The Error Analysis of Numerical Integrators for Deep Neural Network Modeling of Differential Equations

Shunpei Terakawa
Kobe University
Kobe, Japan 657-8501
s-terakawa@stu.kobe-u.ac.jp

Takashi Matsubara
Osaka University
Osaka, Japan 560–8531
matsubara@sys.es.osaka-u.ac.jp

Takaharu Yaguchi
Kobe University
Kobe, Japan 657-8501
yaguchi@pearl.kobe-u.ac.jp

Abstract

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 propose a theoretical framework for investigating the effect of numerical integration on modeling errors and perform the error analysis of the Runge-Kutta methods.

1 Introduction

Data-driven approximation of differential equation by neural networks has a long history [Anastassi, 2014, Cichock and Rolf Unbehauen, 1992, Lagaris et al., 1998, Raissi, 2018, Ramuhalli et al., 2005, Rudd et al., 2014], but it has been attracting limited attention until recently. Deep neural networks have made significant achievements mostly in real-world tasks such as image processing [He et al., 2016] and natural language processing [Devlin et al., 2018]. Recent investigation has revealed that many cutting-edge architectures can be regarded as numerical discretization of differential equations [Lu et al., 2017], leading to a bunch of studies on neural networks for differential equations. [Chen et al., 2018] proposed an automatic differentiation algorithm for a neural network approximating an ordinary differential equation (ODE) and surpassed the memory usage greatly. [Greydanus et al., 2019] approximated an energy function H instead of a time-derivative of a state, and thereby, built a Hamiltonian system, which admits the energy conservation law. Researches on the application of such modeling capability of deep learning to the estimation of the governing equations of physical phenomena have been intensively conducted.

Modeling error in neural ODE models The objective of most of these studies is modeling continuous-time differential equations $\dot{x} = f(x)$ that describes the target physical phenomena by using the neural ODE model (NODE)

$$\dot{x} = f_{\mathrm{NN}}(x), \tag{1}$$

or its extensions [Chen et al., 2018, Chen et al., 2020, Greydanus et al., 2019, Zhong et al., 2020b]. Due to the difficulty of the observation of the values of $\dot{x}$, it would be expected that $x(t)$ at an enough number of t's are observed and hence given as the data. In such a case, numerical integrators (typically, an explicit Runge–Kutta method) are required to integrate the neural network models for

Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020), Vancouver, Canada.

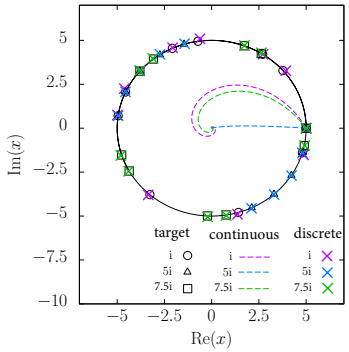
learning and also for predicting the dynamics. However, the employment of the numerical integrators necessarily induce numerical errors, which results in producing non-negligible *modeling errors*. These modeling errors are not a problem if the discrete models can be used as they are; however, this can be serious when the models must be identified as continuous differential equations rather than discrete models, which is a common situation, for example, where the target system is actually an subsystem of a large scale coupled systems. In such a case, the subsystems should be identified as continuous ones because each subsystem may have different timescales and it may not be possible to define a unique time step for the entire system.

New subject of numerical analysis that emerges from deep neural network modeling of ODEs Fig. 1 shows motivating examples, in which we learned the test equation

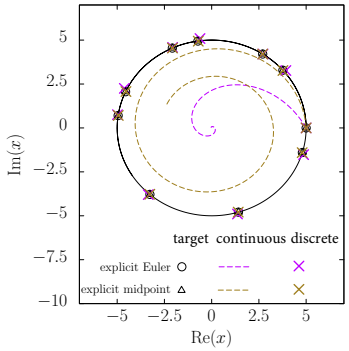
$$\dot{x} = \lambda x, \quad x : t \in \mathbb{R} \mapsto x(t) \in \mathbb{C} \tag{2}$$

