Computer Methods in Applied Mechanics and Engineering 438 (2025) 117856

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Forward and inverse simulation of pseudo-two-dimensional model of lithium-ion batteries using neural networks

Myeong-Su Lee [a,1], Jaemin Oh [b,1], Dong-Chan Lee [c], KangWook Lee [c], Sooncheol Park [c], Youngjoon Hong [a,*]

[a] Seoul National University, 1 Gwanak-ro, Gwanak-gu, 08826, Daejeon, Republic of Korea
[b] Texas A&M University, 3128 TAMU, 77843, College Station, TX, USA
[c] Performance Concept Technology Research Lab, Hyundai Motor Company, 150 Hyundaiyeonguso-Ro, Namyang-Eup, Hwaseong-Si, 18280, Gyeonggi, Republic of Korea

ARTICLE INFO

Keywords:
Physics-informed neural networks
Lithium-ion battery
Pseudo-two-dimensional model
Forward and inverse solver

ABSTRACT

In this work, we address the challenges posed by the high nonlinearity of the Butler–Volmer (BV) equation in forward and inverse simulations of the pseudo-two-dimensional (P2D) model using the physics-informed neural network (PINN) framework. The BV equation presents significant challenges for PINNs, primarily due to the hyperbolic sine term, which renders the Hessian of the PINN loss function highly ill-conditioned. To address this issue, we introduce a bypassing term that improves numerical stability by substantially reducing the condition number of the Hessian matrix. Furthermore, the small magnitude of the ionic flux j often leads to a common failure mode where PINNs converge to incorrect solutions. We demonstrate that incorporating a secondary conservation law for the solid-phase potential ψ effectively prevents such convergence issues and ensures solution accuracy. The proposed methods prove effective for solving both forward and inverse problems involving the BV equation. Specifically, we achieve precise parameter estimation in inverse scenarios and reliable solution predictions for forward simulations.

1. Introduction

Lithium-ion (Li-ion) batteries have become integral to modern technology, powering a wide range of devices from laptops and mobile phones to electric vehicles [1]. The 2019 Nobel Prize in Chemistry, awarded to the pioneers of Li-ion battery technology, highlights its transformative impact on modern life. However, despite their widespread adoption, Li-ion batteries face inherent challenges, including finite lifecycles, performance degradation, and safety risks such as thermal runaway and explosions [2,3]. These limitations, alongside environmental and economic concerns, underscore the critical need for precise monitoring and effective management of battery systems. Ensuring reliability and safety while maximizing performance and lifespan is not only vital for current applications but also for advancing the next generation of energy storage technologies [4].

Effective battery management relies heavily on the ability to accurately predict and understand changes in the battery state under varying operating and environmental conditions [5,6]. In this context, physics-based model simulations have emerged as a powerful tool, enabling accurate battery state estimation without the need for extensive experimental data. These simulations play a pivotal role in solving both forward problems, such as predicting battery performance, and inverse problems, such as identifying key

* Corresponding author.
E-mail addresses: msl3573@snu.ac.kr (M.-S. Lee), jaemin_oh@tamu.edu (J. Oh), hongyj@snu.ac.kr (Y. Hong).
[1] Equal contribution.

https://doi.org/10.1016/j.cma.2025.117856
Received 25 November 2024; Received in revised form 21 January 2025; Accepted 17 February 2025
Available online 2 March 2025

material properties and reaction kinetics from limited data [7,8]. Moreover, the increasing emphasis on digital twin technologies highlights the central role of physics-based models in developing intelligent and efficient battery management systems. By enabling a deeper understanding of battery behavior, these models support not only safer and more sustainable energy storage solutions but also the optimization of battery design and functionality [9]. The pseudo-two-dimensional (P2D) model, also known as the Doyle–Fuller–Newman model, has been widely used in Li-ion battery studies due to its ability to accurately capture the essential electrochemical processes, such as transport and reaction kinetics [10]. The P2D model divides the battery into three distinct regions: the negative electrode, the separator, and the positive electrode. It accounts for the transport of lithium ions through the electrolyte, the diffusion of lithium within solid particles in the electrodes, and the electrochemical reactions that occur at the interfaces between the electrolyte and solid particles. The P2D model consists of coupled partial differential equations (PDEs) to describe the concentration of lithium ions and the electric potentials across these regions, along with the Butler–Volmer (BV) equation to model the kinetics of the electrochemical reactions. Numerous studies have validated the P2D model's accuracy, showing good agreement with experimental data and highlighting its effectiveness in predicting the dynamic behavior of Li-ion batteries [11,12].

Numerical experiments play a crucial role in understanding and optimizing complex systems, including Li-ion battery models. Over the years, numerous efforts have been made to develop and refine classical numerical methods for solving PDEs, such as finite element and finite difference methods. These methods have proven effective in solving the P2D model, providing accurate results under various conditions [13–15]. The availability of software packages in MATLAB [16], Python [17], and Julia [18] has further democratized the use of these methods, making them accessible to researchers and engineers across disciplines. However, the complexity of the underlying equations presents implementation challenges, and these classical methods often struggle with solving inverse problems due to high computational demands and sensitivity to noise. Recently, scientific machine learning (SciML) has emerged as a potential alternative method for solving both forward and inverse problems of PDEs. Especially, physics-informed neural networks (PINNs) incorporate governing physical laws directly into the loss function used for neural network learning, making them particularly well-suited for problems with limited data [19]. PINNs have shown promising results across various scientific domains [20–22]. These applications highlight PINNs' ability to handle complex, multi-scale, and multi-physics problems by leveraging the underlying physics as a constraint during the optimization process. Given this success in other fields, there has been increasing interest in applying SciML techniques, particularly PINNs, to battery simulations. In the context of battery simulations, SciML techniques hold significant potential for capturing hidden physical phenomena, such as transport and reaction dynamics, while also enabling the estimation of critical parameters that are difficult to measure directly.

Despite the notable progress in applying PINNs to various fields of physics and engineering, their applicability across diverse domains still remains underexplored, highlighting the need for further extensive research. In particular, vanilla PINNs have shown significant limitations in addressing highly non-linear phenomena, complex boundary conditions, and stiff problems. These challenges also arise in battery simulations [23]. Specifically, the vanilla PINN approach struggles to effectively solve the P2D model due to the highly nonlinear flux term, known as the BV equation, which describes electrochemical reactions in batteries, as well as the model's multi-scale nature. To address these challenges, several studies have proposed modified approaches. Xue et al. [24] considered the single particle model (SPM), a simplified version of the P2D model, instead of directly solving the full P2D model, to reduce computational complexity and instability. Hassanaly et al. [25] proposed a two-step training strategy: first fitting the neural network to the single particle model (SPM), and then using the learned parameters as initial values for fitting the P2D model with a linearized BV equation. For the nonlinear BV equation, they suggested an additional division step to handle gradient exploding problems. Besides, Zubov et al. [26] studied a simplified P2D model using PINNs to test their software `NeuralPDE.jl`. Chen et al. [27] simulated redox flow batteries with PINNs, formulating a model with 6 governing equations and 24 boundary conditions.

In this work, we address the specific challenges posed by the BV equation in P2D model simulations using PINNs. By designing and employing simplified toy models that replicate the inherent complexities of the BV equation, we systematically analyzed and identified critical issues that arise in battery simulations. Based on these insights, we developed novel strategies to overcome these challenges, enabling the direct solution of the full P2D model with the fully nonlinear BV equation in a single step, without relying on commonly used simplifications such as the SPM or linearized BV equations. The proposed strategies include the following. First, we introduce a bypassing term, which employs a neural network to approximate the stiffest component of the BV equation. This reduces the Hessian spectrum of the PINN loss, significantly improving the conditioning of the optimization problem and stabilizing the training process. Second, we augment the PINN loss function with a secondary conservation law for the solid-phase potential, ensuring that the predicted solution faithfully captures the nonlinear dynamics of the system and avoids convergence to incorrect solutions. Using these strategies, we achieve accurate simulation results. This demonstrates the efficacy of our approach in addressing the computational challenges associated with nonlinear battery models. Furthermore, we demonstrate the effectiveness of the proposed PINN framework in addressing inverse problems related to Li-ion battery models, specifically the estimation of critical geometric parameters such as the total battery length and the length ratio of battery sections. Inverse simulations of this kind are notoriously challenging for classical numerical methods due to high computational costs and sensitivity to noise. By leveraging the strengths of the PINN framework and building on our robust solution to the forward problem, we successfully estimate these parameters directly within the nonlinear P2D model framework. Our method requires minimal modifications to incorporate observational data and avoids the need for simplifying assumptions or additional numerical approximations. These results highlight the capability of our approach to tackle complex inverse problems with improved reliability and computational efficiency.

2. Preliminary

We start with a brief overview of the P2D model, followed by an introduction to the basic concepts of PINNs.

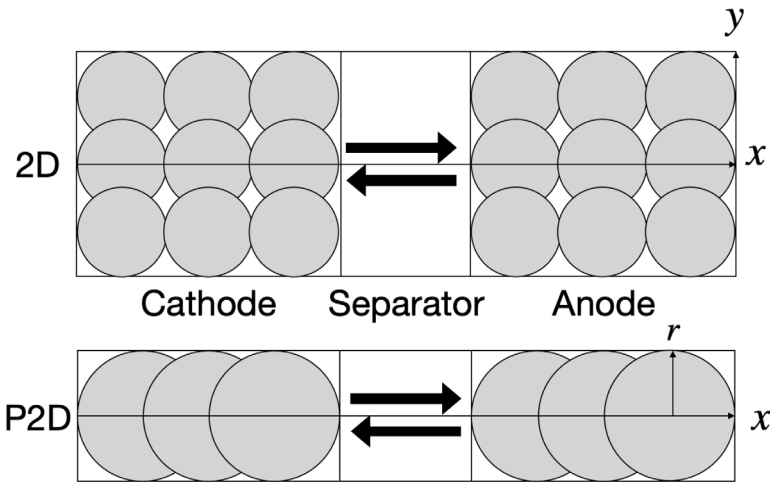

Fig. 1. A schematic of the P2D model.

2.1. Pseudo-two-dimensional model

Lithium-ion battery cells consist of three main components: the positive electrode, separator, and negative electrode. Both the positive and negative electrodes contain solid particles, and the electrolyte permeates the porous structure of these electrodes as well as the separator, allowing lithium ions to move between the electrodes. When the battery is charged or discharged, lithium ions migrate through the electrolyte from the negative electrode to the positive electrode, or vice versa, driven by the potential difference between the electrodes. During this process, lithium ions enter or exit the solid particles within the electrodes, causing changes in lithium concentration inside these particles as a result of electrochemical reactions occurring at the interfaces. These electrochemical reactions and resulting concentration changes play a crucial role in determining the battery's capacity, efficiency, and performance.

The P2D model, proposed in [10], captures the complex dynamics within a Li-ion battery cell by modeling the transport and diffusion of lithium ions along the through-thickness of the cell, denoted as the x-direction. The model assumes that lithium ions move through the electrolyte in the x-direction, while Li-ion diffusion within each spherical solid particle in the electrodes occurs symmetrically along the radial direction r. In this paper, we denote the lengths of the positive electrode, separator, and negative electrode as L_p, L_s, and L_n, respectively. Then, $x \in [0, L_p] =: \mathcal{D}_p$ corresponds to the positive electrode, $x \in [L_p, L_p + L_s] =: \mathcal{D}_s$ represents the separator, and $x \in [L_p + L_s, L_p + L_s + L_n] =: \mathcal{D}_n$ represents the negative electrode. For simplicity, we denote the total length as $L := L_p + L_s + L_n$. For the radial direction within solid particles, the coordinate r ranges from 0 to $\mathcal{R}_p$ in the positive electrode and from 0 to $\mathcal{R}_n$ in the negative electrode, where $\mathcal{R}_p$ and $\mathcal{R}_n$ denote the radii of the solid particles in each electrode, respectively. For descriptive purposes, we provide a schematic representation of the P2D model in Fig. 1.

The primary variables of the P2D model include: (1) the solid-phase Li-ion concentration, $c_{s,i}$ $(i = p, n)$, in the positive and negative electrodes; (2) the solid-phase potential, ψ_i $(i = p, n)$, in the electrodes; (3) the Li-ion concentration, c_i $(i = p, s, n)$, in the electrolyte; and (4) the liquid-phase potential, ϕ_i $(i = p, s, n)$, in the electrolyte. Here, p, s, and n represent the positive electrode, separator, and negative electrode, respectively. Additional variables, such as the overpotential η_i and ionic flux j_i, are also considered. A comprehensive summary of these state variables and other quantities used in the P2D model is provided in Table 1. Using these variables, the governing equations of the P2D model are formulated to describe the time evolution of the state variables as follows.

