

Implementation and (Inverse Modified) Error Analysis for implicitly-templated ODE-nets

Aiqing Zhu^{*}, Tom Bertalan[†], Beibei Zhu[‡], Yifa Tang[§], and Ioannis G. Kevrekidis[¶]

Abstract.

We focus on learning unknown dynamics from data using ODE-nets templated on implicit numerical initial value problem solvers. First, we perform Inverse Modified error analysis of the ODE-nets using unrolled implicit schemes for ease of interpretation. It is shown that training an ODE-net using an unrolled implicit scheme returns a close approximation of an Inverse Modified Differential Equation (IMDE). In addition, we establish a theoretical basis for hyper-parameter selection when training such ODE-nets, whereas current strategies usually treat numerical integration of ODE-nets as a black box. We thus formulate an adaptive algorithm which monitors the level of error and adapts the number of (unrolled) implicit solution iterations during the training process, so that the error of the unrolled approximation is less than the current learning loss. This helps accelerate training, while maintaining accuracy. Several numerical experiments are performed to demonstrate the advantages of the proposed algorithm compared to nonadaptive unrollings, and validate the theoretical analysis. We also note that this approach naturally allows for incorporating partially known physical terms in the equations, giving rise to what is termed “gray box” identification.

Key words. learning dynamics, deep learning, ODE-nets, implicit schemes, neural ODEs

MSC codes. 37M10, 65L06, 65L09, 65P99

1. Introduction. Discovering unknown dynamical systems from observed dynamical data is an established systems task where machine learning has been shown to be remarkably effective. Neural networks f_θ , coined “ODE-nets”, are used to parameterize the unknown governing differential equations; their parameters θ are obtained by minimizing the difference between the observed state time series and the outputs evaluated by numerically solving the ODE governed by the right-hand-side f_θ . Original publications along this line date back to the 1990s [2, 21, 42, 43]. Recently, Neural ODEs [11] substantially revisited these ideas using modern computational tools, and is being applied to more challenging tasks beyond modeling dynamical systems. Here, the adjoint reverse-time equations—introduced as a continuous-time analogue of backpropagation—are employed for computation of gradients. In addition, various related architectures have been proposed [40, 27, 51], and research interest in this direction has been growing to include coupling machine learning with prior knowledge of (some) physics of the underlying systems [6, 7, 10, 29, 30, 50, 53] (see also [42] for a discussion

^{*}LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (zaq@lsec.cc.ac.cn)

[†]Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, Maryland 21211, USA (tom@tombertalan.com)

[‡]School of Mathematics and Physics, University of Science and Technology Beijing, Beijing 100083, China (zhubeibei@lsec.cc.ac.cn)

[§]Corresponding author, LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (tyf@lsec.cc.ac.cn)

[¶]Corresponding author, Department of Chemical and Biomolecular Engineering and Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, Maryland 21211, USA (yannisk@jhu.edu)

on gray-box modeling for incorporating known physics into such learned models).

However, even assuming the best-case convergence of the optimizer and accuracy of the data, the numerical integration of the network used to fit the data can itself introduce a bias into the equations extracted. In this paper, we propose to analyse the influence of the numerical integration scheme template in such learning models.

In the last few decades, modified differential equations (MDEs) and backward error analysis [16, 18, 19, 24, 41, 46, 52] have become well-established tools for analyzing the numerical solution of evolution equations (where we produce approximate trajectories from a true ODE). The main idea of MDEs is to interpret the numerical solution as the exact solution of a perturbed differential equation expressed by a formal series. We can then analyze the MDE, which is easier than the analysis of the discrete numerical solution.

Recently, inspired by MDEs and BE, Inverse Modified Differential Equations (IMDEs) [55] have been proposed; they allow the efficient analysis of numerical schemes applied to the discovery of dynamics (*where we produce an approximate ODE from true trajectories*). By analogy with the MDE (see Figure 1), the IMDE is a perturbed differential equation whose numerical solution matches the exact observed solution (the data). It was shown in [55] that training an ODE-net returns a close approximation of the IMDE, and that some known analysis results of solving ODEs, such as order of convergence, have natural extensions to the field of discovery of dynamics. Other analysis results exist for the discovery of dynamics by

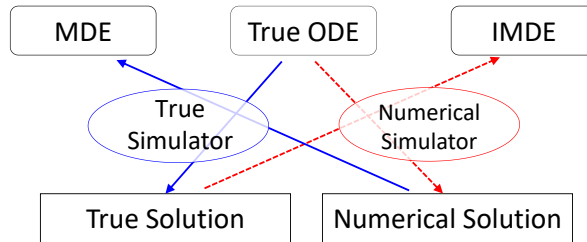


Figure 1. Schematic depiction of the relation between the true model, the Modified and the Inverse Modified Differential Equations (MDE and IMDE respectively). Forward Error Analysis studies the difference between the true and the numerical solution of the model, while Backward, and Inverse Backward Error Analysis examine the difference between the true model and the MDE/IMDE respectively.

combining numerical integrators and deep learning. In [31], a refined framework is established to derive the convergence and stability of Linear Multistep Neural Networks (LMNets) [40] via the characteristic polynomial of classic numerical linear multistep methods (LMM). In addition, an augmented loss function was introduced, based on auxiliary conditions that serve a purpose analogous to the explicit starting step used when performing forward integration with a LMM. It has been shown that the grid error of the global minimizer is bounded by the sum of the discretization error and the approximation error [15].

These analyses concentrate on LMM in LMNets, where all LMM discretization (typically implicit) can be exactly employed, and directly quantify the error between the true governing function and its neural network approximation. The existence of an associated IMDE implies uniqueness of the solution to the learning task (in a concrete sense), and also allows us to analyze the numerical error in ODE-nets. However, the results in [55] only hold when the

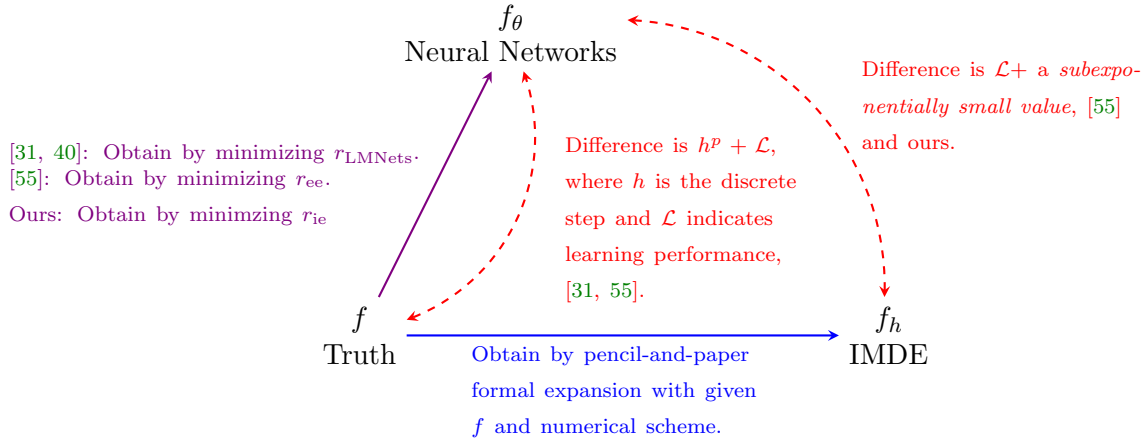


Figure 2. Relationships between different ODEs considered in the backwards analysis literature. A schematic diagram showing existing theoretical analyses and our main results with explicit Euler (ee) and implicit Euler (ie) schemes as examples. The residual of explicit Euler is $r_{ee} = \|\phi_{\Delta t}(x) - (x + \Delta t \cdot f_{\theta}(x))\|_2^2$ where $\{x, \phi_{\Delta t}(x)\}$ are the data. The residual of the LMNets approach to implicit Euler is $r_{LMNets} = \|\phi_{\Delta t}(x) - (x + \Delta t \cdot f_{\theta}(\phi_{\Delta t}(x)))\|_2^2$, given the data. The residual of our implicit Euler is $r_{ie} = \|\phi_{\Delta t}(x) - \text{argsoln}_z\{z = x + \Delta t \cdot f_{\theta}(z)\}\|_2^2$ where z is the network prediction obtained by a root-finding algorithm.

numerical integration is *exactly* evaluated, whereas the implementation of implicit integration in ODE-nets requires a root-finding algorithm, i.e. by unrolling the iterations, so as to obtain an accurate approximate solution. The mutual differences between these existing theoretical analyses and our main results are schematically visualized in Figure 2.

In this paper, we extend the analysis proposed in [55] and perform IMDE analysis for ODE-nets in which we unroll (and truncate) the iterations for solving the implicit scheme within the network architecture. To begin with, we search for a perturbed differential equation, i.e., the IMDE, such that its unrolled implicit integration matches observations of the exact solution of the true system. It is noted that this IMDE now depends on the number of unrolled stages (iteration number) of the unrolled implicit scheme. In addition, we prove that, under reasonable assumptions, training an ODE-net using an unrolled implicit scheme returns an approximation of the corresponding IMDE. As a direct consequence, increasing the iteration number results in a more accurate recovery of the IMDE. Finally, the rate of convergence of ODE-nets using unrolled implicit schemes is also presented. Several experiments are performed to validate the analysis, and the numerical results are in agreement the theoretical findings.

The numerical integration of ODE-nets is typically treated as a black box in current strategies. Here, an unrolling approach to implicit integration requires recurrent calculations; augmenting computational cost and, in particular, memory demands. Based on the analysis results, we establish a theoretical basis for hyper-parameter selection when training ODE-nets. We formulate an adaptive algorithm that monitors the level of error and adapts the iteration number in the training process to accelerate training while maintaining accuracy. In the initial stage of training, a rough approximation target, i.e., a smaller iteration number, is accurate enough for optimization. As learning loss decreases, we increase the iteration number so as

to achieve a more accurate target. Numerical experiments show that the proposed algorithm leads to a 2-3 \times speedup in training without any degradation in accuracy.

1.1. Related works. There have been extensive attempts to determine unknown dynamics using various approaches including symbolic regression [47], Gaussian processes [39], sparse regression [8], statistical learning [34], etc. Among various models, the ODE-nets [44, 38, 11, 43] have been established as powerful tools to model complicated physical phenomena from time series data, and have achieved numerous successes [2, 7, 21, 29, 40, 42, 43]. Recently, researchers have focused on leveraging a continuous-time representation to incorporate physical inductive biases such as symplectic structure [6, 22, 50], the Onsager principle [53], the GENERIC formalism [54] and time-reversal symmetry [29], to name a few, into the learning model.

The implementation of ODE-nets and their variants is inevitably linked with numerical integration. Several libraries such as *torchdiffeq*, *diffraX* and *torchdyn* have been developed to provide standardized differentiable implementations of ODE solvers. Many learning models use the Euler discretization method (e.g. [6, 22]) or higher-order explicit Runge-Kutta methods (e.g. [53]), while some models encoding symplecticity use a symplectic integrator to preserve the special Hamiltonian form (e.g., [12, 48]). The work in [35] proposed a novel stiffness regularization for ODE-nets based on the internal cost of numerical integration. The interplay between learning Neural ODEs and numerical integration is explored in [37], where so-called hypersolvers are introduced for fast inference. A comprehensive study of gradient-based learning with implicit integration was explored in previous work [2], considering unrolling as well as Pineda’s and Almeida’s Recurrent Back-Propagation [36, 1]. In this paper, we focus on the implementation of *unrolled* implicit numerical integration within ODE-nets, its numerical analysis, and the adaptation of the iteration number for the solution of the implicit problem to reduce computational cost.

Recent works [3, 5, 28, 17] proposed various versions of implicit models and demonstrated their empirical success; they directly exploit root-finding algorithms (e.g., fixed-point iteration, Newton-Raphson iteration and its variants, Broyden’s method and Anderson acceleration) to solve for the output in the forward pass. In [4] an auxiliary network was introduced, to provide both the initial value and perform iterative updates to improve inference efficiency. In [20] a novel gradient estimate was proposed, to circumvent computing the exact gradient by implicit differentiation. Although the precise formulations and motivations of these implicit models are quite different, applications of our adaptive algorithm to these implicit models is a promising avenue for future work.

2. Problem setup. Consider the dynamical system

$$(2.1) \quad \frac{d}{dt}\mathbf{y}(t) = f(\mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{x}$$

where $\mathbf{y}(t) \in \mathbb{R}^D$ is the state vector, evolving over time according to the governing function f . Let $\phi_t(\mathbf{x})$ be the exact solution and $\Phi_h(\mathbf{x})$ be the numerical solution (by some initial value problem solving algorithm) with discrete step h . In order to emphasize a specific differential equation, we will add the subscript f and denote ϕ_t as $\phi_{t,f}$ and Φ_h as $\Phi_{h,f}$.

If f and the initial state \mathbf{x} are known, the future states can be predicted by solving the equation (2.1). On the other hand, if the exact governing equation is unknown, but some trajectories are given, ODE-nets model the dynamical system by neural networks and then predict future states via the learned model.

Mathematically, an ODE-net identified right-hand-side leads to the ODE

$$(2.2) \quad \frac{d}{dt} \tilde{\mathbf{y}}(t) = f_\theta(\tilde{\mathbf{y}}(t)), \quad \tilde{\mathbf{y}}(0) = \mathbf{x},$$

where f_θ is the neural network approximating the unknown vector field f . With initial condition \mathbf{x} , an ODE-net predicts the output by solving (2.2) numerically. Given N observed trajectories $\mathbf{x}_n, \phi_{\Delta t}(\mathbf{x}_n), \dots, \phi_{M\Delta t}(\mathbf{x}_n), n = 1, \dots, N$ with time step Δt , the network parameters are determined by minimizing the loss function

$$(2.3) \quad \mathcal{L}_{exact} = \sum_{n=1}^N \sum_{m=1}^M \|\text{ODESolve}(\mathbf{x}_n, f_\theta, m\Delta t) - \phi_{m\Delta t}(\mathbf{x}_n)\|_2^2 / (m\Delta t)^2.$$

Note that variable data step Δt are also possible. $M = 1$ is the classical ‘‘teacher forcing’’; excessive M can be both computationally costly and offer limited benefits especially early in training, when the long-time predictions are especially poor. So, if the training data is in the form of long trajectories, we often divide them into smaller sub-episodes, leading to an M -step teacher forcing scheme [49]. We used $M = 1$ for all of the numerical experiments in this paper except the last, for which we used $M = 10$.