with several values of λ's. This is, in fact, 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. $\lambda \in \mathbb{C}$ characterises the dynamics in the sense that $\mathrm{Re}\ \lambda$ determines the speed of decay (or blowup) and $\mathrm{Im}\ \lambda$ does the speed of oscillation. $f_{\mathrm{NN}} : \mathbb{C} \to \mathbb{C}$ in the model (1) was given by a standard multilayer perceptron. See Appendix B for details. The important point here is that the modeling errors caused by the numerical integrators depend on the characteristics (the value of λ) of the dynamics *in a non-trivial way*. In fact the dynamics with the slowest speed ($\lambda = \mathrm{i}$) is easier to model than that with a moderate speed($\lambda = 5\mathrm{i}$); however, the dynamics with the hishest speed ($\lambda = 7.5\mathrm{i}$) is again easier than the moderate one. Thus there arises a new challenge of numerical analysis: *the learnability analysis*, which will be the 4th subject of numerical analysis after the compatibility, the stability and the convergence analysis. The aim of this paper is providing a framework of such an analysis.

The main contributions of this paper would be: (1) introduction of a framework of theoretical analysis of the modeling errors caused by the numerical integrators, (2) thereby providing the theoretical background for newly developing integrators that are suitable not only for computation but also for modeling.



(a) The results for the various vibration modes at different speeds ($\dot{x} = \lambda x$ with $\lambda = \mathrm{i}, 5\mathrm{i}$ and $7.5\mathrm{i}$).



(b) The results of NODE discretized by the explicit Euler method and the explicit midpoint rule for $\dot{x} = \mathrm{i}x$.

Figure 1: The orbits of the true dynamics, the discrete NODE and the NODE used as continuous models.

The related work includes: as a bridge between deep learning and numerical analysis, Celledoni et al.[Celledoni et al., 2020] investigated the relation between the structure-preserving numerical integrators and deep neural network models. Modeling error analysis of dynamic neural network models as a nonlinear identification problem is performed by Poznyak et al.[Poznyak et al., 1996].

2 Proposed framework of learnability analysis

The proposed framework parallels to 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, 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 (2) by the integrator remains bounded. The analysis of this region helps users to narrow down their candidates of the integrators for the target differential equations according to the characteristics of the equations.

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, thereby playing a similar role to the stability region of the standard stability analysis.

First of all, we confirm the learning method assumed in this paper. We suppose that the target differential equation is learned by the model $\dot{\hat{x}} = \hat{f}(\hat{x};\theta)$, where $\hat{f}$ is a function that is represented by, e.g., a multilayer perceptron and θ denotes the model parameters. As a model, we mainly consider neural network models, but we only assume the universal approximation property for models. For the data, we suppose that only the states x are observable, and therefore the derivatives $\dot{x}$ are not available. In order to focus on the modeling errors caused by numerical integrators, we consider the 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 $\mathcal{D} := \{(x_{\mathrm{d}}^{(n)}, x_{\mathrm{d}}^{(n+1)})\}$, where $x_{\mathrm{d}}^{(n)}$ denotes the data sampled at $t = nh$. For training, $\hat{f}(\hat{x};\theta)$ is assumed to be learned by minimizing $\sum_{(x_{\mathrm{d}}^{(n)}, x_{\mathrm{d}}^{(n+1)})\in\mathcal{D}} \|x_{\mathrm{d}}^{(n+1)} - \hat{x}^{(n+1)}\|$ for a specified norm $\|\cdot\|$, where $\hat{x}^{(n+1)}$ is given as the numerical solution by the concerned integrator: $\hat{x}^{(n+1)} = x_{\mathrm{d}}^{(n)} + h\hat{f}_{\mathrm{numer}}(x_{\mathrm{d}}^{(n)}, \hat{x}^{(n+1)};\theta)$, where $h\hat{f}_{\mathrm{numer}}(x_{\mathrm{d}}^{(n)}, \hat{x}^{(n+1)})$ is the increment numerically computed by the integrator. Following the stability analysis, we focus on the case where the target equation is the test equation (2). In this case, the data set becomes $\mathcal{D} = \{(x_{\mathrm{d}}^{(n)}, x_{\mathrm{d}}^{(n+1)} = \mathrm{e}^{\lambda h} x_{\mathrm{d}}^{(n)})\}$ and the loss function is

$$l(\theta;\mathcal{D}) := \sum_{x_{\mathrm{d}}^{(n)}} \|\mathrm{e}^{\lambda h} x_{\mathrm{d}}^{(n)} - \hat{x}^{(n+1)}\|. \tag{3}$$