Solid phase concentration (Fick's Second Law in Spherical Coordinates): For each $i = p, n$,

$$\frac{\partial c_{s,i}}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 D_{s,i}\frac{\partial c_{s,i}}{\partial r}\right), \qquad r \in (0, \mathcal{R}_i).$$

The boundary conditions are specified as follows:

$$\begin{cases} -D_{s,i}^{\text{eff}}\dfrac{\partial c_{s,i}}{\partial r} = 0, & \text{at } r = 0, \\ -D_{s,i}^{\text{eff}}\dfrac{\partial c_{s,i}}{\partial r} = j_i, & \text{at } r = \mathcal{R}_i. \end{cases}$$

Liquid phase concentration: For $i = p, s, n$, we write

$$\epsilon_i \frac{\partial c_i}{\partial t} = \frac{\partial}{\partial x}\left(D_i^{\text{eff}}\frac{\partial c_i}{\partial x}\right) + (1 - t_+^0)a_i j_i.$$

The boundary conditions are given as follows:

$$-D_p^{\text{eff}}\frac{\partial c_p}{\partial x} = 0 \text{ at } x = 0 \qquad \text{and} \qquad -D_n^{\text{eff}}\frac{\partial c_n}{\partial x} = 0 \text{ at } x = L.$$

Table 1
A list of battery state variables and other quantities.

Quantity	Description	Quantity	Description
ψ	Solid phase potential	R	Gas constant
ϕ	Liquid phase potential	F	Faraday's constant
c_s	Solid phase concentration	$c_{s,\max}$	Maximum value for c_s
c	Liquid phase concentration	c_{ref}	Reference value for c
j	Ionic flux	k	Electrochemical reaction rate constant
T	Temperature	T_{ref}	Reference temperature
D	Diffusion coefficient	D_s	Diffusion coefficient for solid particle
U	Open circuit potential	I_{app}	Applied current density
a	Area	σ	Electronic conductivity of solid matrix
ϵ	Volume fraction	t_+^0	Transference number of Li-ion
η	Overpotential	Brugg	Bruggmann constant
κ	Ionic conductivity	$\bullet^{\text{eff}}$	Effective coefficient: $\bullet \times \epsilon^{\text{Brugg}}$

The interfacial conditions are given to impose continuity and their flux continuity at the interfaces:

$$\begin{cases} c_p = c_s \\ -D_p^{\text{eff}} \dfrac{\partial c_p}{\partial x} = -D_s^{\text{eff}} \dfrac{\partial c_s}{\partial x} \end{cases} \quad x = L_p \quad \text{and} \quad \begin{cases} c_s = c_n \\ -D_s^{\text{eff}} \dfrac{\partial c_s}{\partial x} = -D_n^{\text{eff}} \dfrac{\partial c_n}{\partial x} \end{cases} \quad x = L_p + L_s.$$

Solid Phase Potential (Ohm's Law): For each $i = p, n$,

$$\begin{cases} \sigma_p^{\text{eff}} \dfrac{\partial^2 \psi_p}{\partial x^2} = a_p F j_p & x \in (0, L_p) \\ -\sigma_p^{\text{eff}} \dfrac{\partial \psi_p}{\partial x} = I_{\text{app}} & x = 0, \\ -\sigma_p^{\text{eff}} \dfrac{\partial \psi_p}{\partial x} = 0 & x = L_p \end{cases} \quad \text{and} \quad \begin{cases} \sigma_n^{\text{eff}} \dfrac{\partial^2 \psi_n}{\partial x^2} = a_n F j_n & x \in (0, L_n) \\ -\sigma_n^{\text{eff}} \dfrac{\partial \psi_n}{\partial x} = 0 & x = L_p + L_s, \\ \psi_n = 0 & x = L, \end{cases}$$

where I_{app} is the applied current density.

Liquid phase potential (based on Ohm's Law): For $i = p, s, n$,

$$-\frac{\partial}{\partial x}\left(\kappa_i^{\text{eff}} \frac{\partial \phi_i}{\partial x}\right) + \frac{2RT(1-t_+^0)}{F} \frac{\partial}{\partial x}\left(\kappa_i^{\text{eff}} \frac{\partial \log c_i}{\partial x}\right) = a_i F j_i.$$

The boundary conditions are given as follows:

$$-\kappa_p^{\text{eff}} \frac{\partial \phi_p}{\partial x} = 0 \quad \text{at } x = 0 \quad \text{and} \quad -\kappa_n^{\text{eff}} \frac{\partial \phi_n}{\partial x} = 0 \quad \text{at } x = L.$$

The interfacial conditions are given to impose continuity and their flux continuity at the interfaces:

$$\begin{cases} \phi_p = \phi_s \\ -\kappa_p^{\text{eff}} \dfrac{\partial \phi_p}{\partial x} = -\kappa_s^{\text{eff}} \dfrac{\partial \phi_s}{\partial x} \end{cases} \quad x = L_p \quad \text{and} \quad \begin{cases} \phi_s = \phi_n \\ -\kappa_s^{\text{eff}} \dfrac{\partial \phi_s}{\partial x} = -\kappa_n^{\text{eff}} \dfrac{\partial \phi_n}{\partial x} \end{cases} \quad x = L_p + L_s.$$

The aforementioned state variables are coupled by the ionic flux j_i, which is given by the following BV equation:

$$j_i = \begin{cases} 2k_i \left(c_{s,i,\max} - c_{s,i,\text{surf}}\right)^{0.5} c_{s,i,\text{surf}}^{0.5} c_i^{0.5} \sinh\left(\dfrac{F}{2RT}\eta_i\right) & \text{for } i = p, n \\ 0 & \text{for } i = s \end{cases} \tag{1}$$

where $c_{s,i,\text{surf}} = c_{s,i}(\mathcal{R}_i)$, and η_i represents the over-potential defined as:

$$\eta_i = \psi_i - \phi_i - U_i\left(\frac{c_{s,i}}{c_{s,i,\max}}\right),$$

and U_i denotes the open circuit potential.

Interested readers may consult with [28] for well-posedness, and [13,16] for numerics.

2.2. Physics-informed neural networks

Physics-Informed Neural Networks (PINNs), as proposed in [19], are a deep learning framework designed to solve both forward and inverse problems for partial differential equations (PDEs) within a unified framework.

For simplicity, we consider the following simple example. Let $\Omega \subset \mathbb{R}^{d_{\text{in}}}$ be a domain and $u : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ be the solution to the following differential equation:

$$\mathcal{N}[u](x) = f(x), \quad x \in \Omega,$$
$$\mathcal{B}[u](x) = g(x), \quad x \in \partial\Omega,$$

where $\mathcal{N}$ is a general differential operator and $\mathcal{B}$ is a boundary operator. Note that the boundary condition can also represent initial conditions in time-dependent problems. PINNs approach approximates the solution u by using a neural network approximation $u_{\theta^\star}$, where $\theta^\star$ denotes the optimal set of neural network parameters. These parameters are obtained by solving the following minimization problem:

$$\theta^\star = \arg\min_\theta \mathcal{L}_{\text{PINN}}(\theta),$$

where the loss function $\mathcal{L}_{\text{PINN}}(\theta)$ is defined as:

$$\mathcal{L}_{\text{PINN}}(\theta) = \int_\Omega \left|\mathcal{N}[u_\theta](x) - f(x)\right|_2^2 \mathrm{d}x + \lambda \int_{\partial\Omega} \left|\mathcal{B}[u_\theta](s) - g(s)\right|_2^2 \mathrm{d}\sigma(s). \tag{2}$$

Here, σ is a surface measure, and λ is a weighting parameter balancing the importance of the differential equation and the boundary conditions. In practice, the integrals in (2) are approximated using numerical quadrature methods. Common approaches include the trapezoidal rule [29], Gauss quadrature [30], and Quasi-Monte Carlo methods, such as the Sobol sequence [31]. The choice of the quadrature method can significantly impact the accuracy and efficiency of the PINN approach and may depend on the specific problem and domain characteristics.

3. Conventional application of PINNs to the P2D model

In this section, we first provide a complete presentation of how PINNs are typically applied to PDEs, which we refer to as the "vanilla PINN" approach. We then discuss some issues that can appears in applying the vanilla PINNs and address the issues with existing methods.

3.1. Vanilla PINNs for the P2D model

In this subsection, we present a complete explanation of a vanilla PINN approach to solve the P2D model. The P2D model includes two interfaces: one between the positive electrode and the separator, and the other between the separator and the negative electrode. The interfacial conditions enforce that for the liquid-phase concentration c_i and the potential ψ_i, they and their flux are continuous at the interfaces of both the electrodes and the separator. The flux continuity conditions can cause discontinuities in the first derivatives, which makes it difficult for a single approximate network defined over the whole x-axis to satisfy the flux continuity. To address this, we adopt a *domain decomposition* approach, dividing the x-axis into three subdomains $\mathcal{D}_p$, $\mathcal{D}_s$, and $\mathcal{D}_n$. We then assign a separate neural network to approximate each primary state variable (e.g. $c_{s,i}, c_i, \phi_i, \psi_i$) in each subdomain. In other words, each region $i \in \{p, s, n\}$ is handled by its own network, denoted by

$$\widetilde{c}_{s,i}, \quad \widetilde{c}_i, \quad \widetilde{\phi}_i, \quad \widetilde{\psi}_i.$$

With such approximations for the primary variables by neural networks, the ionic flux j_i is approximated by

$$\widetilde{j}_i = 2k\left(c_{s,\max} - \widetilde{c}_{s,i,\text{surf}}\right)^{0.5} \widetilde{c}_{s,i,\text{surf}}^{0.5} \widetilde{c}_i^{0.5} \sinh\left(\frac{F}{2RT}\left(\widetilde{\phi}_i - \widetilde{\psi}_i - U(\widetilde{c}_{s,i}/c_{\text{s,max}})\right)\right), \tag{3}$$

according to the original relation (1). Then, the PINN loss function is given as follows:

$$\mathcal{L}_{\text{PINN}}(\theta) := \mathcal{L}_{\text{PDE}}(\theta) + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}(\theta) + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}(\theta) + \lambda_{\text{Inter}}\mathcal{L}_{\text{Inter}}(\theta), \tag{4}$$

where θ represents the network parameter and each of the loss function terms and $\lambda_\bullet$ are weighting parameters to be chosen appropriately. Each term in the loss function — $\mathcal{L}_{\text{PDE}}(\theta)$, $\mathcal{L}_{\text{IC}}(\theta)$, $\mathcal{L}_{\text{BC}}(\theta)$, and $\mathcal{L}_{\text{Inter}}(\theta)$ — is conventionally defined to enforce the PDEs, boundary/initial conditions, and the interface conditions into the networks.

Specifically, the PDE residual loss $\mathcal{L}_{\text{PDE}}(\theta)$ can be split into four parts, corresponding to the main variables: $\mathcal{L}_{\text{c}_\text{s},\text{PDE}}(\theta)$, $\mathcal{L}_{\text{c,PDE}}(\theta)$, $\mathcal{L}_{\phi,\text{PDE}}(\theta)$, and $\mathcal{L}_{\psi,\text{PDE}}(\theta)$. For example, the PDE residual loss $\mathcal{L}_{\text{c,PDE}}(\theta)$ for the liquid concentration c is given as

$$\mathcal{L}_{\text{c,PDE}}(\theta) = \sum_{i\in\{p,s,n\}} \int_{[0,\mathcal{T}]\times\mathcal{D}_i\times[0,\mathcal{R}_i]} \left|\epsilon_i \frac{\partial \widetilde{c}_i}{\partial t} - \frac{\partial}{\partial x}\left(D^{\text{eff}}\frac{\partial \widetilde{c}_i}{\partial x}\right) - (1 - t_+^0)a_i\widetilde{j}_i\right|^2 \mathrm{d}t\mathrm{d}x\mathrm{d}r_,.$$

The PDE residual losses for the other state variables are given similarly. Likewise, the initial, boundary, and interface condition losses can also be split into four parts. For instance, the losses for the liquid-phase concentration c_i are given as follows:

$$\mathcal{L}_{\text{c,IC}}(\theta) = \sum_{i=p,s,n} \int_{D_i} \left|\widetilde{c}_i(0,x) - c_i^0(x)\right|^2 \mathrm{d}x,$$
$$\mathcal{L}_{\text{c,BC}}(\theta) = \int_{[0,\mathcal{T}]} \left|D_p^{\text{eff}}\frac{\partial \widetilde{c}_p}{\partial x}(t,0)\right|^2 + \left|D_n^{\text{eff}}\frac{\partial \widetilde{c}_n}{\partial x}(t,L)\right|^2 \mathrm{d}t,$$

$$\mathcal{L}_{\text{c,Inter}}(\theta) = \int_{[0,\mathcal{T}]} \left| D_p^{\text{eff}} \frac{\partial \widetilde{c}_p}{\partial x}(t, L_p) - D_s^{\text{eff}} \frac{\partial \widetilde{c}_s}{\partial x}(t, L_p) \right|^2 + \left| \widetilde{c}_p(t, L_p) - \widetilde{c}_s(t, L_p) \right|^2 + \left| D_s^{\text{eff}} \frac{\partial \widetilde{c}_s}{\partial x}(t, L_p + L_s) - D_n^{\text{eff}} \frac{\partial \widetilde{c}_n}{\partial x}(t, L_p + L_s) \right|^2 + \left| \widetilde{c}_s(t, L_p + L_s) - \widetilde{c}_n(t, L_p + L_s) \right|^2 \mathrm{d}t.$$

The boundary, initial, and interface losses for other variables are defined similarly, following the same manners. While the vanilla PINN approach, in principle, be applied to a wide range of PDEs, PINNs have often been shown to exhibit failure modes [32], particularly for strongly nonlinear PDE systems. Indeed, when applying this straightforward method to the P2D model, one encounters such failure modes due to inherent complexities. In the remainder of this section, we discuss several challenges with the vanilla approach and address them by using standard strategies.

