Nonlinear Dyn (2023) 111:21117–21130
https://doi.org/10.1007/s11071-023-08933-6

ORIGINAL PAPER

Parameter estimation and modeling of nonlinear dynamical systems based on Runge–Kutta physics-informed neural network

Weida Zhai · Dongwang Tao · Yuequan Bao

Received: 24 May 2023 / Accepted: 15 September 2023 / Published online: 1 October 2023
© The Author(s), under exclusive licence to Springer Nature B.V. 2023

Abstract To identify the nonlinear behavior of dynamical systems subjected to external excitations such as earthquakes, explosions, and impacts, a novel method is proposed for dynamical system parameter estimation and modeling based on the combination of the physics-informed neural network (PINN) and Runge–Kutta algorithm. Drawing inspiration from classical numerical integration solution rules for differential equations, a new recurrent neural network architecture is designed for modeling. PINN cells are embedded as the basic integration units of the architecture to introduce prior physics-based biases. And the motion equations of different dynamical systems can be flexibly added to this architecture as soft constraints for neural network training. The method, referred to as the Runge–Kutta physics-informed neural network (RK-PINN), differs from black-box learning, as it models the evolutionary process of nonlinear dynamical systems. The satisfactory parameter estimation capability of the RK-PINN method was demonstrated through two illustrative examples. The results indicate that embedding physics can reduce the data required for training, and the trained network can serve as a surrogate model for the dynamical system to predict future states or responses.

W. Zhai · Y. Bao (✉)
School of Civil Engineering, Harbin Institute of Technology, Harbin 150090, China
e-mail: baoyuequan@hit.edu.cn

D. Tao
Institute of Engineering Mechanics, China Earthquake Administration, Harbin 150080, China

Keywords Physics-informed neural network · Runge–Kutta · Parameter estimation · Modeling · Nonlinear dynamical system

1 Introduction

Most response prediction tasks for dynamical systems can be modeled as initial value problems (IVP) for ordinary differential equations (ODEs). Researchers have tried to solve ODEs to get the expression forms of solutions or trajectories [1]. However, only certain special nonlinear problems can be solved for closed-form analytic solutions. Therefore, solving most differential equations relies on numerical integration algorithms to obtain a numerical solution approximating the true solution. Classic numerical integration algorithms, such as the Euler method [2] and the Runge–Kutta method [3,4], are based on the idea of discretizing continuous problems and moving forward by step along the trajectory. Different from the first-order Euler method, higher-order Runge–Kutta methods can achieve lower truncation errors. For over a century, different truncation orders of Runge–Kutta methods have developed into a class of simple and efficient frame [5,6] for solving ODEs, among which the most common one in applications is the fourth-order Runge–Kutta (RK4) method. As Runge–Kutta methods are linear extrapolation methods, their structures are highly compatible with the Gaussian Process regression algorithm [7] to embed a probabilistic distribution model.

Based on the universal approximation theorem [8], a sufficiently deep neural network can fit any dynamical system theoretically under the drive of big data. According to E's statement [9], the architecture of deep neural networks can be viewed as a numerical differential equation, and network training can be viewed as the optimal representation of solving the differential equation. The residual connection structure of the famous ResNet [10] has the same algorithmic approach as the classical Euler method. Lu's research shows that many effective neural networks can be explained as differential equations, and they proposed a new network architecture based on the linear multi-step method of differential equations [11]. Much research has already been conducted on solving ODEs, meaning many novel network architectures need to be built and validated. The combinations of the Runge–Kutta methods and neural networks have attracted the attention of researchers. Wang et al. proposed a feedforward neural network called the Runge–Kutta neural network (RKNN) to identify unknown dynamical systems [12]. By borrowing from the framework of the numerical integration method to approximate ODEs, the network can achieve higher prediction accuracy and better generalization ability. The Runge–Kutta methods have been extended to other neural networks, such as the Runge–Kutta convolutional neural network (RKCNN) that was proposed for modeling the visual system [13]. ODE-Net and RK-Net [14], derived from RKNN, achieved similar performance to ResNet with fewer parameters. Furthermore, the physical property of sparsity was embedded in machine learning tasks by Runge–Kutta inspired dictionary-based sparse regression to discover nonlinear ODEs [15].

