IEICE
TRANSACTIONS

on Information and Systems

DOI:10.1587/transinf.2023EDL8064

Publicized:2024/07/22

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

LETTER

Loss Function for Deep Learning to Model Dynamical Systems

Takahito YOSHIDA[†a)], **Takaharu YAGUCHI**[††b)], ***Nonmembers*, *and* Takashi MATSUBARA**[†††c)], ***Member***

SUMMARY Accurately simulating physical systems is essential in various fields. In recent years, deep learning has been used to automatically build models of such systems by learning from data. One such method is the neural ordinary differential equation (neural ODE), which treats the output of a neural network as the time derivative of the system states. However, while this and related methods have shown promise, their training strategies still require further development. Inspired by error analysis techniques in numerical analysis while replacing numerical errors with modeling errors, we propose the error-analytic strategy to address this issue. Therefore, our strategy can capture long-term errors and thus improve the accuracy of long-term predictions.

key words: *Deep learning, physical system, partial differential equation, numerical error analysis*

1. Introduction

Neural networks have been used to automatically model various real-world systems, including natural phenomena, human interactions, and industrial machinery. In particular, continuous-time dynamical systems are often described using ordinary differential equations (ODEs) $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u} = f(t, \boldsymbol{u})$ that specify the time t and system state $\boldsymbol{u}$. Recently, a model called n eural ODE has gained popularity among researchers in this field [1], as it combines neural networks to model the time-derivative f with advanced numerical integrators and adjoint methods. In the field of signal processing, several studies have extensively investigated the errors that occur in neural networks for ODEs [2]–[4], albeit they do not apply these insights to learning algorithms.

If pairs of states $\boldsymbol{u}$ and the corresponding time-derivatives $f(t, \boldsymbol{u})$ are given, one can train a neural network f_{NN} such that its output $f_{NN}(t, \boldsymbol{u})$ approaches the time-derivative $f(t, \boldsymbol{u})$ [5]; this is essentially a regression problem. If only initial condition $\boldsymbol{u}(0)$ and terminal condition $\boldsymbol{u}(T)$ are given, one can solve ODE $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u} = f_{NN}(t, \boldsymbol{u})$ defined using neural network f_{NN} from initial condition $\boldsymbol{u}(0)$, obtain the terminal value $\boldsymbol{u}_{NN}(T)$, and then update the parameters such that the terminal value $\boldsymbol{u}_{NN}(T)$ approaches the given terminal condition $\boldsymbol{u}(T)$ [1]. When one considers the whole process to obtain a terminal value $\boldsymbol{u}_{NN}$ using a neural network f_{NN} as a single model, this problem is also a regression problem. Practically, many observations $\{\boldsymbol{u}(t_k)\}$ for $t_0 < t_1 < \cdots < t_K$ are available per a time-series. The same strategy applies to every pair of successive observations $\boldsymbol{u}(t_k)$ and $\boldsymbol{u}(t_{k+1})$ [6], [7]. Another possible strategy is to adjust the parameters such that the predicted states $\boldsymbol{u}_{NN}(t_k)$ at corresponding time points in the initial value problem match observations $\boldsymbol{u}(t_k)$ [8]. It is worth noting that the thorough experiments in reference [8] have verified that the multi step strategy, which captures long-term errors, improves the accuracy of long-term predictions, which led many subsequent studies to employ the same or similar strategies (e.g., [7]). However, the multi-step strategy (as well as the long-term strategy) requires a memory footprint and computational cost proportional to the length of the error calculation. Consequently, when memory usage and computational resources are balanced, a model can learn only a limited number of time series at once, increasing error variance and reducing learning efficiency. Additionally, while the multi-step strategy attempts to minimize errors at each point in time, errors accumulate over time, making it meaningless to reduce them only after they have become large.

In this study, we propose an error-analytic strategy as a new strategy for training neural ODEs to be more accurate in long-term predictions when many observations are given per a time series. The proposed strategy only performs long-term predictions virtually, allowing it to reduce variance while considering long-term errors. Additionally, our proposed method is more logical, aiming to minimize errors that could cause significant future discrepancies as soon as they occur.