As we assumed the universal approximation property of the model, by appropriately choosing the parameters θ, $\hat{f}$ can represent arbitrary functions. Hence in particular $\hat{f}$ can be a linear function $\hat{f}(\hat{x};\theta) = \alpha\hat{x}$ with $\alpha \in \mathbb{C}$, which is in the same class of functions as the target equation $\dot{x} = \lambda x$, that makes the loss function vanished if such an optimal linear function exists. By using such α, we define the learnability coefficient in the following way.

Definition 2.1. *For each α that eliminates* (3)*, we define the learnability coefficient ℓ_α by $\ell_\alpha := |(\alpha - \lambda)/\lambda|$.*

3 The learnability analysis of the Runge–Kutta methods

In this section, we show the learnability coefficient for the general Runge–Kutta methods:

$$\hat{x}^{(n+1)} = \hat{x}^{(n)} + h\sum_{i=1}^{p} b_i k_i, \quad k_i = f(\hat{x}^{(n)} + h\sum_{j=1}^{p} a_{ij} k_j), \tag{4}$$

where p, a_{ij}'s and b_j's are the constants that defines the method (see, e.g., [Butcher, 2016]).

Theorem 3.1. *If the equation* (2) *is discretized by a Runge–Kutta method* (4)*, there exists an α such that the model with $\hat{f}(\hat{x}) = \alpha\hat{x}$ makes the loss function* (3) *vanished. Moreover, α is a solution to*

$$b^\top (I - \alpha h A)^{-1} \mathbb{1}\alpha h - e^{\lambda h} + 1 = 0, \quad \det(I - \alpha h A) \neq 0, \tag{5}$$

where $\mathbb{1} = (1\ 1\ \cdots\ 1)^\top$.

Definition 3.1. *We call the equation* (5) *the learnability equation for the Runge–Kutta method.*

Remark 3.1. *In general, the equation* (5) *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 $z := h\lambda$.*

Remark 3.2. *By expressing the coefficients as a function of $h\lambda$, it is possible to predict the modeling errors when h becomes smaller.*

Example 3.1. *The learnability equation of the explicit Euler method is*

$$h\alpha - \mathrm{e}^{\lambda h} + 1 = 0, \tag{6}$$

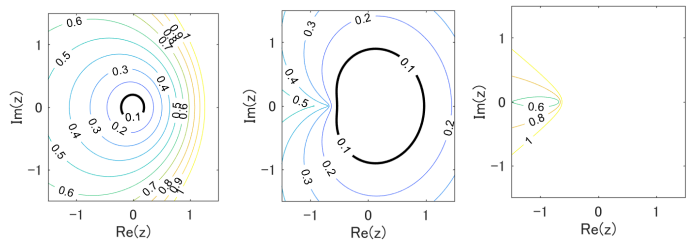
which gives a unique α: $\alpha = \mathrm{e}^{\lambda h} - 1/h$. In addition, the learnability coefficient, which is a relative modeling error, is $\ell_\alpha := \left|\frac{\alpha-\lambda}{\lambda}\right| = \left|\frac{\mathrm{e}^z - 1}{z} - 1\right|$ with $z = h\lambda$.

