AN ERROR ANALYSIS FRAMEWORK FOR NEURAL NETWORK MODELING OF DYNAMICAL SYSTEMS

A PREPRINT

Shunpei Terakawa
Graduate School of System Informatics
Kobe University
Kobe, Hyogo, Japan
s-terakawa@stu.kobe-u.ac.jp

Takashi Matsubara
Graduate School of Engineering Science
Osaka University
Toyonaka, Osaka, Japan
matsubara@sys.es.osaka-u.ac.jp

Takaharu Yaguchi
Graduate School of System Informatics
Kobe University
Kobe, Hyogo, Japan
yaguchi@pearl.kobe-u.ac.jp

December 30, 2021

ABSTRACT

We propose a theoretical framework for investigating a modeling error caused by numerical integration in the learning process of dynamics. Recently, learning equations of motion to describe dynamics from data using neural networks has been attracting attention. During such training, numerical integration is used to compare the data with the solution of the neural network model; however, discretization errors due to numerical integration prevent the model from being trained correctly. In this study, we formulate the modeling error using the Dahlquist test equation that is commonly used in the analysis of numerical methods and apply it to some of the Runge–Kutta methods.

Keywords learning dynamics · modeling error · neural networks · Runge-Kutta methods

1 Introduction

Data-driven approximation of differential equations by neural networks has a long history [Anderson et al. 1996], [Wang and Lin 1998], [Oliveira 2004], [Raissi et al. 2019]. An important application is learning the governing equations of physical phenomena [Greydanus et al. 2019]; for example, in [Greydanus et al. 2019], instead of time-derivatives of the state variables, the energy function is modeled by neural networks, thereby discovering the equation of motion. The objective of most of these studies is modeling continuous-time differential equations \( \dot{x}(t) = f(x) \) that describe the target dynamics by using the neural ODE model \( \dot{x}(t) = f_{NN}(x) \), or its extensions.

Due to the difficulty of the observation of the values of \( \dot{x}(t) \) in some practical situations, it would be expected that \( x(t) \) at enough numbers of \( t \)'s are observed and hence given as the data. In this case, numerical integrators (typically, an explicit Runge–Kutta method) are required to integrate the neural network models for learning and also for predicting the dynamics. However, the employment of the numerical integrators necessarily induces numerical errors, which results in producing non-negligible modeling errors in the learned continuous-time model. In other words, even if the difference between the data and the numerical solutions of the learned model could be reduced to zero, it does not necessarily mean that the model \( f_{NN} \) matches the target dynamics \( f \).

These modeling errors are not a problem if the discrete models can be used as they are; for example, the learned model may be used to predict the dynamics by integrating the model with the same time step as the training data. In such cases, the modeling error due to the discretization has no harmful effect because the model is trained so that it can
reproduce the data as the numerical solutions completely. However, the errors can be serious when the models need to be identified as continuous differential equations rather than discrete models. This is a common situation, for example, where the target system is a subsystem of a large-scale system. In such a case, the subsystems should be identified as continuous ones because each subsystem may have different timescales determined by the data sampling settings and it may not be possible to define a unique time step for the entire system. Even if in simpler cases, many commonly used ODE solvers employ the adaptive step size control, so the same kind of problem could happen.

To reduce the modeling errors, the integrators must be replaced or redesigned according to certain criteria. This paper aims to provide a framework for such an analysis (Figure 1). In summary, the main contributions of this paper are:

1. introduction of a framework of theoretical analysis of the modeling errors caused by the numerical integrators, thereby providing the theoretical background for newly developing integrators that are suitable not only for computation but also for modeling.

Figure 1: An overview of the error analysis of the neural network modeling of ODE systems. We theoretically estimate the modeling error and propose a framework that enables us to select or design a suitable numerical integrator for learning dynamics according to the characteristics of the target dynamics.

2 Proposed framework of learnability analysis

The proposed framework parallels the classical analysis of the stability region. In fact, the problem to be addressed here is to a certain extent similar to the stability analysis of numerical integrators. As is well-known, the stability of each numerical integrator depends on the characteristics of the target differential equations, that is, how rapidly the solution decays and/or how rapidly the solution oscillates. In the stability analysis, the stability region is defined by specifying \( \lambda \in \mathbb{C} \) for which the numerical solution to the Dahlquist test equation

\[
\dot{x}(t) = \lambda x, \quad x : t \in \mathbb{R} \mapsto x(t) \in \mathbb{C}
\]

(1)

by the integrator remains bounded. The test equation is a representative equation for dynamics in the sense that most nonlinear differential equations describing physical phenomena are reduced to this equation by linearization and diagonalization. The analysis of the stability region helps users to narrow down their candidates of the integrators for the target differential equations according to the characteristics of the equations. See, e.g., [Hairer and Wanner, 2013], [Butcher, 2016] for details.