Particularly, our study makes the following contributions. (1) The proposed error-analytic strategy is inspired by error analysis techniques in numerical analysis and is derived by replacing numerical errors with modeling errors. Therefore, it is reasonable to train neural networks that model ODEs than the previous strategies based on a regression problem [1], [6], [7] or RNN [8]. (2) The proposed error-analytic strategy can capture a long-term error (e.g., an error at time t_{k+M} for $M > 1$) and, hence, improve the long-term prediction performance. Hence, it can learn a higher-performing model than strategies that only consider local errors [1], [6], [7]. (3) The proposed error-analytic strategy is applied to a numerical integration only between two observations $\boldsymbol{u}(t_k)$ and $\boldsymbol{u}(t_{k+1})$. Hence, it is more memory and computationally-efficient than strategies that require long numerical integrations [8].

[†] Graduate School of Engineering Science, Osaka University

[††] Graduate School of System Informatics, Kobe University

[†††] Graduate School of Information Science and Technology, Hokkaido University

a) E-mail: yoshida@hopf.sys.es.osaka-u.ac.jp

b) E-mail: yaguchi@pearl.kobe-u.ac.jp

c) E-mail: matsubara@ist.hokudai.ac.jp

2. Background

2.1 Ordinary Differential Equation

Our target is systems described by ODEs $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u} = f(t, \boldsymbol{u})$ for time t and system state $\boldsymbol{u}$ [9]. The function f defines the time derivative $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}$ of state $\boldsymbol{u}$. We assume the state $\boldsymbol{u}$ is fully observable. We can discretize systems described by partial differential equations (PDEs) in space, obtaining systems of ODEs. If f is Lipschitz continuous, there exists a unique solution $\boldsymbol{u}(t)$ given an initial condition $\boldsymbol{u}(0) = \boldsymbol{u}_0$; $\boldsymbol{u}(t) = \boldsymbol{u}_0 + \int_0^t f(\tau, \boldsymbol{u}(\tau))\mathrm{d}\tau$. Obtaining solution $\boldsymbol{u}(t)$ from the initial condition $\boldsymbol{u}_0$ is called an initial value problem. However, it is often difficult to be solved analytically; instead, it is solved numerically. Given a set of time points $t_0 < t_1 < \cdots < t_K$, a numerical solution is typically obtained as $\tilde{\boldsymbol{u}}_{k+1} = \tilde{\boldsymbol{u}}_k + \tilde{\Phi}(f, t_k, \tilde{\boldsymbol{u}}_k, \Delta t_k)$, where $\tilde{\boldsymbol{u}}_k$ is a numerical approximation to state $\boldsymbol{u}(t_k)$ at time t_k, $\Delta t_k = t_{k+1} - t_k$ is the time step size, and $\tilde{\Phi}$ is an increment function defined using a numerical solver. The Euler and Runge-Kutta methods are widely used numerical solvers.

Let us consider a case where a set of $K+1$ observations $\{\boldsymbol{u}_k\}_{k=0}^K = \{\boldsymbol{u}(t_k)\}_{k=0}^K$ at time point t_k is given, but the underlying ODE is unknown. Learning f from data is referred to as system identification in the context of dynamical systems theory [10]. However, it is not easy when f is nonlinear; instead, many previous studies have approximated $\tilde{\Phi}$ using a discrete-time map [10].

2.2 Comparison Training Strategies

Following [8], we define a *single-step strategy*. Given a set of observations $\{\boldsymbol{u}_k\}_{k=0}^K$, the cost function to be minimized $\mathcal{C}$ is given as

$$\mathcal{C}(\boldsymbol{u}) = \frac{1}{K}\sum_{k=0}^{K-1} \mathcal{L}(\boldsymbol{u}_{k+1}, \tilde{\boldsymbol{u}}_{k+1})\Big|_{\tilde{\boldsymbol{u}}_k = \boldsymbol{u}_k}, \qquad (1)$$

where $\mathcal{L}$ denotes a loss function such as the mean squared error (MSE). In hindsight, previous studies employed a single-step strategy. However, as HNNs have shown, the performance of time-derivative approximations and short-term predictions does not often match the performance of long-term predictions owing to the accumulation of modeling errors.

