A computationally efficient implementation of a full and reduced-order electrochemistry-based model for Li-Ion batteries

Citation for published version (APA):

Document license:
CC BY-NC-ND

DOI:
10.1016/j.apenergy.2017.09.025

Document status and date:
Published: 15/12/2017

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 05. Sep. 2020
A computationally efficient implementation of a full and reduced-order electrochemistry-based model for Li-ion batteries

L. Xia\textsuperscript{a}, E. Najafi\textsuperscript{a,⁎}, Z. Li\textsuperscript{b}, H.J. Bergveld\textsuperscript{a,c}, M.C.F. Donkers\textsuperscript{a}

\textsuperscript{a} Department of Electrical Engineering, Eindhoven University of Technology, Netherlands
\textsuperscript{b} Department Earth and Space Science and Engineering, York University, Toronto, Canada
\textsuperscript{c} NXP Semiconductors, Eindhoven, Netherlands

HIGHLIGHTS

- A computationally efficient implementation of the DFN model is proposed.
- Nonlinear model reduction is applied for the first time to the full nonlinear model.
- The implementation uses a particular numerical scheme based on Newton’s method.
- This implementation is a step towards using the DFN model in real-time applications.

ARTICLE INFO

Keywords:
Lithium-ion batteries
Electrochemistry-based model
Partial differential equations
Model order reduction
Proper orthogonal decomposition

ABSTRACT

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 paper 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.

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 application 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 battery [8–10]. These applications would benefit from having a computationally efficient electrochemistry-based battery model.

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 [11] and a simplified multi-particle model via linearisation has been proposed in [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. [13,14]. These solid-state diffusion dynamics...
can be simplified analytically, as was done in, e.g. [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 [15,17], balanced truncations [18], or residue grouping [16], see [19] for an overview. Others use a Galerkin projection based on several Legendre polynomials [20], or a proper orthogonal decomposition (POD) [21]. Still, these latter two methods have only been applied to a linearised DFN model in [20,21]. Even though linearised models are valid for low (dis)charge currents, they become less accurate at higher currents. To summarize, the aforementioned 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. [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 [23]. Numerical methods have been studied in [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 [25]. However, [25] uses a simplification that was proposed in [16], which makes the model less accurate in comparison to the full DFN model. Moreover, [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 paper, 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 [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 [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. [29]. While it has been shown in [27] that the discretised DFN model already allows simulating the DFN model faster than real-time,1 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 paper proposes to apply model order reduction, which was not done in our previous work [27]. In particular, we apply a combination of POD [30] and discrete empirical interpolation method (DEIM) [31] to the full nonlinear DFN model, instead of applying it to a linearised or a priori simplified battery model, as was done in many of the aforementioned papers. We employ the efficient implementation of [27] in the model order reduction and extend the simulation results of [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 a 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 paper is organized as follows. Section 2 outlines the DFN model. Then, a computational procedure is presented in Section 3 that implements the full DFN model. Section 4 introduces a model order reduction technique that simplifies the model and reduces the computational time and required memory. The simulation results of the full-order model implementation are given in Section 5, where computational performance analysis and model validation are discussed. Section 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 7.

2. Model of Lithium-ion batteries

The DFN model [1,2], considered in this paper, is a one-dimensional physics-based electrochemical model of a Li-ion battery. Fig. 1 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. 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.

\[ \frac{\delta c_r}{\delta t} = \frac{D_s}{r^2} \frac{\delta}{\delta r} \left( \delta \left( \frac{\delta c_r}{\delta r} \right) \right) \quad (1a) \]

with boundary conditions

---

1 Faster than real-time means that doing the simulation takes less time than the physical phenomena that are simulated take.
\[
\frac{\partial \phi}{\partial t} + \frac{\partial}{\partial x} \left( \sigma \frac{\partial \phi}{\partial x} \right) = j_{li},
\]
with boundary conditions
\[
-\sigma \frac{\partial \phi}{\partial x} \bigg|_{x=0} = \frac{i_{app}}{A_{surf}}, \quad \sigma \frac{\partial \phi}{\partial x} \bigg|_{x=L} = 0,
\]
for the negative electrode and
\[
-\sigma \frac{\partial \phi}{\partial x} \bigg|_{x=L-t_i} = 0, \quad -\sigma \frac{\partial \phi}{\partial x} \bigg|_{x=0} = \frac{i_{app}}{A_{surf}},
\]
for the positive electrode. In (2), \( \sigma \) denotes the effective conductivity in the solid-phase, and \( \sigma \) denotes the conductivity of solid active material, \( A_{surf} \) is the electrode plate area and \( i_{app} \) is the applied (dis) charge current, where \( i_{app} > 0 \) indicates discharging.