3.2. Issues with vanilla PINNs

This subsection is devoted to addressing several issues that arise when applying the above vanilla approach to solve the P2D model.

Multi-scale issues

One of the challenges in applying PINNs to the P2D model lies in the significant scale differences between the microscopic and macroscopic coordinates, as well as the various physical and chemical constants in the system. The P2D model considers two spatial coordinates: (1) macroscopic coordinates x, representing the through-thickness position across the electrolyte and electrodes, and (2) microscopic coordinates r, representing the radial position within solid particles. These scale differences pose numerical difficulties in the optimization process, particularly when the scales differ by several orders of magnitude. Additionally, the physical and chemical constants, such as diffusion coefficients and conductivity, exhibit vast disparities in scale, further complicating the learning process in the PINN framework.

Such scale-related issues have been widely reported not only in classical numerical methods but also in several SciML methods. One of the simplest yet effective techniques to address these issues is non-dimensionalization. Following this, we adopt a non-dimensionalized form of the P2D model to mitigate the multi-scale issues. Specifically, the coordinate components t, x, and r, and the state variables $c_{s,i}$, c_i, ϕ_i, and ψ_i are rescaled as follows:

$$\hat{t} = \frac{t}{\mathcal{T}}, \quad \hat{x} = \frac{x}{L}, \quad \hat{r} = \frac{r}{\mathcal{R}_i}, \quad \hat{c}_{s,i} = \frac{c_{s,i}}{c_{s,\max}}, \quad \hat{c}_i = \frac{c_i}{c_{\text{ref}}}, \quad \hat{\phi}_i = \frac{\phi_i}{\phi_{\text{ref}}}, \quad \hat{\psi}_i = \frac{\psi_i}{\psi_{\text{ref}}},$$

which yields $\hat{t},\ \hat{x},\ \hat{r} \in [0,1]$, $\hat{c}_{s,i} \in [0,1]$, and $\hat{c}_i, \hat{\phi}_i, \hat{\psi}_i \sim \mathcal{O}(1)$. Following this non-dimensionalization, the rescaled governing equations are given by:

Solid phase concentration: For $i = p, s, n$,

$$\hat{r} \frac{\partial \hat{c}_{s,i}}{\partial \hat{t}} = \frac{\mathcal{T} D_{s,i}}{R_i^2} \left(2 \frac{\partial \hat{c}_{s,i}}{\partial \hat{r}} + \frac{\partial^2 \hat{c}_{s,i}}{\partial \hat{r}^2} \right).$$

The boundary conditions are specified as follows:

$$\begin{cases} \dfrac{\partial \hat{c}_{s,i}}{\partial \hat{r}} = 0 & \text{at } \hat{r} = 0, \\ \dfrac{\partial \hat{c}_{s,i}}{\partial \hat{r}} = \dfrac{\mathcal{R}_i j_i}{-D_{s,i}^{\text{eff}}} & \text{at } \hat{r} = 1, \end{cases}$$

Liquid phase concentration: For each $i = p, s, n$,

$$\frac{\partial \hat{c}_i}{\partial \hat{t}} = \frac{D_i^{\text{eff}} \mathcal{T}}{L^2 \epsilon_i} \frac{\partial^2 \hat{c}_i}{\partial \hat{x}^2} + \frac{(1 - t_+^0) \mathcal{T} a_i j_i}{\epsilon_i c_{\text{ref}}}.$$

The boundary conditions are given as follows:

$$\frac{\partial \hat{c}_p}{\partial \hat{x}} = 0 \quad \text{at } \hat{x} = 0 \quad \text{and} \quad \frac{\partial \hat{c}_n}{\partial \hat{x}} = 0 \quad \text{at } \hat{x} = 1.$$

The interfacial conditions are given to impose continuity and their flux continuity at the interfaces:

$$\begin{cases} \hat{c}_p = \hat{c}_s \\ -D_p^{\text{eff}} \dfrac{\partial \hat{c}_p}{\partial \hat{x}} = -D_s^{\text{eff}} \dfrac{\partial \hat{c}_s}{\partial \hat{x}} \end{cases} \quad \hat{x} = \frac{L_p}{L} \quad \text{and} \quad \begin{cases} c_s = c_n \\ -D_s^{\text{eff}} \dfrac{\partial c_s}{\partial x} = -D_n^{\text{eff}} \dfrac{\partial c_n}{\partial x} \end{cases} \quad \hat{x} = \frac{L_p + L_s}{L}.$$

Solid phase potential: For $i = p, n$,

$$\begin{cases} \dfrac{\partial^2 \hat{\psi}_p}{\partial \hat{x}^2} = \dfrac{L^2 a_p F j_p}{\sigma_p^{\text{eff}} \psi_{\text{ref}}} & \hat{x} \in (0, L_p/L) \\ \dfrac{\partial \hat{\psi}_p}{\partial \hat{x}} = \dfrac{L^2 I_{\text{app}}}{-\sigma_p^{\text{eff}} \psi_{\text{ref}}} & \hat{x} = 0, \\ \dfrac{\partial \hat{\psi}_p}{\partial \hat{x}} = 0 & \hat{x} = L_p/L \end{cases} \quad \text{and} \quad \begin{cases} \dfrac{\partial^2 \hat{\psi}_n}{\partial x^2} = \dfrac{L^2 a_n F j_n}{\sigma_n^{\text{eff}} \psi_{\text{ref}}} & \hat{x} \in (0, L_n/L) \\ \dfrac{\partial \hat{\psi}_n}{\partial \hat{x}} = 0 & \hat{x} = (L_p + L_s)/L, \\ \hat{\psi}_n = 0 & \hat{x} = 1. \end{cases}$$

Liquid phase potential: For $i = p, s, n$,

$$-\frac{\partial}{\partial \hat{x}}\left(\kappa_i^{\text{eff}} \frac{\partial \hat{\phi}_i}{\partial \hat{x}}\right) + \frac{2RT(1-t_+^0)}{F} \frac{\partial}{\partial \hat{x}}\left(\kappa_i^{\text{eff}} \frac{\partial \log \hat{c}_i}{\partial \hat{x}}\right) = L^2 a_i F j_i.$$

The boundary conditions are given as follows:

$$-\kappa_p^{\text{eff}} \frac{\partial \hat{\phi}_p}{\partial \hat{x}} = 0 \quad \text{at } \hat{x} = 0 \quad \text{and} \quad -\kappa_n^{\text{eff}} \frac{\partial \hat{\phi}_n}{\partial \hat{x}} = 0 \quad \text{at } \hat{x} = 1.$$

The interfacial conditions are given as follows:

$$\begin{cases} \hat{\phi}_p = \hat{\phi}_s \\ -\kappa_p^{\text{eff}} \dfrac{\partial \hat{\phi}_p}{\partial \hat{x}} = -\kappa_s^{\text{eff}} \dfrac{\partial \hat{\phi}_s}{\partial \hat{x}} \end{cases} \quad \hat{x} = \frac{L_p}{L} \quad \text{and} \quad \begin{cases} \hat{\phi}_s = \hat{\phi}_n \\ -\kappa_s^{\text{eff}} \dfrac{\partial \hat{\phi}_s}{\partial \hat{x}} = -\kappa_n^{\text{eff}} \dfrac{\partial \hat{\phi}_n}{\partial \hat{x}} \end{cases} \quad \hat{x} = \frac{L_p + L_s}{L}.$$

Butler–Volmer equation With the rescaled state variables $\hat{c}_{s,i}$, $\hat{c}_i$, $\hat{\phi}_i$, and $\hat{\psi}_i$, the formula for the reaction rate j_i is also rewritten as follows:

$$j_i = \begin{cases} 2k c_{s,i,\max}^{0.5} c_{\text{ref}}^{0.5} \left(1 - \hat{c}_{s,i,\text{surf}}\right)^{0.5} \hat{c}_{s,i,\text{surf}}^{0.5} \hat{c}_i^{0.5} \sinh\left(\dfrac{F}{2RT} \eta_i\right) & \text{for } i = p, n \\ 0 & \text{for } i = s \end{cases}$$

where η_i is given as:

$$\eta_i = \psi_{\text{ref}} \hat{\psi}_i - \phi_{\text{ref}} \hat{\phi}_i - U_i\left(\hat{c}_{s,i}\right).$$

Throughout this paper, we will use $c_{\text{ref}} = 10^3$, and $\phi_{\text{ref}} = \psi_{\text{ref}} = 1$. In all the following sections, we will consider only the rescaled form of the P2D model, obtained via the above non-dimensionalization. For simplicity, we omit the $\hat{\cdot}$ notation, and all variables will be referred to in their rescaled forms.

Initial and boundary conditions

As shown in the PINN loss (4), PINNs generally enforce boundary and initial conditions on the neural network through penalty-type loss functions. This approach, commonly referred to as the soft constraint method or penalty method, has been effective in many cases. However, the imbalance between the PDE residual loss and the penalty loss can lead to training instability and failure, even for relatively simple cases [33].

To address this issue, an alternative approach is to redesign the neural network architecture so that it inherently satisfies the boundary and initial conditions, eliminating the need for additional penalty loss terms. This method, commonly known as the hard constraint method, has been widely applied in PINN applications [34]. Adopting this hard-constraint approach, we reconstruct the network architectures in such a way that it automatically satisfies some of the boundary and initial conditions, as follows:

The liquid concentration We use the following structure of network to approximate the liquid concentration c_i:

$$\widetilde{c}_i(t, x; \theta) = c_i^0(x) \exp\left(t \cdot \text{MLP}(t, x; \theta)\right), \qquad \text{for } i = p, s, n,$$

which directly enforce that the approximate solution $\widetilde{c}(t, x; \theta)$ satisfies the initial condition:

$$\widetilde{c}_i(0, x; \theta) = c_i^0(x), \qquad \text{for } i = p, s, n.$$

The solid concentration For the solid concentration $c_{s,p}$ in the positive electrode, we use

$$\widetilde{c}_{s,p}(t, r, x; \theta) = 1 - \exp\left(-\left(\sqrt{\log\left(\frac{1}{1 - c_{s,p}^0(r, x)}\right)} + t \cdot \text{MLP}(t, r, x; \theta)\right)^2\right),$$

and for the solid concentration in the negative electrode,

$$\widetilde{c}_{s,n}(t, r, x; \theta) = \exp\left(-\left(\sqrt{\log\left(\frac{1}{c_{s,n}^0(r, x)}\right)} + t \cdot \text{MLP}(t, r, x; \theta)\right)^2\right).$$

From the construction of approximate solutions, it follows that

$$\widetilde{c}_{s,i}(0, r, x; \theta) = c^0_{s,p}(x) = c^0_{s,i}(r, x) \qquad \text{for } i = p, n.$$

The solid potential For the solid potential ψ_n in the negative electrode, we utilize

$$\widetilde{\psi}_n(t, x; \theta) = (1 - x) \cdot \mathrm{MLP}(t, x; \theta),$$

which implies that

$$\widetilde{\psi}_n(t, 1; \theta) = 0,$$

where we note that each $\mathrm{MLP}(\cdot; \theta)$ represents a different fully-connected neural network.

Through this hard-constraint method, the approximate solutions automatically satisfy the initial conditions and Dirichlet boundary conditions without soft constraints. However, for Neumann boundary conditions, we do not apply the hard-constraint method and instead continue using penalty methods, to be consistent with interfacial conditions which will be discussed below.