To learn long-term dynamics, we considered a baseline called *long-step strategy*. From an initial condition $\boldsymbol{u}_k$, neural ODE solves an initial value problem for M steps and obtains state $\tilde{\boldsymbol{u}}_{k+M}$. Then, the cost function $\mathcal{C}$ is given as

$$\mathcal{C}(\boldsymbol{u}) = \frac{1}{K-M+1}\sum_{k=0}^{K-M} \mathcal{L}(\boldsymbol{u}_{k+M}, \tilde{\boldsymbol{u}}_{k+M})\Big|_{\tilde{\boldsymbol{u}}_k = \boldsymbol{u}_k}. \qquad (2)$$

However, a long-term prediction is inherently difficult. Let us consider the predicted results obtained through the single-step strategy and the long-step strategy in the convergence system shown in Figure 1. Although the m-th step prediction result obtained using the long-step strategy may be better than that obtained using the single-step strategy, there is a possibility that the preceding process deviates from the ground truth. Therefore, it is conceivable that the results obtained using the long-step strategy may deteriorate.

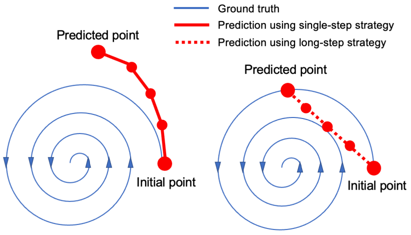
Ground truth
Prediction using single-step strategy
Prediction using long-step strategy
Predicted point
Initial point
Predicted point
Initial point

Fig. 1 Conceptual diagram of predicted results using the single-step strategy (left) and long-step strategy (right) in a convergence system. Considering the case where $m = 4$ for the long-step strategy, the results obtained using the strategy may deviate from the ground truth in the preceding steps, even if the predicted result at the m-th step is accurate.

Chen et al. (2018) proposed a *multi-step strategy* inspired by the typical training strategy of RNN. Using M-step predictions, the parameters are updated by minimizing the error at each step, that is, by minimizing

$$\mathcal{C}(\boldsymbol{u}) = \frac{1}{K-M+1}\sum_{k=0}^{K-M}\left(\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}(\boldsymbol{u}_{k+m}, \tilde{\boldsymbol{u}}_{k+m})\Big|_{\tilde{\boldsymbol{u}}_k = \boldsymbol{u}_k}\right). \qquad (3)$$

However, an M-step prediction requires M times the computation time and memory of a 1-step prediction. Although the adjoint method reduces memory consumption, it further increases computation time, and the final performance may decrease because of numerical errors during learning. Moreover, it is unclear whether the strategy for RNN is suited for neural ODE.

Contrary to these strategies, our proposed error-analytic strategy solves an initial value problem only for a single-step and dynamically weights samples based on the estimated long-term error.

3. Method

3.1 Modeling Error Analysis

In this section, we introduce the proposed error-analytic strategy for training neural ODEs. First, we analyzed the modeling error of neural ODEs. We suppose a target system to be described by an ODE $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u} = f(t, \boldsymbol{u})$, but we can observe only snapshots of states $\boldsymbol{u}_k$ at time points t_k for $k = 0, \ldots, K$. For simplicity, the time-derivative f is independent of time t, and the step size $\Delta t_k = t_{k+1} - t_k$ is constant Δt. Then, we define a discrete-time map $\Phi(\boldsymbol{u}_k) = \boldsymbol{u}_{k+1} - \boldsymbol{u}_k$.

Let f_{NN} denote a model to be trained, which can be neural ODE, HNN, and others. Combined with a numerical solver, a discrete-time map $\tilde{\Phi}(f_{NN}, t, \boldsymbol{u}, \Delta t)$ is defined. For simplicity, we refer to the discrete-time map as neural ODE and denote it by $\tilde{\Phi}_{NN}(\boldsymbol{u})$. Because the neural ODE $\tilde{\Phi}_{NN}$ is not identical to the true discrete-time map Φ, we define a remainder function $R(\boldsymbol{u}) := \tilde{\Phi}_{NN}(\boldsymbol{u}) - \Phi(\boldsymbol{u})$, which corresponds to a single-step (i.e., local) modeling error of the neural ODE $\tilde{\Phi}_{NN}$. The single-step strategy aims to minimize the remainder function R.