**Li-ion concentration in the electrolyte phase \( c_e \).** The concentration of Li-ion in the electrolyte phase \( c_e \) as a function of \( x \in [0,L] \) and \( t \in \mathbb{R}^+ \) is given by
\[
c_e \frac{\partial c_e}{\partial t} + \frac{\partial}{\partial x} \left( \kappa \frac{\partial c_e}{\partial x} \right) = -j_{li},
\]
with boundary conditions
\[
\kappa \frac{\partial c_e}{\partial x} \bigg|_{x=0} = \kappa \frac{\partial c_e}{\partial x} \bigg|_{x=L} = 0,
\]
where \( \kappa \) denotes the porosity (i.e., the electrolyte phase volume fraction), \( \kappa \) denotes the effective diffusion coefficient of Li-ion in the electrolyte phase, such that \( \kappa \) is the Li-ion diffusion coefficient in the electrolyte phase and \( \kappa \) is the Bruggeman porosity exponent, and \( \kappa \) is the transference number of Li-ion.

**Potential of the electrolyte phase \( \phi_e \).** The potential of the electrolyte phase \( \phi_e \) is given by
\[
\frac{\partial \phi_e}{\partial t} + \frac{\partial}{\partial x} \left( \alpha \frac{\partial \phi_e}{\partial x} \right) = -j_{li},
\]
with boundary conditions
\[
\alpha \frac{\partial \phi_e}{\partial x} \bigg|_{x=0} = \alpha \frac{\partial \phi_e}{\partial x} \bigg|_{x=L} = 0,
\]
where \( \alpha \) is the volumetric fraction of the active material and \( \alpha \) is the Bruggeman porosity exponent. Moreover, the electrolyte ionic diffusion coefficient \( \kappa_0^e \) is described as
\[
\kappa_0^e = \frac{2RT\sigma \kappa_{eff}^e}{F} \left( 1 - \frac{\partial \ln f_e}{\partial \ln c_e} \right),
\]
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 paper.

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
\[
j_{li} = a_n i_0 \exp \left( \frac{\alpha_n F}{RT} \right) \exp \left( -\frac{\alpha_n F}{RT} \right),
\]
where \( i_0 \) is the exchange current density given by
\[
i_0 = k_0 c_e^{\alpha_n}(c_{\text{max}}^{\alpha_n} - c_e^{\alpha_n})^{\alpha_n} c_e^{\alpha_n},
\]
in which \( k_0 \) is the kinetic rate constant, \( \alpha_n \) is the anodic transfer coefficient, \( \alpha_n \) is the cathodic transfer coefficient, \( c_{\text{max}} \) is the maximum concentration in the solid-phase, \( c_e(x,t) = c_e(R_c x,t) \) is the concentration at the surface of the spherical particle. Moreover, the variable \( \eta \) in (6) denotes the over-potential and is defined as
\[
\eta = \frac{\phi_e - \phi_a - U}{},
\]
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 \( c_{\text{surf}} \).

Finally, the (measurable) terminal battery voltage is given by
\[
V(t) = \phi_e(L,t) - \phi_a(0,t) - \frac{R}{A_{surf}} i_{app}(t),
\]
where \( R \) 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. Model implementation

In this section, an efficient implementation of the full model (1)-(9) is proposed. The model 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 paper uses a similar discretisation scheme as in [25], the numerical method for solving the algebraic equations is different.