In this paper, the choice for the ODE solver consists of s compositions of a numerical scheme, i.e.,

$$\text{ODESolve}(\mathbf{x}, f_\theta, m\Delta t) = \underbrace{\Phi_{h, f_\theta} \circ \dots \circ \Phi_{h, f_\theta}}_{ms \text{ compositions}}(\mathbf{x}) = (\Phi_{h, f_\theta})^{ms}(\mathbf{x}),$$

where $h = \Delta t/s$ is the discrete step. A common choice of numerical scheme Φ_h is the Runge-Kutta method, which is formulated as

$$(2.4a) \quad \mathbf{v}_i = \mathbf{x} + h \sum_{j=1}^I a_{ij} f_\theta(\mathbf{v}_j) \quad i = 1, \dots, I$$

$$(2.4b) \quad \Phi_{h, f_\theta}(\mathbf{x}) = \mathbf{x} + h \sum_{i=1}^I b_i f_\theta(\mathbf{v}_i).$$

A Runge-Kutta method (2.4) is explicit only if $a_{ij} = 0$ for $i \leq j$. Otherwise it is implicit, and the output has to be computed iteratively. For example, we could use fixed-point iteration (successive substitution) with fixed iteration number L , in which case the approximation of

(2.4), denoted by $\Phi_{h,g}^L(\mathbf{x})$, is given by

$$(2.5) \quad \begin{aligned} \mathbf{v}_i^0 &= \mathbf{x} \quad i = 1, \dots, I, \\ \mathbf{v}_i^l &= \mathbf{x} + h \sum_{j=1}^I a_{ij} f_\theta(\mathbf{v}_j^{l-1}) \quad i = 1, \dots, I, \quad l = 1, \dots, L. \\ \Phi_{h,f_\theta}^L(\mathbf{x}) &= \mathbf{x} + h \sum_{i=1}^I b_i f_\theta(\mathbf{v}_i^L). \end{aligned}$$

Newton-Raphson iteration is available as an alternative approach for solving the implicit equation (2.4a), where the approximation using L iterations of (2.4), denoted as $\Phi_{h,g}^L(\mathbf{x})$, is given by

$$(2.6) \quad \begin{aligned} \mathbf{v}_i^0 &= \mathbf{x} \quad i = 1, \dots, I, \\ \mathbf{v}_i^l &= \mathbf{x} + h \sum_{j=1}^I a_{ij} (f_\theta(\mathbf{v}_j^{l-1}) + f'_\theta(\mathbf{v}_j^{l-1})(\mathbf{v}_j^l - \mathbf{v}_j^{l-1})) \quad i = 1, \dots, I, \quad l = 1, \dots, L. \\ \Phi_{h,f_\theta}^L(\mathbf{x}) &= \mathbf{x} + h \sum_{i=1}^I b_i f_\theta(\mathbf{v}_i^L). \end{aligned}$$

The second of the equations in (2.6) is equivalent to the l^{th} Newton step, where we know \mathbf{v}_j^{l-1} for $j = 1, \dots, I$, and we solve (for each l) $D \times I$ linear equations $F'(\mathbf{V}^{l-1}) \cdot (\mathbf{V}^{l-1} - \mathbf{V}^l) = F(\mathbf{V}^{l-1})$ to obtain \mathbf{v}_j^l for $j = 1, \dots, I$. Specifically,

$$\mathbf{V}^{l-1} = \begin{pmatrix} \mathbf{v}_1^{l-1} \\ \mathbf{v}_2^{l-1} \\ \vdots \\ \mathbf{v}_I^{l-1} \end{pmatrix}, \quad \mathbf{V}^l = \begin{pmatrix} \mathbf{v}_1^l \\ \mathbf{v}_2^l \\ \vdots \\ \mathbf{v}_I^l \end{pmatrix}, \quad F(\mathbf{V}^{l-1}) = \begin{pmatrix} \mathbf{v}_1^{l-1} - \mathbf{x} - h \sum_{j=1}^I a_{1j} f_\theta(\mathbf{v}_j^{l-1}) \\ \mathbf{v}_2^{l-1} - \mathbf{x} - h \sum_{j=1}^I a_{2j} f_\theta(\mathbf{v}_j^{l-1}) \\ \vdots \\ \mathbf{v}_I^{l-1} - \mathbf{x} - h \sum_{j=1}^I a_{Ij} f_\theta(\mathbf{v}_j^{l-1}) \end{pmatrix},$$

and

$$F'(\mathbf{V}^{l-1}) = \begin{pmatrix} \mathbf{I}_{D \times D} - h a_{11} f'_\theta(\mathbf{v}_1^{l-1}) & -h a_{12} f'_\theta(\mathbf{v}_2^{l-1}) & \cdots & -h a_{1I} f'_\theta(\mathbf{v}_I^{l-1}) \\ -h a_{21} f'_\theta(\mathbf{v}_1^{l-1}) & \mathbf{I}_{D \times D} - h a_{22} f'_\theta(\mathbf{v}_2^{l-1}) & \cdots & -h a_{2I} f'_\theta(\mathbf{v}_I^{l-1}) \\ \vdots & \vdots & \ddots & \vdots \\ -h a_{I1} f'_\theta(\mathbf{v}_1^{l-1}) & -h a_{I2} f'_\theta(\mathbf{v}_2^{l-1}) & \cdots & \mathbf{I}_{D \times D} - h a_{II} f'_\theta(\mathbf{v}_I^{l-1}) \end{pmatrix}.$$

In either case, ((2.5) or (2.6)), all of the operations (including any Jacobian evaluations or inversions) have forward and backward implementations in established automatic differentiation packages, and the practical loss function we optimize is given as

$$(2.7) \quad \mathcal{L}_{unrolled} := \sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n) - \phi_{m\Delta t}(\mathbf{x}_n)\|_2^2 / (m\Delta t)^2.$$

3. Inverse Modified Error Analysis. The discovery of dynamics using ODE-nets is essentially an inverse process. As (direct) Modified Differential Equations (MDEs) were well-established for the numerical analysis of differential equations, the idea of a formal extension in Inverse Modified Differential Equations (IMDEs) should prove particularly useful to the study error analysis for ODE-nets [55]. In this section, we will extend the results in [55] to unrolled implicit schemes.

3.1. Inverse Modified Differential Equations of unrolled implicit schemes. An IMDE is a perturbed differential equation of the form

$$\frac{d}{dt}\tilde{\mathbf{y}}(t) = f_h(\tilde{\mathbf{y}}(t)) = f_0(\tilde{\mathbf{y}}) + hf_1(\tilde{\mathbf{y}}) + h^2f_2(\tilde{\mathbf{y}}) + \dots,$$

such that formally

$$(3.1) \quad \Phi_{h,f_h}(\mathbf{x}) = \phi_{h,f}(\mathbf{x}),$$

where the identity is understood in the sense of the formal power series in h . To obtain f_h of an unrolled implicit scheme (2.5), we can expand both sides of (3.1) into the corresponding Taylor series around $h = 0$. First,

$$(3.2) \quad \begin{aligned} \phi_{h,f}(\mathbf{x}) = & \mathbf{x} + hf(\mathbf{x}) + \frac{h^2}{2}f'f(\mathbf{x}) + \frac{h^3}{6}(f''(f,f)(\mathbf{x}) + f'f'f(\mathbf{x})) \\ & + \frac{h^4}{24}(f'''(f,f,f)(\mathbf{x}) + 3f''(f'f,f)(\mathbf{x}) + f'f''(f,f)(\mathbf{x}) + f'f'f'f(\mathbf{x})) + \dots \end{aligned}$$

Here, $f'(\mathbf{x})$ is a linear map (the Jacobian); the second order derivative $f''(\mathbf{x})$ is a symmetric bilinear map; and so on for higher order derivatives described as tensors. We remark that a general expansion (3.2) can be obtained by Lie derivatives. Next, we expand the unrolled implicit scheme (2.5) as

$$(3.3) \quad \Phi_{h,f_h}^L(\mathbf{x}) = \mathbf{x} + hd_{1,f_h}(\mathbf{x}) + h^2d_{2,f_h}(\mathbf{x}) + h^3d_{3,f_h}(\mathbf{x}) + \dots,$$

where the functions d_{j,f_h} are given –and typically composed of f_h and its derivatives–, and can be calculated by applying B-series [9] on equation (2.5). For consistent integrators, we have

$$d_{1,f_h}(\mathbf{x}) = f_h(\mathbf{x}) = f_0(\mathbf{x}) + hf_1(\mathbf{x}) + h^2f_2(\mathbf{x}) + \dots.$$

Furthermore, in $h^i d_{i,f_h}(\mathbf{x})$, the powers of h of the terms containing f_k is at least $k + i$. Thus, the coefficient of h^{k+1} in (3.3) is

$$f_k + \dots,$$

where the “ \dots ” indicates residual terms composed of f_j with $j < k$ and their derivatives. A comparison of equal powers of h (3.2) and (3.3) then yields recursively the functions f_k in

terms of f and its derivatives. Some examples are included in [Appendix A](#) to illustrate this process. Here, we denote the truncation as

$$f_h^K(\mathbf{y}) = \sum_{k=0}^K h^k f_k(\mathbf{y}).$$

The IMDE is obtained by paper-and-pencil formal expansion given f and a numerical scheme of choice, and thus is inaccessible in practice due to the unknown true governing function. Nevertheless, we will be able to conclude the uniqueness of the solution of the learning task and analyse the numerical integration in ODE-nets.

3.2. Main results. We now show that, under reasonable assumptions, training an ODE-net using an unrolled implicit scheme returns a close approximation of the IMDE for the underlying numerical method.

We first set some notation: For a compact subset $\mathcal{K} \subset \mathbb{C}^D$ and the complex ball $\mathcal{B}(\mathbf{x}, r) \subset \mathbb{C}^D$ of radius $r > 0$ centered at $\mathbf{x} \in \mathbb{C}^D$, we define the r -dilation of \mathcal{K} as $\mathcal{B}(\mathcal{K}, r) = \bigcup_{\mathbf{x} \in \mathcal{K}} \mathcal{B}(\mathbf{x}, r)$. We will work with l_∞ - norm on \mathbb{C}^D , denote $\|\cdot\| = \|\cdot\|_\infty$, and for an analytic vector field f , define

$$\|f\|_{\mathcal{K}} = \sup_{\mathbf{x} \in \mathcal{K}} \|f(\mathbf{x})\|.$$

Now we present the main result, which implies that the unrolled implicit ODE-net approximates the IMDE.

Theorem 3.1 (The unrolled approximation approaches the IMDE). *Consider the dynamical system (2.1), a consistent implicit Runge-Kutta scheme Φ_h (2.4), and its unrolled approximation Φ_h^L ((2.5) or (2.6)). Let f_θ be the network learned by optimizing (2.7). For $\mathbf{x} \in \mathbb{R}^D$, $r_1, r_2 > 0$, we denote*

$$(3.4) \quad \mathcal{L} = \|(\Phi_{h, f_\theta}^L)^s - \phi_{sh, f}\|_{\mathcal{B}(\mathbf{x}, r_1)} / \Delta t,$$

and suppose the true vector field f and the learned vector field f_θ are analytic and satisfy $\|f\|_{\mathcal{B}(\mathbf{x}, r_1+r_2)} \leq m$, $\|f_\theta\|_{\mathcal{B}(\mathbf{x}, r_1+r_2)} \leq m$. Then, there exists a uniquely defined vector field f_h^K , i.e., the truncated IMDE of Φ_h^L , such that, if $0 < \Delta t < \Delta t_0$,

$$(3.5) \quad \|f_\theta(\mathbf{x}) - f_h^K(\mathbf{x})\| \leq c_1 m e^{-\gamma_1 / \Delta t^{1/q}} + \frac{e}{e-1} \mathcal{L},$$

where the integer $K = K(h)$ and the constants Δt_0 , q , γ_1 , c_1 depend only on m/r_1 , m/r_2 , s , Φ_h and the implicit solver¹

Proof. The proof can be found in [Appendix B.1](#). ■

¹The constants here depend on the choice of solver (specifically, on the constants b_1, b_2, b_3 in [Assumption B.1](#)). However, since the first term in (3.5) is very small, the constants contained have little effect on the results.

Here, the first term on the right hand side of (3.5) is sub-exponentially small. The \mathcal{L} defined in (3.4) can be regarded as a form of generalization of the learning loss (2.3) when $M = 1$ (loss (2.3); with different M we have equivalent convergence due to the following Lemma 3.2). In this paper we mainly focus on numerical schemes, and thus we will not further quantify \mathcal{L} . Provided we make the additional assumption that there are sufficient many data points, the network is sufficiently large and the training finds a neural network with perfect performance, then the learning loss converges to zero and the difference between the learned ODE and the truncated IMDE converges to near-zero (as per (3.5)). We therefore claim that f_θ is a close approximation of f_h^K .

Next, we show that the teacher-forcing loss (i.e., setting $M = 1$ in $\mathcal{L}_{unrolled}$ of (2.7)) is bounded by the M -step shooting loss on the same data, and thus these two have the equivalent convergence.

Lemma 3.2 (The M -step shooting loss and the teacher-forcing loss have equivalent convergence). *Let $\mathcal{T} = \{\phi_{m\Delta t}(\mathbf{x}_n)\}_{1 \leq n \leq N, 0 \leq m \leq M-1}$ be the total observed data, then, there exist constants C_1, C_2 , such that*

$$(3.6) \quad C_1 \cdot \mathcal{L}_{unrolled} \leq \sum_{\mathbf{x} \in \mathcal{T}} \|(\Phi_{h, f_\theta}^L)^s(\mathbf{x}) - \phi_{sh, f}(\mathbf{x})\|_2^2 / \Delta t^2 \leq C_2 \cdot \mathcal{L}_{unrolled}.$$

Proof. The proof can be found in Appendix B.2. ■

Since we consider variable M in section 2, we perform the analysis that follows for $M = 1$, and use Lemma 3.2 to extend to different choices of M .

Next, we have the following Theorem 3.3, which indicates that increasing the iteration number L is equivalent to adjusting the approximation target to gradually approach the true target with the help of Theorem 3.1.

Theorem 3.3 (Increasing the iteration number L is equivalent to adjusting the approximation target to gradually approach the true target). *Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4) and denote the IMDE² of Φ_h as $\hat{f}_h = \sum_{k=0}^{\infty} h^k \hat{f}_k$, and the corresponding IMDE via unrolled approximation Φ_h^L ((2.5) or (2.6)) as $f_h = \sum_{k=0}^{\infty} h^k f_k$, respectively. Then*

$$\hat{f}_h - f_h = \mathcal{O}(h^{L^*+1}), \text{ i.e., } \hat{f}_k = f_k \text{ for } k = 0, \dots, L^*,$$

where $L^* = L$ for the unrolled approximation using fixed-point iteration (2.5) and $L^* = 2^{L+1} - 2$ for the unrolled approximation using Newton-Raphson iteration (2.6).

Proof. The proof can be found in Appendix B.3. ■

Additionally, with the tools of IMDEs, we can obtain the order of convergence for learning ODEs with unrolled implicit integration:

Theorem 3.4 (Order of convergence for learning ODEs). *With the notation and under the conditions of Theorem 3.1 and Theorem 3.3, if Φ_h is of order p , i.e., $\Phi_h(\mathbf{x}) = \phi_h(\mathbf{x}) +$*

²If we suspect that this sum does not converge with $L \rightarrow \infty$, we can still study the this sum formally, truncating it according to Theorem 3.1.

$\mathcal{O}(h^{p+1})$, and $L^* + 1 \geq p$, then,

$$\|f_\theta(\mathbf{x}) - f(\mathbf{x})\| \leq c_2 m h^p + \frac{e}{e-1} \mathcal{L},$$

where the constant c_2 depends only on m/r_1 , m/r_2 , s , Φ_h and the implicit solver.

Proof. The proof can be found in [Appendix B.4](#). ■

4. Implementation of implicit scheme. As discussed in [section 2](#), one has to exploit a root-finding algorithm to solve the implicit equation (2.4a) for an implementation of (2.4). However, a drawback is that the iteration number, or stopping criterion, should usually be determined in advance and fixed during training. According to [Theorem 3.1](#) and [Theorem 3.3](#), different iteration numbers lead to different approximation targets and increasing the iteration number results in a more accurate target. Therefore, our goal is to provide an *adaptive algorithm* that increases the iteration number L , such that the error of the unrolled approximation is less than the current learning loss, thereby increasing computational efficiency while preserving accuracy.

Algorithm 4.1 Training with adaptive iteration

- 1: Initialization: L , and a neural network f_θ with trainable parameters.
 - 2: **for** each training epoch **do**
 - 3: Compute Loss = $\frac{1}{D \cdot N \cdot M} \mathcal{L}_{unrolled}$, where D is the dimension.
 - 4: Let $\theta \leftarrow \text{optimizer}(\theta, \text{lr}, \frac{\partial \text{Loss}}{\partial \theta})$ to update neural network parameters, where lr is the learning rate.
 - 5: **if** adjust iteration number **then**
 - 6: $\delta = \frac{1}{D \cdot N \cdot M} \sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h, f_\theta}^{L+1})^{ms}(\mathbf{x}_n) - (\Phi_{h, f_\theta}^L)^{ms}(\mathbf{x}_n)\|_2^2 / (m\Delta t)^2$.
 - 7: **if** Loss < $c\delta$ **then**
 - 8: Increase the iteration number L .
 - 9: **end if**
 - 10: **end if**
 - 11: **end for**
-

Next, we present the error quantification for an ODE-net using an unrolled implicit scheme, which will form the cornerstone for the following adaptive algorithm.

Lemma 4.1 (Convergence of the (“inner”) implicit iteration). *Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4) and its approximation Φ_h^L using fixed-point iteration (2.5) or Newton-Raphson iteration (2.6). Then,*

$$\begin{aligned} \mathcal{L}_{exact}^{\frac{1}{2}} &:= \left(\sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h, f_\theta}^{L+1})^{ms}(\mathbf{x}_n) - \phi_{m\Delta t}(\mathbf{x}_n)\|_2^2 / (m\Delta t)^2 \right)^{\frac{1}{2}} \\ &\leq \mathcal{L}_{unrolled}^{\frac{1}{2}} + \left(\sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h, f_\theta}^{L+1})^{ms}(\mathbf{x}_n) - (\Phi_{h, f_\theta}^L)^{ms}(\mathbf{x}_n)\|_2^2 / (m\Delta t)^2 \right)^{\frac{1}{2}} + \mathcal{O}(h^{(L+1)^*+1}), \end{aligned}$$

where $L^* = L$ for the unrolled approximation using fixed-point iteration (2.5) and $L^* = 2^{L+1} - 2$ for the unrolled approximation using Newton-Raphson iteration (2.6).