Interfacial conditions

Unlike boundary or initial conditions, enforcing interfacial conditions through a hard-constraint method is not straightforward, since at least two approximate solutions from different networks are involved at one interface condition. Based on lessons from the additive Schwarz method, it is reasonable to expect that naively imposing interface conditions could lead to reduced information exchange and potentially slower convergence due to limited communication between the individual neural networks.

To explore how PINNs handle complex interface conditions, we tested a simplified toy example to represent the geometry of a Lithium-ion battery:

$$\begin{cases} u_x = 0, \quad \phi_x = 0 & x = 0, \\ u_t = \dfrac{4}{\pi} u_{xx} - \dfrac{\pi^2}{4}\phi + \cos\left(\dfrac{\pi}{2}x\right)\cos(t), \quad \phi_{xx} = u & x \in (0, 1), \\ u = v, \quad 4u_x = v_x, \quad \phi = 0 & x = 1, \\ v_t = \dfrac{1}{4\pi^2} v_{xx} - \sin(2\pi x)\,(\sin(t) + \cos(t)) & x \in (1, 2), \\ v = w, \quad v_x = 2w_x, \quad \psi = 0 & x = 2, \\ w_t = \dfrac{1}{\pi^2} w_{xx} - \psi - \sin(\pi x)\cos(t), \quad \psi_{xx} = w & x \in (2, 3), \\ w = w_{\mathrm{sol}}, \quad \psi_x = 0 & x = 3. \end{cases} \tag{5}$$

The exact solution for the equations is given by:

$$\begin{aligned} u_{\mathrm{sol}}(x, t) &= \cos\left(\frac{\pi}{2}x\right)\sin(t), \\ v_{\mathrm{sol}}(x, t) &= -\sin(2\pi x)\sin(t), \\ w_{\mathrm{sol}}(x, t) &= -\sin(\pi x)\sin(t), \\ \phi_{\mathrm{sol}}(t, x) &= -\frac{4}{\pi^2}\cos\left(\frac{\pi}{2}x\right)\sin(t), \\ \psi_{\mathrm{sol}}(t, x) &= \frac{1}{\pi^2}\sin(\pi x)\sin(t). \end{aligned} \tag{6}$$

This toy example represents a simplified scenario that reflects the essential dynamics of a Lithium-ion battery system. It enables us to validate the performance of the domain decomposition method in handling interfacial conditions with discontinuous derivatives. By applying appropriate boundary and interface conditions, this setup captures the behavior of the electrodes and the separator.

To solve this system using PINNs, we use five neural networks—one for each variable. Interface conditions are incorporated into the PINN loss by adding appropriate penalties. Fig. 2 shows the result of the simulation. Despite the complexity of the system, which includes four boundary conditions, six interface conditions, and five PDEs (5), the PINN framework successfully approximates the exact solution with only the penalty method on the interfacial conditions. Through this toy example, we found that a simpler approach — adding soft penalties — is surprisingly sufficient to ensure proper information exchange between the subdomains.

3.3. Limitations of the conventional approach

In this section, we presented a vanilla PINN approach for solving the P2D model and then applied existing strategies to address certain issues with that approach. The resulting conventional PINN framework is illustrated in Fig. 3. Notably, all initial conditions are already satisfied by construction through the hard constraint method, making any additional penalty terms for initial conditions unnecessary. However, while these strategies mitigate several difficulties that may arise in solving the P2D model, they remain insufficient for fully resolving complexities of the P2D model. In particular, when using a conventional PINN setup, the initial training loss can be as large as $O(10^{75})$, leading to severe instability and ultimately causing convergence to fail. In the following section, we focus on the challenges posed by the BV equation, which we believe is a key reason why this conventional approach can fail.

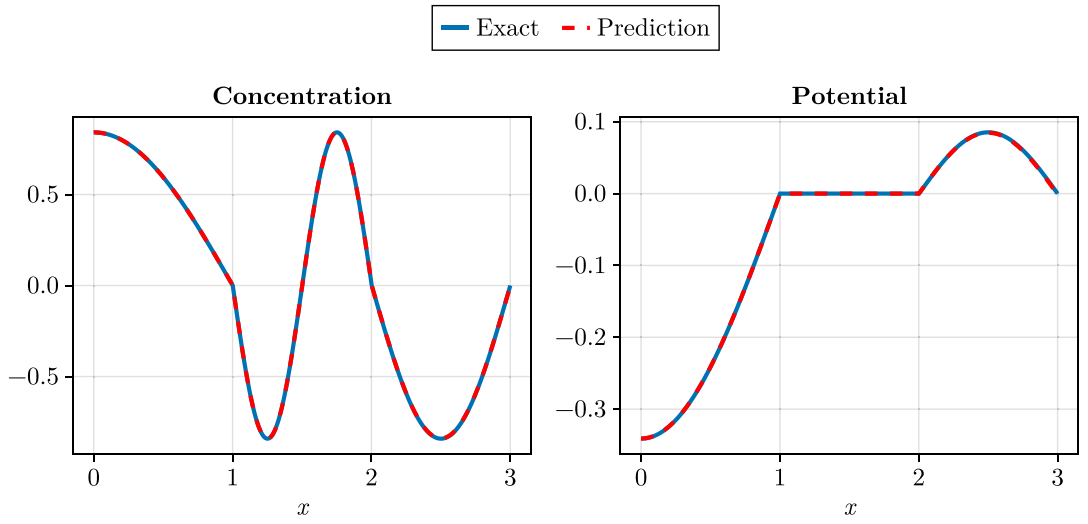

Fig. 2. Comparison of the PINN result for (5) to the exact solution (6). The left figure presents *concentration* variables u, v, and w. The right figure shows *potential* variables ϕ and ψ. Note that potential values are zero in $[1,2]$, to imitate the separator.

4. Challenges related to the Butler–Volmer equation

In this section, we examine the remaining issues of the conventional PINN approach in Section 3, which lead to an excessively large final loss and ultimately fail to yield accurate solutions. We have identified that these problems stem primarily from the nonlinearity of the BV equation (1) in the P2D model. This strong nonlinearity makes the BV equation one of the most challenging parts of the P2D model to address using PINNs.

4.1. Sensitivity of the BV equation with respect to η

We recall the BV equation in its rescaled form:

$$j_i \;=\; 2k\, c_{s,i,\max}\, c_{\mathrm{ref}}^{0.5} \left(1 - c_{s,i,\mathrm{surf}}\right)^{0.5} c_{s,i,\mathrm{surf}}^{0.5}\, c_i^{0.5}\, \sinh\Bigg(\underbrace{\frac{F}{2RT}}_{\mathcal{G}}\, \eta_i \Bigg), \tag{7}$$

where $\eta_i = \psi_i - \phi_i - U_i$. Notably, the constant $\mathcal{G}$ inside the hyperbolic sine function in (7) is approximately 19.5. As shown in the left plot of Fig. 4, the graph of $f(\eta) = \sinh(19.5\eta)$ demonstrates a very fast growth rate as η increases. We note that small changes in the approximate solutions ϕ_p and ψ_p can lead to large fluctuations in the loss function, exacerbating instability during training. The impact of this issue is illustrated in the upper right panel of Fig. 4, which shows the loss behavior obtained using the conventional PINN approach described earlier. During the early stages of training, the initial loss reaches a magnitude of approximately $O(10^{75})$. In contrast, when η is manually set to zero, the starting loss value drops to approximately $O(10^{-1})$, clearly demonstrating the ill-conditioning posed by the BV equation.

4.2. Discrepancy between loss and accuracy

A small PINN loss does not always imply that the corresponding approximate solution is sufficiently close to the true solution of the target PDEs. Such discrepancies between loss and accuracy have been reported in the literature [32,35]. In the context of the P2D model, this issue is especially relevant for the BV equation. We recall the rescaled governing equation for the solid potential ψ_p:

$$\frac{\partial^2 \psi_p}{\partial x^2} \;=\; \frac{L^2 a_p F\, j_p}{\sigma_p^{\mathrm{eff}}\, \psi_{\mathrm{ref}}}, \quad x \in (0,1),$$

where the source term on the right-hand side is:

$$\underbrace{\frac{2k\, c_{s,p,\max}\, c_{\mathrm{ref}}^{0.5}\, L^2\, a_p\, F}{\sigma^{\mathrm{eff}}\, \psi_{\mathrm{ref}}} \left(1 - c_{s,p,\mathrm{surf}}\right)^{0.5} c_{s,p,\mathrm{surf}}^{0.5}\, c_p^{0.5}}_{=:\mathcal{H}}\; \sinh\Big(\tfrac{F}{2RT}\, \eta_p\Big), \tag{8}$$

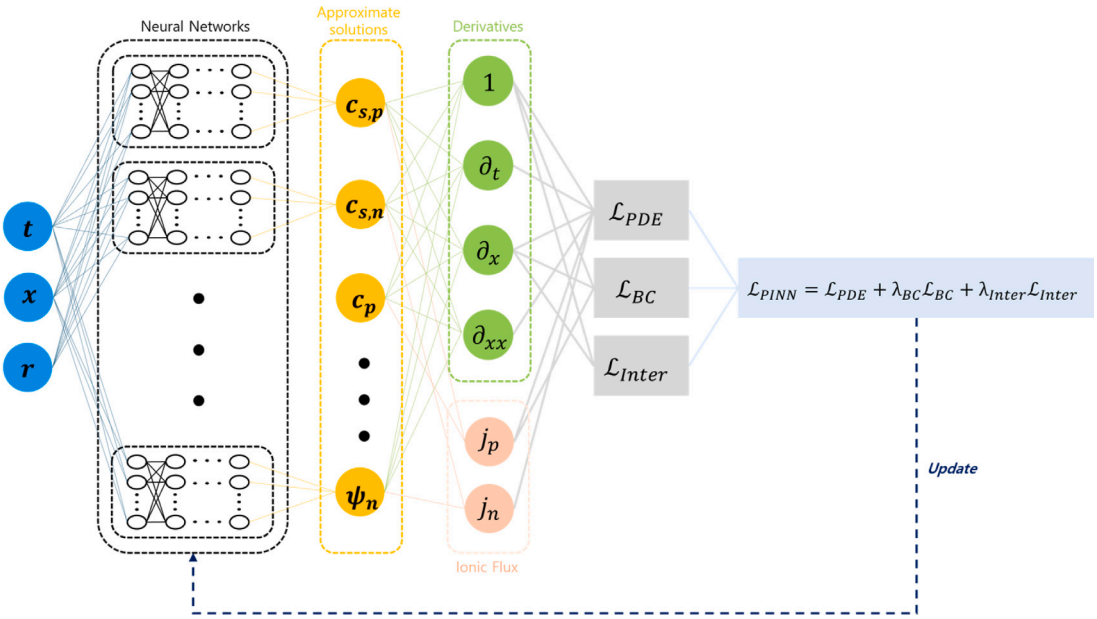

Fig. 3. A schematic diagram of the PINN architecture for the P2D model of a Li-ion battery cell.

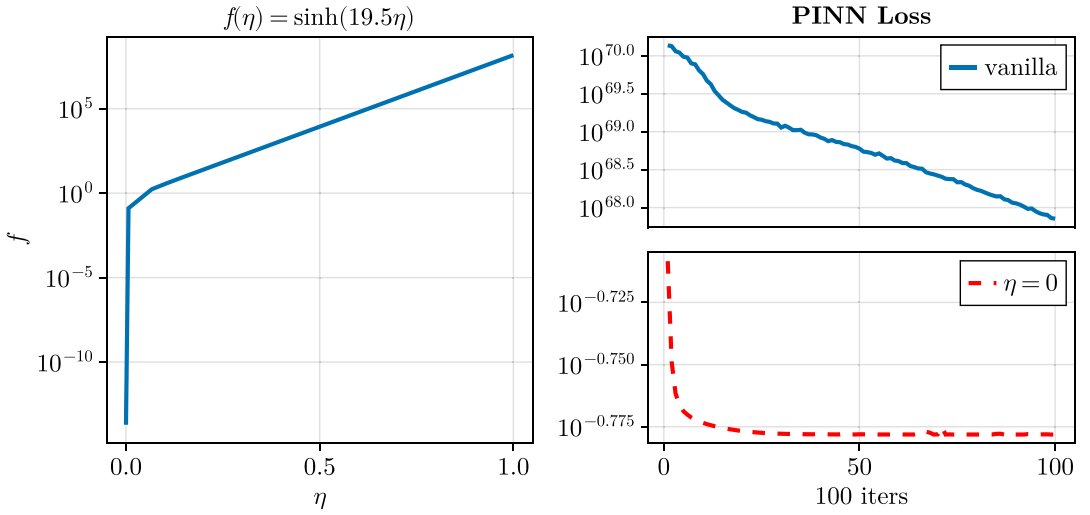

Fig. 4. The left panel displays the graph of the function $f(\eta) = \sinh(19.5\eta)$, while the right panel illustrates how errors in η are amplified when evaluating the PINN loss. The solid line represents the baseline, whereas the dashed line corresponds to the scenario where $\eta = 0$ is manually enforced. These results highlight why the nonlinear BV equation (1) is the primary contributor to the large PINN loss values.

according to the definition of j_p in (1). Notably, the coefficient $\mathcal{H}$ can be much smaller in magnitude than other rescaled variables (on the order of 1). Although the hyperbolic sine function grows exponentially for large inputs, it behaves almost linearly near the origin. In most practical situations, the over-potential η remains small, so $\sinh\left(\frac{F}{2RT}\eta_p\right) \approx \frac{F}{2RT}\eta_p$. Consequently, when $\mathcal{H}$ is also very small, the right-hand side of (8) changes only slightly even if η_p (or ψ_p) is perturbed.