#### 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 \( \eta = ((i-1)\delta_i,i \in [1,...,n_i]) \), with interval size \( \delta_i = R/(n_i-1) \), where \( n_i \) 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 \( \delta_j(i), i \in [1,...,n_x] \), with interval size \( \delta_j \) that can vary with \( i \), 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 \( \sum_{i=1}^{n_i} \delta_i = L, \sum_{i=1}^{n_i} \delta_i = \delta_j, \sum_{k=1}^{n_x} \delta_j(i) = \delta_x \), \( n_x \) denotes the number of intervals (control volumes) along the x-dimension in the negative electrode, separator, and positive electrode, respectively and satisfy \( n_x = n_s + n_s + n_p \). The spatial discretisation of (1) using FDM is given by
\[
\frac{d}{dt} c_{(i,j)}(t) = \begin{cases} 
\frac{2Îm \delta_j^{(2)}-Îm^{(3)}(i,j)}{\delta_j} & \text{for } i = 1, \\
\frac{2Îm \delta_j^{(n_x-1)}-Îm^{(n_x)}(i,j)}{n_x \delta_j} + \frac{\delta_j}{n_x \delta_j} & \text{for } i \notin [2,...,n_x-1],
\end{cases}
\]
where the time variable \( t \) is omitted from this expression for compactness.
of the notation. Moreover, \( c_{ij}^{(l)}(t) = c_{ij}(x_i, x_j, t) \), with \( x_i \in \{ \sum_{j=1}^{k-1} g_j^{(l)} \} \).

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

\[
\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x_a} \left( \alpha_c \frac{\partial u}{\partial x_a} + \alpha_f \ln c_i \right) + \alpha_j \phi(t),
\]

with boundary conditions

\[
\frac{\partial \phi}{\partial x_a} \bigg|_{x=a} = \alpha_4, \quad \frac{\partial \phi}{\partial x_a} \bigg|_{x=b} = \alpha_5.
\]

Namely, (2) is retrieved by choosing \( \omega = \phi, \alpha_4 = 0, \alpha_5 = \alpha = \alpha = \alpha = 0 \); (2b) by \( \omega = \alpha_c, \alpha_4 = 0, \alpha_5 = \alpha \); (2c) by \( \omega = \alpha_f, \alpha_4 = \alpha, \alpha_5 = 0 \). Similarly, (3) is formalised by choosing \( \omega = \phi, \alpha_4 = \alpha, \alpha_5 = \alpha \); (4) by \( \omega = \phi, \alpha_4 = \alpha, \alpha_5 = \alpha \).

The spatial discretisation of (11) using the FVM yields

\[
\alpha_5^2 \frac{\partial^2 u}{\partial x_a^2} \bigg|_{x=a} = \alpha_4, \quad \alpha_5^2 \frac{\partial^2 u}{\partial x_a^2} \bigg|_{x=b} = \alpha_5.
\]

Finally, (4) corresponds to \( \omega = \phi, \alpha_4 = \alpha, \alpha_5 = \alpha \).

The strategy for solving the nonlinear algebraic Eq. (14) at sample times \( t_j \), that we use in this paper, 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 Eqs. (14) can be solved by solving

\[
x^{(k+1)}(t_j) = x^{(k)}(t_j) - J(x^{(k)}(t_j))^{-1} F(x^{(k)}(t_j)),
\]

where \( k \in [0,1,\ldots, K] \) 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 \( J(x^{(k)}(t_j)) \) and \( F(x^{(k)}(t_j)) \) 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 Eq. (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

\[
x^{(k+1)}(t_j) = x^{(k)}(t_j) - J(x^{(k)}(t_j))^{-1} F(x^{(k)}(t_j)), \quad \forall i \in [1,4], \quad \forall \gamma \in (0,1),
\]

where \( I = [\mathbf{1}]_4 \) denotes the identity matrix and \( D^{(i)} \) denotes the diagonal matrix with entries \( D_{ii} = 1 \) and \( D_{ij} = 0, \forall i \neq j \).
given tolerance $\varepsilon$ when
\[ \|x^{(i+1)}(t) - x^{(i)}(t)\| \leq \varepsilon \quad \text{for} \quad i \in \{1, \ldots, 4\}, \]
(17)
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 $i \in \{1, \ldots, 4\}$ in which (16) is solved can be changed. This might influence the computational efficiency, which will be discussed in detail later.

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 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.

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 basis functions that contain the most relevant features from the system of interest, in the sense that $x_i(t) \equiv \hat{U}_i \hat{x}(t)$ for every $x_i(t), i \in \{1, \ldots, 4\}$, as in (13b), by a suitable choice of the matrix $\hat{U}_i \in \mathbb{R}^{n \times k_i}$, where $k_i \ll n$. The orthonormal matrix $\hat{U}_i$ is obtained by solving
\[
\min_{\hat{U}_i \in \mathbb{R}^{n \times k_i}} \sum_{j=1}^{m} \|x_j(t) - \hat{U}_i \hat{x}(t)\|_2^2, \tag{18}
\]
subject to $\hat{U}_i^T \hat{U}_i = I_3$, using a snapshot of state trajectories $x_i(t)$ at sample times $t_j = \beta_j$, with $i \in \{1, \ldots, 4\}$ and $j \in \{1, \ldots, m\}$.