Proof. The proof can be found in Appendix B.5. ■

According to this inequality, we formulate our adaptive Algorithm 4.1. The core idea is to monitor the level of error, and adapt the iteration number in the training process according to Lemma 4.1. Essentially, Algorithm 4.1 adjusts the approximation target, i.e., the IMDE of Φ_h^L , to gradually approach the true target, i.e., the IMDE of Φ_h , see Figure 3 for an illustration.

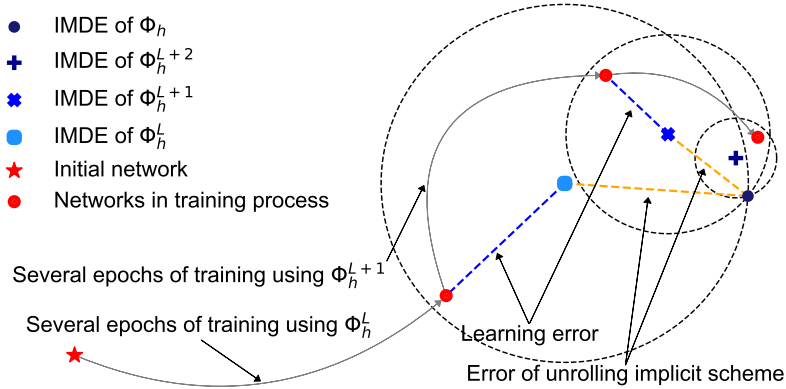


Figure 3. Illustration of the proposed adaptive algorithm. We initially employ a smaller iteration number L to train the neural network, and we gradually increase L as the learning error decreases for more precise approximation target.

5. Numerical examples. In this section, several examples are used to demonstrate the performance of the proposed algorithm and verify the theoretical analysis. We use the PyTorch library to implement Algorithm 4.1 to train our neural networks. For a given implicit solver (e.g., fixed-point iteration or Newton-Raphson iteration), we can store and backpropagate through all the iterations to obtain exact gradients for optimization. For all experiments except the last one, we generate the state data by numerically solving the dynamical system using a high order integrator with a tiny adaptive step. The trajectories are “split” so that their length M is 1. The last of our experiments uses real-world data [47]. In this last case, due to the measurement errors and other non-ideal effects, we set the length of divided trajectories to $M = 10$. After training, we simulate the learned system using a high resolution numerical solver and compare it against the true system solution. Specifically, the numerical solver for generating data and solving learned system is the fourth-order Runge-Kutta method with a finer time step of size $0.01 \cdot \Delta t$.

5.1. Linear ODEs. We first present some numerical results for two-dimensional *linear* ODEs, to verify that training an ODE-net using an unrolled implicit scheme returns an approximation of the IMDE. All examples are taken from [51].

For each test in this subsection, the training data is composed of 100 points generated from a uniform distribution over a computational domain \mathcal{D} ; each paired with its time- Δt flow; i.e., $\{\mathbf{x}_n, \phi_{\Delta t}(\mathbf{x}_n)\}_{n=1}^{100}$, with $\mathbf{x}_n \sim \text{Uniform}(\mathcal{D})$. The ODE solver that used to learn the ODE was

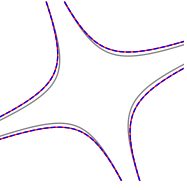
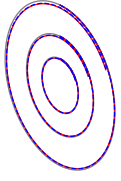
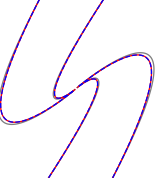
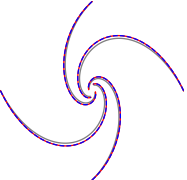
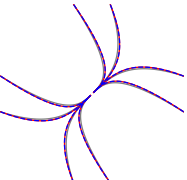
Phase portrait	True system	Learned system	IMDE	Settings
	Saddle point $\frac{d}{dt}p = p + q - 2$ $\frac{d}{dt}q = p - q$	$\frac{d}{dt}p = 1.1035p$ $+ 1.0033q$ $- 2.1068$ $\frac{d}{dt}q = 1.0033p$ $- 0.9032q$ $- 0.1002$	$\frac{d}{dt}p = 1.1035p$ $+ 1.0033q$ $- 2.1068$ $\frac{d}{dt}q = 1.0033p$ $- 0.9032q$ $- 0.1002$	$\mathcal{D} = [0, 2]^2$ $\Delta t = 0.1$ $s = 1$ $L = 0$ All schemes are equivalent in this case.
	Center point $\frac{d}{dt}p = p + 2q$ $\frac{d}{dt}q = -5p - q$	$\frac{d}{dt}p = 0.9651p$ $+ 1.9607q$ $+ 0.0000$ $\frac{d}{dt}q = -4.9017p$ $- 0.9956q$ $+ 0.0000$	$\frac{d}{dt}p = 0.9609p$ $+ 1.9568q$ $+ 0.0000$ $\frac{d}{dt}q = -4.8920p$ $- 0.9959q$ $+ 0.0000$	$\mathcal{D} = [-1, 1]^2$ $\Delta t = 0.12$ $s = 1$ Implicit Trapezoidal using fixed-point iteration with $L = 1$
	Improper node $\frac{d}{dt}p = p - 4q$ $\frac{d}{dt}q = 4p - 7q$	$\frac{d}{dt}p = 0.8222p$ $- 3.7709q$ $+ 0.0000$ $\frac{d}{dt}q = 3.7709p$ $- 6.7197q$ $+ 0.0000$	$\frac{d}{dt}p = 0.8270p$ $- 3.7771q$ $+ 0.0000$ $\frac{d}{dt}q = 3.7771p$ $- 6.7272q$ $+ 0.0000$	$\mathcal{D} = [-1, 1]^2$ $\Delta t = 0.12$ $s = 1$ Implicit Midpoint using fixed-point iteration with $L = 2$
	Spiral point $\frac{d}{dt}p = -p - q - 1$ $\frac{d}{dt}q = 2p - q + 5$	$\frac{d}{dt}p = -0.9729p$ $- 1.0503q$ $- 0.8955$ $\frac{d}{dt}q = 2.1006p$ $- 0.9729q$ $+ 5.1742$	$\frac{d}{dt}p = -0.9729p$ $- 1.0504q$ $- 0.8954$ $\frac{d}{dt}q = 2.1008p$ $- 0.9729q$ $+ 5.1745$	$\mathcal{D} = [-3, -1] \times [0, 2]$ $\Delta t = 0.05$ $s = 1$ Implicit Euler using fixed-point iteration with $L = 3$
	Nodal sink $\frac{d}{dt}p = -2 + q - 2$ $\frac{d}{dt}q = p - 2q + 1$	$\frac{d}{dt}p = -2.3368p$ $+ 1.2743q$ $- 2.3368$ $\frac{d}{dt}q = 1.2743p$ $- 2.3368q$ $+ 1.2743$	$\frac{d}{dt}p = -2.3366p$ $+ 1.2741q$ $- 2.3366$ $\frac{d}{dt}q = 1.2741p$ $- 2.3366q$ $+ 1.2741$	$\mathcal{D} = [-2, 0] \times [-1, 1]$ $\Delta t = 0.12$ $s = 1$ Implicit Euler using Newton-Raphson iteration with $L = 1$

Table 1

Discovery of linear systems. The leftmost column shows the phase portraits of the true, learned and modified systems. The four columns on the right give the corresponding ODEs as well as the experiment details. Here, L is the fixed iteration number, and $L = 0$ means that there is no iteration; i.e., the scheme reduces to forward Euler $\Phi_{h,f}^0(\mathbf{x}) = \mathbf{x} + hf(\mathbf{x})$.

chosen to be a single composition ($s = 1$) of a chosen unrolled implicit scheme with a fixed iteration number L . For the linear case, the employed neural networks have a single linear layer, i.e., we learn an affine transformation $f_\theta(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{D \times D}$, $\mathbf{b} \in \mathbb{R}^D$ are the $D^2 + D$ learnable parameters. We use full-batch Adam optimization [32] with a learning rate of 0.01 to update the parameters 10^4 times.

The detailed computational settings, descriptions of the systems and the corresponding numerical and analysis results are presented in Table 1. Note that Newton-Raphson iteration with $L = 1$ can exactly solve the implicit linear equation and thus higher iterations are not discussed. As shown in the phase portraits, the ODE-nets accurately capture the evolution of the corresponding IMDE. In addition, the trajectories of the learned systems are closer to those of the Modified systems than to the those of the true ODE, which confirms that training an ODE-net returns an approximation of the IMDE.

5.2. Damped pendulum problem. We now consider the damped pendulum problem,

$$\begin{aligned} \frac{d}{dt}p &= -\alpha p - \beta \sin q, \\ \frac{d}{dt}q &= p, \end{aligned}$$

where $\alpha = 0.2$ and $\beta = 8.91$. We generate 90 and 10 trajectories from $t = 0$ to $t = 4$ for the

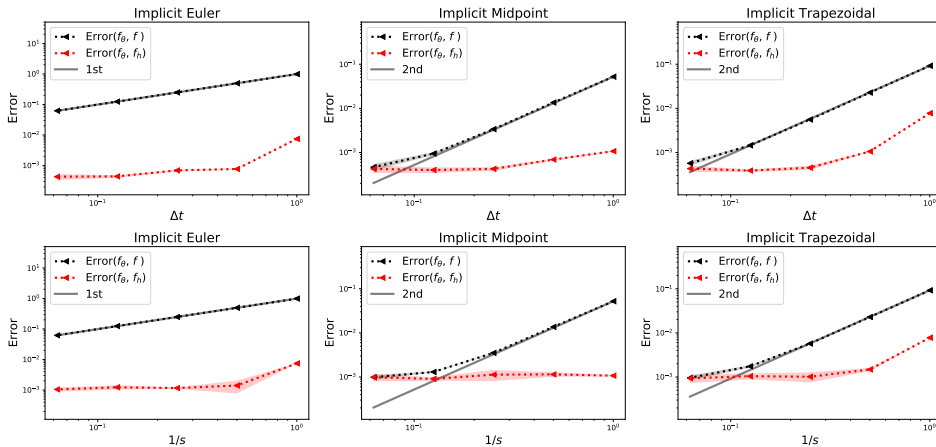


Figure 4. *Convergence rate test with respect to $h = \Delta t/s$ for learning the damped pendulum system.* Here, Δt is the data step size, h is the numerical scheme step size, and therefore $s = \Delta t/h$ is the number of scheme compositions used. Since $\text{error} \sim h^p = (\Delta t/s)^p$, we set $1/s$ and Δt as the horizontal coordinates to show convergence with respect to h . As we sweep Δt , we keep h (i.e. s) constant, and vice-versa. We see that the $\text{Error}(f_\theta, f)$ is more than the $\text{Error}(f_\theta, f_h)$. The order of $\text{Error}(f_\theta, f)$ with respect to h is consistent with the order of the employed numerical schemes. The results are obtained by taking the mean of 5 independent experiments, and the shaded region represents one standard deviation.

training data and test data respectively, with initial points randomly sampled from a uniform distribution on $\mathcal{D} = [-1.5, 0] \times [-4, 0]$. For each trajectory, $4/\Delta t + 1$ data points at equidistant time steps Δt are selected and grouped in $4/\Delta t$ successive $M = 1$ pairs.

Here we employ fixed-point iteration to solve the implicit equation. We use a feedforward neural network with two hidden layers to represent the unknown vector field, i.e.,

$$f_\theta(\mathbf{x}) = \mathbf{W}_3 \tanh(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3,$$

where $\mathbf{W}_1 \in \mathbb{R}^{128 \times D}$, $\mathbf{W}_2 \in \mathbb{R}^{128 \times 128}$, $\mathbf{W}_3 \in \mathbb{R}^{D \times 128}$, $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^{128}$, $\mathbf{b}_3 \in \mathbb{R}^D$ are the $256D + 128^2 + 256 + D$ learnable parameters and $D = 2$ is the state dimension. Results are collected after 10^5 parameter updates using full-batch Adam optimization; the learning rate is set to decay exponentially with, linearly decreasing power from 10^{-2} to 10^{-4} . We also include comparisons with the fixed iteration number setting, where we apply $L = 5$ iterations.

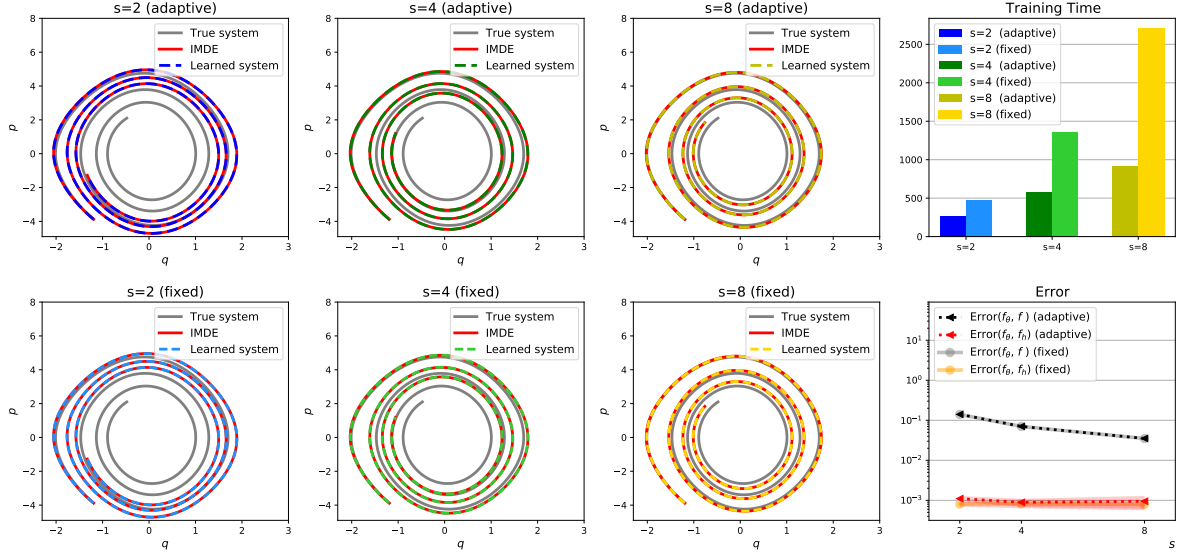


Figure 5. Results for learning damped pendulum problem. The integration of the learned system always matches that of the derived IMDE more than the truth, but the adaptive algorithm extracts this IMDE more quickly.

We first verify the convergence rate with respect to the step size h . Here, we evaluate the average error between f_θ and f and between f_θ and f_h in the l_∞ - norm, i.e.,

$$(5.1) \quad \begin{aligned} \text{Error}(f_\theta, f) &= \frac{1}{|\mathcal{T}_{test}|} \sum_{\mathbf{x}_n \in \mathcal{T}_{test}} \|f_\theta(\mathbf{x}_n) - f(\mathbf{x}_n)\|, \\ \text{Error}(f_\theta, f_h) &= \frac{1}{|\mathcal{T}_{test}|} \sum_{\mathbf{x}_n \in \mathcal{T}_{test}} \|f_\theta(\mathbf{x}_n) - f_h(\mathbf{x}_n)\|, \end{aligned}$$

where $\mathcal{T}_{test} = \{\phi_{m\Delta t}(\mathbf{x}_n)\}_{1 \leq n \leq 10, 0 \leq m \leq 4/\Delta t, \Delta t=0.01}$ is the total test data when $\Delta t = 0.01$. We assign various step sizes Δt as $\Delta t = 0.01 \cdot 2^k$, $k = 0, \dots, 4$ with fixed composition number $s = 1$; we also use several composition numbers $s = 2^k$, $k = 0, \dots, 4$ with fixed horizon $\Delta t = 0.16$, respectively. The errors are recorded in Figure 4. It can be seen that the

$\text{Error}(f_\theta, f)$ is markedly more than the $\text{Error}(f_\theta, f_h)$, indicating that the learned ODE-net returns an approximation of the particular IMDE rather than the true ODE. In addition, the order of $\text{Error}(f_\theta, f)$ with respect to h is consistent with the order of the employed numerical schemes when h is relatively large, since the learning error dominates the overall error for an accurate solver.