Our final goal is to minimize a long-term (i.e., global) modeling error. Inspired by the forward error analysis, we use the Taylor expansion to express the long-term modeling error. The states are updated by the true and modeled dynamics as $\boldsymbol{u}_{k+1} = (\Phi+I)(\boldsymbol{u}_k)$, $\tilde{\boldsymbol{u}}_{k+1} = (\Phi+R+I)(\tilde{\boldsymbol{u}}_k)$, where I denotes the identity map. In the same manner, the states after m steps are obtained as $\boldsymbol{u}_{k+m} = (\Phi+I)^m(\boldsymbol{u}_k)$, $\tilde{\boldsymbol{u}}_{k+m} = (\Phi+R+I)^m(\boldsymbol{u}_k)$. We assumed that an observation $\boldsymbol{u}_k$ at time t_k is given and set $\tilde{\boldsymbol{u}}_k = \boldsymbol{u}_k$. Let $e_{n,n+m}$ denote the m-step error from the n-th step. By definition, $e_{k,k} = 0$ and $e_{k,k+1} = (\Phi+R+I)(\boldsymbol{u}_k) - (\Phi+I)(\boldsymbol{u}_k) = R(\boldsymbol{u}_k)$. The m-step error $e_{k,k+m}$ is

$$\begin{aligned} e_{k,k+m} &= \tilde{\boldsymbol{u}}_{k+m} - \boldsymbol{u}_{k+m} \\ &= (\Phi+R+I)(\tilde{\boldsymbol{u}}_{k+m-1}) - \boldsymbol{u}_{k+m} \\ &= (\Phi+R+I)(\boldsymbol{u}_{k+m-1}) \\ &\quad + \tfrac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}}(\tilde{\boldsymbol{u}}_{k+m-1} - \boldsymbol{u}_{k+m-1}) \\ &\quad - \boldsymbol{u}_{k+m} + o(|\tilde{\boldsymbol{u}}_{k+m-1} - \boldsymbol{u}_{k+m-1}|) \\ &= \boldsymbol{u}_{k+m} + R(\boldsymbol{u}_{k+m-1}) \\ &\quad + \tfrac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}}(e_{k,k+m-1}) \\ &\quad - \boldsymbol{u}_{k+m} + o(|e_{k,k+m-1}|), \\ &= e_{k+m-1,k+m} + \tfrac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}}(e_{k,k+m-1}) \\ &\quad + o(|e_{k,k+m-1}|). \end{aligned} \tag{4}$$

At $(k+m)$-th step, a local error $e_{k+m-1,k+m}$ is added in, and the accumulated error $e_{k,k+m-1}$ is enlarged (or shrunken) by factor $\frac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}}$. By applying the above equation recursively from the zero-step error $e_{k,k}$, the m-step error $e_{k,k+m}$ can be obtained. However, the above equation requires solving an initial value problem for m steps to obtain the m-step error. Moreover, obtaining the Jacobian matrix $\frac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}}$ is computationally expensive, although recent automatic differentiation libraries make it possible [11].

3.2 Error Approximations

To reduce the computation time, we employed the following approximations. We assumed that the remainder function R is less significant compared to main dynamics $\Phi + I$. Because of the linearity of differentiation, $\frac{\partial(\Phi+R+I)}{\partial \boldsymbol{u}_{k+m-1}} = \frac{\partial(\Phi+I)}{\partial \boldsymbol{u}_{k+m-1}} + \frac{\partial R}{\partial \boldsymbol{u}_{k+m-1}} \simeq \frac{\partial(\Phi+I)}{\partial \boldsymbol{u}_{k+m-1}}$. Moreover, the denominator and numerator are approximated as finite differences in the observations. That is, $\frac{\partial(\Phi+I)}{\partial \boldsymbol{u}_{k+m-1}} = \mathrm{diag}(\frac{\boldsymbol{u}_{k+m+1} - \boldsymbol{u}_{k+m-1}}{\boldsymbol{u}_{k+m} - \boldsymbol{u}_{k+m-1}}) + I$. Let J_m denote the Jacobian matrix. Then, J_m is obtained as

$$J_{m-1} = \frac{\partial(\Phi+I)}{\partial \boldsymbol{u}_{k+m-1}} \simeq \frac{\boldsymbol{u}_{k+m+1} - \boldsymbol{u}_{k+m}}{\boldsymbol{u}_{k+m} - \boldsymbol{u}_{k+m-1}} \tag{5}$$

Table 1 Results on the CH and the KdV equations.