Different from the integration methods to calculate the IVP, the data-driven black-box modeling approaches for dynamical systems have developed rapidly in recent years [16–21]. Traditional numerical methods for modeling and predicting responses of dynamic systems require knowledge of the physical properties, such as mass, stiffness, and damping matrices, as well as accurate analytical models for the nonlinear components. However, this is often difficult to achieve, as the complex dynamic systems in practical engineering applications are difficult to describe by simple mathematical differential equations accurately. At the same time, high-dimensional numerical integration and the coupling of multiple degrees of freedom in dynamic characteristics can lead to high computational costs and divergent solutions. However, recurrent neural network (RNN) [22] has shown remarkable performance in sequence modeling tasks. Dynamical recurrent neural network (DRNN) was used to model single-input–single-output nonlinear systems without requiring exact observation knowledge [23]. The famous long short-term memory (LSTM) [24] network adds a forget gate mechanism to RNN, which slows down gradient vanishing and makes longer time series prediction possible. Chen et al. proposed a data-driven method [25] based on LSTM to predict the seismic response of a 42-story reinforced concrete frame-core structure. Furthermore, LSTM can effectively learn the dynamic behavior of strong nonlinear chaotic systems such as the Lorenz system [26]. However, the deeper the network model, which improves the ability to fit complex nonlinearity, the more difficult it is to explain. If a model lacks the incorporation of physical information, it may predict inconsistently or implausibly owing to poor generalization beyond training data.

From another perspective, embedding physical information into the network is closely related to nonlinear parameter estimation problems. It is a dual problem of IVP, intending to infer the characteristics and parameters of a dynamical system by observing trajectories. Kalman filter (KF) [27] and its variants, such as extended Kalman filter (EKF) [28] and unscented Kalman filter (UKF) [29], are widely used in dynamical system tracking. Markov chain Monte Carlo (MCMC) algorithm is used to calculate of the likelihood function in non-Gaussian state estimation [30]. However, these methods are all based on Bayesian inference, which involves the serious difficulty of the choice of the prior distribution [31]. If the prior probability differs significantly from reality, it can lead to misestimation. Empirical formulas are artificially introduced in the description of the nonlinear dynamical evolution process, which is insufficient to fit the complex nonlinear behavior perfectly. PINN provides a new parameter estimation approach, seamlessly integrating data and physical information, even including formulas with partially missing or uncertain. The underlying physical principles of the system can be learned through theoretical constraints. Empirical formulas are replaced by neural networks driven by experimental data with physical laws. Generally, mathematical equations are added to the loss function of traditional neural networks as soft penalty constraints through automatic differentiation [32]. This provides a flexible platform for introducing physics-based inductive biases [33]. Raissi validated deep hidden physics models on problems such as the Burgers', Korteweg–de Vries, and Navier–Stokes equations [34]. PINN inspired by Runge–Kutta can solve forward and inverse problems involving nonlinear partial differential equations [35]. Zhang et al. proposed the physics-informed multi-LSTM network to metamodel structures under nonlinear seismic excitation with limited available training datasets [36]. A data-driven framework based on physics-integrated deep learning was established to identify the underlying nonlinear normal modes (NNMs) from response data only [37]. Physical-informed machine learning that makes the black-box network interpretable is a promising area in the future.

The motivation of this work is to find a generic workflow for modeling and identifying nonlinear dynamical systems with partially unknown physical information based on data-driven and physics-driven. This workflow not only addresses prediction tasks but also employs neural networks to estimate the characteristic parameters of nonlinear systems. For problems that can be represented by ODEs, the unknown physical information is represented as parameters to be estimated. The proposed method allows for the estimation of relevant system parameters during the network training process, facilitating the integration of black-box neural network models with known mathematical expressions. This method enhances the physical interpretability of the network modeling, while theoretically improving the accuracy of prediction task by incorporating prior knowledge.

This article is organized as follows. Section 2 introduces the architecture of RK-PINN, taking the architecture of RK4-PINN as an example. The parameters expressed from the PINN using automatic differentiation are added to the loss function with time-invariant properties. The effect of the physical loss term on network gradient learning will be analyzed. In Sect. 3, the performance of RK4-PINN was verified by the Lorenz chaotic system and a single degree of freedom (SDOF) nonlinear numerical example. Section 4 summarizes the conclusions. The data and codes used in this paper will be available on GitHub at https://github.com/VVeida/RK4_PINN after the paper is published.