Next, we simulate the exact solution from $t = 0$ to $t = 8$ using the initial condition $y_0 = (-3.876, -1.193)$. We show in [Figure 5](#) the exact trajectories of the true system, the corresponding IMDE, and the right-hand-sides learned by ODE-net for different schemes, where $\Delta t = 0.01, s = 2, 4, 8$. For all integrations, the ODE-net accurately captures the evolution of the corresponding IMDE, which again implies that the learned ODE-net returns an approximation of the IMDE. When using a small learning time step $h = \Delta t/s$, the difference between the IMDE and the original equation is reduced, and thus the ODE-net tends to learn the true system. In addition, we record the error and the training time on the right side of [Figure 5](#). It is observed that the proposed adaptive iteration algorithm is remarkably faster than the non-adaptive, direct implementation, and requires less training wall-clock time to reach similar accuracy.

5.3. Glycolytic oscillator. As an example of an initial value solver employing the Newton-Raphson iteration, we consider a model of oscillations in yeast glycolysis [13]. The model describes the concentrations of seven biochemical species and is defined by

$$\begin{aligned} \frac{d}{dt}S_1 &= J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \\ \frac{d}{dt}S_2 &= 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 - 2S_5, \\ \frac{d}{dt}S_3 &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \\ \frac{d}{dt}S_4 &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\ \frac{d}{dt}S_5 &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\ \frac{d}{dt}S_6 &= -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \\ \frac{d}{dt}S_7 &= \psi \kappa (S_4 - S_7) - k S_7, \end{aligned}$$

where the ground truth parameters are taken from Table 1 in [13].

In this example, training data consists of 20 simulations which start at $(1 + \delta) \cdot \mathbf{x}_0$, where $\mathbf{x}_0 = (1.125, 0.95, 0.075, 0.16, 0.265, 0.7, 0.092)^\top = (S_1, \dots, S_7)^\top$ and δ is uniformly sampled from $[-0.2, 0.2]$. On each trajectory, 500 pairs of snapshots at $(i\Delta t, (i+1)\Delta t)$, $i = 0, \dots, 499$, $\Delta t = 0.01$ are used as training data. While the Newton-Raphson iteration is used, and s is fixed to 2, the chosen model architecture and hyperparameters are the same as in [subsection 5.2](#).

After training, we record the training time and the error in [Table 2](#); the error between f_θ and f is evaluated via (5.1) with $\mathcal{T}_{\text{test}} = \{\phi_{m\Delta t}(\mathbf{x}_0)\}_{0 \leq m \leq 500}$, while the error between the

Methods	Implicit Midpoint (adaptive)	Implicit Midpoint (fixed)	Implicit Trapezoidal (adaptive)	Implicit Trapezoidal (fixed)
Training time	1763 \pm 87	5879 \pm 224	2264 \pm 92	6357 \pm 167
Error(f_θ, f)	2.91e-2 \pm 2.37e-3	2.91e-2 \pm 2.76e-3	4.05e-2 \pm 3.14e-3	4.09e-2 \pm 3.21e-3
Error($\phi_{T,f_\theta}, \phi_{T,f}$)	7.62e-3 \pm 4.10e-3	7.63e-3 \pm 3.70e-3	1.19e-2 \pm 5.30e-3	8.74e-3 \pm 5.46e-3

Table 2

The training time (in seconds) and global error for learning the glycolytic oscillator. The results are recorded in the form of mean \pm standard deviation based on 10 independent training. The proposed adaptive algorithm markedly decrease training time with no compromise in accuracy.

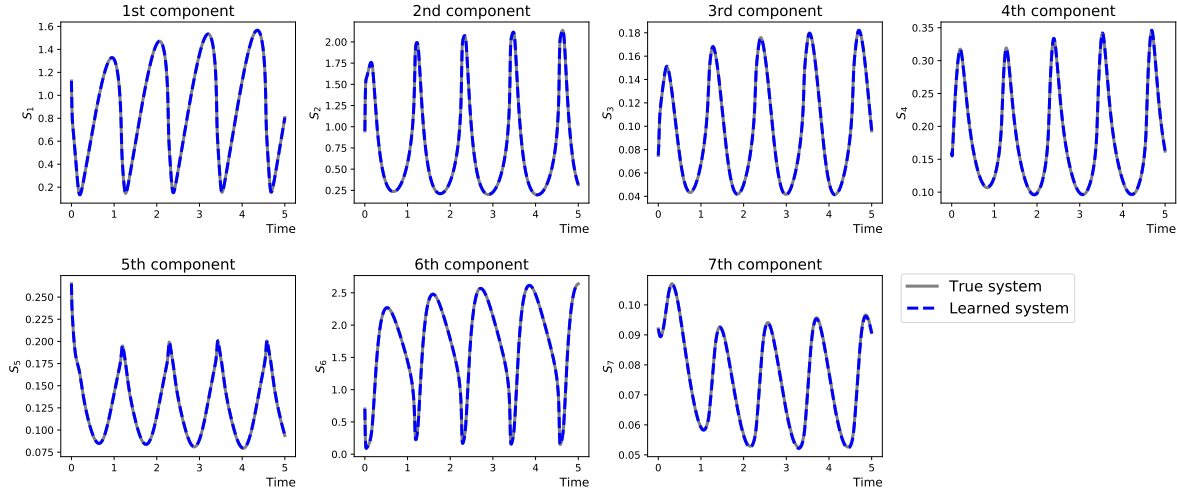


Figure 6. *Exact and learned dynamics of glycolytic oscillator. For this evaluation experiment, we keep $\Delta t = 0.01$ as in training, but set $M = 500$ rather than $M = 1$ to show that the long-term dynamics are accurate. Additionally, the initial condition used is not itself present in the training dataset.*

learned and exact trajectories are evaluated by

$$\text{Error}(\phi_{T,f_\theta}, \phi_{T,f}) = \frac{\Delta t}{T} \sum_{m=1}^{T/\Delta t} \|\phi_{m\Delta t, f_\theta}(\mathbf{x}_0) - \phi_{m\Delta t, f}(\mathbf{x}_0)\|.$$

As can be seen from Table 2, the proposed algorithm leads to a 2-3 \times speedup in training without noticeable degradation in accuracy.

In addition, we use an implicit midpoint scheme to learn the system from initial condition \mathbf{x}_0 and depict the learned and exact dynamics in Figure 6. It can be seen that the system learned using the proposed algorithm correctly captures the form of the dynamics, indicating that the performance of our approach is still promising for moderately high-dimensional equation discovery.

5.4. Learning real-world dynamics. Finally, we use real-world data [47] to verify that the proposed algorithm can learn accurate dynamics and predict future behavior in real-world

problems. This data consists of about 500 points a single trajectory of two coupled oscillators. We use the first 3/4 of the trajectory for training, and the remainder for testing. Here we assign $s = 1$, and set the length of divided trajectories to $M = 10$ rather than 1 due to measurement errors and other non-ideal effects. We train models by using full-batch Adam with a learning rate of 10^{-4} over 5000 epochs.

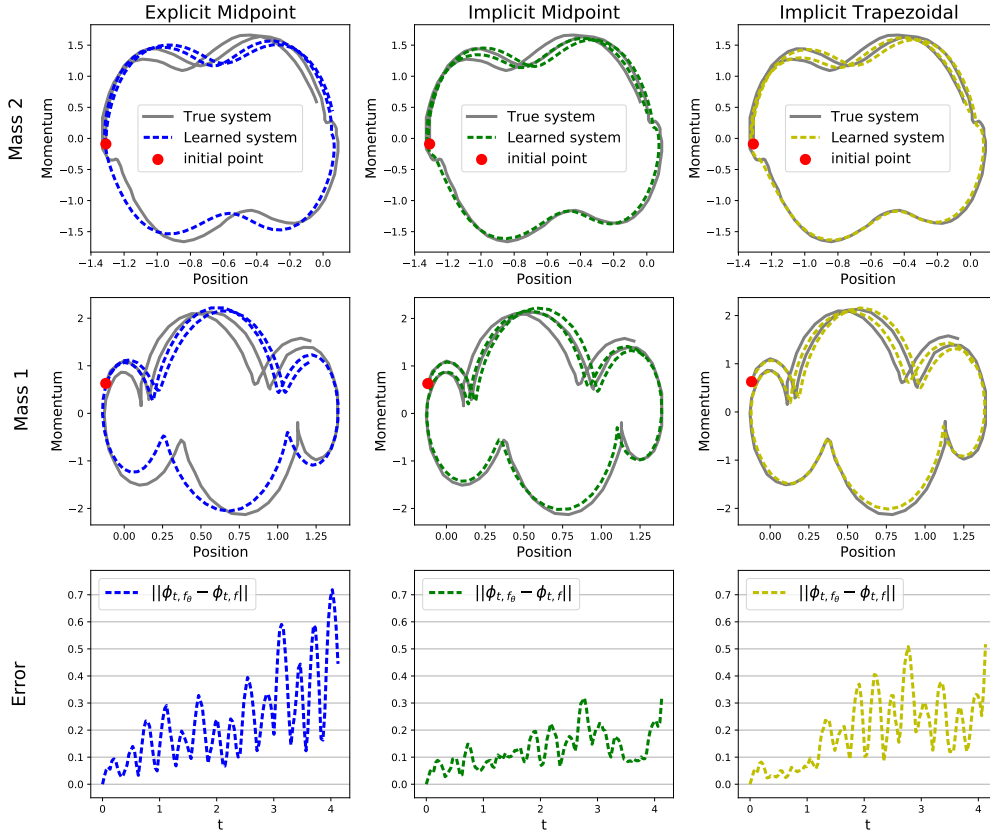


Figure 7. *Results of learning the real-world dynamics.* We find that the fine details of the trajectory around position 0.0, momentum 0.1 are preserved more faithfully by the two implicit methods than by the one explicit. This effect is conserved across multiple experiments.

We use the last point in the training data as the initial point to simulate the learned system, and depict the learned dynamics and test trajectory in Figure 7. Despite the measurement errors and other non-ideal effects, we see that the proposed algorithm still performs robustly. In addition, while all test schemes are of order 2, the use of implicit schemes for identification preserves the phase portrait more accurately and the implicit midpoint method achieves the lowest prediction error. The use of implicit schemes also permits the incorporation of geometric properties such as symplecticity, symmetry and reversibility. Although their necessity has not been mathematically proven, the results in Figure 7 show empirically better results with implicit training schemes.

Summary. Machine learning via ODE-nets provides data-driven approaches to model and predict the dynamics of physical systems from data. Since the models are typically trained on discrete data, we have to perform a numerical integration to evaluate a loss for training. In this paper we extend previous work [55], in which we defined the inverse modified differential equation (IMDE). We prove that training an ODE-net templated on an unrolled implicit scheme returns an approximation of a particular IMDE. In addition, we show that the convergence with discrete step h is of order p , where p is the order of the numerical integrator. Numerical experiments support the theoretical findings.

In addition, for learning with neural networks templated on implicit numerical integration, we propose and implement an adaptive algorithm that adjusts the iteration number of unrolled implicit integration during the training process to accelerate training. Instead of treating numerical integration of ODE-nets as a black box, our algorithm allows for finding the cheapest iteration number via monitoring the errors of the implicit solver and the learning loss. Numerical experiments show that the proposed algorithm leads to a $2 - 3\times$ speedup in training without any degradation in accuracy. Finally, we remark that our method naturally applies to the approaches based on ODE-net incorporating partially known physical terms (i.e., “gray box” identification). [42, 33]

Several challenges remain to be addressed in future work. First, the Newton-Raphson iteration (2.6) requires solving a linear equation, which makes scaling to high-dimensional equation discovery expensive. One possible direction is to do Newton-Raphson steps with an iterative algorithm such as GMRES [45].

Second, our algorithm uses the interplay between the training and numerical integration to adapt the stopping criterion. Such an idea can also be extended to efficient adaptive time-step methods, where the IMDE for adaptive steps still remains open.

Third, in classical initial value solvers, it is well known that implicit schemes have better stability [26], and allow for geometric properties such as symplecticity, symmetry and reversibility [25]. We would like to further explore in future work how these well-known forward-integration properties of implicit methods produce benefits in implicitly-templated ODE-nets over the merely explicitly-templated ones.

Finally, while we provide a rigorous grounding for the proposed adaptive Algorithm 4.1, this suggests a family of further adaptive methods for accelerating the neural identification of ODEs from data; these should better exploit existing intuition and experience about the tradeoffs inherent in the methods available in the literature. For instance, in the same way that the current learning loss sets a ceiling on the useful iteration number, we might find that switching from unrolled ODE-nets to adjoint-differentiation NODEs does make sense, but possibly only later in the training process. This program could be taken further towards a meta-learning approach, where an agent is trained to make such hyperparameter decisions online, during the training of the target ODE network. [23, 14]

Appendix A. Calculation of IMDE.

A.1. Linear ODEs. Consider a linear IVP

$$(A.1) \quad \frac{d}{dt}\mathbf{y}(t) = f(\mathbf{y}(t)) = \mathbf{A}\mathbf{y}(t), \quad \mathbf{y}(0) = \mathbf{x},$$

where $\mathbf{y}(t), \mathbf{b} \in \mathbb{R}^D$ and $\mathbf{A} \in \mathbb{R}^{D \times D}$ is invertible. It's solution at time h can be given as

$$(A.2) \quad \phi_{h,f}(\mathbf{x}) = e^{\mathbf{A}h} \mathbf{x} = \sum_{k=0}^{\infty} \frac{h^k}{k!} \mathbf{A}^k \mathbf{x} = \left(\sum_{k=0}^{\infty} \frac{(-h)^k}{k!} \mathbf{A}^k \right)^{-1} \mathbf{x}.$$

We now consider learning with implicit Euler scheme,

$$\mathbf{v}_1 = \mathbf{x} + hf(\mathbf{v}_1), \quad \Phi_{h,f_h}(\mathbf{x}) = \mathbf{x} + hf_h(\mathbf{v}_1)$$

we have

$$\Phi_{h,f_h}(\mathbf{x}) = (\mathbf{I}_D - hf_h)^{-1}(\mathbf{x})$$

where \mathbf{I}_D is the identity map. By $\Phi_{h,f_h} = \phi_{h,f}$, we deduce that

$$f_h(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{(-1)^k h^k}{(k+1)!} \mathbf{A}^{k+1} \mathbf{x} = \mathbf{A}_h \mathbf{x},$$

Additionally, if $f(\mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{b}$, we have $f_h(\mathbf{x}) = \mathbf{A}_h \mathbf{x} + \mathbf{A}_h \mathbf{A}^{-1} \mathbf{b}$ by linear transformation $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{A}^{-1} \mathbf{b}$.