Example 3.2. *For the learnability equation of the explicit midpoint method*

$$\hat{x}^{(n+1)} = \hat{x}^{(n)} + hk, \quad k = \alpha(\hat{x}^{(n)} + \frac{h}{2}\alpha\hat{x}^{(n)})$$

admits the two solutions $\alpha = (-1 \pm \sqrt{2\mathrm{e}^{\lambda h} - 1})/h$. *Among these two solutions,* $\alpha_+ = (-1 + \sqrt{2\mathrm{e}^{\lambda h} - 1})/h$ *is a 2nd-order approximation to* λ. *In fact, the Taylor series expansion yields* $\alpha_+ = (-1 + \sqrt{2\mathrm{e}^{\lambda h} - 1})/h = \lambda + \lambda^3 h^2/6 + O(h^3)$. *Meanwhile,* $\alpha_- = (-1 - \sqrt{2\mathrm{e}^{\lambda h} - 1})/h$ *is not an approximation since* $\alpha_- = (-1 - \sqrt{(\lambda h + 1)^2})/h = -2/h - \lambda - \lambda^3 h^2/6 + O(h^3)$. *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 Fig. 2. As is expected, the errors are smaller for the explicit midpoint method than for the Euler method when the model corresponding to α_+ 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 λ goes in the negative direction on the real axis. Next, we compare the modeling errors of the models actually trained using the multilayer perceptrons with the theoretical values. Because for the neural network models, the model function $\hat{f}$ may not be linear, we regard the average value of $\hat{x}^{(n+1)}/x_{\mathrm{d}}^{(n)}$ as an estimation of α. Actually, for each model these values were almost constant. Figs. 3 and 4 show the results for the Euler method and the explicit midpoint method, respectively. The actual measured modeling errors are very close to theoretical values, in particular, to the desirable α_+ for the midpoint method. Although it is theoretically unclear why α_+ was learned, this implies the reliability of our theory.

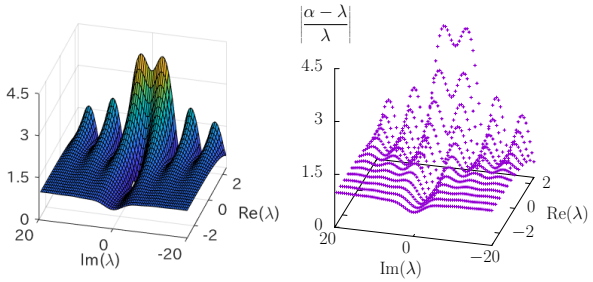


(a) Explicit Euler method. (b) Explicit midpoint method (α_+). (c) Explicit midpoint method (α_-).

Figure 2: The contour lines of the learnability coefficients.

4 Concluding remarks

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. Because reducing the step size that is determined by the sample rate of the data is not easy, to reduce the modeling errors, the integrators must be replaced or redesigned. To appropriately select and/or design numerical methods, an evaluation criteria for the errors is required. In this paper, we have introduced the learnability coefficient as such a criterion along with the detailed analysis of Runge–Kutta methods.

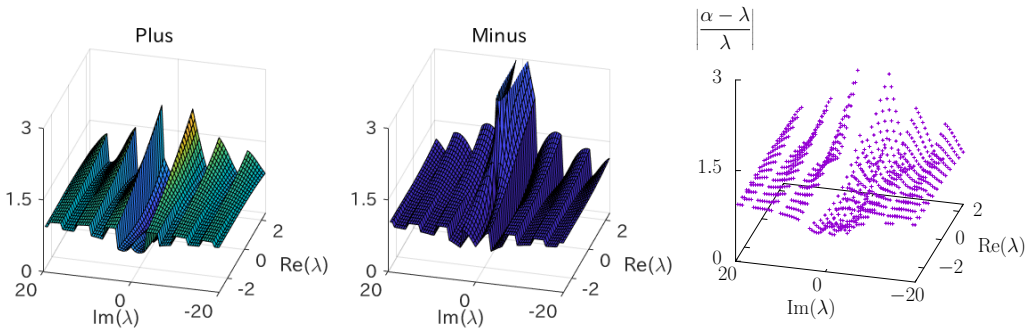


(a) The learnability coefficients. (b) Relative modeling errors of actually trained NODE models.

Figure 3: Comparison of the theoretical results for the Euler method with the relative errors of actually trained models.

Broader Impact

The most significant contribution of this paper would be to demonstrate the need for a new theory of numerical analysis, that is, *the learnability*, which is the fourth property after the three main topics of mathematical research on numerical analysis: the consistency, the stability and the convergence. In this sense, this paper bridges the gap between deep learning and numerical analysis and, furthermore, broadens the study of mathematical research on numerical integrators.