The POD basis $\hat{U}_i$ in (18) can be constructed from the left singular vector of the snapshot matrix $x_i = [x_i(t_1), \ldots, x_i(t_m)] \in \mathbb{R}^{n \times m}$ obtained by a singular value decomposition (SVD). Here, the SVD of $x_i$ is given by
\[
x_i = U_i \Sigma_i V_i^T, \tag{19}
\]
where $U_i \in \mathbb{R}^{n \times k_i}$ and $V_i \in \mathbb{R}^{m \times m}$ are the left and right singular vectors, respectively. Moreover
\[
\Sigma_i = \begin{bmatrix}
\hat{\Sigma}_l & 0_{k_l \times (m-k_l)} \\
0_{m-k_l \times k_l} & 0_{m-k_l \times (m-k_l)}
\end{bmatrix} \in \mathbb{R}^{m \times m}, \tag{20}
\]
where $\hat{\Sigma}_l = \text{diag}(\sigma_l(1), \ldots, \sigma_l(k_l)) \in \mathbb{R}^{k_l \times k_l}$ is a diagonal matrix containing the singular values, where $k_l = \text{rank}(x_i)$. The optimal solution of (18), denoted by $\hat{U}_i \in \mathbb{R}^{n \times k_i}$, is given by the matrix formed by the first $k_i$ left columns of $U_i$. The choice of the matrix $\hat{U}_i$ provides a bound on the approximation error, because
\[
\sum_{j=1}^{m} \|x_j(t) - \hat{U}_i \hat{x}(t)\|_2^2 = \sum_{k_{i+1}}^{k} \|c_{k_i+1}\|^2. \tag{21}
\]
This means that the accuracy of the approximation is directly determined by the number of columns of $U_i \in \mathbb{R}^{n \times k_i}$ used to construct $\hat{U}_i \in \mathbb{R}^{n \times k_i}$. It should be noted that the choice of snapshot influences the accuracy of the reduced-order model (ROM), which will be discussed in detail in Section 6.

The model described by Eq. (14) can now be reduced by applying the POD-Galerkin projection to every function $F(x_i(t))$, i.e., $U_i^T F(U_i \hat{x}(t)) = 0$. For (14b), the ROM equation becomes
\[
\ddot{\hat{x}}(t) + A(t) \hat{x}(t) - \hat{w} \hat{U}_i f(U_i \hat{x}(t)) + \hat{U}_i^T f(U_i \hat{x}(t)) \hat{z} + \hat{U}_i^T U_i \hat{x}(t-1) = 0_{n \times 1}, \tag{22}
\]

The ROM equations for (14c)–(14e) are obtained similarly. The linear terms of (22) and the other reduced basis models of (14c)–(14e) have a computational complexity that depends on the number of columns $q_i$ of $U_i$ used to construct $\hat{U}_i$ 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 $f_i(x_i(t))$ 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.

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
\[
f_i(t) \approx \hat{W}_i c_i(t), \tag{23}
\]
for some matrix $\hat{W}_i \in \mathbb{R}^{n \times k}$ and coefficient vector $c_i(t)$. The matrix $\hat{W}_i \in \mathbb{R}^{n \times k}$ is obtained by applying an SVD on a snapshot matrix of the nonlinear functions $[f(x_i(t_1)), \ldots, f(x_i(t_m))]$, where $\hat{W}_i$ is constructed from the first $k$ columns of the left singular vector $\hat{W}_i = [w_1, \ldots, w_k] \in \mathbb{R}^{n \times k}$ of the snapshot matrix. The coefficient vector $c_i(t)$ can be calculated by some selected interpolation indices, given by
\[
P_i^T f_i(t) \approx P_i^T \hat{W}_i c_i(t), \tag{24}
\]
where $P_i = [e_i^{(0)}, \ldots, e_i^{(k_i)}] \in \mathbb{R}^{n \times k}$ is a matrix with $e_i^{(0)} = [0 \ldots 010 \ldots 0]^T \in \mathbb{R}^n$ and $e_i^{(k_i)}$ is the interpolation index, which can be calculated using the algorithm presented in Algorithm 1, extracted from [31]. This leads to an approximation of (23), described as
\[
f_i(t) \approx \hat{W}_i c_i(t) = \hat{W}(P_i^T \hat{W}_i)^{-1} P_i^T f_i(t). \tag{25}
\]

Algorithm 1. Discrete empirical interpolation method [31].

