separator and cathode [1]. Model reduction and implementation into MATLAB 2018a were applied to the p2D model used in this work according to previous works [3]. The Parabolic Profile approximation [4] for the discretization of the particle domain was used together with finite difference method (FDM) for the remaining spatial discretization of the differential algebraic equation system. Crank Nicolson [5] formulation was used for the time discretization and an iterative Newton-Raphson scheme [6] for the overall solving process. No side reactions or multiple particle sizes were included in this p2D model. While the complete electrochemical model presents many advantages such as giving detailed insight into the electrochemical process inside the battery cell, it is not suitable for real-time simulation due to its high computational cost. To simulate the electro-thermal behaviour of a battery pack including many cells connected in series and parallel, a simplified cell model must be used and at the same time it must be able to represent accurately the electro-thermal behaviour of the cell.
4.3 REDUCED-ORDER ELECTRO-THERMAL MODEL

4.3.1 PROPOSED MODEL

Several types of simplified models, which are real-time compatible, are available to represent the electro-thermal behaviour of the battery (e.g., equivalent circuit model [7][8][9][10], black box model [11]). In our study, an equivalent circuit model is chosen because it is simple, with a good compromise of accuracy/simplicity and has already been implemented in Simcenter Amesim, which is a multi-physical simulation software of Siemens.

Figure 44 shows the reduced-order electro-thermal model used in our study. The electrical behaviour of the cell is represented with an equivalent circuit model consisted of different elements: (1) a voltage source to represent the open circuit voltage (OCV); (2) a resistance $R_{ohm}$ to represent the ohmic resistance responsible for the instantaneous voltage drop when the cell current changes; and (3) several RC circuits $R_{diff[i]}/C_{diff[i]}$ to represent the dynamic behaviour of the cell mainly related to the diffusion of Lithium-ion inside the battery cell. The OCV is a function of the SoC and the temperature. Other elements in Figure 44 are functions of the SoC, the temperature and the current.

![Figure 44 Reduced order circuit model](image)

The SoC is determined by the coulomb counting as follow:

$$SoC = SoC_{init} - \frac{1}{Q_{cell}} \int \eta_{cell} I \, dt$$  \hspace{1cm} (1)

Where $SoC_{init}$ is the initial SoC; $I$ is the current; $Q_{cell}$ is the cell capacity; $\eta_{cell}$ is the faradic efficiency which is set to 1 in our study. The voltage of the cell is calculated by equation as follow:

$$U_{cell} = OCV - \Delta U_{ohm} - \sum_{i=1}^{NR} \Delta U_{diff,i}$$  \hspace{1cm} (2)

Where:
- The OCV is calculated with equation as follow to consider the hysteresis behaviour of the OCV in battery:

$$OCV = OCV_d + F_{hys} \cdot (OCV_c - OCV_d) + \frac{du}{dt} \cdot (T_{cell} - T_{ref})$$  \hspace{1cm} (3)

where $OCV_c$ and $OCV_d$ are open circuit voltages measured in charge and in discharge at the reference temperature $T_{ref}$; $\frac{du}{dt}$ is the entropic coefficient; $F_{hys}$ is the hysteresis factor with value varying between 0 and 1 and is calculated by equations:

\[
\begin{align*}
\frac{dF_{hys}}{dt} &= \frac{3 \Delta SoC}{\Delta SoC_{hys}} \cdot (1 - F_{hys}), \hspace{1cm} \text{during charge} \\
\frac{dF_{hys}}{dt} &= \frac{3 \Delta SoC}{\Delta SoC_{hys}} \cdot F_{hys}, \hspace{1cm} \text{during discharge}
\end{align*}
\]

With $\Delta SoC_{hys}$ the state of charge variation necessary for full charge/discharge open circuit voltage transition. An example of the open circuit voltage transition is shown in Figure 45. The initial
open circuit voltage starts from point (1); during a charge, the open circuit voltage joins progressively its high boundary defined by $OCV_c$; during a discharge, the open circuit voltage joins progressively the low boundary defined by $OCV_d$.