A.2. General nonlinear ODEs. For a nonlinear ODE,

$$\frac{d}{dt} \mathbf{y}(t) = f(\mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{x},$$

we first expand the exact solution:

$$(A.3) \quad \begin{aligned} \phi_{h,f}(\mathbf{x}) &= \mathbf{x} + hf(\mathbf{x}) + \frac{h^2}{2} f'f(\mathbf{x}) + \frac{h^3}{6} (f''(f, f)(\mathbf{x}) + f'f'f(\mathbf{x})) \\ &+ \frac{h^4}{24} (f'''(f, f, f)(\mathbf{x}) + 3f''(f'f, f)(\mathbf{x}) + f'f''(f, f)(\mathbf{x}) + f'f'f'f(\mathbf{x})) + \dots \end{aligned}$$

As an example, the numerical scheme is chosen to be implicit Euler scheme, using Newton-Raphson iteration with $L = 1$,

$$\mathbf{v}_1^0 \equiv \mathbf{x}, \quad \mathbf{v}_1^1 = \mathbf{x} + hf_h(\mathbf{v}_1^0) + hf'_h(\mathbf{v}_1^0)(\mathbf{v}_1^1 - \mathbf{v}_1^0), \quad \Phi_{h,f_h}^1(\mathbf{x}) \equiv \mathbf{x} + hf_h(\mathbf{v}_1^1).$$

We expand it as

$$\begin{aligned} \Phi_{h,f_h}^1(\mathbf{x}) &= \mathbf{x} + hf_h(\mathbf{x}) + h^2 f'_h f_h(\mathbf{x}) + \frac{h^3}{2} f''_h(f_h, f_h)(\mathbf{x}) + h^3 f'_h f'_h f_h(\mathbf{x}) \\ &+ \frac{h^4}{6} f'''_h(f_h, f_h, f_h)(\mathbf{x}) + h^4 f''_h(f'_h f_h, f_h)(\mathbf{x}) + h^4 f'_h f'_h f'_h f_h(\mathbf{x}) + \dots \end{aligned}$$

Substituting

$$f_h = f_0 + hf_1 + h^2 f_2 + h^3 f_3 \dots$$

yields

$$\begin{aligned}
\Phi_{h,f_h}^2(\mathbf{x}) &= \mathbf{x} + hf_0 + h^2(f_1(\mathbf{x}) + f'_0f_0(\mathbf{x})) \\
&\quad + h^3(f_2(\mathbf{x}) + f'_1f_0(\mathbf{x}) + f'_0f_1(\mathbf{x}) + \frac{1}{2}f''_0(f_0, f_0)(\mathbf{x}) + f'_0f'_0f_0(\mathbf{x})) \\
&\quad + h^4(f_3(\mathbf{x}) + f'_1f_1(\mathbf{x}) + f'_0f_2(\mathbf{x}) + f'_2f_0(\mathbf{x}) + \frac{1}{2}f''_1(f_0, f_0)(\mathbf{x}) + f''_0(f_1, f_0)(\mathbf{x}) \\
&\quad + f'_1f'_0f_0(\mathbf{x}) + f'_0f'_1f_0(\mathbf{x}) + f'_0f'_0f_1(\mathbf{x})) \\
&\quad + \frac{1}{6}f'''_0(f_0, f_0, f_0)(\mathbf{x}) + f''_0(f'_0f_0, f_0)(\mathbf{x}) + f'_0f'_0f'_0f_0(\mathbf{x}) + \dots.
\end{aligned}$$

Comparing like powers of h with expression (A.3) yields recurrence relations for functions f_k , i.e.,

$$\begin{aligned}
f_0(\mathbf{y}) &= f(\mathbf{y}), \\
f_1(\mathbf{y}) &= \frac{1}{2}f'f(\mathbf{y}) - f'_0f_0(\mathbf{y}) = -\frac{1}{2}f'f(\mathbf{y}), \\
f_2(\mathbf{y}) &= \frac{1}{6}(f''(f, f)(\mathbf{y}) + f'f'f(\mathbf{y})) - (f'_1f_0(\mathbf{y}) + f'_0f_1(\mathbf{y}) + \frac{1}{2}f''_0(f_0, f_0)(\mathbf{y}) + f'_0f'_0f_0(\mathbf{y})) \\
&= \frac{1}{6}f''(f, f)(\mathbf{y}) + \frac{1}{6}f'f'f(\mathbf{y}), \\
f_3(\mathbf{y}) &= \frac{1}{24}(f'''(f, f, f)(\mathbf{y}) + 3f''(f'f, f)(\mathbf{y}) + f'f''(f, f)(\mathbf{y}) + f'f'f'f(\mathbf{y})) \\
&\quad - (f'_1f_1(\mathbf{y}) + f'_0f_2(\mathbf{y}) + f'_2f_0(\mathbf{y}) + \frac{1}{2}f''_1(f_0, f_0)(\mathbf{y}) + f''_0(f_1, f_0)(\mathbf{y})) \\
&\quad + f'_1f'_0f_0(\mathbf{y}) + f'_0f'_1f_0(\mathbf{y}) + f'_0f'_0f_1(\mathbf{y}) \\
&\quad + \frac{1}{6}f'''_0(f_0, f_0, f_0)(\mathbf{y}) + f''_0(f'_0f_0, f_0)(\mathbf{y}) + f'_0f'_0f'_0f_0(\mathbf{y}) \\
&= -\frac{1}{24}f'''(f, f, f)(\mathbf{y}) - \frac{1}{8}f''(f'f, f)(\mathbf{y}) + \frac{11}{24}f'f''(f, f)(\mathbf{y}) - \frac{1}{24}f'f'f'f(\mathbf{y}), \\
&\quad \vdots
\end{aligned}$$

Note that Newton-Raphson iteration with $L = 1$ can exactly solve the implicit linear equation. In the linear case (A.1), $f' = \mathbf{A}$ and all terms involving f'' and all higher-order derivatives are 0, giving us

$$\begin{aligned}
(A.4) \quad f_0(\mathbf{y}) &= \mathbf{A} \cdot \mathbf{y} & f_1(\mathbf{y}) &= -\frac{1}{2}\mathbf{A}^2 \cdot \mathbf{y} \\
f_2(\mathbf{y}) &= \frac{1}{6}\mathbf{A}^3 \cdot \mathbf{y} & f_3(\mathbf{y}) &= -\frac{1}{24}\mathbf{A}^4 \cdot \mathbf{y}
\end{aligned}$$

so we recover approximately (A.2) via

$$\begin{aligned}
 f_h(\mathbf{y}) &= f_0(\mathbf{y}) + hf_1(\mathbf{y}) + h^2f_2(\mathbf{y}) + h^3f_3(\mathbf{y}) + \dots \\
 (A.5) \quad &= \left(\frac{h^0}{1}\mathbf{A} + \frac{-h}{2}\mathbf{A}^2 + \frac{h^2}{6}\mathbf{A}^3 - \frac{1h^3}{24}\mathbf{A}^4 + \dots \right) \cdot \mathbf{y} \\
 &\approx \sum_{k=0}^{\infty} \frac{(-1)^k h^k}{(k+1)!} \mathbf{A}^{k+1} \cdot \mathbf{y}.
 \end{aligned}$$

We next present an example employing fixed-point iteration with $L = 2$,

$$(A.6) \quad \mathbf{v}_1^0 \equiv \mathbf{x}, \quad \mathbf{v}_1^1 = \mathbf{x} + hf(\mathbf{x}), \quad \mathbf{v}_1^2 = \mathbf{x} + hf(\mathbf{v}_1^1), \quad \Phi_{h,f_h}^2(\mathbf{x}) \equiv \mathbf{x} + hf_h(\mathbf{v}_1^2).$$

We then use B-series to expand $\Phi_{h,f_h}^2(\mathbf{x})$:

$$\begin{aligned}
 \Phi_{h,f_h}^2(\mathbf{x}) &= \mathbf{x} + hf_h(\mathbf{x}) + h^2f'_hf_h(\mathbf{x}) + \frac{h^3}{2}f''_h(f_h, f_h)(\mathbf{x}) + h^3f'_hf'_hf_h(\mathbf{x}) \\
 &\quad + \frac{h^4}{6}f'''_h(f_h, f_h, f_h)(\mathbf{x}) + h^4f''_h(f'_hf_h, f_h)(\mathbf{x}) + \frac{h^4}{2}f'_hf''_h(f_h, f_h)(\mathbf{x}) + \dots.
 \end{aligned}$$

And similarly we have

$$\begin{aligned}
 \Phi_{h,f_h}^2(\mathbf{x}) &= \mathbf{x} + hf_0 + h^2(f_1(\mathbf{x}) + f'_0f_0(\mathbf{x})) \\
 &\quad + h^3(f_2(\mathbf{x}) + f'_1f_0(\mathbf{x}) + f'_0f_1(\mathbf{x}) + \frac{1}{2}f''_0(f_0, f_0)(\mathbf{x}) + f'_0f'_0f_0(\mathbf{x})) \\
 &\quad + h^4(f_3(\mathbf{x}) + f'_1f_1(\mathbf{x}) + f'_0f_2(\mathbf{x}) + f'_2f_0(\mathbf{x}) + \frac{1}{2}f''_1(f_0, f_0)(\mathbf{x}) + f''_0(f_1, f_0)(\mathbf{x}) \\
 &\quad + f'_1f'_0f_0(\mathbf{x}) + f'_0f'_1f_0(\mathbf{x}) + f'_0f'_0f_1(\mathbf{x})) \\
 &\quad + \frac{1}{6}f'''_0(f_0, f_0, f_0)(\mathbf{x}) + f''_0(f'_0f_0, f_0)(\mathbf{x}) + \frac{1}{2}f'_0f''_0(f_0, f_0)(\mathbf{x}) + \dots.
 \end{aligned}$$

Comparing like powers of h with expression (A.3), we obtain that

$$\begin{aligned}
f_0(\mathbf{y}) &= f(\mathbf{y}), \\
f_1(\mathbf{y}) &= \frac{1}{2}f'f(\mathbf{y}) - f'_0f_0(\mathbf{y}) = -\frac{1}{2}f'f(\mathbf{y}), \\
f_2(\mathbf{y}) &= \frac{1}{6}(f''(f, f)(\mathbf{y}) + f'f'f(\mathbf{y})) - (f'_1f_0(\mathbf{y}) + f'_0f_1(\mathbf{y}) + \frac{1}{2}f''_0(f_0, f_0)(\mathbf{y}) + f'_0f'_0f_0(\mathbf{y})) \\
&= \frac{1}{6}f''(f, f)(\mathbf{y}) + \frac{1}{6}f'f'f(\mathbf{y}), \\
f_3(\mathbf{y}) &= \frac{1}{24}(f'''(f, f, f)(\mathbf{y}) + 3f''(f'f, f)(\mathbf{y}) + f'f''(f, f)(\mathbf{y}) + f'f'f'f(\mathbf{y})) \\
&\quad - (f'_1f_1(\mathbf{y}) + f'_0f_2(\mathbf{y}) + f'_2f_0(\mathbf{y}) + \frac{1}{2}f''_1(f_0, f_0)(\mathbf{y}) + f''_0(f_1, f_0)(\mathbf{y}) \\
&\quad + f'_1f'_0f_0(\mathbf{y}) + f'_0f'_1f_0(\mathbf{y}) + f'_0f'_0f_1(\mathbf{y})) \\
&\quad + \frac{1}{6}f'''(f_0, f_0, f_0)(\mathbf{y}) + f''_0(f'_0f_0, f_0)(\mathbf{y}) + \frac{1}{2}f'_0f''_0(f_0, f_0)(\mathbf{y})) \\
&= -\frac{1}{24}f'''(f, f, f)(\mathbf{y}) - \frac{1}{8}f''(f'f, f)(\mathbf{y}) - \frac{1}{24}f'f''(f, f)(\mathbf{y}) + \frac{23}{24}f'f'f'f(\mathbf{y}), \\
&\quad \vdots
\end{aligned}$$

Appendix B. Proofs.

B.1. Proof of Theorem 3.1 (The unrolled approximation approaches the IMDE). The proof of Theorem 3.1 is obtained as the proof of Theorem 3.1 in [55] under the following Assumption B.1. Here we sketch the main idea in the notation used there, and then show that Assumption B.1 holds.

Assumption B.1 (Assumptions for numerical schemes). For analytic g , \hat{g} satisfying $\|g\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$, $\|\hat{g}\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$, there exist constants b_1, b_2, b_3 that depend only on the scheme Φ_h and composition number S such that the unrolled approximation Φ_h^L satisfies

- for $|h| \leq h_0 = b_1 r/m$, $(\Phi_{h, \hat{g}}^L)^S$, $(\Phi_{h, g}^L)^S$ are analytic on \mathcal{K} .
- for $|h| \leq h_0$,

$$\|(\Phi_{h, \hat{g}}^L)^S - (\Phi_{h, g}^L)^S\|_{\mathcal{K}} \leq b_2 S |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, r)}.$$

- for $|h| < h_1 < h_0$,

$$\|\hat{g} - g\|_{\mathcal{K}} \leq \frac{1}{S|h|} \|(\Phi_{h, \hat{g}}^L)^S - (\Phi_{h, g}^L)^S\|_{\mathcal{K}} + \frac{b_2|h|}{h_1 - |h|} \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, b_3 S h_1 m)}.$$

Lemma B.2 (Choice of truncation and estimation of error for IMDE). Let $f(\mathbf{y})$ be analytic in $\mathcal{B}(\mathcal{K}, r)$ and satisfies $\|f\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$. Suppose the numerical scheme Φ_h and its approximation Φ_h^L satisfy Assumption B.1. Take $\eta = \max\{6, \frac{b_2+1}{29} + 1\}$, $\zeta = 10(\eta - 1)$, $q = -\ln(2b_2)/\ln 0.912$ and let K be the largest integer satisfying

$$\frac{\zeta(K - p + 2)^q |h|m}{\eta r} \leq e^{-q}.$$

If $|h|$ is small enough such that $K \geq p$, then the truncated IMDE satisfies

$$\begin{aligned} \|(\Phi_{h, f_h^K}^L)^S - \phi_{Sh, f}\|_{\mathcal{K}} &\leq b_2 \eta m e^{2q-qp} |Sh| e^{-\gamma/|Sh|^{1/q}}, \\ \|\sum_{k=p}^K h^k f_k\|_{\mathcal{K}} &\leq b_2 \eta m \left(\frac{\zeta m}{b_1 r}\right)^p (1 + 1.38^q d_p) |h|^p, \\ \|f_h^K\|_{\mathcal{K}} &\leq (\eta - 1)m, \end{aligned}$$

where $\gamma = \frac{q}{e} \left(\frac{b_1 r}{\zeta m}\right)^{1/q}$, $d_p = p^{qp} e^{-q(p-1)}$.

Proof. According the Lemma B.1 in [55], the IMDE of the $(\Phi_h^L)^S$ coincides with the IMDE of Φ_h^L . In addition, via regarding compositions $(\Phi_h^L)^S$ as a one-step integrator, the estimates are obtained as in the proof of Lemma B.4 in [55]. Special constants including 0.912 and 1.38 are also explained in [55]. \blacksquare

Proof of Theorem 3.1. According to Lemma B.2, we have that

$$(B.1) \quad \delta := \frac{1}{\Delta t} \|(\Phi_{h, f_\theta})^S - (\Phi_{h, f_h^K})^S\|_{\mathcal{B}(\mathbf{x}, r_1)} \leq \mathcal{L} + c m e^{-\gamma/\Delta t^{1/q}}, \quad \|f_h^K\|_{\mathcal{B}(\mathbf{x}, r_1)} < (\eta - 1)m,$$

where $c = b_2 \eta e^q$. Let

$$h_1 = (eb_2 + 1)h, \quad M = (\eta - 1)m.$$

By the third term of Assumption B.1, we deduce that for $0 \leq j \leq r_1/b_3 S h_1 M$,

$$\|f_\theta - f_h^K\|_{\mathcal{B}(\mathbf{x}, j b_3 S h_1 M)} \leq \delta + e^{-1} \|f_\theta - f_h^K\|_{\mathcal{B}(\mathbf{x}, (j+1) b_3 S h_1 M)}.$$

As in the proof of Theorem 3.1 in [55], we obtain that

$$\|f_\theta(\mathbf{x}) - f_h^K(\mathbf{x})\| \leq e^{-\hat{\gamma}/\Delta t} \|f_\theta - f_h^K\|_{\mathcal{B}(\mathbf{x}, r_1)} + \frac{\delta}{1 - \lambda},$$

where $\hat{\gamma} = \frac{r_1}{(eb_2 + 1)b_3 M}$. And thus we conclude that

$$\|f_\theta(\mathbf{x}) - f_h^K(\mathbf{x})\| \leq c_1 m e^{-\gamma/\Delta t^{1/q}} + C \mathcal{L},$$

where $C = e/(e - 1)$ and c_1 is a constant satisfying $c_1 \geq C \cdot c + \eta e^{\gamma/\Delta t^{1/q} - \hat{\gamma}/\Delta t}$. \blacksquare

Next, to complete the proof of Theorem 3.1, it suffices to show that unrolled implicit Runge-Kutta scheme Φ_h (2.4) using fixed-point iteration (2.5) or Newton-Raphson iteration (2.6) both satisfy Assumption B.1.