Require: $\hat{W}_i = [w_1^{(0)}, \ldots, w_k^{(0)}] \in \mathbb{R}^{n \times k}$ linearly independent
1: $[\hat{\phi}, \hat{\phi}^{(0)}] = \text{max}[w_1^{(0)}]$.
2: $\hat{W}_i = [w_1^{(0)}], P_i = [\hat{\phi}^{(0)}], \hat{\phi}_i = [\hat{\phi}^{(0)}]$
3: for $i = 2$ to $m$ do
4: $[\hat{\phi}, \hat{\phi}^{(i)}] = \text{max}[w_i^{(i)}], \hat{W}_i = [w_i^{(i)}], P_i = [P_i e_i^{(0)}], \hat{\phi}_i = [\hat{\phi}^{(i)}]$
5: end for
7: return $\hat{\phi}_i = [\hat{\phi}^{(0)}, \ldots, \hat{\phi}^{(m)}]^T \in \mathbb{R}^n$

The DEIM allows the nonlinear parts in Eq. (14), $f_i(x_i(t))$ for $i \in \{1, \ldots, 4\}$, to be approximated by
\[
f_i(x_i(t)) \approx \hat{W}_i (P_i^T \hat{W}_i)^{-1} P_i f_i(x_i(t)) = \hat{W}_i (P_i^T \hat{W}_i)^{-1} f_i(P_i x_i(t)), \tag{26}
\]
and the approximation for the Jacobian of the nonlinear term in (15) becomes
\[
J_i(x_i(t)) \approx \hat{U}_i^T \hat{W}_i (P_i^T \hat{W}_i)^{-1} J_i(P_i \hat{U}_i x(t)) \hat{U}_i, \tag{27}
\]
where
\[
J_i(P_i x_i) = J_i(x_i) = \text{diag}(f'(x_i^{(1)}), \ldots, f'(x_i^{(k)})). \tag{28}
\]
and $x_i(t) = P_i x_i(t)$, which can be computed with a small selected number of interpolation indices. It should be noted that the equality in (28) is due to a one-to-one correspondence between the state vector $x_i$ and the nonlinear vector function given in Eq. (14), i.e., $f_i: x_i^{(i)} \rightarrow f_i^{(i)}$. 

1289
This is because of the FVM and FDM structures. Furthermore, \( \dot{\mathbf{U}}/\mathbf{W}/\dot{\mathbf{W}} \) does not depend on time, meaning that it can be pre-computed and the complexity in evaluating the nonlinearities of (22) and the other POD-based equations become proportional to the small selected number of interpolation indices \( k_i \).

### 5. Simulation study of proposed numerical scheme

In this section, computational performance and accuracy of the numerical scheme of Section 3 are studied for a 6 (A h) Li-ion hybrid electric vehicle (HEV) battery with all the parameters summarised in Table 1, extracted from [16,25]. It should be noted that all simulation results are specific for the battery (and its parameters) under consideration. The methodology proposed in this paper and the analysis techniques presented below are general and can be repeated for cells with different model parameters.

#### 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.

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 \( n_i \) 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. Fig. 2a illustrates a comparison of the battery voltages with different size of grids for each phase during a constant 5 C-rate discharge, starting from 100% SOC. Note that the depth-of-discharge (DOD) is defined as DOD = 1 – SOC. It can be observed from Fig. 2a 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 Fig. 2a. However, as can be seen from Fig. 2a, the number of spatial discretisation steps of the solid particle (i.e., \( n_i \)) does play a significant role in accuracy. Namely, increasing \( n_i \) from \( n_i = 44 \) to \( n_i = 110 \), the solution changes, where we assume that \( n_i = 110 \) is the most accurate result in this case. According to Fig. 2a, increasing \( n_i \) from \( n_i = 44 \) to \( n_i = 88 \) leads to a significant change in voltage, while increasing \( n_i \) from \( n_i = 88 \) to \( n_i = 110 \) only leads to a small change in voltage. This conclusion is, at least for the battery type used in this paper, 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 \( n_n = 12, n_s = 6, n_p = 8 \) and \( n_i = 88 \), resulting a 1832nd-order model.