2 Design of architecture of RK4-PINN for dynamic systems

2.1 Problem statement and the Runge–Kutta methods

A continuous time-invariant dynamical system, linear or nonlinear, can be described by the following ODE:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) \tag{1}$$

where $\boldsymbol{x}(t)$ is the state vector and $\dot{\boldsymbol{x}}(t) = \partial \boldsymbol{x}(t)/\partial t$ holds for all $t \in [t_0, t_T]$. Equation (1) relies on the dynamical evolution function f, assumed to be time-invariant, on its right-hand side to capture the system's dynamic behavior. Although only autonomous systems are discussed here to introduce the Runge–Kutta method, the proposed approach can also be extended to certain specific non-autonomous systems [38]. As the second illustrative example, if the system explicit time components are solely contained within known external inputs, a time-invariant evolution function f can be separated from the state-space equation as well. The IVP, given an initial condition $\boldsymbol{x}(0)$, aims to obtain the corresponding solution

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} f(\boldsymbol{x}(\tau))\mathrm{d}\tau \tag{2}$$

However, not all differential equations have analytical solutions. The classical numerical integration methods are used to approximate the analytical solution. Time $t \in [t_0, t_T]$ is discretized into N time steps, where each time step has a length denoted by $h > 0$.

$$t_n = t_0 + nh \tag{3}$$

where $0 \le n \le N$ is an integer. The continuous problem is then discretized to approximate the value of $\boldsymbol{x}(t)$ at each time step n.

$$\begin{cases} \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \\ \boldsymbol{x}_{n+1} = \Phi(\boldsymbol{x}_n) \end{cases} \tag{4}$$

where Φ is a given integration scheme related to f.

The Runge–Kutta methods are the most widely used single-step explicit integration schemes that operate integration on small contiguous subintervals and predict the state of the system step by step. As mentioned above, the Runge–Kutta methods are linear extrapolations based on Taylor series expansion. A k-order Runge–Kutta method can be written as

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h \sum_{i=1}^{k} c_i \boldsymbol{s}_i \tag{5}$$

where $\boldsymbol{s}_i$ is the first-order derivative values of special points on the subinterval and c_i is the coefficient of their linear combination. In contrast to the Taylor formula, a series of first-order derivative values are combined to obtain the same prediction as the power series expansion at each time step. The Runge–Kutta method is characterized by its order k that determines the accuracy and computational cost. As shown in Fig. 1, the most commonly used is the RK4 method

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{h}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right) \tag{6}$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= f\left(\boldsymbol{x}_n\right) \\ \boldsymbol{k}_2 &= f\left(\boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k}_1\right) \\ \boldsymbol{k}_3 &= f\left(\boldsymbol{x}_n + \frac{h}{2}\boldsymbol{k}_2\right) \\ \boldsymbol{k}_4 &= f\left(\boldsymbol{x}_n + h\boldsymbol{k}_3\right) \end{aligned} \tag{7}$$

2.2 Design of the RK4-PINN

From Fig. 1, the key to calculating the dynamical response by the RK method lies in utilizing the evolution function f. The first-order derivative values are recursively fed back into f four times when estimating at each time step. However, f is typically unknown or only partially known in applications. In order to predict the trajectories of unknown systems, RKNN [12], which first combined the Runge–Kutta method and neural network, used the multilayer perceptron network or radial basis function network. This work takes a further step by aiming to obtain the dynamical behavior using the novel PINN, named RK-PINN. Taking RK4-PINN as an example, its computation flow is shown in Fig. 2.

As shown in Fig. 2, inspired by RK4, the general structure of RK4-PINN is similar to Fig. 1. In order to learn the uncertain function f, PINN cells replace its position. It is worth noting that PINN cells have the same structure with consistent trainable parameters θ. Therefore, only one code implementation for PINN is required in the program. It means that the calculating consumption of the RK4-PINN network is consistent with only one cell.