Model	**CH equation**	**KdV equation**
single-step	9.66×10^{-3}	3.74×10^{-1}
long-step	5.38×10^{-2} $(m = 13)$	3.73×10^{-1} $(m = 6)$
multi-step	4.29×10^{-2} $(m = 14)$	3.59×10^{-1} $(m = 8)$
proposed	$\mathbf{3.68 \times 10^{-3}}$ $(\mathbf{m = 15})$	$\mathbf{3.05 \times 10^{-1}}$ $(\mathbf{m = 6})$

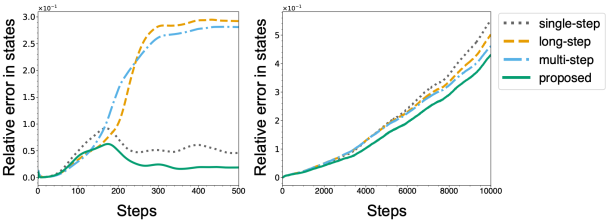

Fig. 2 Mean absolute errors of states for the CH equation (left) and KdV equation (right). The results of both the CH and KdV equation are shown for the best m values.

Assuming that $o(|e_{k,k+m-1}|)$ in Equation (4) is less significant and $e_{k+m-1,k+m} \simeq e_{k+m-2,k+m-1}$, the m-step error $e_{k,k+m}$ is rewritten as

$$\begin{aligned} e_{k,k+m} &\simeq e_{k+m-1,k+m} + J_m(e_{k,k+m-1}) \\ &= e_{k+m-1,k+m} + J_{m-1}e_{k+m-2,k+m-1} \\ &\quad + J_{m-1}J_{m-2}e_{k,k+m-2} \\ &\simeq (I + J_{m-1} + \cdots + J_{m-1} \cdots J_1)e_{k,k+1} \\ &= (\sum_{i=1}^{m-1} \prod_{j=1}^{i} J_{m-j} + I)e_{k,k+1} \quad (m > 1) \end{aligned} \tag{6}$$

Our proposed strategy uses the cost function $\mathcal{C}$ of the following form with the m-step error $e_{k,k+m}$:

$$\mathcal{C} = \frac{1}{KM}\sum_{k=0}^{K}\left((\sum_{i=1}^{m-1} \prod_{j=1}^{i} J_{m-j} + I)e_{k,k+1} \right). \tag{7}$$

With $m = 1$, the cost function has the same form as the single-step strategy.

4. Experiments and Results

We evaluated our strategy on the PDE systems, namely the Cahn-Hilliard (CH) and KdV equations. We used the HNN++ for the CH system and neural ODE for the KdV system. The neural ODE treats the output of a neural network as the time derivative of the input [1]. The HNN++ takes the gradient of a neural network for obtaining the time derivative, guaranteeing the underlying geometric properties. To emphasize the effectiveness of the error-analytic strategy, we employed a simpler integration method, namely the explicit midpoint method (RK2) for the CH and KdV equation. For comparison, we also trained these models using the single-step, long-step, and multi-step strategies.

After training, we solved the initial value problems and obtained the MSE between the ground truth and states predicted by trained models. Table 1 summarizes the results, providing a median of ten trials. For the PDE systems, the error-analytic strategy improved the prediction accuracy. For the CH equation, the prediction of the long-step and multi-step strategies exhibited large errors, indicating that learning

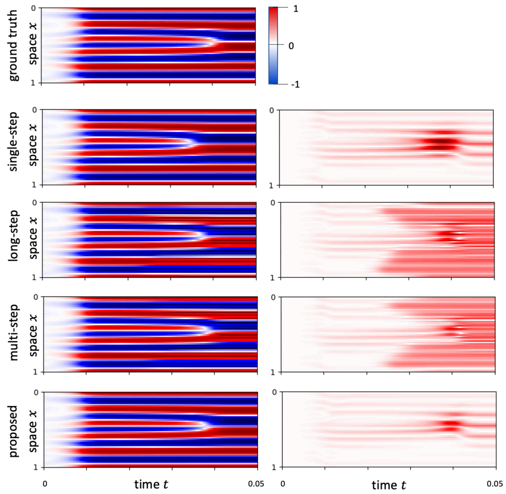

Fig. 3 CH equation predicted by the HNN++ with the RK2 method. (left) Ground truth $\boldsymbol{u}_k$ and predicted state $\tilde{\boldsymbol{u}}_k$. (right) The squared error between the ground truth and predicted state. The results of the long-step, multi-step, and proposed strategies are with $m = 13$, $m = 14$, and $m = 15$, respectively.