In addition, the influence of the convergence tolerance \( \delta_t \) and temporal discretisation step \( \delta_t \) is studied, while both affecting the accuracy and computation time. Fig. 2b 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 \( \delta_t = 10^{-2} \) and \( \delta_t = 0.7 \) (s), indicated by the red dashed line, and the case in which \( \delta_t = 10^{-4} \) 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 \( \delta_t = 10^{-4} \) and \( \delta_t = 0.2 \) (s) takes 1095 (s), while with \( \delta_t = 10^{-2} \) and \( \delta_t = 0.7 \) (s) takes only 613 (s). It should also be noted from Fig. 2b that increasing the temporal discretisation step \( \delta_t \) causes the model fails to converge at the end of simulation (i.e., around DOD = 0.85), albeit at an improved computational

---

**Table 1**

Model parameters of a 6 (A h) Li-ion HEV battery, extracted from [16,25].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Neg. electrode</th>
<th>Separator</th>
<th>Pos. electrode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thickness (cm)</td>
<td>( \delta )</td>
<td>50 \times 10^{-4}</td>
<td>25.4 \times 10^{-4}</td>
<td>36.4 \times 10^{-4}</td>
</tr>
<tr>
<td>Particle radius (cm)</td>
<td>( R_p )</td>
<td>1 \times 10^{-4}</td>
<td>1 \times 10^{-4}</td>
<td>0.580</td>
</tr>
<tr>
<td>Active material volume fraction</td>
<td>( \varepsilon_p )</td>
<td>0.867</td>
<td>0.372</td>
<td></td>
</tr>
<tr>
<td>Kinetic constant (A cm(^{-3}) cm(^{-1}))</td>
<td>( k_p )</td>
<td>1.38 \times 10^{-2}</td>
<td>1.2 \times 10^{-3}</td>
<td>1.2 \times 10^{-3}</td>
</tr>
<tr>
<td>Electrolyte volume fraction</td>
<td>( \varepsilon_e )</td>
<td>0.352</td>
<td>0.5</td>
<td>0.330</td>
</tr>
<tr>
<td>Maximum solid-phase concentration (mol cm(^{-3}))</td>
<td>( \varepsilon_{max} )</td>
<td>16.1 \times 10^{-5}</td>
<td>23.9 \times 10^{-5}</td>
<td></td>
</tr>
<tr>
<td>Stoichiometry at 0% SOC</td>
<td>( W_{0.5-2.5} )</td>
<td>0.126</td>
<td>0.363</td>
<td></td>
</tr>
<tr>
<td>Stoichiometry at 100% SOC</td>
<td>( W_{0.5-2.0} )</td>
<td>0.676</td>
<td>0.580</td>
<td></td>
</tr>
<tr>
<td>Average electrolyte concentration (mol cm(^{-3}))</td>
<td>( \varepsilon_t )</td>
<td>1.2 \times 10^{-5}</td>
<td>1.2 \times 10^{-3}</td>
<td>1.2 \times 10^{-3}</td>
</tr>
<tr>
<td>Charge-transfer coefficients</td>
<td>( \alpha_{\varepsilon,\varepsilon_t} )</td>
<td>0.5,0.5</td>
<td>0.5,0.5</td>
<td></td>
</tr>
<tr>
<td>SEI layer film resistance (( \Omega ) cm(^2))</td>
<td>( R_{SEI} )</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Solid-phase Li diffusion coefficient (cm(^2) s(^{-1}))</td>
<td>( D_\varepsilon )</td>
<td>2.0 \times 10^{-12}</td>
<td>3.7 \times 10^{-12}</td>
<td></td>
</tr>
<tr>
<td>Solid-phase conductivity (S cm(^{-1}))</td>
<td>( \sigma )</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Electrolyte-phase Li(^+) diffusion coefficient</td>
<td>( D_{\varepsilon} )</td>
<td>2.6 \times 10^{-6}</td>
<td>2.6 \times 10^{-6}</td>
<td></td>
</tr>
<tr>
<td>Bruggeman porosity exponent</td>
<td>( p )</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Electrolyte activity coefficient</td>
<td>( J_0 )</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Li(^+) transference number</td>
<td>( \ell_0 )</td>
<td>0.363</td>
<td>0.363</td>
<td></td>
</tr>
</tbody>
</table>
performance. We aimed at finding the largest γ that would lead to successful convergence of the numerical scheme. The fastest computation time is achieved by taking the sequence c_1, c_2, c_3, c_4 and damping coefficient γ = 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 2