Keeping these points in mind, let us consider a simple example of how inaccurate solutions can still yield small PINN losses. Suppose $\widetilde{\psi}_p$ differs from the true solution ψ_p by a constant shift $C = O(1)$, i.e., $\widetilde{\psi}_p(x) = \psi_p(x) + C$. Then, $\widetilde{\psi}_p$ can produce nearly the same small loss as ψ_p, even though $\widetilde{\psi}_p$ is far from the true solution. The reasons are:

- **PDE residual:** The left-hand side of PDE involves only $\frac{\partial^2 \psi_p}{\partial x^2}$ so that a constant shift C disappears in that side. Also, if $\mathcal{H}$ is sufficiently small, the source term changes very slightly when shifting ψ_p by C. Therefore, the shifted one may still lead to a very small PDE residual loss.

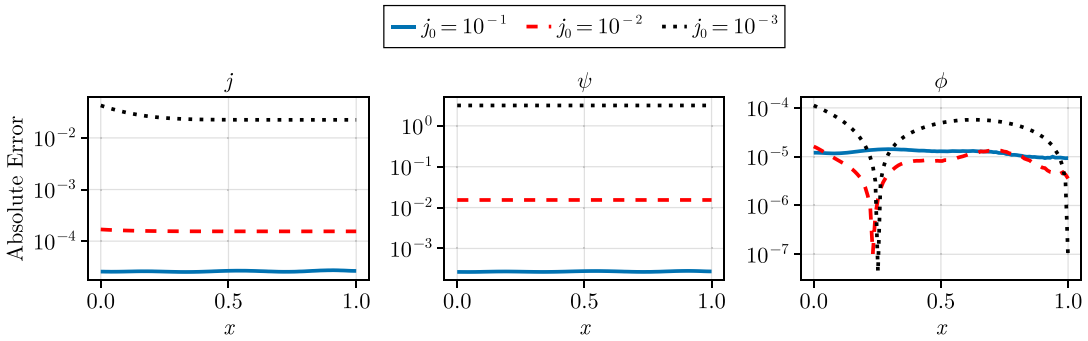

Fig. 5. PINN results for the toy model (9) with $j_0 \in \{10^{-1}, 10^{-2}, 10^{-3}\}$. The vertical axes represent absolute error.

- **Neumann boundary conditions:** The relevant boundary conditions are Neumann conditions so that a constant shift does not change the boundary loss.

An inaccuracy in ψ_p can distort other state variables and degrade the solution's overall accuracy, yet the PINN loss function may fail to detect it. In other words, even if the PINN loss is minimized to a very small value, the approximate solutions may still deviate substantially from the true solution. This discrepancy between loss and accuracy complicates the application of PINNs to the P2D model, underscoring the need for additional strategies to mitigate it.

In addition, we present the following toy example to illustrate the practical impact of how the small coefficient $\mathcal{H}$ affects the accuracy of PINN approaches in a simpler setting:

$$\begin{cases} \phi_{xx} &= j, \quad x \in (0,1), \\ \phi_x &= -10^{-4}, \quad x = 0, \\ \phi &= 0, \quad x = 1, \end{cases} \quad \text{and} \quad \begin{cases} \psi_{xx} + f &= j, \quad x \in (0,1), \\ \psi_x &= 0, \quad x \in \{0,1\}, \end{cases} \tag{9}$$

where the source term j is given by

$$j = j_0\left(e^{\psi-\phi+U} - 1\right), \qquad f(x) = (10^{-3} - 10^{-4})(1-x)^8 + 6x - 3,$$
$$U(x) = x^2(x-1.5) + 4 - 10^{-5}(1-x)^{10} + \log\left(j_0^{-1}(10^{-3} - 10^{-4})(1-x)^8 + 1\right),$$

and j_0 is a parameter to be chosen. The exact solutions for ϕ and ψ are:

$$\phi(x) = 10^{-5}(1-x)^{10}, \quad \psi(x) = -x^2(x-1.5) - 4.$$

Here, j_0 plays a role similar to $\mathcal{H}$ in the original P2D model.

Let us now consider a shifted function $\widetilde{\psi} = \psi + C$. Using a Taylor expansion, we have:

$$e^{\widetilde{\psi}-\phi+U} - 1 \;\approx\; e^{\psi-\phi+U} - 1 + C\, e^{\psi-\phi+U} \;=\; \frac{j}{j_0} + C\, e^{\psi-\phi+U}.$$

Let $\widetilde{j} = j_0\left(e^{\widetilde{\psi}-\phi+U} - 1\right)$. Then

$$\widetilde{j} \approx j + C\, j_0\, e^{\psi-\phi+U}.$$

Since

$$e^{\psi-\phi+U} \;=\; j_0^{-1}(10^{-3} - 10^{-4})(1-x)^8 + 1 \;\le\; j_0^{-1}(10^{-3} - 10^{-4}) + 1,$$

the shifted function $\widetilde{\psi}$ alters the source term j by approximately $j_0 C + 10^{-3} C$. Hence, the PDE residual of $\widetilde{\psi}$ becomes

$$|\widetilde{\psi}_{xx} + f - \widetilde{j}| = |j - \widetilde{j}| \quad (\text{since } \widetilde{\psi}_{xx} = \psi_{xx}),$$
$$= O\left(j_0 C + 10^{-3} C\right),$$

while the boundary loss remains zero (the boundary conditions for ψ are Neumann). According to this observation, when $j_0 = 10^{-1}$, the PDE residual loss from $\widetilde{\psi}$ is on the order of $O(10^{-2})$. However, if j_0 is much smaller (e.g., 10^{-3}), the PDE residual would be below $O(10^{-6})$. In typical PINN training, it is generally infeasible to reduce the loss function exactly to zero; the loss typically converges to about 10^{-4} or 10^{-5}. Therefore, if the PINN loss by $\widetilde{\psi}$ is smaller than the achieved loss threshold, the PINN may be unable to distinguish between the true solution ψ and the shifted function $\widetilde{\psi}$. In fact, the experimental results in Fig. 5 confirm our observation. Specifically, for a relatively large $j_0 = 10^{-1}$, the PINN provides the correct solution easily, but for a smaller $j_0 = 10^{-3}$, the trained solution is inaccurate. This confirms that the smaller j_0 is, the more PINNs struggle to distinguish between the true solution and a shifted one.

This toy example illustrates a key challenge related to the smallness of $\mathcal{H}$ of the BV equation, which hinder the PINN's ability to accurately capture the true solution.

Remark 1. It is worth highlighting the study by Hassanaly et al. [36], which encountered similar issues when training PINNs for the single-particle model (SPM) under a nonlinear BV equation. Their solution involved a two-step training approach—first training on a linearized BV equation, then transferring the learned parameters to the nonlinear model for initialization. While innovative, that method introduces extra hyperparameters requiring careful tuning, and their results are not extensively detailed. In contrast, we focus on developing an end-to-end training strategy that avoids these additional complexities, offering a simpler and more robust alternative for handling the nonlinear BV equation in the P2D model.

5. Strategies for the nonlinear Butler–Volmer equation

As discussed in the previous sections, the highly nonlinear nature of the BV equation can cause significant difficulties in applying PINNs to solve the P2D model. In this section, we propose two strategies aimed at mitigating these challenges: the introduction of a bypassing term to stabilize the training process and the addition of secondary conservation laws to improve the accuracy of solutions.

5.1. Bypassing term

The instability caused by the exponential growth of the sinh term in the BV equation, particularly when applied to PINNs, can be attributed to the sensitivity of the overpotential η to changes in the network parameters. In fact, the PINN approach presented in Section 3 approximates the overpotential $\widetilde{\eta}$ using multiple approximate solutions $\widetilde{\psi}$, $\widetilde{\phi}$, $\widetilde{c}_s$, according to the definition of overpotential $\psi - \phi - U(c_{s,\mathrm{surf}}/c_{s,\max})$. In this way, $\widetilde{\eta}$ depends on several networks, making it not straightforward to initialize $\widetilde{\eta}$ at a small value. Consequently, the approximate ionic flux $\widetilde{j}_i$ become highly sensitive to errors in any of the associated approximations, which results in an instable training. In addition, because each network is also involved in different types of losses terms, $\widetilde{\eta}$ can be affected by various parts of the overall loss function, leading to difficulties in maintaining a proper approximation. For this, we propose a novel approach to reduce the sensitivity of η during the optimization process, making it possible to handle the fully non-linear BV equation.

Specifically, we introduce a neural network, β_θ, which replaces the entire term inside the hyperbolic sine function in the BV equation. In this approach, the approximate reaction rate $\widetilde{j}_i$ in the PINN loss is computed as

$$\widetilde{j}_i = 2k c_{s,i,\max} c_{\mathrm{ref}}^{0.5} \left(1 - \widetilde{c}_{s,i,\mathrm{surf}}\right)^{0.5} \widetilde{c}_{s,i,\mathrm{surf}}^{0.5} \widetilde{c}_i^{0.5} \sinh\left(\beta_\theta\right) \tag{10}$$

instead of the formulation in (3). Then, an additional penalty term $\mathcal{L}_\beta$ is incorporated into the PINN loss to ensure β_θ accurately approximates $(F/2RT)\eta$:

$$\mathcal{L}_\beta := \int_\Omega \left(\beta_\theta - \frac{F}{2RT}\widetilde{\eta}\right)^2 dx. \tag{11}$$

We refer to this strategy as *bypassing*. By assigning a dedicated single network to approximate the overpotential η, we isolate the approximation inside the hyperbolic sine function from the other networks and prevent large interference among several types of loss terms. Furthermore, this design facilitates initializing β_θ at an $O(1)$ scale, which helps stabilize the early stages of training. Altogether, these mitigate the risk of exploding loss values.

In [37], Wang et al. observed that an imbalance between the gradients of the PDE loss ($\nabla_\theta \mathcal{L}_{\mathrm{PDE}}$) and the boundary condition loss ($\nabla_\theta \mathcal{L}_{\mathrm{BC}}$) can lead to a wide range of eigenvalues in the Hessian matrix, resulting in ill-conditioned training dynamics. This emphasizes the importance of maintaining a balanced gradient distribution for effective training. From this point of view, we can verify the effectiveness of the bypassing strategy. By conducting an eigenvalue analysis, we observed a significant reduction in the condition number after introducing the bypassing term. Although memory constraints required us to perform this analysis with a smaller neural network compared to the one used for the full P2D model in Section 6.1, the results still sufficiently demonstrate the effectiveness of the proposed strategy. The results are visualized in Fig. 6.

Remark 2. Our bypassing strategy is inspired by observations from previous studies. Han et al. [14] successfully solved the P2D model using a finite difference method with backward Euler time stepping. Notably, their approach involved treating η and j as independent unknowns, despite these quantities being defined by algebraic equations. We hypothesize that this choice was made to address stability issues inherent to the BV equation. Building on their insight, we adopted a similar concept by introducing a neural network, β_θ, to approximate a scaled version of η.

5.2. Secondary conservation laws

As mentioned in Section 4, due to the small scaling factor $\mathcal{H}$ in the BV term (7), the conventional PINN loss struggles to capture the true battery state, particularly for ψ_p. A function that is quite far from the true solution can still result in a very small loss, and further reducing this loss would require significant computational resources because the corresponding loss value is already small. In other words, minimizing the PINN loss presented in Section 3 is insufficient for accurately approximating the true solution.