(a) Learnability coefficient (α_+) (b) Learnability coefficient (α_-) (c) Relative errors of actually trained NODE models.

Figure 4: Comparison of the theoretical results for the explicit midpoint method with the relative modeling errors of actually trained models.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: JST CREST Grant Number JPMJCR1914, JSPS KAKENHI Grant Number 19K20344, 20K11693.

References

[Anastassi, 2014] Anastassi, A. A. (2014). Constructing Runge-Kutta methods with the use of artificial neural networks. *Neural Computing and Applications*, 25(1):229–236.

[Butcher, 2016] Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations, Third Edition*. John Wiley & Sons, Ltd., Chichester.

[Celledoni et al., 2020] , Celledoni, E, Ehrhardt M. J, Etmann, C, McLachlan, R, I, Owren, B. and Schönlieb C.B. and Sherry, F. (2020). Structure preserving deep learning. *arXiv*, pages 1–42.

[Chen et al., 2018] Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D., Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–19.

[Chen et al., 2020] Chen, Z., Zhang, J., Arjovsky, M., and Bottou, L. (2020). Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR)*, pages 1–23.

[Cichock and Rolf Unbehauen, 1992] Cichock, A. and Rolf Unbehauen, D. (1992). Neural Networks for Solving Systems of Linear Equations and Related Problems. *IEEE Transactions on Circuits and Systems I*, 39(2):124–138.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, pages 1–15.

[Greydanus et al., 2019] Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–16.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

[Lagaris et al., 1998] Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000.

[Lu et al., 2017] Lu, Y., Zhong, A., Li, Q., Dong, B.. (2017). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. arXiv, 2017.

[Poznyak et al., 1996] Poznyak, A. S., Sanchez E. N., and Acosta, G. (1996). Error analysis for nonlinear system identification using dynamic neural networks, In *Proceedings Mexico-USA Collaboration in Intelligent Systems Technologies.*, pages 403–407.

[Raissi, 2018] Raissi, M. (2018). Deep Hidden Physics Models : Deep Learning of Nonlinear Partial Differential Equations. *Journal of Machine Learning Research*, 19:1–24.

[Ramuhalli et al., 2005] Ramuhalli, P., Udpa, L., and Udpa, S. S. (2005). Finite-element neural networks for solving differential equations. *IEEE Transactions on Neural Networks*, 16(6):1381–1392.

[Rudd et al., 2014] Rudd, K., Muro, G. D., and Ferrari, S. (2014). A constrained backpropagation approach for the adaptive solution of partial differential equations. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3):571–584.

[Zhong et al., 2020b] Zhong, Y. D., Dey, B., and Chakraborty, A. (2020b). Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations (ICLR)*, pages 1–17.

Appendices

A Proofs

Proof of Theorem 3.1. Suppose that the loss function (3) vanishes for the model with

$$\hat{f}(\hat{x}) = \alpha\hat{x}.$$

If this equation is discretized by the Runge–Kutta method with the initial condition $\hat{x}(nh) = x_{\mathrm{d}}^{(n)}$, it holds

$$\hat{x}^{(n+1)} = x_{\mathrm{d}}^{(n)} + h\sum_{i=1}^{p} b_i k_i, \quad k_i = \alpha(x_{\mathrm{d}}^{(n)} + h\sum_{j=1}^{p} a_{ij}k_j)), \tag{7}$$

which is rewritten as

$$\hat{x}^{(n+1)} = x_{\mathrm{d}}^{(n)} + hb^{\top}k, \quad k = \alpha(x_{\mathrm{d}}^{(n)}\mathbb{1} + hAk).$$

Assuming that $\det(I - \alpha hA) \neq 0$, we get

$$\hat{x}^{(n+1)} = x_{\mathrm{d}}^{(n)} + \alpha x_{\mathrm{d}}^{(n)} hb^{\top}(I - \alpha hA)^{-1}\mathbb{1}.$$