<table>
<thead>
<tr>
<th>Order of subsystems</th>
<th>Maximum damping coefficient γ</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_1, c_2, c_3, c_4</td>
<td>1</td>
<td>271.668</td>
</tr>
<tr>
<td>c_2, c_3, c_1, c_4</td>
<td>1</td>
<td>281.948</td>
</tr>
<tr>
<td>c_3, c_2, c_4, c_1</td>
<td>1</td>
<td>292.480</td>
</tr>
<tr>
<td>c_4, c_3, c_1, c_2</td>
<td>1</td>
<td>326.850</td>
</tr>
<tr>
<td>c_3, c_1, c_2, c_4</td>
<td>1</td>
<td>357.362</td>
</tr>
<tr>
<td>c_1, c_3, c_2, c_4</td>
<td>1</td>
<td>359.004</td>
</tr>
<tr>
<td>c_4, c_2, c_1, c_3</td>
<td>1</td>
<td>480.177</td>
</tr>
<tr>
<td>c_2, c_4, c_1, c_3</td>
<td>1</td>
<td>493.219</td>
</tr>
<tr>
<td>c_1, c_4, c_2, c_3</td>
<td>1</td>
<td>513.315</td>
</tr>
<tr>
<td>c_3, c_4, c_1, c_2</td>
<td>1</td>
<td>516.376</td>
</tr>
<tr>
<td>c_4, c_1, c_3, c_2</td>
<td>0.85</td>
<td>534.828</td>
</tr>
<tr>
<td>c_1, c_2, c_4, c_3</td>
<td>0.85</td>
<td>545.520</td>
</tr>
<tr>
<td>c_2, c_1, c_4, c_3</td>
<td>0.85</td>
<td>546.984</td>
</tr>
<tr>
<td>c_1, c_3, c_2, c_4</td>
<td>0.8</td>
<td>585.164</td>
</tr>
<tr>
<td>c_2, c_3, c_1, c_4</td>
<td>0.8</td>
<td>388.594</td>
</tr>
<tr>
<td>c_3, c_1, c_2, c_4</td>
<td>0.8</td>
<td>408.385</td>
</tr>
<tr>
<td>c_4, c_1, c_3, c_2</td>
<td>0.8</td>
<td>457.297</td>
</tr>
<tr>
<td>c_1, c_4, c_2, c_3</td>
<td>0.8</td>
<td>572.978</td>
</tr>
<tr>
<td>c_2, c_4, c_1, c_3</td>
<td>0.8</td>
<td>653.273</td>
</tr>
<tr>
<td>c_1, c_2, c_4, c_3</td>
<td>0.8</td>
<td>657.245</td>
</tr>
<tr>
<td>c_2, c_1, c_4, c_3</td>
<td>0.8</td>
<td>688.611</td>
</tr>
<tr>
<td>c_3, c_4, c_1, c_2</td>
<td>0.75</td>
<td>600.031</td>
</tr>
<tr>
<td>c_1, c_3, c_2, c_4</td>
<td>0.75</td>
<td>663.374</td>
</tr>
<tr>
<td>c_2, c_1, c_3, c_4</td>
<td>0.75</td>
<td>675.431</td>
</tr>
<tr>
<td>c_3, c_2, c_1, c_4</td>
<td>0.75</td>
<td>728.444</td>
</tr>
</tbody>
</table>

As we observed in Section 3, the numerical scheme (16) allows for changing the order of subsystems i ∈ [1,...,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 γ for which the algorithm converges, in the sense that (17) is achieved for k < K, with K = 1000. In general, a large damping coefficient γ 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 tol = 10^{-2} and temporal discretisation step δt = 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 γ ranges from 0.8 to 0.85, 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.