To mitigate this discrepancy between loss and accuracy, we introduce additional constraints based on secondary conservation laws to more accurately approximate the reaction rates $\widetilde{j}_i$, which, in turn, guide the approximate solutions toward the true solution. Specifically, we enforce the following secondary conservation laws:

$$\frac{La_p F}{I_{\mathrm{app}}} \int_0^{\hat{L}_p} j_p \, dx = 1, \quad \text{and} \quad \frac{La_n F}{I_{\mathrm{app}}} \int_{\hat{L}_p + \hat{L}_s}^1 j_n \, dx = -1, \tag{12}$$

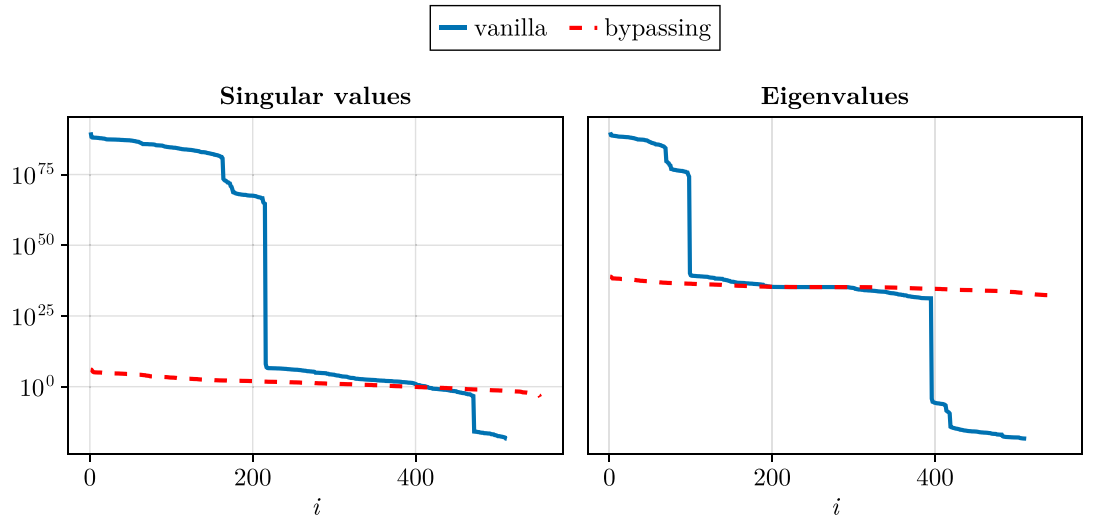

Fig. 6. Distribution of singular values and eigenvalues of the Hessian matrix. The solid line represents the distribution without the bypassing term, while the dashed line represents the distribution with the bypassing term. The introduction of the bypassing term significantly narrows the spectrum, indicating improved conditioning. The condition numbers were approximately $2.017 \cdot 10^{108}$ without the bypassing term and $4.599 \cdot 10^{9}$ with the bypassing term.

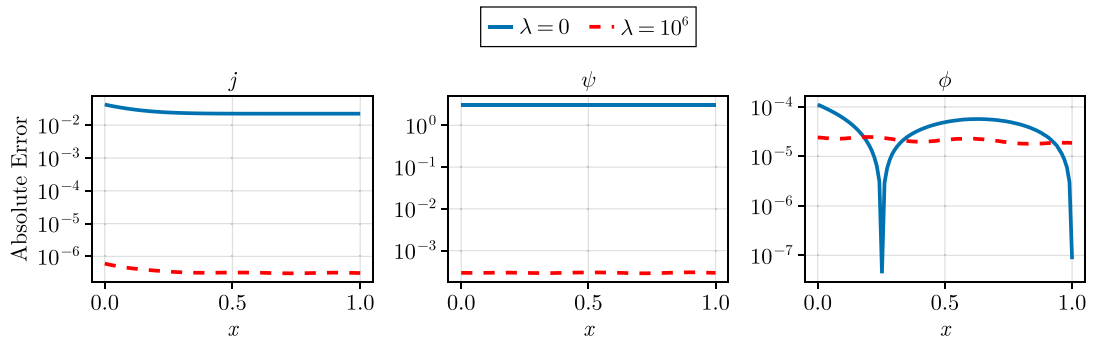

Fig. 7. Results for $j_0 = 10^{-3}$ after adding $\lambda L_{SC}(\theta)$ to the PINN loss. Solid lines represent the baseline, whereas dotted lines show the result of adding the secondary conservation law. For j and ψ, the error magnitudes are reduced by several orders of magnitude.

which are derived by integrating 3.2. These constraints help ensure the correct computation of the reaction rates, which in turn guides the approximation of $\widetilde{\psi}_p$ and other key variables in the P2D model. Our approach, which incorporates the bypassing terms and secondary conservation laws, is illustrated in Fig. 8.

For the toy example presented in Section 4, we can see that this strategy helps prevent the network from converging to a shifted or incorrect solution. For the toy example, the corresponding secondary conservation can be expressed as:

$$\int_0^1 j \, dx = 10^{-4}. \tag{13}$$

As previously discussed, using a typical PINN loss alone is not sufficient for solving the toy example effectively, especially when dealing with the small factor j_0. However, by incorporating the additional constraints (13) into the PINN loss function, the PINNs can predict the solution more accurately, as shown in Fig. 7.

As demonstrated in the toy example, by combining secondary conservation laws with the PINN framework, we can mitigate the challenges posed by the BV equation and improve the network's ability to capture the true solution with greater accuracy, despite the computational challenges.

5.3. Ablation study

We performed an ablation study to evaluate the individual and combined effectiveness of the methods proposed in this section. The results are summarized in Table 2.

The Baseline configuration represents the simplest implementation of the PINN framework presented in Section 3, without both the bypassing term and the secondary conservation law. The B configuration includes the bypassing term to address the stiffness

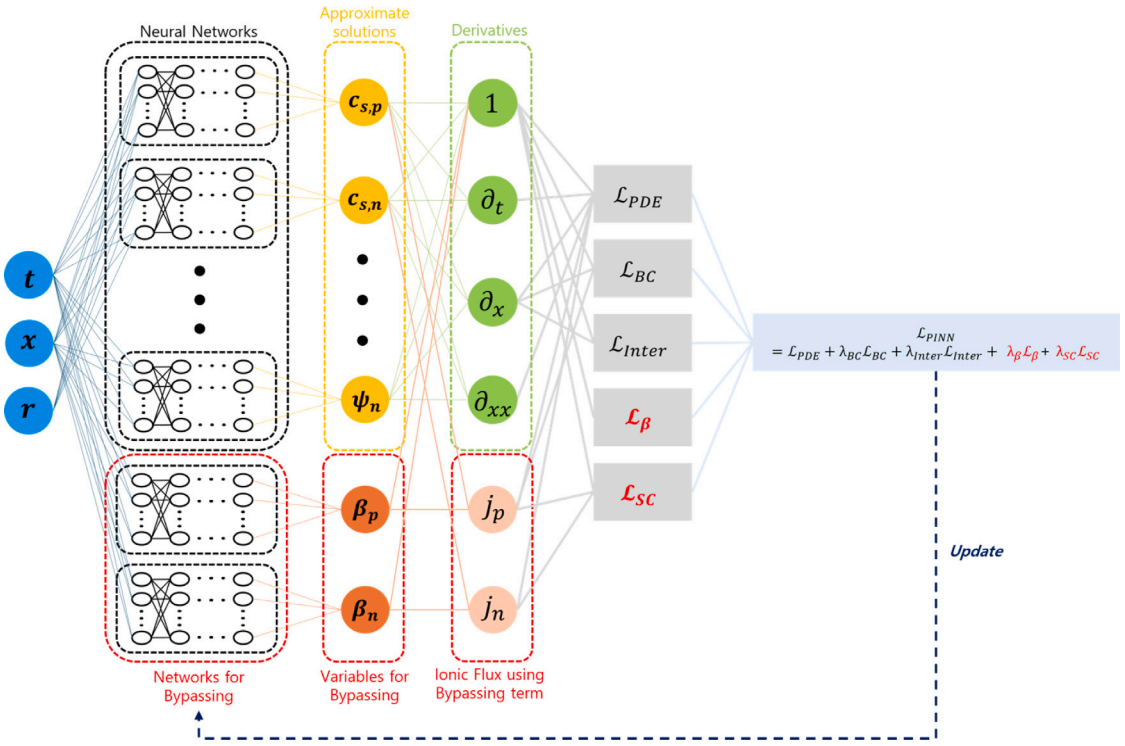

Fig. 8. A schematic diagram of our proposed PINN architecture for the P2D model of a Li-ion battery cell.

Table 2
Ablation study results showing relative L^2 errors. B: Using the bypassing term to approximate the stiffest component of the BV equation. SC: Using the secondary conservation law to enhance solution accuracy by preventing convergence to trivial solutions. B + SC: Using both strategies simultaneously.

Methods	ψ	c_s	ϕ	c	Time (s)	Final loss
Baseline	8.69e−1	7.60e−1	5.18e−1	1.89e−1	31049	1.63e+48
B	2.24e−1	8.96e−1	8.88e−1	2.11e−1	**4729**	7.32e−8
SC	8.69e−1	7.61e−1	5.17e−1	1.89e−1	50578	1.63e+48
B + SC	**4.84e−3**	**1.31e−2**	**2.31e−2**	**6.51e−3**	6352	9.15e−6

of the BV equation, whereas the SC configuration incorporates the secondary conservation law into the PINN loss to mitigate the discrepancy between loss and accuracy.

Necessity of bypassing terms

The Baseline and SC configurations both exhibit extremely large losses (on the order of 10^{48}) and require over 8 h of training, as neither addresses the high sensitivity of the BV equation. This underscores the need to mitigate the ill-conditioning posed by the nonlinear BV term. By employing the bypassing method, we successfully reduce the PINN loss to 7.32×10^{-8}, indicating that the instability by the BV equation has been effectively resolved. The bypassing method remains effective when combined with the secondary conservation law, achieving a still low loss value of 9.15×10^{-6}.

Necessity of secondary conservation laws

Although the bypassing term ensures faster training and a small loss, the accuracy achieved with the bypassing method alone is still insufficient. In other words, achieving a small value for the PINN loss (4) with bypassing loss (11) does not guarantee accuracy. As noted in Section 5.2, the secondary conservation laws are required to resolve this discrepancy. Indeed, once a sufficiently small loss is achieved through the bypassing method, the secondary conservation plays an important role in resolving the discrepancy between loss and accuracy, improving accuracy to the order of 10^{-2} or lower.

Overall, our ablation study shows that neither method is redundant; both the bypassing term and the secondary conservation law are essential for an efficient and accurate simulation of the P2D model. This result underscores the robustness of our proposed framework in overcoming the challenges posed by nonlinear BV equations in PINNs.

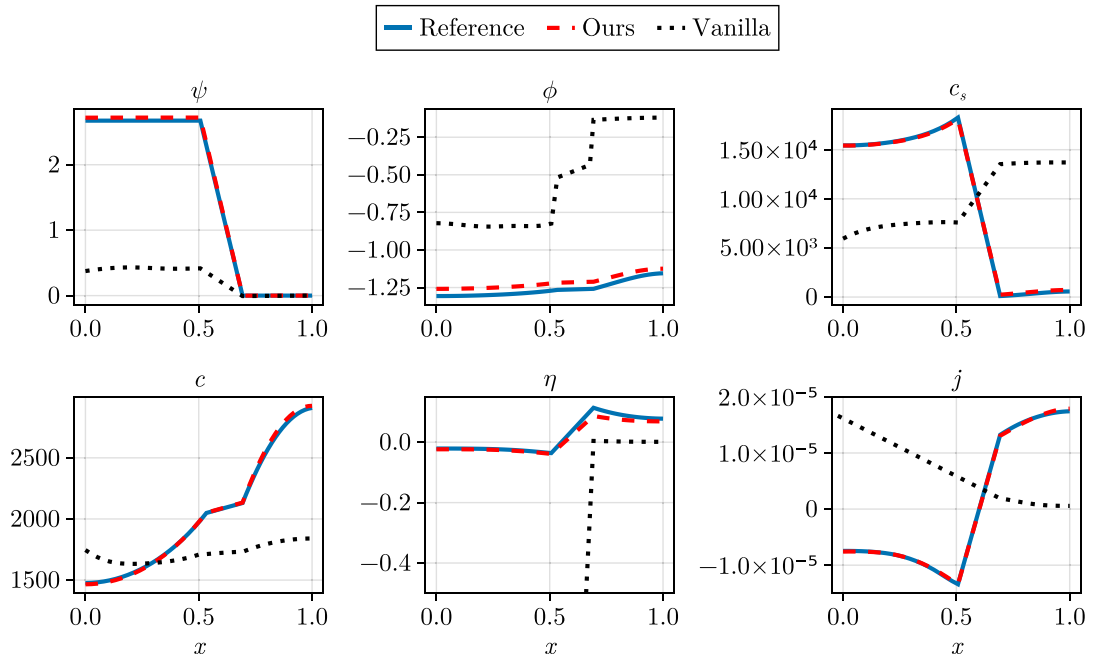

Fig. 9. Battery state variables over the spatial domain, evaluated at $t = 3500$. Our proposed method closely matches the reference solution, whereas vanilla PINNs fail to provide accurate predictions.