The input $\boldsymbol{x}_n$ enters the PINN cell four times recurrently at each time step, in conjunction with the corresponding weight coefficients c_i of RK4. Based on the idea of the single-step method, the output of RK4-PINN is the state prediction for the next time step $\tilde{\boldsymbol{x}}_{n+1}$, where the upper tilde represents the estimated value from the network. The mapping function corresponding to RK4-PINN can be noted as $\mathcal{F}$. The network parameters are trained by minimizing the prediction loss calculated as the mean squared error (MSE).

$$\begin{aligned} \text{Loss}_{\text{pred}} &= \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_{n+1} - \tilde{\boldsymbol{x}}_{n+1}\right)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_{n+1} - \mathcal{F}\left(\boldsymbol{x}_n;\theta\right)\right)^2 \end{aligned} \tag{8}$$

If the loss function only contains $\text{Loss}_{\text{pred}}$, the network is merely a black-box mapping without interpretability. It becomes crucial to embed physical information into the loss function. The role of the PINN cell denoted as g is to approximate f by implementing the following function:

$$\tilde{\boldsymbol{x}}_n = g\left(\boldsymbol{x}_n;\theta\right) \tag{9}$$

Automatic differentiation, one of the critical components of PINN, is a computational technique for automatically computing the derivatives of a function with respect to its inputs. Unknown parameters denoted as $\boldsymbol{\mu}_{\text{phy}}$ represent the unknown physical information of a dynamical system. These unknown time-invariant parameters can be calculated by

$$\boldsymbol{\mu}_{\text{phy},n} = AD_f\left(\boldsymbol{x}_n, g\left(\boldsymbol{x}_n;\theta\right)\right) \tag{10}$$

where AD_f denotes the automatic differentiation function based on prior physical knowledge of f. The focus research object of this work is nonlinear dynamical systems with time-invariant parameters. The time-invariant property indicates that for different steps n, the calculated values of $\boldsymbol{\mu}_{\text{phy},n}$ are the same. Therefore, the loss function Loss_{phy} that embeds physical information is in the form of coefficient of variation as

$$\begin{aligned} \text{Loss}_{\text{phy}} &= c_v^2\left(\boldsymbol{\mu}_{\text{phy}}\right) \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{\mu}_{\text{phy,n}} - \overline{\boldsymbol{\mu}}_{\text{phy}}\right)^2/\overline{\boldsymbol{\mu}}_{\text{phy}}^2 \end{aligned} \tag{11}$$

where c_v is the coefficient of variation and

$$\overline{\boldsymbol{\mu}}_{\text{phy}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\mu}_{\text{ply,n}} \tag{12}$$

Besides constraining the network training by setting the $c_v^2\left(\boldsymbol{\mu}_{\text{ply}}\right)$ to be zero, the Loss $_{\text{phy}}$ can also use Kullback–Leibler (KL) divergence to gradually push the distribution of $\boldsymbol{\mu}_{\text{phy},n}$, denoted as $q\left(\boldsymbol{\mu}_{\text{phy}}\right)$, toward a smaller