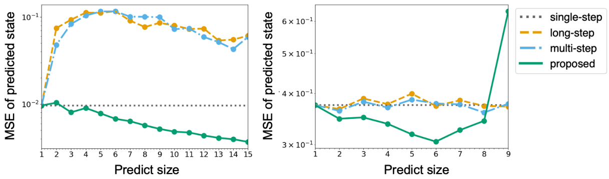

Fig. 4 MSE of (left) the CH equation predicted by the HNN++ with the RK2 and (right) the KdV equation predicted by the NODE with the RK2 method by the varying prediction step size m. The vertical axis is on a logarithmic scale.

by long-term prediction is unsuitable for the CH equation. However, for the KdV equation, the accuracy improved when the multi-step strategy was used.

Figure 2 shows the relative error along long test trajectories. Using the error-analytic strategy, we observed that the errors were relatively less extensive in the last steps than in the other strategies for both the CH and KdV equations. We also visualized the prediction results of the CH equation, which is depicted in Figure 3. Without the error-analytic strategy, the prediction exhibited large errors and awkward noises for the CH equation. Figure 4 shows the results when varying the prediction step size m. For the CH equation, the proposed method suggests that the prediction accuracy improved even when exceeding 15, by setting an upper limit of m. As for the long-step and multi-step methods, a significant decrease in prediction accuracy was observed for $m = 2$. Concerning the KdV equation, the proposed method showed the best results for $m = 6$, with a tendency for the prediction accuracy to deteriorate when m is increased beyond the value.

Furthermore, we evaluated our strategy on noisy data. Table 2 summarizes the results obtained for the CH and KdV equations with noise. Based on the obtained results, it is evident that when noise is introduced in both datasets, the prediction accuracy of our proposed method noticeably decreases. However, when applied to the CH equation, the long-step and multi-step methods exhibit improved performance when dealing with noisy data compared to noise-free scenarios. This indicates the usefulness of these methods for noisy data.

Table 2 Results on CH and KdV equations with noise.

Model	CH equation	KdV equation
single-step	4.08×10^{-2}	3.82×10^{0}
long-step	5.35×10^{-3} $(m=15)$	2.00×10^{0} $(m=8)$
multi-step	$\mathbf{1.77\times10^{-3}}$ $\mathbf{(m=15)}$	$\mathbf{1.92\times10^{0}}$ $\mathbf{(m=8)}$
proposed	4.56×10^{-2} $(m=15)$	7.88×10^{0} $(m=8)$

5. Conclusion

We introduced an error-analytic approach as a novel method to enhance the accuracy of long-term predictions when training time series datasets. We evaluated the error-analytic strategy on the PDEs and demonstrated that a model trained using the proposed strategy outperformed the state-of-the-art baselines. The theoretical analysis of the improved performance of our proposed strategy remains a topic for future research. Since it handles sampled data (i.e., time series that is discretized in time), methods designed for continuous-time dynamics, such as the Hamilton-Jacobi-Bellman equation, cannot be directly applied. In addition, it may be possible to integrate the error analysis in previous studies [2]–[4] with our proposed strategy, achieving a refined performance.

Acknowledgment

This study was partially supported by JST PRESTO (JPMJPR21C7), JST CREST (JPMJCR1914), JST ASPIRE (JPMJAP2329),and JSPS KAKENHI (19H04172, 19K20344, 24K15105), Japan.

References

[1] T.Q. Chen et al., “Neural Ordinary Differential Equations,” NeurIPS, pp.1–19, 2018.

[2] H. Nam et al., “Error estimation using neural network technique for solving ordinary differential equations,” Advances in Continuous and Discrete Models, vol.2022, 2022.

[3] C. Filici, “Error estimation in the neural network solution of ordinary differential equations,” Neural networks : the official journal of the International Neural Network Society, pp.614–7, 2010.

[4] A.S. Dogra, “Error estimation and correction from within neural network differential equation solvers,” 2020.

[5] S. Greydanus et al., “Hamiltonian Neural Networks,” NeurIPS, pp.1–16, 2019.

[6] T. Matsubara et al., “Deep Energy-Based Modeling of Discrete-Time Physics,” NeurIPS, 2020.

[7] S. Liang et al., “Stiffness-aware neural network for learning Hamiltonian systems,” ICLR, pp.1–16, 2022.

[8] Z. Chen et al., “Symplectic Recurrent Neural Networks,” ICLR, pp.1–23, 2020.

[9] E. Hairer et al., Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, Springer Series in Computational Mathematics, vol.31, Springer-Verlag, Berlin/Heidelberg, 2006.

[10] O. Nelles, Nonlinear System Identification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[11] A. Paszke et al., “Automatic differentiation in PyTorch,” Autodiff Workshop on Advances in NIPS, pp.1–4, 2017.