- The ohmic voltage drop $\Delta U_{ohm}$ is calculated with equation:

$$\Delta U_{ohm} = I \cdot R_{ohm}$$ (5)

- The voltage drop $\Delta U_{diff,i}$ for each of the RC circuits ($i = 1, 2, ..., N_{RC}$) is calculated with equation:

$$\frac{d(\Delta U_{diff,i})}{dt} = -\frac{\Delta U_{diff,i}}{R_{diff,i}C_{diff,i}} + \frac{I}{C_{diff,i}}$$ (6)

![Figure 45 OCV transition between its high and low boundaries](image)

The heat generated by the cell ($H_{gen}$) during its operation includes the heat related to the entropic coefficient ($\phi_{dv}$), the hysteresis loss ($\phi_{hys}$), the ohmic loss ($\phi_{ohm}$) and the diffusion loss ($\phi_{diff}$). The heat generated is calculated with equations as follow:

$$H_{gen} = \phi_{dv} + \phi_{hys} + \phi_{ohm} + \phi_{diff}$$

$$\phi_{dv} = I \cdot \frac{dU_{dv}}{dt} \cdot (T_{cell} + 273.15)$$

$$\phi_{hys} = -I \cdot \left(\frac{OCV_c + OCV_d}{2} - OCV\right)$$

$$\phi_{ohm} = -I^2 \cdot R_{ohm}$$

$$\phi_{diff} = -I \cdot \sum_{i=1}^{N_{RC}} \Delta U_{diff,i}$$ (7)

The thermal behavior of the battery cell is represented with a simple thermal model consisted of different elements as shown in Figure 46: (1) a heat source to represent the heat generated by the cell; (2) a thermal capacity of the cell; (3) a thermal convection resistance; and (4) a temperature source to represent the ambient temperature $T_{amb}$. The cell temperature $T_{cell}$ is calculated with equation:

$$\frac{dT_{cell}}{dt} = -\frac{A \cdot h_{cov}}{m_{cell} \cdot C_p} \cdot (T_{cell} - T_{amb}) + \frac{H_{gen}}{m_{cell} \cdot C_p}$$ (8)

Where $m_{cell}$ is the cell mass and $C_p$ is the cell specific heat.

![Figure 46 Thermal model of the cell](image)

Table 7 summarizes the parameters needed for the model. The battery model in Simcenter Amesim also includes other features such as aging and thermal runaway modelling, which are not used in our
study. Detailed information of the model can be found in the help documentation of Simcenter Amesim [12].

**Table 7 Parameters for the reduced order circuit model**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_{\text{cell}}$</td>
<td>Cell capacity</td>
<td>Ah</td>
</tr>
<tr>
<td>$OCV_c$</td>
<td>OCV in charge at the reference temperature</td>
<td>V</td>
</tr>
<tr>
<td>$OCV_d$</td>
<td>OCV in discharge at the reference temperature</td>
<td>V</td>
</tr>
<tr>
<td>$\Delta SoC_{\text{hys}}$</td>
<td>State of charge variation for full charge and discharge open circuit voltage transition</td>
<td>%</td>
</tr>
<tr>
<td>$dU/dT$</td>
<td>Entropic coefficient</td>
<td>V/K</td>
</tr>
<tr>
<td>$R_{\Omega \text{hm}}$</td>
<td>Ohmic resistance</td>
<td>Ohm</td>
</tr>
<tr>
<td>$R_{\text{diff}[i]}$</td>
<td>Diffusion resistance</td>
<td>Ohm</td>
</tr>
<tr>
<td>$C_{\text{diff}[i]}$</td>
<td>Diffusion capacitance</td>
<td>F</td>
</tr>
<tr>
<td>$C_p$</td>
<td>Specific heat of the cell</td>
<td>J/kg/\text{K}</td>
</tr>
<tr>
<td>$h_{\text{cov}}$</td>
<td>Convective heat exchange coefficient</td>
<td>W/m$^2$/\text{K}</td>
</tr>
<tr>
<td>$S_{\text{conv}}$</td>
<td>Convective heat exchange area</td>
<td>m$^2$</td>
</tr>
<tr>
<td>$m_{\text{cell}}$</td>
<td>mass of the cell</td>
<td>kg</td>
</tr>
</tbody>
</table>
4.3.2 Model Calibration