Because $x_{\mathrm{d}}^{(n+1)} = \mathrm{e}^{\lambda \mathrm{i}} x_{\mathrm{d}}^{(n)}$, in order for $\|\hat{x}^{(n+1)} - x_{\mathrm{d}}^{(n+1)}\|$ to be zero for all $x_{\mathrm{d}}^{(n)}$, it must hold

$$\mathrm{e}^{\lambda \mathrm{i}} x_{\mathrm{d}}^{(n)} = x_{\mathrm{d}}^{(n)} + \alpha x_{\mathrm{d}}^{(n)} hb^{\top}(I - \alpha hA)^{-1}\mathbb{1}.$$

for all $x_{\mathrm{d}}^{(n)}$. Hence, α should satisfy

$$\mathrm{e}^{\lambda \mathrm{i}} - 1 = \alpha hb^{\top}(I - \alpha hA)^{-1}\mathbb{1}.$$

□

Proof of Theorem 3.2. From the learnability equation, we have

$$b^{\top}(I - \frac{\alpha}{\lambda}\lambda hA)^{-1}\mathbb{1}\frac{\alpha}{\lambda}\lambda h - e^{\lambda h} + 1 = 0, \quad \det(I - \frac{\alpha}{\lambda}\lambda hA) \neq 0.$$

Therefore, we get

$$b^{\top}(I - \frac{\alpha}{\lambda}zA)^{-1}\mathbb{1}\frac{\alpha}{\lambda}z - e^{\lambda h} + 1 = 0, \quad \det(I - \frac{\alpha}{\lambda}zA) \neq 0,$$

which shows that α/λ and hence $\alpha/\lambda - 1$ are functions of z. □

B The details of the motivating examples

We explain the details of the numerical examples (Figs 1a and 1b) that motivated this study. The multilayer perceptron used in these tests has two fully-connection layers. 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 trained these neural networks in the following way. For convenience of illustration, we set the time step to the unit value: $\Delta t = 1$. For each λ, first we prepared the training data as tuples

$$\{(x_1, x_0) \mid x_1 = \exp(\lambda\Delta t)x_0\}$$

where we uniformly randomly sampled 10000 points for x_0's from $-10 \leq \mathrm{Re}\, x_0 \leq 10, -10 \leq \mathrm{Im}\, x_0 \leq 10$. Second, (1) was numerically integrated by using the explicit Euler method and the explicit midpoint rule to generate the data set

$$\{(x_0, \hat{x}_1) \mid \hat{x}_1 \text{ is the numerical one-step solution from } x_0 \text{ bythe integrator}\}.$$

After that, the neural networks were trained by minimizing the the mean squared error

$$\frac{1}{M} \sum_{x_0} \|x_1 - \hat{x}_1\|_2^2,$$

where M is the number of the data.

To investigate the dependence of the modeling errors on the characteristics of dynamics, we changed the values of λ to $\mathrm{i}, 5\mathrm{i}$ and $7.5\mathrm{i}$. As explained in Introduction, $\lambda \in \mathbb{C}$ characterises the differential equation in the sense that $\mathrm{Re}\ \lambda$ and $\mathrm{Im}\ \lambda$ correspond to the speed of decay (or blowup) and the speed of oscillation, respectively. Figure 1a shows the orbits of the discrete and the continuous models. The colored points represent the validation data and the results of the discrete model. The dashed lines are the true solutions of the learned model computed by the Runge–Kutta method with a tiny time step. Remarkably the non-trivial dependence of the modeling errors on the value of λ is observed. In fact the modeling errors for $\lambda = \mathrm{i}, 7.5\mathrm{i}$ are apparently much smaller than that for $\lambda = 5\mathrm{i}$.

Secondly, we compared the dependence of the errors on the integrators by learning the dynamics with the explicit Euler method and the explicit midpoint method. Then, we set $\lambda = \mathrm{i}$ and the other conditions are same as before. The results are shown in Figure 1b. As expected, the modeling error of the explicit midpoint method is much smaller than that of the Euler method.

From the above observations we numerically confirm that the modeling errors depends both on the characteristics of the dynamics (i.e., the value of λ) and the numerical integrators. These results motivate development of a framework of analysis of the modeling errors of numerical integrators with respect to the type of dynamics.