Lemma B.3 (Fixed-point iteration obeys Assumption B.1). *Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4) and its approximation via fixed-point iteration Φ_h^L (2.5), denote*

$$\mu = \sum_{i=1}^I |b_i|, \quad \kappa = \max_{1 \leq i \leq I} \sum_{j=1}^I |a_{ij}|.$$

Let g, \hat{g} be analytic in $\mathcal{B}(\mathcal{K}, r)$ and satisfy $\|g\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$, $\|\hat{g}\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$. Then, for $|h| \leq h_0 = r/(2(S\mu + \kappa)m)$ and $\mathbf{x} \in \mathcal{K}$, the compositions $(\Phi_{h,g}^L)^S(\mathbf{x})$, $(\Phi_{h,\hat{g}}^L)^S(\mathbf{x})$ are analytic and

$$\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}} \leq (e-1)(S\mu + \kappa)|h|\|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, r)}.$$

In addition, for $|h| < h_1 \leq h_0$,

$$\|\hat{g} - g\|_{\mathcal{K}} \leq \frac{\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}}}{S|h|} + \frac{(e-1)(\mu + \kappa/S)|h|\|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)h_1 m)}}{h_1 - |h|}.$$

Proof. For $\mathbf{y} \in \mathcal{B}(\mathcal{K}, r/2)$ and $\|\Delta\mathbf{y}\| \leq 1$, the function $\alpha(z) = g(\mathbf{y} + z\Delta\mathbf{y})$ is analytic for $|z| \leq r/2$ and bounded by m . By Cauchy's estimate, we obtain

$$\|g'(\mathbf{y})\Delta\mathbf{y}\| = \|\alpha'(0)\| \leq 2m/r,$$

and $\|g'(\mathbf{y})\| \leq 2m/r$ for $\mathbf{y} \in \mathcal{B}(\mathcal{K}, r/2)$ in the operator norm. Similarly, $\|\hat{g}'(\mathbf{y})\| \leq 2m/r$ for $\mathbf{y} \in \mathcal{B}(\mathcal{K}, r/2)$.

As in (A.6), when using fixed-point iteration to unroll Runge-Kutta method (2.5), the solutions are recursively obtained by

$$\begin{cases} \mathbf{v}_i^{0,s} = \mathbf{x}^{L,s}, & \mathbf{x}^{L,0} = \mathbf{x} \\ \mathbf{v}_i^{l,s} = \mathbf{x}^{L,s} + h \sum_{j=1}^I a_{ij} g(\mathbf{v}_j^{l-1,s}), \\ \mathbf{x}^{L,s+1} = \mathbf{x}^{L,s} + h \sum_{i=1}^I b_i g(\mathbf{v}_i^{L,s}), \end{cases} \quad \begin{cases} \hat{\mathbf{v}}_i^{0,s} = \hat{\mathbf{x}}^{L,s}, & \hat{\mathbf{x}}^{L,0} = \mathbf{x} \\ \hat{\mathbf{v}}_i^{l,s} = \hat{\mathbf{x}}^{L,s} + h \sum_{j=1}^I a_{ij} \hat{g}(\hat{\mathbf{v}}_j^{l-1,s}), \\ \hat{\mathbf{x}}^{L,s+1} = \hat{\mathbf{x}}^{L,s} + h \sum_{i=1}^I b_i \hat{g}(\hat{\mathbf{v}}_i^{L,s}), \end{cases}$$

where $l = 1, \dots, L$, $s = 0, \dots, S-1$. For $|h| \leq h_0 = r/(2(S\mu + \kappa)m)$ and $\mathbf{x} \in \mathcal{K}$, we can readily check that

$$\begin{aligned} \mathbf{x}^{L,s}, \hat{\mathbf{x}}^{L,s} &\in \mathcal{B}(\mathcal{K}, s\mu m|h|) \text{ for } s = 0, \dots, S, \\ \mathbf{v}_i^{l,s}, \hat{\mathbf{v}}_i^{l,s} &\in \mathcal{B}(\mathcal{K}, (s\mu + \kappa)m|h|) \text{ for } s = 0, \dots, S-1, l = 1, \dots, L, i = 1, \dots, I. \end{aligned}$$

Denote $V^{l,s} = \max_{1 \leq i \leq I} \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^{l,s}\|$, $X^s = \|\mathbf{x}^{L,s} - \hat{\mathbf{x}}^{L,s}\|$, we have

$$\begin{aligned} \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^{l,s}\| &\leq |h| \sum_{j=1}^I |a_{ij}| (\|g(\mathbf{v}_j^{l-1,s}) - g(\hat{\mathbf{v}}_j^{l-1,s})\| + \|g(\hat{\mathbf{v}}_j^{l-1,s}) - \hat{g}(\hat{\mathbf{v}}_j^{l-1,s})\|) + X^s \\ &\leq |h|\kappa \frac{2m}{r} V^{l-1,s} + |h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)} + X^s. \end{aligned}$$

Thus we obtain

$$V^{l,s} \leq |h|\kappa \frac{2m}{r} V^{l-1,s} + |h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)} + X^s.$$

As a result, we have

$$\begin{aligned}
 (B.2) \quad V^{L,s} &\leq (|h|\kappa \frac{2m}{r})^L V^{0,s} + \frac{1 - (|h|\kappa \frac{2m}{r})^L}{1 - |h|\kappa \frac{2m}{r}} X^s + \frac{1 - (|h|\kappa \frac{2m}{r})^L}{1 - |h|\kappa \frac{2m}{r}} |h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)} \\
 &\leq \frac{1}{1 - |h|\kappa \frac{2m}{r}} X^s + \frac{1}{1 - |h|\kappa \frac{2m}{r}} |h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)}.
 \end{aligned}$$

In addition,

$$\begin{aligned}
 (B.3) \quad \|\mathbf{x}^{L,s+1} - \hat{\mathbf{x}}^{L,s+1}\| &\leq X^s + |h| \sum_{i=1}^s |b_i| (\|g(\mathbf{v}_j^{L,s}) - g(\hat{\mathbf{v}}_j^{L,s})\| + \|g(\hat{\mathbf{v}}_j^{L,s}) - \hat{g}(\hat{\mathbf{v}}_j^{L,s})\|) \\
 &\leq X^s + |h|\mu \frac{2m}{r} V^{L,s} + |h|\mu \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)}.
 \end{aligned}$$

These estimates, together with (B.2), indicate that

$$(B.4) \quad X^{s+1} \leq (1 + \frac{|h|\mu \frac{2m}{r}}{1 - |h|\kappa \frac{2m}{r}}) X^s + (\frac{|h|\mu \frac{2m}{r}}{1 - |h|\kappa \frac{2m}{r}} \kappa + \mu) |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)},$$

Therefore, we deduce that

$$\begin{aligned}
 X^S &\leq \frac{(1 + \frac{|h|\mu \frac{2m}{r}}{1 - |h|\kappa \frac{2m}{r}})^S - 1}{\frac{|h|\mu \frac{2m}{r}}{1 - |h|\kappa \frac{2m}{r}}} (\frac{|h|\mu \frac{2m}{r}}{1 - |h|\kappa \frac{2m}{r}} \kappa + \mu) |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)} \\
 &\leq (e - 1)(S\mu + \kappa) |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)m|h|)}.
 \end{aligned}$$

where we have used the fact $|h|\mu \frac{2m}{r} / (1 - |h|\kappa \frac{2m}{r}) \leq 1/S$.

Finally, using Cauchy's estimate, we deduce that $h_1 \leq h_0$

$$\left\| \frac{d^i}{dh^i} ((\Phi_{h,\hat{g}}^L)^S(\mathbf{x}) - (\Phi_{h,g}^L)^S(\mathbf{x})) \Big|_{h=0} \right\| \leq \frac{i! \cdot (e - 1)(S\mu + \kappa) \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)h_1m)}}{h_1^{i-1}}.$$

By the analyticity and triangle inequality, we obtain that for $|h| < h_1$,

$$\begin{aligned}
 &\|(\Phi_{h,\hat{g}}^L)^S(\mathbf{x}) - (\Phi_{h,g}^L)^S(\mathbf{x})\| \\
 &\geq S|h| \|\hat{g}(\mathbf{x}) - g(\mathbf{x})\| - \sum_{i=2}^{\infty} \left\| \frac{h^i}{i!} \frac{d^i}{dh^i} ((\Phi_{h,\hat{g}}^L)^S(\mathbf{x}) - (\Phi_{h,g}^L)^S(\mathbf{x})) \Big|_{h=0} \right\| \\
 &\geq S|h| \|\hat{g}(\mathbf{x}) - g(\mathbf{x})\| - (e - 1)(S\mu + \kappa) |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)h_1m)} \sum_{i=2}^{\infty} \left(\frac{|h|}{h_1} \right)^{i-1}.
 \end{aligned}$$

Therefore, we have

$$(B.5) \quad \|\hat{g} - g\|_{\mathcal{K}} \leq \frac{\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}}}{S|h|} + \frac{(e - 1)(\mu + \kappa/S) |h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu + \kappa)h_1m)}}{h_1 - |h|},$$

which concludes the proof. ■

Lemma B.4 (Newton-Raphson iteration obeys Assumption B.1). *Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4), and its approximation using Newton-Raphson iteration Φ_h^L (2.6), denote*

$$\mu = \sum_{i=1}^I |b_i|, \quad \kappa = \max_{1 \leq i \leq I} \sum_{j=1}^I |a_{ij}|.$$

Let g, \hat{g} be analytic in $\mathcal{B}(\mathcal{K}, r)$ and satisfy $\|g\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$, $\|\hat{g}\|_{\mathcal{B}(\mathcal{K}, r)} \leq m$. Then, for $|h| \leq h_0 = r/(2(S\mu + 3.5\kappa)m)$ and $\mathbf{x} \in \mathcal{K}$, the compositions $(\Phi_{h,g}^L)^S(\mathbf{x})$, $(\Phi_{h,\hat{g}}^L)^S(\mathbf{x})$ are analytic and

$$\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}} \leq (e-1)\mu S|h|\|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, r)}.$$

In addition, for $|h| < h_1 \leq h_0$,

$$(B.6) \quad \|\hat{g} - g\|_{\mathcal{K}} \leq \frac{\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}}}{S|h|} + \frac{(e-1)(\mu + \kappa/S)|h|\|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu+3\kappa)h_1m)}}{h_1 - |h|}.$$

We note that (B.5) and (B.6) differ only in the constants 3.5 and 3 used (which are both 1 in the FP case).

Proof. For $\mathbf{y} \in \mathcal{B}(\mathcal{K}, r/2)$ and $\|\Delta\mathbf{y}\| \leq 1$, the function $\alpha(z) = g(\mathbf{y} + z\Delta\mathbf{y})$ is analytic for $|z| \leq r/2$ and bounded by m . By Cauchy's estimate, we obtain

$$\|g'(\mathbf{y})\Delta\mathbf{y}\| = \|\alpha'(0)\| \leq 2m/r, \quad \|g''(\mathbf{y})(\Delta\mathbf{y}, \Delta\mathbf{y})\| = \|\alpha''(0)\| \leq 4m/r^2$$

and thus $\|g'(\mathbf{y})\| \leq 2m/r$, $\|g''(\mathbf{y})\| \leq 4m/r^2$ for $\mathbf{y} \in \mathcal{B}(\mathcal{K}, r/2)$ in the operator norm. Similar estimates hold for \hat{g} .

When using Newton-Raphson iteration to unroll Runge-Kutta method (2.5), the solutions are recursively obtained by

$$\begin{cases} \mathbf{v}_i^{0,s} = \mathbf{x}^{L,s}, & \mathbf{x}^{L,0} = \mathbf{x} \\ \mathbf{v}_i^{l,s} = \mathbf{x}^{L,s} + h \sum_{j=1}^I a_{ij} (g(\mathbf{v}_j^{l-1,s}) + g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s})), \\ \mathbf{x}^{L,s+1} = \mathbf{x}^{L,s} + h \sum_{i=1}^I b_i g(\mathbf{v}_i^{L,s}), \end{cases}$$

$$\begin{cases} \hat{\mathbf{v}}_i^{0,s} = \hat{\mathbf{x}}^{L,s}, & \hat{\mathbf{x}}^{L,0} = \mathbf{x} \\ \hat{\mathbf{v}}_i^{l,s} = \hat{\mathbf{x}}^{L,s} + h \sum_{j=1}^I a_{ij} (\hat{g}(\hat{\mathbf{v}}_j^{l-1,s}) + \hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\hat{\mathbf{v}}_j^{l,s} - \hat{\mathbf{v}}_j^{l-1,s})), \\ \hat{\mathbf{x}}^{L,s+1} = \hat{\mathbf{x}}^{L,s} + h \sum_{i=1}^I b_i \hat{g}(\hat{\mathbf{v}}_i^{L,s}), \end{cases}$$

where $l = 1, \dots, L$, $s = 0, \dots, S-1$. Denote $U^{l,s} = \max_{1 \leq i \leq I} \|\mathbf{v}_i^{l,s} - \mathbf{v}_i^{l-1,s}\|$, we have

$$\begin{aligned} & \|\mathbf{v}_i^{l,s} - \mathbf{v}_i^{l-1,s}\| \\ & \leq h \sum_{j=1}^I |a_{ij}| \|g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_i^{l,s} - \mathbf{v}_j^{l-1,s}) + g(\mathbf{v}_j^{l-1,s}) - g(\mathbf{v}_j^{l-2,s}) - g'(\mathbf{v}_j^{l-2,s})(\mathbf{v}_i^{l-1,s} - \mathbf{v}_j^{l-2,s})\| \\ & \leq m_1 \kappa h U^{l,s} + m_2 \kappa h (U^{l-1,s})^2 \end{aligned}$$

where $m_1 = \max_{s,l,i} \|g'(\mathbf{v}_j^{l-1,s})\|$, $m_2 = \max_{s,l,i} \sup_{\theta \in [0,1]} \|g''(\theta \mathbf{v}_j^{l-1,s} + (1-\theta)\mathbf{v}_j^{l-2,s})\|$. Let $m_0 = \max_{s,i} |g(\mathbf{x}^{L,s})|$, we have

$$U^{l,s} \leq \frac{m_2 \kappa h}{1 - m_1 \kappa h} (U^{l-1,s})^2 \leq \left(\frac{m_2 \kappa h}{1 - m_1 \kappa h}\right)^{2^{l-1}-1} (U^{1,s})^{2^{l-1}} \leq \left(\frac{m_2 \kappa h}{1 - m_1 \kappa h}\right)^{2^{l-1}-1} \left(\frac{m_0 \kappa h}{1 - m_1 \kappa h}\right)^{2^{l-1}}.$$

Therefore, we can inductively check that for $s = 0, \dots, S-1$, $l = 1, \dots, L$, $i = 1, \dots, I$,

$$\begin{aligned} \mathbf{x}^{L,s}, \hat{\mathbf{x}}^{L,s} & \in \mathcal{B}(\mathcal{K}, (s-1)|h|\mu m), \quad \mathbf{v}_i^{l,s}, \hat{\mathbf{v}}_i^{l,s} \in \mathcal{B}(\mathcal{K}, (s-1)|h|\mu m + |h|\kappa m / (1 - m_1 \kappa h)), \\ m_0 & \leq m, \quad m_1 \leq 2m/r. \end{aligned}$$

Denote $V^{l,s} = \max_{1 \leq i \leq I} \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^{l,s}\|$, $X^s = \|\mathbf{x}^{L,s} - \hat{\mathbf{x}}^s\|$, we have

$$\begin{aligned} & \|g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s}) - \hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\hat{\mathbf{v}}_j^{l,s} - \hat{\mathbf{v}}_j^{l-1,s})\| \\ & \leq \|g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s}) - \hat{g}'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s})\| \\ & \quad + \|\hat{g}'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s}) - \hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s})\| \\ & \quad + \|\hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s}) - \hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\hat{\mathbf{v}}_j^{l,s} - \hat{\mathbf{v}}_j^{l-1,s})\| \\ & \leq \|g - \hat{g}\|_{\mathcal{B}(\mathcal{K}, (s-1)|h|\mu m + 2|h|\kappa m / (1 - m_1 \kappa |h|))} + \frac{4|h|\kappa m^2}{r^2(1 - m_1 \kappa |h|)} V^{l-1,s} + \frac{2m}{r} (V^{l-1,s} + V^{l,s}) \\ & \leq \|g - \hat{g}\|_{\mathcal{B}(\mathcal{K}, (s-1)|h|\mu m + 3|h|\kappa m)} + \frac{m}{r} V^{l-1,s} + \frac{2m}{r} (V^{l-1,s} + V^{l,s}), \end{aligned}$$

where the last inequality holds by the fact that $3\kappa m|h| \leq r/2$.