The reduced order circuit model is calibrated with simulated data from the electrochemical model. 2 types of test profile are used to generate the simulated data:

- **Test 1: Pulses test.** As shown on Figure 47, this profile discharges the cell from 100% to 0% SoC. It includes several groups of short-duration (< 2 s) charge and discharge pulses at different current levels. Between two groups of pulses, a long-duration discharge with a constant current is used to decrease 5% SoC of the cell. 2 level of currents are used alternatively for the long-duration discharge (1C and 0.5C). Each long-duration discharge is followed by a long rest period (1800 s) to stabilize the cell voltage.

- **Test 2: Charge test.** As shown on Figure 48, this profile is consisted of long-duration charges with 2 levels of current alternatively (1C and 0.5C) to charge the cell from 0% to 95% SoC. Each long-duration charge increases 5% SoC of the cell and is followed by a long rest period (1800 s).

These test profiles are designed on the one hand to facilitate the parameter identification with the Battery Identification Assistant tool in Simcenter Amesim[12]. On the other hand, the duration for the pulses and the current levels for the long-duration discharges or charges are chosen to minimize the temperature variation of the cell during the test. These profiles can be easily adapted to test other battery cells.

In our study, the test profiles were applied to the electrochemical model at 3 ambient temperatures (5, 25 and 45 °C), for which the electrochemical model has been characterized. Figure 47 and Figure 48 shows the examples of the simulated data from the electrochemical model at 25 °C for the 2 test profiles respectively.

![Figure 47 Test 1: pulses test](image-url)
The parameters in Table 7 are identified with the Battery Identification Assistant tool in Simcenter Amesim. The detail is explained as follow.

### 4.3.2.1 OCV<sub>d</sub> and OCV<sub>c</sub>

The OCV<sub>d</sub> and OCV<sub>c</sub> are identified with the relaxation phase after each long-duration discharge or charge. The value at the end of each relaxation after discharge (Figure 48) is considered as the OCV value in discharge. Figure 49 shows an example of the OCV<sub>d</sub> identified at the 3 temperatures. The same process allows to get the OCV<sub>c</sub> at the 3 temperatures by using the charge profile in Figure 48. 25 °C is chosen as the reference temperature in our study to calculate the open circuit voltage with equation (3).

![OCV identified at different temperatures](image)

**Figure 49 OCVd identified at different temperatures**

### 4.3.2.2 ΔSoC<sub>hys</sub>

Figure 50 shows the hysteresis between the OCV<sub>d</sub> and OCV<sub>c</sub> at 25 °C. The experimental test results of a Li-ion cell in [13] shows that the state of charge variation (ΔSoC<sub>hys</sub>) necessary for full charge/discharge OCV transition is from 15% to 25% in most of the case. In the absence of tests to identify ΔSoC<sub>hys</sub>, its value is set arbitrarily to 15% in our study.
4.3.2.3 \( \frac{dU}{dT} \)

The entropic coefficient curve can be calculated by using the OCV curves identified previously as follow:

\[
\frac{dU}{dT}\bigg|_x^{T_1,T_2} = \frac{OCV_x(T_2) - OCV_x(T_1)}{T_1 - T_2}
\]

(9)

Where \( T_1 \) and \( T_2 \) are 2 different temperatures; \( OCV_x \) can be \( OCV_d \) or \( OCV_c \). In our case, the OCV is identified at 3 temperatures in discharge and in charge. There are therefore 6 possible combinations to calculate 6 curves of entropic coefficient as illustrated in Table 8. The average curve of the 6 entropic coefficient curves is the one to be used in the reduced order circuit model. The average curve is shown in Figure 51.