Following this approach, we propose a framework for analyzing the learnability of numerical integrators. More precisely, we will introduce the learnability coefficient, which characterizes the dynamics of which the given numerical integrator is suitable for modeling.

First of all, we confirm the learning process assumed in this paper. We suppose that the target differential equation is learned by the model \( \hat{x}(t) = \hat{f}(\hat{x}; \theta) \), where \( \hat{f} \) is a function that is represented by, e.g., a multilayer perceptron and \( \theta \) denotes the model parameters. As a model, we mainly consider neural network models, but we only assume the universal approximation property for the model (e.g., [Hornik et al., 1989]).

For the data, we suppose that only the states \( x \) are observable, and therefore the derivatives \( \dot{x} \) are not available. To focus on the modeling errors caused by numerical integrators, we consider an ideal situation, where a sufficient amount of the noise-free data are given and they are sampled at a fixed sampling rate \( 1/h \), thereby supposing that the data are given as a set of pairs \( D := \{(x_d^{(n)}, x_d^{(n+1)})\} \), where \( x_d^{(n)} \) denotes the data sampled at \( t = nh \).

For training, \( \hat{f}(\hat{x}; \theta) \) is assumed to be learned by minimizing \( \sum_{(x_d^{(n)}, x_d^{(n+1)}) \in D} \| x_d^{(n+1)} - \hat{x}^{(n+1)} \| \) for a specified norm \( \| \cdot \| \), where \( \hat{x}^{(n+1)} \) is given as the numerical solution by the concerned integrator: \( \hat{x}^{(n+1)} = x_d^{(n)} + h \hat{f}_{\text{numer}}(x_d^{(n)}, \hat{x}^{(n+1)}; \theta) \), where \( h \hat{f}_{\text{numer}}(x_d^{(n)}, \hat{x}^{(n+1)}) \) is the increment numerically computed by the integrator.
Following the stability analysis, we focus on the case where the target equation is the Dahlquist test equation \( \text{(1)} \). In this case, the data set becomes \( \mathcal{D} = \{ x^{(n)}, x^{(n+1)} = e^{h\lambda} x^{(n)} \} \) and the loss function is
\[
 l(\theta; \mathcal{D}) := \sum_{x} ||e^{h\lambda} x^{(n)} - \hat{x}^{(n+1)}||. \tag{2}
\]

As we assumed the universal approximation property of the model, \( \hat{f} \) can represent arbitrary functions by appropriately choosing the parameters \( \theta \). Hence in particular \( \hat{f} \) can be a linear function \( f(x; \theta) = \alpha x \) with \( \alpha \in \mathbb{C} \), which is in the same class of functions as the target equation \( \hat{x} = \lambda x \). In fact, there exists an optimal linear function that reduces the loss function to zero (see Theorem 3.1). By using such \( \alpha \), we define the learnability coefficient in the following way.

**Definition 2.1.** For each \( \alpha \) that eliminates \( \text{(1)} \), we define the learnability coefficient \( \ell_\alpha \) by
\[
 \ell_\alpha := \frac{\alpha - \lambda}{\lambda},
\]

We also define the relative error of the real part and the imaginary part of the \( \alpha \) independently as the learnability coefficients for decaying and oscillating part.

**Definition 2.2.** For each \( \alpha \) that eliminates \( \text{(1)} \), we define the componentwise learnability coefficients \( \ell_{R\alpha} \) and \( \ell_{I\alpha} \) for the real part and the imaginary part respectively by
\[
 \ell_{R\alpha} := \left| \frac{\text{Re} \alpha - \text{Re} \lambda}{\text{Re} \lambda} \right|, \quad \ell_{I\alpha} := \left| \frac{\text{Im} \alpha - \text{Im} \lambda}{\text{Im} \lambda} \right|.
\]

### 3 The learnability analysis of the Runge–Kutta methods

In this section, we show the learnability coefficient for the general Runge–Kutta methods:
\[
 \hat{x}^{(n+1)} = \hat{x}^{(n)} + h \sum_{i=1}^{p} b_i k_i, \quad k_i = f(\hat{x}^{(n)} + h \sum_{j=1}^{p} a_{ij} k_j), \tag{3}
\]

where \( p \), \( a_{ij} \)'s, and \( b_j \)'s are the constants that define the method (see, e.g., Butcher [2016]). The matrix and the vector defined by \( a_{ij} \) and \( b_j \) are respectively denoted by \( A \) and \( b \).