Subsequently, we deduce that

$$\begin{aligned} & \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^{l,s}\| \\ & \leq X^s + |h| \sum_{j=1}^s |a_{ij}| (\|g(\mathbf{v}_j^{l-1,s}) - \hat{g}(\hat{\mathbf{v}}_j^{l-1,s})\| + \|g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s}) - \hat{g}'(\hat{\mathbf{v}}_j^{l-1,s})(\hat{\mathbf{v}}_j^{l,s} - \hat{\mathbf{v}}_j^{l-1,s})\|) \\ & \leq |h|\kappa \frac{5m}{r} V^{l-1,s} + |h|\kappa \frac{2m}{r} V^{l,s} + 2|h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (s-1)|h|\mu m + 3|h|\kappa m)} + X^s. \end{aligned}$$

Thus we obtain

$$V^{l,s} \leq \frac{|h|\kappa \frac{5m}{r}}{1 - |h|\kappa \frac{2m}{r}} V^{l-1,s} + \frac{|h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (s-1)|h|\mu m + 3|h|\kappa m)} + X^s}{1 - |h|\kappa \frac{2m}{r}}.$$

As a result, we have

$$(B.7) \quad V^{L,s} \leq \frac{1}{1 - |h|\kappa\frac{7m}{r}} X^s + \frac{1}{1 - |h|\kappa\frac{7m}{r}} |h|\kappa \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu+3\kappa)m|h|)}.$$

Due to the similarity of estimates (B.2) and (B.7), it is now possible to carry over the results of fixed-point iteration to Newton-Raphson iteration. Here, the analogous estimates are given as

$$\begin{aligned} X^S &\leq (e-1)(S\mu + \kappa)|h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu+3\kappa)m|h|)}, \\ \|\hat{g} - g\|_{\mathcal{K}} &\leq \frac{\|(\Phi_{h,\hat{g}}^L)^S - (\Phi_{h,g}^L)^S\|_{\mathcal{K}}}{S|h|} + \frac{(e-1)(\mu + \kappa/S)|h| \|\hat{g} - g\|_{\mathcal{B}(\mathcal{K}, (S\mu+3\kappa)h_1m)}}{h_1 - |h|}. \end{aligned}$$

The proof is completed. ■

B.2. Proof of Lemma 3.2 (The M -step shooting loss and the teacher-forcing loss have equivalent convergence). In the following, we seek to prove a double inequality of the form $c_1A \leq B \leq c_2A$, and, broadly speaking, do this by showing (1) that $B \leq c_2A$, and (2) that $A \leq c_3B$ with $c_1 = 1/c_3$.

Proof. Denote by C_L the Lipschitz constant of $(\Phi_{h,f_\theta}^L)^s$, we have

$$\begin{aligned} \sum_{\mathbf{x} \in \mathcal{T}} \|(\Phi_{h,f_\theta}^L)^s(\mathbf{x}) - \phi_{sh,f}(\mathbf{x})\|_2^2 &= \sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h,f_\theta}^L)^s \circ \phi_{(m-1)\Delta t, f}(\mathbf{x}_n) - \phi_{m\Delta t, f}(\mathbf{x}_n)\|_2^2 \\ &\leq \sum_{n=1}^N \sum_{m=1}^M 2\|(\Phi_{h,f_\theta}^L)^s \circ \phi_{(m-1)\Delta t, f}(\mathbf{x}_n) - (\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n)\|_2^2 + 2\|(\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n) - \phi_{m\Delta t, f}(\mathbf{x}_n)\|_2^2 \\ &\leq 2(C_L^2 + 1) \cdot M^2 \cdot \sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n) - \phi_{m\Delta t, f}(\mathbf{x}_n)\|_2^2 / m^2. \end{aligned}$$

In addition,

$$\begin{aligned} &\|(\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n) - \phi_{m\Delta t, f}(\mathbf{x}_n)\|_2^2 / m^2 \\ &\leq \sum_{i=0}^{m-1} \|(\Phi_{h,f_\theta}^L)^{(m-i)s} \circ \phi_{i\Delta t, f}(\mathbf{x}_n) - (\Phi_{h,f_\theta}^L)^{(m-i-1)s} \circ \phi_{(i+1)\Delta t, f}(\mathbf{x}_n)\|_2^2 \\ &\leq \sum_{i=0}^{m-1} C_L^{2(m-i-1)} \|(\Phi_{h,f_\theta}^L)^s \circ \phi_{i\Delta t, f}(\mathbf{x}_n) - \phi_{(i+1)\Delta t, f}(\mathbf{x}_n)\|_2^2 \\ &\leq \sum_{i=1}^M C_L^{2(M-1)} \|(\Phi_{h,f_\theta}^L)^s \circ \phi_{(i-1)\Delta t, f}(\mathbf{x}_n) - \phi_{i\Delta t, f}(\mathbf{x}_n)\|_2^2. \end{aligned}$$

Therefore, we conclude that

$$\begin{aligned}
 & \sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h,f_\theta}^L)^{ms}(\mathbf{x}_n) - \phi_{m\Delta t, f}(\mathbf{x}_n)\|_2^2 / m^2 \\
 \text{(B.8)} \quad & \leq \sum_{n=1}^N \sum_{m=1}^M \sum_{i=1}^M C_L^{2(M-1)} \|(\Phi_{h,f_\theta}^L)^s \circ \phi_{(i-1)\Delta t, f}(\mathbf{x}_n) - \phi_{i\Delta t, f}(\mathbf{x}_n)\|_2^2 \\
 & \leq C_L^{2(M-1)} \cdot M \cdot \sum_{\mathbf{x} \in \mathcal{T}} \|(\Phi_{h,f_\theta}^L)^s(\mathbf{x}) - \phi_{sh, f}(\mathbf{x})\|_2^2.
 \end{aligned}$$

The proof is completed. ■

B.3. Proof of Theorem 3.3 (Increasing the iteration number L is equivalent to adjusting the approximation target to gradually approach the true target). We first demonstrate the convergence of both fixed-point iteration (2.5) and Newton-Raphson iteration (2.6) for multiple compositions, which will be also used for the proof of Lemma 4.1.

Lemma B.5 (Multiple compositions of fixed-point iteration converges). Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4) and its approximation using fixed-point iteration Φ_h^L (2.5). Denote

$$\mu = \sum_{i=1}^I |b_i| \quad \kappa = \max_{1 \leq i \leq I} \sum_{j=1}^I |a_{ij}|.$$

Then, for any continuously differentiable g and initial value \mathbf{x} , there exists remainder term $R = \mathcal{O}(h^{L+3})$ such that

$$\|(\Phi_{h,g}^L)^S(\mathbf{x}) - (\Phi_{h,g})^S(\mathbf{x})\|_\infty \leq S \|g(\mathbf{x})\| \|g'(\mathbf{x})\|^{L+1} \mu \kappa^{L+1} h^{L+2} + R.$$

Proof. The solution of $(\Phi_{h,g})^S(\mathbf{x})$ and $(\Phi_{h,g}^L)^S(\mathbf{x})$ with initial value \mathbf{x} are respectively given by

$$\begin{cases} \mathbf{x}^0 = \mathbf{x}, \\ \mathbf{v}_i^s = \mathbf{x}^s + h \sum_{j=1}^I a_{ij} g(\mathbf{v}_j^s), \\ \mathbf{x}^{s+1} = \mathbf{x}^s + h \sum_{i=1}^I b_i g(\mathbf{v}_i^s), \end{cases} \quad \begin{cases} \mathbf{x}^{L,0} = \mathbf{x}, \quad \mathbf{v}_i^{0,s} = \mathbf{x}^{L,s}, \\ \mathbf{v}_i^{l,s} = \mathbf{x}^{L,s} + h \sum_{j=1}^I a_{ij} g(\mathbf{v}_j^{l-1,s}), \\ \mathbf{x}^{L,s+1} = \mathbf{x}^{L,s} + h \sum_{i=1}^I b_i g(\mathbf{v}_i^{L,s}), \end{cases}$$

where $s = 0, \dots, S-1$, $l = 1, \dots, L$, $i = 1, \dots, I$ and $(\Phi_{h,g})^S(\mathbf{x}) = \mathbf{x}^S$, $(\Phi_{h,g}^L)^S(\mathbf{x}) = \mathbf{x}^{L,S}$. Denote $V^{l,s} = \max_i \|\mathbf{v}_i^{l,s} - \mathbf{v}_i^s\|$, we have

$$\|\mathbf{v}_i^{l,s} - \mathbf{v}_i^s\| \leq m_1 \kappa h \cdot V^{l-1,s} + \|\mathbf{x}^{L,s} - \mathbf{x}^s\|,$$

where $m_1 = \max_{s,l,i} \|g(\mathbf{v}_i^s) - g(\mathbf{v}_i^{l,s})\| / \|\mathbf{v}_i^s - \mathbf{v}_i^{l,s}\|$. As a result,

$$\begin{aligned} V^{L,s} &\leq (m_1 \kappa h)^L \cdot V^{0,s} + \frac{1 - (m_1 \kappa h)^L}{1 - m_1 \kappa h} \|\mathbf{x}^{L,s} - \mathbf{x}^s\| \\ &\leq (m_1 \kappa h)^L \cdot m_0 \kappa h + \frac{1}{1 - m_1 \kappa h} \|\mathbf{x}^{L,s} - \mathbf{x}^s\|, \end{aligned}$$

where $m_0 = \max_{s,i} |g(\mathbf{v}_i^s)|$. In addition, we deduce that

$$\begin{aligned} \|\mathbf{x}^{L,s+1} - \mathbf{x}^{s+1}\| &\leq \|\mathbf{x}^{L,s} - \mathbf{x}^s\| + m_1 \mu h \cdot V^{L,s} \\ &\leq \left(1 + \frac{m_1 \mu h}{1 - m_1 \kappa h}\right) \|\mathbf{x}^{L,s} - \mathbf{x}^s\| + m_0 m_1^{L+1} \mu \kappa^{L+1} h^{L+2}. \end{aligned}$$

Finally, we obtain that

$$\begin{aligned} \|\mathbf{x}^{L,S} - \mathbf{x}^S\| &\leq \left(1 + \frac{m_1 \mu h}{1 - m_1 \kappa h}\right)^S \|\mathbf{x}^{L,0} - \mathbf{x}^0\| + \frac{\left(1 + \frac{m_1 \mu h}{1 - m_1 \kappa h}\right)^S - 1}{\frac{m_1 \mu h}{1 - m_1 \kappa h}} m_1^{L+1} m_0 \mu \kappa^{L+1} h^{L+2} \\ &\leq \frac{\left(1 + \frac{m_1 \mu h}{1 - m_1 \kappa h}\right)^S - 1}{\frac{m_1 \mu h}{1 - m_1 \kappa h}} \cdot m_0 m_1^{L+1} \mu \kappa^{L+1} h^{L+2} \\ &= S \|g(\mathbf{x})\| \|g'(\mathbf{x})\|^{L+1} \mu \kappa^{L+1} h^{L+2} + \mathcal{O}(h^{L+3}). \end{aligned}$$

The proof is complete. ■

Lemma B.6 (Multiple compositions of Newton-Raphson iteration converges.) *Consider a consistent implicit Runge-Kutta scheme Φ_h (2.4) and its approximation using Newton-Raphson iteration Φ_h^L (2.6). Then, for any twice continuously differentiable g and initial value \mathbf{x} , there exist remainder term $R = \mathcal{O}(h^{2L+1+1})$ such that*

$$\|(\Phi_{h,g}^L)^S(\mathbf{x}) - (\Phi_{h,g})^S(\mathbf{x})\|_\infty \leq S \|g'(\mathbf{x})\| \left(\frac{\|g''(\mathbf{x})\|}{1 - \|g'(\mathbf{x})\| \kappa h}\right)^{2L-1} \|g(\mathbf{x})\|^{2L} \mu \kappa^{2L+1-1} h^{2L+1} + R,$$

where μ and κ are constants defined in Lemma B.5.

Proof. The solution of $(\Phi_{h,g})^S(\mathbf{x})$ and $(\Phi_{h,g}^L)^S(\mathbf{x})$ with initial value \mathbf{x} are respectively given by

$$\begin{cases} \mathbf{x}^0 = \mathbf{x}, \\ \mathbf{v}_i^s = \mathbf{x}^s + h \sum_{j=1}^I a_{ij} g(\mathbf{v}_j^s), \\ \mathbf{x}^{s+1} = \mathbf{x}^s + h \sum_{i=1}^I b_i g(\mathbf{v}_i^s), \end{cases} \quad \begin{cases} \mathbf{x}^{L,0} = \mathbf{x}, \quad \mathbf{v}_i^{0,s} = \mathbf{x}^{L,s}, \\ \mathbf{v}_i^{l,s} = \mathbf{x}^{L,s} + h \sum_{j=1}^I a_{ij} (g(\mathbf{v}_j^{l-1,s}) + g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l,s} - \mathbf{v}_j^{l-1,s})), \\ \mathbf{x}^{L,s+1} = \mathbf{x}^{L,s} + h \sum_{i=1}^I b_i g(\mathbf{v}_i^{L,s}), \end{cases}$$

where $s = 0, \dots, S-1$, $l = 1, \dots, L$, $i = 1, \dots, I$ and $(\Phi_{h,g})^S(\mathbf{x}) = \mathbf{x}^S$, $(\Phi_{h,g}^L)^S(\mathbf{x}) = \mathbf{x}^{L,S}$.

Let

$$\hat{\mathbf{v}}_i^s = \mathbf{x}^{L,s} + h \sum_{j=1}^I a_{ij} g(\hat{\mathbf{v}}_j^s), \text{ for } i = 1, \dots, I, \quad s = 0, \dots, S-1,$$

and $m_0 = \max_{s,i} |g(\hat{\mathbf{v}}_i^s)|$, $m_1 = \max_{s,l,i} \|g'(\mathbf{v}_j^{l-1,s})\|$, $m_2 = \max_{s,l,i} \sup_{\theta \in [0,1]} \|g''(\theta \mathbf{v}_j^{l-1,s} + (1-\theta)\hat{\mathbf{v}}_j^s)\|/2$, $\hat{V}^{l,s} = \max_i \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^s\|$, we have that

$$\begin{aligned} \|\mathbf{v}_i^{l,s} - \hat{\mathbf{v}}_i^s\| &\leq h \sum_{j=1}^I |a_{ij}| \|g(\mathbf{v}_j^{l-1,s}) + g'(\mathbf{v}_j^{l-1,s})(\hat{\mathbf{v}}_j^s - \mathbf{v}_j^{l-1,s}) - g(\hat{\mathbf{v}}_j^s) + g'(\mathbf{v}_j^{l-1,s})(\mathbf{v}_j^{l-1,s} - \hat{\mathbf{v}}_j^s)\| \\ &\leq m_2 \kappa h (\hat{V}^{l-1,s})^2 + m_1 \kappa h \hat{V}^{l,s}, \end{aligned}$$

which implies that $\hat{V}^{l,s} \leq (\frac{m_2 \kappa h}{1 - m_1 \kappa h})^{2^l - 1} (m_0 \kappa h)^{2^l}$. Let $\tilde{m}_1 = \max_{s,i} \|g(\hat{\mathbf{v}}_i^s) - g(\mathbf{v}_i^s)\|/\|\hat{\mathbf{v}}_i^s - \mathbf{v}_i^s\|$ and $\tilde{V}^s = \max_i \|\hat{\mathbf{v}}_i^s - \mathbf{v}_i^s\|$, we have that

$$\|\hat{\mathbf{v}}_i^s - \mathbf{v}_i^s\| \leq \|\mathbf{x}^{L,s} - \mathbf{x}^s\| + \tilde{m}_1 \kappa h \tilde{V}^s,$$

which implies that $\tilde{V}^s \leq \|\mathbf{x}^{L,s} - \mathbf{x}^s\|/(1 - \tilde{m}_1 \kappa h)$. Therefore, we conclude that

$$V^{l,s} \leq \hat{V}^{l,s} + \tilde{V}^s \leq (\frac{m_2 \kappa h}{1 - m_1 \kappa h})^{2^l - 1} (m_0 \kappa h)^{2^l} + \frac{1}{1 - \tilde{m}_1 \kappa h} \|\mathbf{x}^{L,s} - \mathbf{x}^s\|.$$

In addition, similarly to [Lemma B.5](#), we have that

$$\begin{aligned} \|\mathbf{x}^{L,s+1} - \mathbf{x}^{s+1}\| &\leq \|\mathbf{x}^{L,s} - \mathbf{x}^s\| + m_1 \mu h \cdot V^{L,s} \\ &\leq (1 + \frac{m_1 \mu h}{1 - \tilde{m}_1 \kappa h}) \|\mathbf{x}^{L,s} - \mathbf{x}^s\| + m_1 (\frac{m_2}{1 - m_1 \kappa h})^{2^L - 1} m_0^{2^L} \mu \kappa^{2^{L+1} - 1} h^{2^{L+1}}. \end{aligned}$$

and thus

$$\begin{aligned} &\|\mathbf{x}^{L,S} - \mathbf{x}^S\| \\ &\leq (1 + \frac{m_1 \mu h}{1 - \tilde{m}_1 \kappa h})^S \|\mathbf{x}^{L,0} - \mathbf{x}^0\| + \frac{(1 + \frac{m_1 \mu h}{1 - \tilde{m}_1 \kappa h})^S - 1}{\frac{m_1 \mu h}{1 - \tilde{m}_1 \kappa h}} m_1 (\frac{m_2}{1 - m_1 \kappa h})^{2^L - 1} m_0^{2^L} \mu \kappa^{2^{L+1} - 1} h^{2^{L+1}} \\ &\leq S \|g'(\mathbf{x})\| (\frac{\|g''(\mathbf{x})\|}{1 - \|g'(\mathbf{x})\| \kappa h})^{2^L - 1} \|g(\mathbf{x})\|^{2^L} \mu \kappa^{2^{L+1} - 1} h^{2^{L+1}} + \mathcal{O}(h^{2^{L+1} + 1}), \end{aligned}$$

which completes the proof. ■

We next present the proof of [Theorem 3.3](#).