5.2 Accuracy and performance analysis of the numerical model

Let us now compare the model implementation presented in Section 3 to the ones used in [25,16]. As a first step, a simulation of a constant 5 C-rate discharge, starting from 50% SOC, has been done. Fig. 3 depicts the simulation result of the electrode surface concentration c_ϕ, the electrolyte concentration c_e and the reaction current density j_{\text{reaction}} across the battery at various times, with convergence tolerance tol = 10^{-2} and temporal discretisation step δt = 0.7 (s). When comparing this simulation results with the results presented in [16], it is concluded that the results are quite similar, with very slight differences 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:

1. An iterative method with a 1832nd-order model described in [25];
2. An implementation using (16) for the 1832nd-order model with the same simplification used in [25] that we call simplified implementation;
3. 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 Eqs. (56)–(58) in [16], instead of using the surface concentration $c_{x,s}$ as given in Table 1. This leads to an open-circuit potential that is independent of the spatial variable $x$. Furthermore, Eq. (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 blue2 solid line in Fig. 4a, 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 paper. The simulated voltage using this simplification is given by the green dash-dotted line in Fig. 4a. In addition, the result of implementation done by Matlab solver ‘fsolve’ is indicated by the black dotted line.

It can be observed from Fig. 4a 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 (unimplified) model implementation (in which the open-circuit potential is computed using $c_{x,s}/c_{x,max}$) 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 $\delta = 1$ (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 paper and 3653 (s) by Matlab solver ‘fsolve’.

To further investigate the difference between the model presented in [25] and the model proposed in this paper, another simulation is done using a dynamic (dis) charge profile, shown in Fig. 4b. 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 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 paper 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 Section 3 can provide a computation time that is 3–5 times faster than real-time and 14–17 times faster than Matlab solver ‘fsolve’.

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 4, implemented by the numerical scheme of Section 3, are studied using the same HEV battery from Section 5. The simulation results of the ROM are compared to the
full-order model with state dimension 1832, presented in the previous section.

6.1. Study of snapshot and number of reduced basis

As mentioned in Section 4.1, 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 Fig. 4b is chosen to construct the snapshot in the simulation study. Moreover, some scaling of the current amplitude and temporal discretisation are applied to Fig. 4b 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 δt constant. Fig. 5 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 top figure in Fig. 5) and absolute error between the full order model and the reduced order models (cf. the bottom figure in Fig. 5). 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 × 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.

After analysis of the C-rates, the influence of the temporal discretisation step δt on the accuracy and computation time is studied. With the same time length, the size of the snapshot matrix increases as δt 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 δt = 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

\[
\text{Energy} = \frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 99.9999%,
\]

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 \(cs, \phi_s\) and \(\phi_f\) becomes \(q_1 = 156, q_2 = 26, q_3 = 26\) and \(q_4 = 20\), respectively. However, changing the number of basis of \(cs\) does play a significant role in accuracy. The accuracy and computation time decrease as the number of reduced basis of \(cs\) 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 \(cs\) 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 5.1, 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. Fig. 7 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 top figure in Fig. 7) and absolute error between the full order model and the reduced order models (cf. the bottom figure in Fig. 7). 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 $\text{tol} = 0.5$.

### 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. 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 $n_c = 1760, n_k = 26, n_x = 26$ and $n_p = 20$. According to the simulation study in Section 6.1, the POD-DEIM-based ROM is chosen as a 100th-order model with number of states $n_c = 88, n_k = 4, n_x = 4$ and $n_p = 4$.

Fig. 8 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 Fig. 9. 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.

Fig. 10 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 $c_{\text{surf}}$, the electrolyte concentration $c_{\text{e}}$, electrode potential $\phi_e$, and electrolyte potential $\phi_s$ have the same performance as Fig. 3 except some small deviations on both sides of the battery. The electrolyte concentration $c_{\text{e}}$ arrives at a steady-state distribution at the both sides of the battery a bit later than Fig. 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.

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 paper, 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 a 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 paper can be applied to more accurate battery models, leading to a more accurate reduced order model.

Acknowledgements

This work has received financial support from the Horizon 2020 programme of the European Union under the grant ‘Integrated Components for Complexity Control in affordable electrified cars (3Ccar-662192)’ and under the grant ‘Electric Vehicle Enhanced Range, Lifetime And Safety Through INGenious battery management (EVERLASTING-713771)’. The authors would like to thank Prof. Siel Weiland, Dr. Dmitri Danilov and Dr. Michiel Hochstenbach from the Eindhoven University of Technology, Netherlands, for their fruitful discussions.

References

[34] Lu J. Development of fast one-dimensional model for prediction of coupled electrochemical-thermal behavior of Lithium-ion batteries. Bachelor’s thesis, The Ohio State University; 2013.