**Theorem 3.1 (main result).** If the equation \( \text{(1)} \) is discretized by a Runge–Kutta method \( \text{(3)} \), there exists an \( \alpha \) such that the model with \( f(\hat{x}) = \alpha \hat{x} \) reduces the loss function \( \text{(2)} \) to zero. Moreover, \( \alpha \) is a solution to
\[
 1 + h\alpha b^\top(I - h\alpha A)^{-1}1 - e^{h\lambda} = 0, \quad \det(I - h\alpha A) \neq 0, \quad 1 = (11 \cdots 1)^\top. \tag{4}
\]

**Proof of Theorem 3.1** Suppose that the loss function \( \text{(2)} \) vanishes for the model with \( f(\hat{x}) = \alpha \hat{x} \). If this equation is discretized by the Runge–Kutta method with the initial condition \( \hat{x}(nh) = x^{(n)}_d \), the following equation holds
\[
 \hat{x}^{(n+1)} = x^{(n)}_d + h b k, \quad k = \alpha(x^{(n)}_d 1 + hA k).
\]

If \( \det(I - h\alpha A) \neq 0 \) and \( ||\hat{x}^{(n+1)} - x^{(n+1)}_d|| \) is zero for all \( x^{(n)}_d \), we get \( \hat{x}^{(n+1)} = e^{h\lambda} x^{(n)}_d = x^{(n)}_d + x^{(n)}_d h\alpha b^\top(I - h\alpha A)^{-1}1. \) Hence, \( \alpha \) should satisfy \( e^{h\lambda} = 1 + h\alpha b^\top(I - h\alpha A)^{-1}1. \) \( \square \)

**Definition 3.1.** We call equation \( \text{(4)} \) the learnability equation for the Runge–Kutta method.

**Remark 3.1.** In general, equation \( \text{(4)} \) admits \( p \) solutions and hence \( p \) learnability coefficients exist for a Runge–Kutta method with \( p \) stages. In particular, the model is not uniquely determined when trained as assumed in this paper; see the examples below.

**Theorem 3.2.** For the Runge–Kutta methods, the learnability coefficient is a function of \( z := h\lambda \).

**Proof of Theorem 3.2** From the learnability equation, we have \( 1 + (\alpha/\lambda)h\lambda b^\top(I - (\alpha/\lambda)h\lambda A)^{-1}1 - e^{h\lambda} = 0, \det(I - (\alpha/\lambda)h\lambda A) \neq 0 \). Therefore, we get \( 1 + (\alpha/\lambda) z b^\top(I - (\alpha/\lambda) z A)^{-1}1 - e^z = 0, \det(I - (\alpha/\lambda) z A) \neq 0 \), which shows that \( \alpha/\lambda \) and hence \( \alpha/\lambda = 1 \) are functions of \( z \). \( \square \)

**Theorem 3.3.** For the Runge–Kutta methods, the componentwise learnability coefficients are functions of \( z := h\lambda \) and \( z^2 := h\lambda^2 \).
Proof of Theorem 3.3. We show the proof for $\ell_R^\alpha$. If $\alpha$ is the solution of the learnability equation for $\lambda$, then $\pi$ is also the solution of the equation for $\lambda$. Thus, in the same way as the proof of Theorem 3.2, $\pi/\lambda$ is shown to be a function of $\pi := h\lambda$. On the other hand, a part of the definition of $\ell_R^\alpha$ is rearranged in $\text{Re } \alpha/\text{Re } \lambda = \alpha + \pi \pi/\lambda + \lambda = (\alpha + \pi/\lambda) - 1 + \pi/\lambda$, which shows that $\text{Re } \alpha/\text{Re } \lambda$ and hence $|\text{Re } \alpha/\text{Re } \lambda - 1|$ is a function of $\pi$ and $\alpha$.

Remark 3.2. This scaling property of the learnability coefficients is important as evaluation criteria for designing numerical integrators for learning dynamics. Evaluation criteria should be determined by the modeling errors, while the modeling errors depend on $h$. However, because changing $h$ will change the measured performance of the integrators, naive evaluation criteria that depend on $h$ are not appropriate for designing numerical integrators. Thus evaluation criteria with certain invariance with respect to $h$ are indispensable.