Proof of Theorem 3.3. We first prove that the statement holds for fixed-point iteration by induction. First, the case when $k = 0$ is obvious since $f = \hat{f}_0 = f_0$. Suppose now that $\hat{f}_k = f_k$ for $0 \leq k \leq K \leq L - 1$, then

$$\hat{f}_h^K = \sum_{k=0}^K h^k \hat{f}_k = \sum_{k=0}^K h^k f_k = f_h^K$$

By Lemma B.5, we have

$$(B.10) \quad \Phi_{h, \hat{f}_h^K} - \Phi_{h, f_h^K}^L = \mathcal{O}(h^{L+2}).$$

We rewrite the calculation procedure of IMDE as

$$\begin{aligned} \phi_{h,f} - \Phi_{h, \hat{f}_h^K} &= h^{K+2} \hat{f}_{K+1} + \mathcal{O}(h^{K+3}), \\ \phi_{h,f} - \Phi_{h, f_h^K}^L &= h^{K+2} f_{K+1} + \mathcal{O}(h^{K+3}). \end{aligned}$$

Subtracting above two equations and substituting (B.10), we conclude that $\hat{f}_{K+1} = f_{K+1}$, which completes the induction.

In addition, for Newton-Raphson iteration, by Lemma B.6, repeating the above induction implies $\hat{f}_k = f_k$ for $0 \leq k \leq 2^{L+1} - 2$. The proof is completed. ■

B.4. Proof of Theorem 3.4 (Order of convergence for learning ODEs).

Proof. The proof is a direct consequence of Theorem 3.1, Theorem 3.3 and the following Lemma. ■

Lemma B.7 (IMDE power series for a p^{th} order integrator has first error term of order h^p). Suppose that the integrator $\Phi_h(\mathbf{x})$ with discrete step h is of order $p \geq 1$, then, the IMDE obeys

$$\frac{d}{dt} \tilde{\mathbf{y}} = f_h(\tilde{\mathbf{y}}) = f(\tilde{\mathbf{y}}) + h^p f_p(\tilde{\mathbf{y}}) + \dots$$

Proof. The proof can be found in [55]. ■

B.5. Proof of Lemma 4.1 (Convergence of the (“inner”) implicit iteration).

Proof. By Minkowski’s inequality, we obtain that for neural network f_θ ,

$$\begin{aligned} \mathcal{L}_{exact}^{\frac{1}{2}} &\leq \mathcal{L}_{unrolled}^{\frac{1}{2}} + \mathcal{R}_L, \\ \mathcal{L}_{exact}^{\frac{1}{2}} &\leq \mathcal{L}_{unrolled}^{\frac{1}{2}} + \left(\sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h, f_\theta}^L(\mathbf{x}_n))^{ms} - (\Phi_{h, f_\theta}^{L+1}(\mathbf{x}_n))^{ms}\|_2^2 / (m\Delta t)^2 \right)^{\frac{1}{2}} + \mathcal{R}_{L+1}, \end{aligned}$$

where

$$\mathcal{R}_L = \left(\sum_{n=1}^N \sum_{m=1}^M \|(\Phi_{h, f_\theta}^L(\mathbf{x}_n))^{ms} - (\Phi_{h, f_\theta}(\mathbf{x}_n))^{ms}\|_2^2 / (m\Delta t)^2 \right)^{\frac{1}{2}}.$$

According to Lemma B.5 and Lemma B.6, we have $\mathcal{R}_L = \mathcal{O}(h^{L^*+1})$ where $L^* = L$ for the unrolled approximation using fixed-point iteration (2.5) and $L^* = 2^{L+1} - 2$ for the unrolled approximation using Newton-Raphson iteration (2.6) and thus complete the proof. ■

REFERENCES

- [1] L. B. ALMEIDA, *A learning rule for asynchronous perceptrons with feedback in a combinatorial environment*, in IEEE First International Conference on Neural Networks, IEEE, 1987, pp. 608–618.
- [2] J. ANDERSON, I. KEVREKIDIS, AND R. RICO-MARTINEZ, *A comparison of recurrent training algorithms for time series analysis and system identification*, Computers & chemical engineering, 20 (1996), pp. S751–S756.
- [3] S. BAI, J. Z. KOLTER, AND V. KOLTUN, *Deep equilibrium models*, in 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 2019, pp. 688–699.
- [4] S. BAI, V. KOLTUN, AND J. Z. KOLTER, *Neural deep equilibrium solvers*, in International Conference on Learning Representations, 2022.
- [5] J. BEHRMANN, W. GRATHWOHL, R. T. Q. CHEN, D. DUVENAUD, AND J. JACOBSEN, *Invertible residual networks*, in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, vol. 97, PMLR, 2019, pp. 573–582.
- [6] T. BERTALAN, F. DIETRICH, I. MEZÍČ, AND I. G. KEVREKIDIS, *On learning hamiltonian systems from data*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 29 (2019), p. 121107.
- [7] A. BOTEV, A. JAEGLER, P. WIRNSBERGER, D. HENNES, AND I. HIGGINS, *Which priors matter? benchmarking models for learning latent dynamics*, in 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks, 2021.
- [8] S. L. BRUNTON, J. L. PROCTOR, AND J. N. KUTZ, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the national academy of sciences, 113 (2016), pp. 3932–3937.
- [9] P. CHARTIER, E. HAIRER, AND G. VILMART, *Algebraic structures of b-series*, Foundations of Computational Mathematics, 10 (2010), pp. 407–427.
- [10] R. CHEN AND M. TAO, *Data-driven prediction of general hamiltonian dynamics via learning exactly-symplectic maps*, in Proceedings of the 38th International Conference on Machine Learning (ICML 2021), vol. 139, PMLR, 2021, pp. 1717–1727.
- [11] T. CHEN, Y. RUBANOVA, J. BETTENCOURT, AND D. DUVENAUD, *Neural ordinary differential equations*, in 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018, pp. 6572–6583.
- [12] Z. CHEN, J. ZHANG, M. ARJOVSKY, AND L. BOTTOU, *Symplectic recurrent neural networks*, in 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 2020.
- [13] B. C. DANIELS AND I. NEMENMAN, *Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression*, Plos One, 10 (2014).
- [14] D. T. DONCEVIC, A. MITSOS, Y. GUO, Q. LI, F. DIETRICH, M. DAHMEN, AND I. G. KEVREKIDIS, *A recursively recurrent neural network (R2N2) architecture for learning iterative algorithms*, 2022, <https://arxiv.org/abs/2211.12386>.
- [15] Q. DU, Y. GU, H. YANG, AND C. ZHOU, *The discovery of dynamics via linear multistep methods and deep learning: error estimation*, SIAM Journal on Numerical Analysis, 60 (2022), pp. 2014–2045.
- [16] T. EIROLA, *Aspects of backward error analysis of numerical ODEs*, Journal of Computational and Applied Mathematics, 45 (1993), pp. 65–73.
- [17] L. EL GHAOU, F. GU, B. TRAVACCA, A. ASKARI, AND A. TSAI, *Implicit deep learning*, SIAM Journal on Mathematics of Data Science, 3 (2021), pp. 930–958.
- [18] K. FENG, *Formal power series and numerical algorithms for dynamical systems*, in Proceedings of international conference on scientific computation, Hangzhou, China, Series on Appl. Math. Singapore: World Scientific, vol. 1, 1991, pp. 28–35.
- [19] K. FENG, *Formal dynamical systems and numerical algorithms*, SERIES ON APPLIED MATHEMATICS, 4 (1993), pp. 1–10.
- [20] Z. GENG, X.-Y. ZHANG, S. BAI, Y. WANG, AND Z. LIN, *On training implicit models*, in Advances in Neural Information Processing Systems, 2021.
- [21] R. GONZÁLEZ-GARCÍA, R. RICO-MARTINEZ, AND I. G. KEVREKIDIS, *Identification of distributed parameter systems: A neural net based approach*, Computers & chemical engineering, 22 (1998), pp. S965–S968.
- [22] S. GREYDANUS, M. DZAMBA, AND J. YOSINSKI, *Hamiltonian neural networks*, in Advances in Neural Information Processing Systems 32, 2019, pp. 15353–15363.
- [23] Y. GUO, F. DIETRICH, T. S. BERTALAN, D. T. DONCEVIC, M. DAHMEN, I. G. KEVREKIDIS, AND Q. LI, *Personalized algorithm generation: A case study in learning ODE integrators*, SIAM J. Sci. Comput.,

- 44 (2021), pp. 1911–.
- [24] E. HAIRER AND C. LUBICH, *The life-span of backward error analysis for numerical integrators*, *Numerische Mathematik*, 76 (1997), pp. 441–462.
- [25] E. HAIRER, C. LUBICH, AND G. WANNER, *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31, Springer Science & Business Media, 2006.
- [26] E. HAIRER AND G. WANNER, *Solving ordinary differential equations II*, vol. 375, Springer Berlin Heidelberg, 1996.
- [27] P. HU, W. YANG, Y. ZHU, AND L. HONG, *Revealing hidden dynamics from time-series data by ODENet*, *Journal of Computational Physics*, 461 (2022), p. 111203.
- [28] Z. HUANG, S. BAI, AND J. Z. KOLTER, *(Implicit)²: Implicit layers for implicit representations*, 35th Conference on Neural Information Processing Systems (NeurIPS 2021), 34 (2021).
- [29] I. HUH, E. YANG, S. J. HWANG, AND J. SHIN, *Time-reversal symmetric ODE network*, in 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 2020.
- [30] P. JIN, Z. ZHANG, A. ZHU, Y. TANG, AND G. E. KARNIADAKIS, *Sympnets: Intrinsic structure-preserving symplectic networks for identifying hamiltonian systems*, *Neural Networks*, 132 (2020), pp. 166–179.
- [31] R. T. KELLER AND Q. DU, *Discovery of dynamics using linear multistep methods*, *SIAM Journal on Numerical Analysis*, 59 (2021), pp. 429–455.
- [32] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, in 3rd International Conference on Learning Representations, 2014.
- [33] R. J. LOVELETT, J. L. AVALOS, AND I. G. KEVREKIDIS, *Partial observations and conservation laws: Gray-box modeling in biotechnology and optogenetics*, *Industrial & Engineering Chemistry Research*, 59 (2020), pp. 2611–2620, <https://doi.org/10.1021/acs.iecr.9b04507>, <https://doi.org/10.1021/acs.iecr.9b04507>, <https://arxiv.org/abs/https://doi.org/10.1021/acs.iecr.9b04507>.
- [34] F. LU, M. ZHONG, S. TANG, AND M. MAGGIONI, *Nonparametric inference of interaction laws in systems of agents from trajectory data*, *Proceedings of the National Academy of Sciences*, 116 (2019), pp. 14424–14433.
- [35] A. PAL, Y. MA, V. B. SHAH, AND C. V. RACKAUCKAS, *Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics*, in Proceedings of the 38th International Conference on Machine Learning, ICML 2021, vol. 139, PMLR, 2021, pp. 8325–8335.
- [36] F. J. PINEDA, *Generalization of back-propagation to recurrent neural networks*, *Physical Review Letters*, 59 (1987), pp. 2229–2232, <https://doi.org/10.1103/physrevlett.59.2229>, <https://doi.org/10.1103/physrevlett.59.2229>.
- [37] M. POLI, S. MASSAROLI, A. YAMASHITA, H. ASAMA, AND J. PARK, *Hypersolvers: Toward fast continuous-depth models*, in 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada., 2020.
- [38] I. G. K. R. RICO-MARTÍNEZ AND K. KRISCHER, *Nonlinear system identification using neural networks: dynamics and instabilities*, Elsevier Science, 1995, ch. 16.
- [39] M. RAISSI AND G. E. KARNIADAKIS, *Hidden physics models: Machine learning of nonlinear partial differential equations*, *Journal of Computational Physics*, 357 (2018), pp. 125–141.
- [40] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Multistep neural networks for data-driven discovery of nonlinear dynamical systems*, arXiv preprint arXiv:1801.01236, (2018).
- [41] S. REICH, *Backward error analysis for numerical integrators*, *SIAM Journal on Numerical Analysis*, 36 (1999), pp. 1549–1570.
- [42] R. RICO-MARTÍNEZ, J. ANDERSON, AND I. KEVREKIDIS, *Continuous-time nonlinear signal processing: a neural network based approach for gray box identification*, in Proceedings of IEEE Workshop on Neural Networks for Signal Processing, IEEE, 1994, pp. 596–605.
- [43] R. RICO-MARTÍNEZ AND I. G. KEVREKIDIS, *Continuous time modeling of nonlinear systems: A neural network-based approach*, in IEEE International Conference on Neural Networks, IEEE, 1993, pp. 1522–1525.
- [44] R. RICO-MARTÍNEZ, K. KRISCHER, I. KEVREKIDIS, M. KUBE, AND J. HUDSON, *Discrete- vs. continuous-time nonlinear signal processing of cu electrodisolution data*, *Chemical Engineering Communications*, 118 (1992), pp. 25–48, <https://doi.org/10.1080/00986449208936084>.
- [45] Y. SAAD AND M. H. SCHULTZ., *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, *SIAM Journal on scientific and statistical computing*, 7 (1986), pp. 856–869.

- [46] J. M. SANZ-SERNA, *Symplectic integrators for hamiltonian problems: an overview*, Acta numerica, 1 (1992), pp. 243–286.
- [47] M. SCHMIDT AND H. LIPSON, *Distilling free-form natural laws from experimental data*, Science, 324 (2009), pp. 81–85.
- [48] P. TOTH, D. J. REZENDE, A. JAEGLE, S. RACANIÈRE, A. BOTEV, AND I. HIGGINS, *Hamiltonian generative networks*, in 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 2020.
- [49] R. J. WILLIAMS AND D. ZIPSER, *A learning algorithm for continually running fully recurrent neural networks*, Neural computation, 1 (1989), pp. 270–280.
- [50] K. WU, T. QIN, AND D. XIU, *Structure-preserving method for reconstructing unknown hamiltonian systems from trajectory data*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3704–A3729.
- [51] K. WU AND D. XIU, *Numerical aspects for approximating governing equations using data*, Journal of Computational Physics, 384 (2019), pp. 200–221.
- [52] H. YOSHIDA, *Recent progress in the theory and application of symplectic integrators*, Qualitative and Quantitative Behaviour of Planetary Systems, (1993), pp. 27–43.
- [53] H. YU, X. TIAN, E. WEINAN, AND Q. LI, *Onsagernet: Learning stable and interpretable dynamics using a generalized onlager principle*, Physical Review Fluids, 6 (2021), p. 114402.
- [54] Z. ZHANG, Y. SHIN, AND G. E. KARNIADAKIS, *Gfinns: Generic formalism informed neural networks for deterministic and stochastic dynamical systems*, arXiv preprint arXiv:2109.00092, (2021).
- [55] A. ZHU, P. JIN, B. ZHU, AND Y. TANG, *On numerical integration in neural ordinary differential equations*, in Proceedings of the 39th International Conference on Machine Learning (ICML 2022), vol. 162, PMLR, 2022, pp. 27527–27547.