6. Numerical experiments

We performed numerical experiments for both forward and inverse problems of the P2D model to evaluate the effectiveness of our proposed methods. In our numerical experiments, we used a neural network architecture with a width of 32 and a depth of 4. The optimization process followed a two-step approach, combining ADAM [38] and L-BFGS [39] optimizers, implemented in the Optax package [40] and the JAXopt package [41], respectively. Collocation points were dynamically sampled for ADAM but kept fixed during L-BFGS, using 2^{14} points at each step. The ADAM optimizer employed a cosine decay learning rate schedule, starting from 10^{-4}, and the L-BFGS history size was set to 50. Our network utilized the sine activation function, initialized with the SIREN scheme [42]. For reference solution, we used $\Delta t = 1$ for time-stepping up to the final time $\tau = 3500$, and $N_x = 50$ for spatial discretization. Computations were carried out on an Nvidia A100 GPU, and the framework JAX [43] was used for automatic differentiation, while visualizations were created using Makie.jl [44].

6.1. Forward problem

We present the PINN simulation results for the forward problem of the P2D model, demonstrating the effectiveness of our proposed methods in addressing the challenges posed by the fully nonlinear BV equation (1). The simulations were performed using 100k iterations of the Adam optimizer, followed by 100k iterations of the L-BFGS optimizer. Figs. 9 and 10 illustrate the spatial distribution and temporal evolution of key battery state variables, evaluated at $t = 3500$ and $x = 0$, respectively. Unlike vanilla PINNs, which fail to produce meaningful predictions and deviate significantly from the reference solution, our proposed approach delivers accurate results. Appendix includes results for additional experiments with different physical parameters. This close agreement between our PINN solution and the reference solution not only validates the robustness of our computational strategies but also highlights their predictive accuracy over long time horizons.

The learning curves for the PINN optimization are shown in Fig. 11. The left panel illustrates the convergence behavior of the Adam optimizer, while the right panel depicts the results for L-BFGS. Our approach achieves a relative error on the order of 10^{-2}, demonstrating sublinear convergence, which is characteristic of PINN optimization problems. To the best of our knowledge, this is the first successful PINN simulation of a Li-ion battery incorporating the fully nonlinear BV equation (1). The ability of our method to accurately predict battery behavior underscores the high predictive power and reliability of our computational strategy. These results provide a strong foundation for future research into advanced deep-learning methods for complex battery simulations.

6.2. Inverse problem

One advantage of the PINN framework is its ability to estimate model parameters with minimal implementation modifications. Given observational data $\{u(x_i)\}_{i=1}^{N}$, we incorporate a data fitting term into the PINN loss $\mathcal{L}_{\text{PINN}}$ (2) and treat the unknown parameters

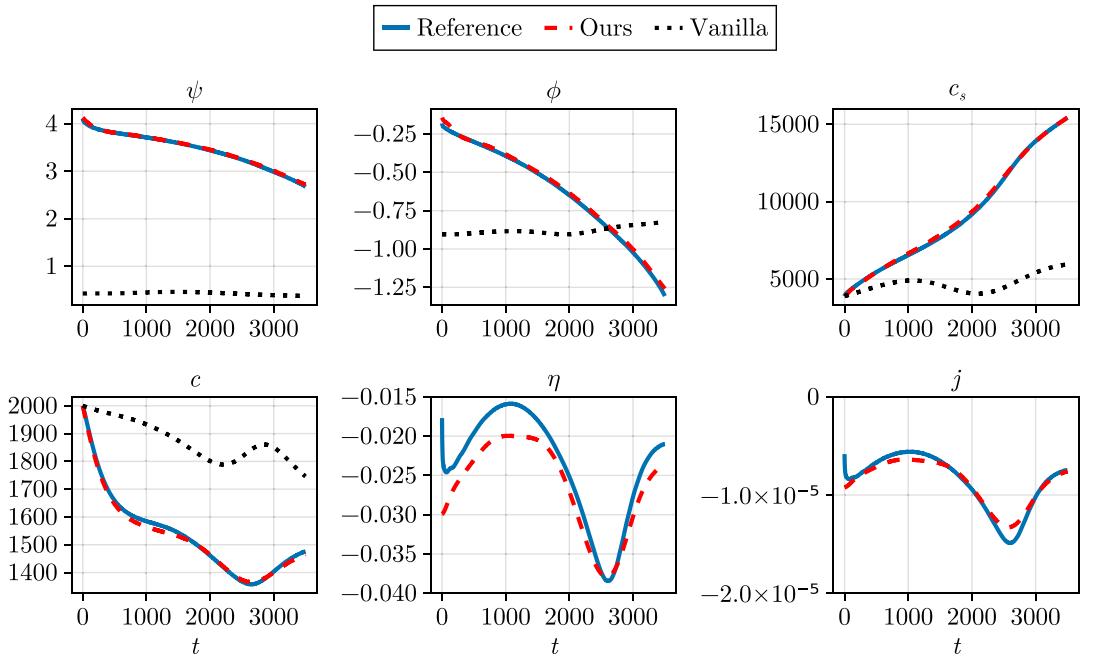

Fig. 10. Battery state variables over time, evaluated at the left end of the battery. Our method demonstrates excellent agreement with the reference solution.

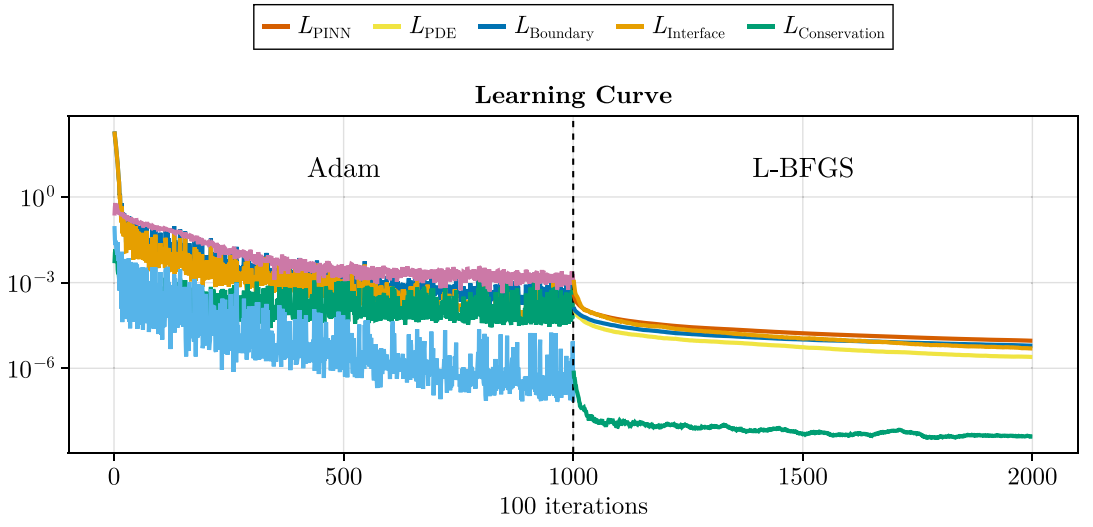

Fig. 11. Learning curves. The left panel shows the loss and error trajectories during optimization with the Adam optimizer, while the right panel shows the corresponding trajectories for the L-BFGS optimizer. Please refer to the online version for colored curves.

α as learnable parameters. This leads to the following loss function for the inverse problem:

$$\mathcal{L}(\theta,\alpha)=\int_{\Omega}\left|\mathcal{N}_{\alpha}[u_{\theta}](x)-f(x)\right|_2^2\,dx+\lambda\int_{\partial\Omega}\left|\mathcal{B}_{\alpha}[u_{\theta}](s)-g(s)\right|_2^2\,d\sigma(s)+\lambda_{\text{data}}\sum_{i=1}^{N}\left(u_{\theta}(x_i)-u(x_i)\right)^2.$$

The minimizer $(u_{\theta^\star},\alpha^\star)$ of this loss function will yield a solution $(u_{\theta^\star},\alpha^\star)$ that satisfies the governing PDE and boundary conditions, while also fitting the observed data closely, thus providing a solution to the inverse problem. With a more reliable method for solving the forward problem now in place, we are prepared to address inverse problems using the reference solution as our data. Our focus is on inverse problems related to battery length, which is considered a crucial factor in the performance of Li-ion batteries [45]. In the following, we perform experiments on three inverse tasks: estimating the battery length (Table 3), the length ratio (Table 4), and both battery length and length ratio simultaneously (Table 5). These tasks are evaluated under two scenarios: the first uses noise-free observational data, while the second introduces 5% Gaussian noise into the observational data. Our proposed

Table 3
Relative errors (%) for the total length (L) estimation problem.

Method	w/o Noise	w/ Noise
Baseline	69.53	69.53
B	44.71	44.85
SC	69.59	69.59
B + SC	**1.51**	**1.58**

Table 4
Relative errors (%) for the length ratio (L_p, L_n) estimation problem.

Method	w/o Noise		w/ Noise	
	L_p	L_n	L_p	L_n
Baseline	26.14	4.65	26.14	4.68
B	**2.58**	**4.33**	**2.71**	**4.30**
SC	17.10	10.01	17.10	10.23
B + SC	3.21	4.64	3.10	4.81

Table 5
Relative errors (%) for the total length and length ratio (L, L_p, L_n) estimation problem.

Method	w/o Noise			w/ Noise		
	L	L_p	L_n	L	L_p	L_n
Baseline	69.66	77.67	40.01	69.66	77.67	39.98
B	52.01	10.86	13.45	52.19	10.74	13.10
SC	69.51	77.57	39.16	69.51	77.57	39.13
B + SC	**8.85**	**7.23**	**6.26**	**8.89**	**7.39**	**6.19**

methodology demonstrates strong predictive performance across all scenarios. Notably, even in the presence of noisy observational data, the predictions remain comparable to those obtained with clean data. This robustness to noise underscores the reliability and effectiveness of our approach in solving inverse problems, even under challenging conditions.

Battery length The first example involves estimating the total battery length, L, while maintaining a fixed ratio of $L_p : L_s : L_n = 1.74 : 0.52 : 1.0$. Although this problem involves only a single parameter, it is nontrivial due to its impact on 11 coefficients in the P2D model after non-dimensionalization, particularly those associated with spatial partial derivatives along x. To parameterize L, we expressed it as $\hat{L} \cdot 10^{-4}$ with an initial value of $\hat{L} = 1$. After 150,000 iterations of the Adam optimizer, we obtained $\hat{L} = 3.21$, corresponding to a relative error of 1.51% compared to the true value $L = 3.26$. The entire computation was completed in approximately 30 min.

Length ratio The second example focuses on estimating the ratio of the battery section lengths, $L_p : L_s : L_n$, while keeping the total battery length fixed at $L = 3.26 \cdot 10^{-4}$. This task introduces additional complexity due to the dynamically changing domain lengths during optimization, which limits the effectiveness of the L-BFGS optimizer. To simplify the parameterization, we used two parameters, ρ_p and ρ_n, since the constraint $\rho_p + \rho_s + \rho_n = 1$ reduces the problem to two degrees of freedom. The initial values were set to $\hat{\rho}_p = \hat{\rho}_n = 0.4$. The B + SC method estimated $\hat{L}_p = \hat{\rho}_p \cdot L = 1.68 \cdot 10^{-4}$ and $\hat{L}_n = \hat{\rho}_n \cdot L = 0.9536 \cdot 10^{-4}$, achieving relative errors of 3.21% and 4.64% for the positive and negative electrode lengths, respectively. The computation was completed in approximately 40 min.

Battery length and length ratio Finally, we extended the problem to simultaneously estimate both the battery length and the length ratio. This combined task represents a significantly more challenging problem due to the increased dimensionality and the inherent coupling between the two parameters. Accurate estimation in such scenarios requires the framework to handle dynamic changes in both domain lengths and parameter dependencies throughout the optimization process. Starting with the same initialization for $\hat{L}, \hat{\rho}_p$, and $\hat{\rho}_n$, the B + SC method estimated $\hat{L} = 2.971$, $\hat{L}_p = 1.614 \cdot 10^{-4}$, and $\hat{L}_n = 0.9374 \cdot 10^{-4}$. The relative errors for these estimates were $8.85 \cdot 10^{-2}$, $7.23 \cdot 10^{-2}$, and $6.26 \cdot 10^{-2}$, respectively.