**Table 8 six combinations to calculate entropic coefficient**

<table>
<thead>
<tr>
<th>( OCV_x )</th>
<th>( OCV_d )</th>
<th>( OCV_d )</th>
<th>( OCV_d )</th>
<th>( OCV_c )</th>
<th>( OCV_c )</th>
<th>( OCV_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 ) (°C)</td>
<td>5</td>
<td>25</td>
<td>45</td>
<td>5</td>
<td>25</td>
<td>45</td>
</tr>
<tr>
<td>( T_2 ) (°C)</td>
<td>25</td>
<td>45</td>
<td>5</td>
<td>25</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>( \frac{dU}{dT} \bigg</td>
<td>_x^{T_1,T_2} )</td>
<td>( \frac{dU}{dT} \bigg</td>
<td>_{5,25}^{T_1,T_2} )</td>
<td>( \frac{dU}{dT} \bigg</td>
<td>_{25,45}^{T_1,T_2} )</td>
<td>( \frac{dU}{dT} \bigg</td>
</tr>
</tbody>
</table>

4.3.2.4 \( R_{Ohm,d} \) and \( R_{Ohm,c} \)

The instantaneous ohmic resistances are identified with the groups of short-duration pulses (Figure 47). These pulses allow to get the ohmic resistance value at different SoCs, currents and temperatures. Figure 52 shows the result of the ohmic resistance identified at 25 °C.
4.3.2.5 RC circuits

By using the long-duration discharges and charges in Figure 47 and Figure 48, the RC circuits \( R_{diff[i]}/C_{diff[i]} \) are identified with the Battery Identification Assistant tool of Simcenter Amesim. Instead of identifying directly several RC circuits, the tool identifies a Warburg impedance represented by 2 parameters (a diffusion resistance \( R_{ss} \) and a time coefficient \( T_c \)). The identification process of the Warburg impedance is similar to the ones in [14][15]. Once the Warburg impedance is identified, it can be easily approximated by different number of RC circuits in the Simcenter Amesim battery model with the help of the RC transformation tool[12]. Figure 53 shows the \( R_{ss} \) and \( T_c \) identified at 25 °C.

4.3.2.6 Thermal parameters

The parameter values of the thermal model are the same as the ones used in the electrochemical model as shown in Table 9.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_p )</td>
<td>791.86</td>
<td>J/kg/K</td>
</tr>
<tr>
<td>( h_{cov} )</td>
<td>27.5087</td>
<td>W/m(^2)/K</td>
</tr>
<tr>
<td>( S_{conv} )</td>
<td>0.00421525</td>
<td>m(^2)</td>
</tr>
<tr>
<td>( m_{cell} )</td>
<td>0.04622</td>
<td>kg</td>
</tr>
</tbody>
</table>

4.3.3 Model Validation

Figure 54 shows the simulation sketch in Simcenter Amesim to evaluate the performance of the reduced-order circuit model. The sketch includes several elements such as (A) the battery model which represent the electrical model in Figure 44; (B) a convective heat exchange component and a thermal mass to represent the thermal model in Figure 46; (C) a resistance to represent the connector.
resistance in the electrochemical model; (D) a virtual test bench to deliver current to the battery model and measure the voltage; (E) 3 tables which import the current, voltage and temperature data generated by the electrochemical model. This sketch allows to compare the estimation of the two models on the battery voltage and temperature for the same current profile.