4 Examples

Example 4.1. The learnability equation of the explicit Euler method is $1 + h\alpha - e^{h\lambda} = 0$, which gives a unique $\alpha$: $\alpha = e^{h\lambda} - 1/h$. In addition, the learnability coefficient, which is a relative modeling error, is $\ell_\alpha = |(e^z - 1)/z - 1|$ with $z = h\lambda$.

Example 4.2. For the explicit midpoint method, the learnability equation becomes $1 + h\alpha + h^2\alpha^2 - e^{h\lambda} = 0$ and admits the two solutions: $\alpha = (-1 \pm \sqrt{2e^{h\lambda} - 1})/h$, where the square root represents the principal value. Among these two solutions, calculating the Taylor series expansion around $\lambda = 0$, $\alpha_+ := (-1 + \sqrt{2e^{h\lambda} - 1})/h$ is a 2nd-order approximation to $\lambda$. Meanwhile, $\alpha_- := (-1 - \sqrt{2e^{h\lambda} - 1})/h$ is not an approximation. This means that the model with the explicit midpoint method is not uniquely identifiable and, moreover, the learned model may be completely different from the true dynamics.

The contour lines of the learnability coefficients for the above methods are shown in Figure 2. As is expected, the errors are smaller for the explicit midpoint method than for the Euler method when the model corresponding to $\alpha_+$ is learned. Meanwhile, it can be seen from the figure that the midpoint method is not effective for dynamics with strong damping since the error increases as $\lambda$ goes in the negative direction on the real axis.

Example 4.3. The learnability equation of the classical 4th order Runge–Kutta method is $1 + h\alpha + h^2\alpha^2 + h^3\alpha^3 + h^4\alpha^4 - e^{h\lambda} = 0$. We computed all of the solutions of the equation numerically and extracted the solution closest to $\lambda$ as $\alpha$. The contour lines of the learnability coefficient computed in the above way are shown in Figure 3.

Example 4.3 well explains the result of the learning test of the equation with $\lambda = 1.5i$, $h = 1$ shown in Figure 4. The experiment was conducted under the following settings.

The neural network model $f_{NN}$ is a multilayer perceptron, consisting of a fully-connection hidden layer. The input and output layers have 2 units that correspond to the real part and the imaginary part of the input and the output. The number of hidden units was 200. We used tanh as the activation function. We used the Adam method for training and specified the learning rate as 0.001. Since we used the neural network models, the model function $f$’s were not
linear, we regarded the average value of $\hat{x}_{d}^{(n+1)}/x_{d}^{(n)}$ over all $x_{d}^{(n)}$’s used in the training process as an estimation of $\alpha$. Actually, for each model, these values were almost constant.

For the $\lambda$, we prepared the training data as tuples $\{(x_{1}, x_{0}) \mid x_{1} = e^{h\lambda x_{0}}\}$ where we uniformly randomly sampled 10000 points for $x_{0}$’s from $-10 \leq \text{Re} x_{0} \leq 10, -10 \leq \text{Im} x_{0} \leq 10$. After that, the neural network was trained by minimizing the mean squared error $M^{-1} \sum x_{0} \|x_{1} - \hat{x}_{1}\|^2$, where $M$ is the number of the data and $\hat{x}_{1}$ is the numerical one-step solution of the model $f_{NN}$ using the classical 4th order Runge–Kutta method.

The black line shows the real part of the exact solution of the target dynamics. The blue line is the predicted dynamics using $\alpha$ above, and the red one is the actually learned dynamics $f_{NN}$ in the experiment calculated using scipy.integrate.solve_ivp with RK45 option. These all oscillate at similar frequencies, but the amplitude of the blue and red ones increases unlike the black one. The behavior is consistent with the learnability coefficients for the real and imaginary parts. In fact, $\ell_{\lambda}^{R}$ is relatively small at $\lambda = 1.5$, explaining the small errors in the frequencies; on the other hand, $\ell_{\lambda}^{R}$ takes large values around the imaginary axis a little away from the origin, which results in the errors in the amplitudes.

Figure 3: The contour lines of the learnability coefficients of the classical 4th order Runge–Kutta method. The $\alpha$ used to calculate the coefficients is found numerically, unlike the case of Euler method and the explicit midpoint method.

Figure 4: A comparison between the dynamics actually learned and the dynamics predicted by the theoretical results.

Figure 5: The learnability coefficient of the method determined by (5).

5 Application for designing numerical schemes

A motivation for the introduction of the learnability coefficient is designing numerical integrators for learning differential equations. To this end, we propose a procedure for designing integrators.