The results throughout 3 different inverse problems demonstrate the robustness of the B + SC method in tackling complex coupled inverse problems.

6.3. A note on implementation

The P2D model tightly interconnects battery state variables. For instance, calculating the ionic flux j necessitates computing β, c, and $c_{s,\text{surf}}$ for the BV equation. While frameworks like JAX-PI [37] offer readable implementations, they can be computationally inefficient due to redundant calculations. As shown in Fig. 12(a), a JAX-PI-style implementation repeatedly computes j, hindering its reuse for the potential and surface concentration PDEs. To optimize computational efficiency, we adopt a strategy that prioritizes minimizing network forward passes. By initially calculating all required quantities collectively and then assembling the PDE residuals, we significantly reduce redundant computations. This approach, illustrated in Fig. 12(b), is expected to yield significant

```python
import jax

def psi_pde(p2d, net, t, x):
    psi_xx = jax.jacfwd(jax.jacfwd(net.psi, 1), 1)(t, x)
    cs_surf = net.cs(t, 1.0, x)
    c = net.c(t, x)
    beta = net.beta(t, x)
    j = p2d.compute_j(beta, cs_surf, c)
    return psi_xx - (p2d.L**2 * p2d.a * F * j)
...
```

(a) Implementation like JAX-PI.

```python
def compute_quantities(net, t, x):
    psi_xx = jax.jacfwd(jax.jacfwd(net.psi, 1), 1)(t, x)
    cs_surf = net.cs(t, 1.0, x)
    c = net.c(t, x)
    beta = net.beta(t, x)
    ...
    return {"psi_xx":psi_xx, "cs_surf":cs_surf, "c":c, "beta":beta, ...}

def compute_psi_pde(p2d, psi_xx, j):
    return psi_xx - (p2d.L**2 * p2d.a * F * j)
...

def compute_pdes(p2d, quantities):
    j = p2d.compute_j(quantities["beta"], quantities["cs_surf"], quantities["c"])
    psi_pde = compute_psi_pde(quantities["psi_xx"], j)
    cs_pde = compute_cs_pde(quantities["cs_rr"], quantities["cs_r"], j)
    ...
    return pdes
```

(b) Our implementation

Fig. 12. Python pseudo-code.

speedups. Our implementation demonstrates this, achieving roughly 1.5 times the performance of a naive approach (14 iterations per second compared to 9).

7. Discussion and conclusion

This study has addressed the challenges of simulating Li-ion batteries using the P2D model in combination with PINNs and has proposed effective strategies to overcome these obstacles. Our methods enable reliable and accurate simulations of the P2D model, incorporating the fully nonlinear BV equation. By developing an enhanced forward solver, we successfully tackled inverse problems related to battery section lengths, achieving remarkable accuracy and computational efficiency. Despite the inherent complexity of these problems, our framework delivers results that hold significant engineering relevance for the design and optimization of Li-ion batteries. These achievements highlight the practical applicability of our approach and its ability to solve inverse problems that might not be infeasible for traditional numerical methods. However, there is still room for further development. For instance, extending our framework to incorporate multiphysics phenomena such as thermal effects or aging, which are currently excluded from our equations, presents an exciting and challenging avenue for future research. Additionally, building on our forward problem solver to develop an operator network capable of real-time predictions is another promising direction.

CRediT authorship contribution statement

Myeong-Su Lee: Writing – original draft, Methodology, Conceptualization. **Jaemin Oh:** Writing – original draft, Software, Methodology, Conceptualization. **Dong-Chan Lee:** Data curation. **KangWook Lee:** Data curation. **Sooncheol Park:** Conceptualization. **Youngjoon Hong:** Writing – original draft, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table A.6
Test results (relative L^2 errors) for different discharge rates.

Discharge rate	ψ	c_s	ϕ	c
1C	4.84e−3	1.31e−2	2.31e−2	6.51e−3
0.5C	4.87e−3	2.43e−2	1.28e−2	3.58e−3
2C	1.65e−2	4.69e−2	7.15e−2	6.02e−3

Acknowledgments

The work of M.-S. Lee was supported by Basic Science Research Programs through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2023-00244475 and RS-2024-00462755). The work of Y. Hong was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2021R1A2C1093579), the Korean government (MSIT) (RS-2023-00219980), and Hyundai Motor Company. This work was supported by the National Supercomputing Center with supercomputing resources including technical support (KSC-2023-CRE-0334).

Appendix. Additional experiments

In addition to the original 1C discharge rate which corresponds to $\tau = 3500$ and $I_{\text{app}} = -17.5$, we test our methods to two different discharge rates, 0.5C and 2C. Specifically, 0.5C discharge rate is $\tau = 7000$ and $I_{\text{app}} = -8.75$, and 2C discharge rate corresponds to $\tau = 1550$ and $I_{\text{app}} = -35$. Table A.6 illustrates our method's ability to simulate different discharge rates.

Data availability

No data was used for the research described in the article.

References

[1] G. Zubi, R. Dufo-López, M. Carvalho, G. Pasaoglu, The lithium-ion battery: State of the art and future perspectives, Renew. Sustain. Energy Rev. 89 (2018) 292–308.
[2] T.M. Bandhauer, S. Garimella, T.F. Fuller, A critical review of thermal issues in lithium-ion batteries, J. Electrochem. Soc. 158 (3) (2011) R1.
[3] V. Etacheri, R. Marom, R. Elazari, G. Salitra, D. Aurbach, Challenges in the development of advanced li-ion batteries: A review, Energy Environ. Sci. 4 (9) (2011) 3243–3262.
[4] C.M. Costa, J.C. Barbosa, R. Gonçalves, H. Castro, F.J. Del Campo, S. Lanceros-Méndez, Recycling and environmental issues of lithium-ion batteries: Advances, challenges and opportunities, Energy Storage Mater. 37 (2021) 433–465.
[5] R. Xiong, L. Li, J. Tian, Towards a smarter battery management system: A critical review on battery state of health monitoring methods, J. Power Sources 405 (2018) 18–29.
[6] Y. Wang, J. Tian, Z. Sun, L. Wang, R. Xu, M. Li, Z. Chen, A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems, Renew. Sustain. Energy Rev. 131 (2020) 110015.
[7] M. Nikdel, et al., Various battery models for various simulation studies and applications, Renew. Sustain. Energy Rev. 32 (2014) 477–485.
[8] G.L. Plett, Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 3. State and parameter estimation, J. Power Sources 134 (2) (2004) 277–292.
[9] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, D.U. Sauer, Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation, J. Energy Storage 30 (2020) 101557.
[10] M. Doyle, T.F. Fuller, J. Newman, Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell, J. Electrochem. Soc. 140 (6) (1993) 1526.
[11] M. Doyle, J. Newman, A.S. Gozdz, C.N. Schmutz, J.-M. Tarascon, Comparison of modeling predictions with experimental data from plastic lithium ion cells, J. Electrochem. Soc. 143 (6) (1996) 1890.
[12] K. Kumaresan, G. Sikha, R.E. White, Thermal model for a li-ion cell, J. Electrochem. Soc. 155 (2) (2007) A164.
[13] L. Cai, R.E. White, Mathematical modeling of a lithium ion battery with thermal effects in COMSOL Inc. Multiphysics (MP) software, J. Power Sources 196 (14) (2011) 5985–5989.
[14] R. Han, C. Macdonald, B. Wetton, A fast solver for the pseudo-two-dimensional model of lithium-ion batteries, 2021, arXiv preprint arXiv:2111.09251.
[15] J. Mora-Paz, High-order transient multidimensional simulation of a thermo-electro-chemo-mechanical model for lithium-ion batteries, 2024, arXiv preprint arXiv:2403.16928.
[16] M. Torchio, L. Magni, R.B. Gopaluni, R.D. Braatz, D.M. Raimondo, Lionsimba: A matlab framework based on a finite volume model suitable for Li-ion battery design, simulation, and control, J. Electrochem. Soc. 163 (7) (2016) A1192.
[17] V. Sulzer, S.G. Marquis, R. Timms, M. Robinson, S.J. Chapman, Python battery mathematical modelling (PyBaMM), J. Open Res. Softw. 9 (1) (2021).
[18] M.D. Berliner, D.A. Cogswell, M.Z. Bazant, R.D. Braatz, Methods—PETLION: Open-source software for millisecond-scale porous electrode theory-based lithium-ion battery simulations, J. Electrochem. Soc. 168 (9) (2021) 090504.
[19] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[20] Q. He, D. Barajas-Solano, G. Tartakovsky, A.M. Tartakovsky, Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport, Adv. Water Resour. 141 (2020) 103610.
[21] H. Jo, H. Hong, H.J. Hwang, W. Chang, J.K. Kim, Density physics-informed neural networks reveal sources of cell heterogeneity in signal transduction, Patterns 5 (2) (2024).
[22] H. Son, M. Lee, A PINN approach for identifying governing parameters of noisy thermoacoustic systems, J. Fluid Mech. 984 (2024) A21.

[23] J. Ostanek, M. Parhizi, J. Jeevarajan, A novel method for alleviating numerical stiffness in li-ion thermal abuse models, J. Power Sources Adv. 23 (2023) 100123.
[24] C. Xue, B. Jiang, J. Zhu, X. Wei, H. Dai, An enhanced single-particle model using a physics-informed neural network considering electrolyte dynamics for lithium-ion batteries, Batteries 9 (10) (2023) 511.
[25] M. Hassanaly, P.J. Weddle, R.N. King, S. De, A. Doostan, C.R. Randall, E.J. Dufek, A.M. Colclasure, K. Smith, PINN surrogate of li-ion battery models for parameter inference. Part II: Regularization and application of the pseudo-2D model, 2023, arXiv preprint arXiv:2312.17336.
[26] K. Zubov, Z. McCarthy, Y. Ma, F. Calisto, V. Pagliarino, S. Azeglio, L. Bottero, E. Luján, V. Sulzer, A. Bharambe, et al., NeuralPDE: Automating physics-informed neural networks (PINNs) with error approximations, 2021, arXiv preprint arXiv:2107.09443.
[27] W. Chen, Y. Fu, P. Stinis, Physics-informed machine learning of redox flow battery based on a two-dimensional unit cell model, J. Power Sources 584 (2023) 233548.
[28] J.I. Díaz, D. Gómez-Castro, A.M. Ramos, On the well-posedness of a multiscale mathematical model for lithium-ion batteries, Adv. Nonlinear Anal. 8 (1) (2018) 1132–1157.
[29] L.N. Trefethen, J.A.C. Weideman, The exponentially convergent trapezoidal rule, SIAM Rev. 56 (3) (2014) 385–458.
[30] G.H. Golub, J.H. Welsch, Calculation of Gauss quadrature rules, Math. Comp. 23 (106) (1969) 221–230.
[31] I.M. Sobol', On the distribution of points in a cube and the approximate evaluation of integrals, Zh. Vychisl. Mat. Mat. Fiz. 7 (4) (1967) 784–802.
[32] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[33] J.C. Wong, C.C. Ooi, A. Gupta, Y.-S. Ong, Learning in sinusoidal spaces with physics-informed neural networks, IEEE Trans. Artif. Intell. 5 (3) (2022) 985–1000.
[34] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105–B1132, http://dx.doi.org/10.1137/21M1397908.
[35] C. Wang, S. Li, D. He, L. Wang, Is L^2 physics informed loss always suitable for training physics informed neural network? Adv. Neural Inf. Process. Syst. 35 (2022) 8278–8290.
[36] M. Hassanaly, P.J. Weddle, R.N. King, S. De, A. Doostan, C.R. Randall, E.J. Dufek, A.M. Colclasure, K. Smith, PINN surrogate of li-ion battery models for parameter inference. Part I: Implementation and multi-fidelity hierarchies for the single-particle model, 2023, arXiv preprint arXiv:2312.17329.
[37] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081.
[38] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[39] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1) (1989) 503–528.
[40] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, F. Viola, The DeepMind JAX ecosystem, 2020, URL http://github.com/google-deepmind.
[41] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, J.-P. Vert, Efficient and modular implicit differentiation, 2021, arXiv preprint arXiv:2105.15183.
[42] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, Adv. Neural Inf. Process. Syst. 33 (2020) 7462–7473.
[43] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: Composable transformations of Python+NumPy programs, 2018, URL http://github.com/google/jax.
[44] S. Danisch, J. Krumbiegel, Makie.jl: Flexible high-performance data visualization for Julia, J. Open Source Softw. 6 (65) (2021) 3349, http://dx.doi.org/10.21105/joss.03349.
[45] J. Kang, Y. Jia, G. Zhu, J.V. Wang, B. Huang, Y. Fan, How electrode thicknesses influence performance of cylindrical lithium-ion batteries, J. Energy Storage 46 (2022) 103827.