![Simulation sketch for model validation in Simcenter Amesim](image)

**Figure 54 Simulation sketch for model validation in Simcenter Amesim**

To evaluate the performance of the reduced order circuit model, a validation test profile in Figure 55 has been applied to the reduced order circuit model and the electrochemical model. The profile includes a constant current charge phase to represent fast charge situation; several long-duration discharges to set the cell at different SoCs; and several dynamic cycles at different SoCs to represent the situation during driving.

![Validation test profile](image)

**Figure 55 Validation test profile**

As explained in paragraph 4.3.2, the number of the RC circuit can be easily set in the reduced order model with the help of the RC transformation tool in Simcenter Amesim. Simulations with 5 different number of RC circuits ($N_{RC} = 1$ to 5) have therefore been done. Figure 56, Figure 57 and Figure 58 show the comparison of the two models on the estimation of battery voltage and temperature at respectively 5, 25 and 45 °C, with $N_{RC} = 1$ in the reduced order circuit model. These figures show that the reduced order circuit model (ROC model) can reproduce correctly the voltage and temperature behavior of the electrochemical model (EC model).
Figure 56 Validation test at 5 °C of ambient temperature ($N_{RC} = 1$)

Figure 57 Validation test at 25 °C of ambient temperature ($N_{RC} = 1$)
To evaluate qualitatively the performance of the reduced order circuit model, the RMS (root mean squared) errors for voltage and temperature estimation are calculated by equation as follow:

$$e_{RMS} = \sqrt{\frac{\sum_{k=1}^{N_s} (V_{ECM,k} - V_{ROM,k})^2}{N_s}}$$  \hspace{1cm} (10)

Where $V_{ECM,k}$ is the voltage or temperature of the electrochemical model; $V_{ROM,k}$ is the voltage or temperature of the reduced order circuit model; $N_s$ is the number of the voltage or temperature data sample. Table 10 and Table 11 present the RMS errors of the voltage and temperature estimation according to the number of RC circuits used. The results show that the reduced order circuit model with 1 RC circuit is sufficient to simulate correctly the electrochemical model. Adding additional RC circuits doesn’t improve significantly the estimation precision. The RMS errors of the reduced order circuit model are similar to the ones reported in other works in the literature[7][9].

**Table 10 RMS error for voltage estimation**

<table>
<thead>
<tr>
<th>RMS error of voltage</th>
<th>1 RC (mV)</th>
<th>2 RC (mV)</th>
<th>3 RC (mV)</th>
<th>4 RC (mV)</th>
<th>5 RC (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 °C</td>
<td>36</td>
<td>36</td>
<td>35</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>25 °C</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>45 °C</td>
<td>29</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>

**Table 11 RMS error for temperature estimation**

<table>
<thead>
<tr>
<th>RMS error of temperature</th>
<th>1 RC (°C)</th>
<th>2 RC (°C)</th>
<th>3 RC (°C)</th>
<th>4 RC (°C)</th>
<th>5 RC (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 °C</td>
<td>0.55</td>
<td>0.55</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
</tr>
<tr>
<td>25 °C</td>
<td>0.32</td>
<td>0.32</td>
<td>0.32</td>
<td>0.32</td>
<td>0.31</td>
</tr>
<tr>
<td>45 °C</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
</tr>
</tbody>
</table>
4.4 Conclusions

In this chapter, a simple method is proposed to get a reduced order electro-thermal cell model from a complex electrochemical cell model. The method requires only 2 tests per temperature to the electrochemical model to collect the simulated data of voltage and temperature. An equivalent circuit model is then identified from the simulated data. The method has been applied to get the reduced order model from a P2D electrochemical-thermal model calibrated for a nickel-rich, silicon-graphite lithium-ion cell of 3.35 Ah. The reduced order model can reproduce correctly the voltage and temperature behavior of the electrochemical model according to the validation tests. The method is generic so can be applied with minor modification to any complex electrochemical models calibrated for other battery cells.
4.5 References


[12] Help document of ESSBATCA01 – advanced equivalent circuit model of battery cell, Simcenter Amesim V17, Siemens, 2018