First, a region $\Omega$ on which the learning coefficient $l_{\alpha}(\lambda)$ should be small must be specified according to the characteristics of the target phenomena; for example, if we want to model dissipative phenomena, $\Omega$ should contain the negative real axis.

Second, because the learning coefficient $l_{\alpha}(\lambda)$ must satisfy (4), instead of specifying the Runge–Kutta method by the Butcher tableau, we design the corresponding equation (4). In fact, the term $1 + h\alpha b^{\top}(I - h\alpha A)^{-1}I$ in this equation is essentially the stability polynomial of the Runge–Kutta method [Hairer and Wanner 2013]. Realization methods of
Runge–Kutta methods from the given stability polynomial have been developed. For example, in the Lebedev method [Lebedev 1989, 1994] the desired numerical integrator is implemented as a composition of a series of simple numerical schemes like the explicit Euler method.

As an illustration, we consider numerical integrators for learning dissipative differential equations; we want to design integrators of which the learnability coefficient $l_{x}(\lambda)$ is small on an interval $[-r_0, 0)$ with $r_0 > 0$ as large as possible.

First, we rewrite (4) to
$$1 + \mu z b^\top (I - zA)^{-1} z - e^{\lambda} = 0,$$

where $\mu = \alpha / \lambda$ and $z = h \lambda$. Because $\mu = 1$ is preferable, we want the absolute value of the left-hand side of $1 + z b^\top (I - zA)^{-1} z - e^{\lambda} = 0$ to be small on $[-r_0, 0)$. If $r_0$ is sufficiently large and hence $e^{\lambda}$ is small, we can consider $1 + z b^\top (I - zA)^{-1} z$ for simplicity. For this function to be small, we set this function to oscillate around zero while satisfying the conditions for ensuring that the associated Butcher tableau certainly defines a numerical integrator. This approach is employed to design a class of highly stable explicit Runge–Kutta methods for dissipative differential equations; in those researches, it is known that the Chebyshev polynomials are optimal solutions in the sense that large $r_0$ can be used.

For example, the 2-stage method is given by
$$1 + b^\top (I - zA)^{-1} z = 1 + z + \frac{1}{8} z^2.$$ (5)

The learnability coefficient of this method is shown in Figure 5, in which we can confirm the quite better performance on the negative real axis than the explicit midpoint method, which is also a 2-stage method. For higher stage methods defined by the Chebyshev polynomials, see Hairer and Wanner [2013].

### 6 Conclusions

In recent years, methods for constructing differential equation models from data by using deep neural networks have been widely studied. In such methods, the models are often discretized by numerical integrators when learning, but the effects of the discretization have not been well studied theoretically. To appropriately select and/or design numerical integrators, evaluation criteria for the errors are required. In this paper, we have introduced the learnability coefficient as such a criterion along with the detailed analysis of Runge–Kutta methods and designed a method for learning dissipative systems. As future work, further investigation on the uniqueness of the learned model is needed.

### Acknowledgements

This work was supported by the JST CREST [Grant Number JPMJCR1914], JST PRESTO [Grant Number JP-MJPR21C7] and JSPS KAKENHI [Grant Number 20K11693].

### References

J. S. Anderson, I. G. Kevrekidis, and R. Rico-Martinez. A comparison of recurrent training algorithms for time series analysis and system identification. *Computers & Chemical Engineering*, 20:S751–S756, Jan 1996. ISSN 0098-1354. doi:10.1016/0098-1354(96)00133-0 [URL](https://www.sciencedirect.com/science/article/pii/0098135496001330).

Yi-Jen Wang and Chin-Teng Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2):294–307, 1998. doi:10.1109/72.661124.

R. Oliveira. Combining first principles modelling and artificial neural networks: a general framework. *Computers & Chemical Engineering*, 28(5):755–766, 2004. ISSN 0098-1354. doi:https://doi.org/10.1016/j.compchemeng.2004.02.014 [ESCAPE 13].

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 15379–15389, 2019.

Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential - Algebraic Problems*. Springer Science & Business Media, 2013.

J. C. Butcher. *Numerical Methods for Ordinary Differential Equations, Third Edition*. John Wiley & Sons, Ltd., Chichester, 2016.
Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080.

V I Lebedev. Explicit difference schemes with time-variable steps for solving stiff systems of equations. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 4(2), 1989.

V I Lebedev. How to solve stiff systems of differential equations by explicit methods. *Numerical Methods and Applications*, pages 45–80, 1994.