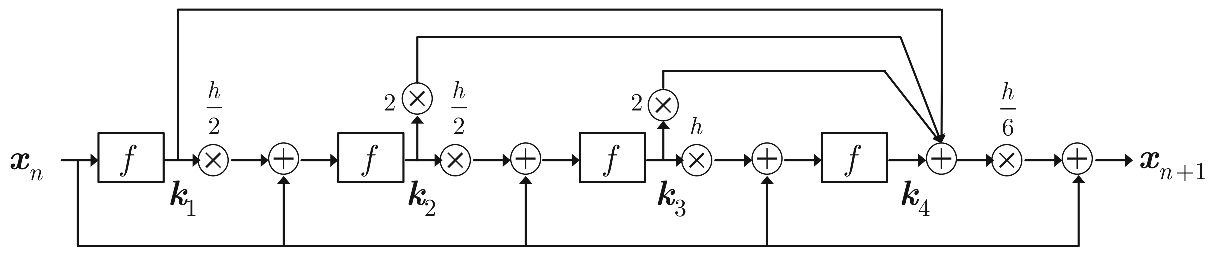

Fig. 1 Flowchart of RK4

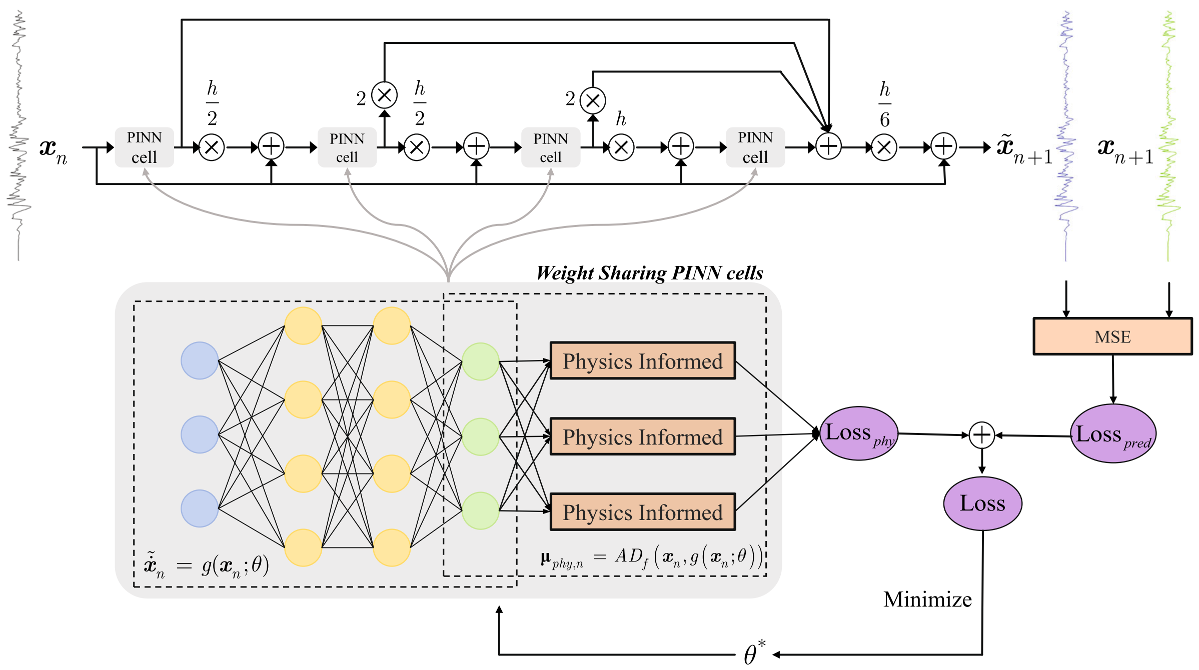

Fig. 2 Flowchart of RK4-PINN

pre-set variance Gaussian distribution $p\left(\boldsymbol{\mu}_{\text{phy}}\right)$.

$$\text{Loss}_{\text{phy}} = KL\left(p(\boldsymbol{\mu}) \| q\left(\boldsymbol{\mu}_{\text{phy}}\right)\right) \tag{13}$$

The Loss_{phy} in the form of c_v is chosen in this work because it is more meaningful in terms of time-invariant properties and is the extreme case of the KL form.

In summary, the loss function is primarily comprised of two parts: the prediction loss and the physical time-invariant loss, which respectively correspond to data-driven and physics-driven approaches.

$$\text{Loss} = \alpha\, \text{Loss}_{\text{pred}} + \beta\, \text{Loss}_{\text{phy}} \tag{14}$$

where α and β are the weight coefficients of the corresponding loss part, respectively.

2.3 Determination of loss weighting coefficients

Equation (14) defines a joint loss function based on both data and physical prior knowledge, which is used to train RK4-PINN. The learning process of the network becomes a multi-objective optimization problem. The model can simultaneously utilize data-driven and physics-driven information by minimizing the loss function via gradient-based optimizers. Thus, taking appropriate values for α and β is crucial, which keep the balance between training data and physics.

As it is widely acknowledged, the ability of $\text{Loss}_{\text{pred}}$ is to make the neural network fit the dynamical system as closely as possible. Therefore, it is worthwhile to study the effect of Loss_{phy} designed in this work.

Assume that the neural network g used to represent the PINN cell has m layers with the tanh activation function, and the output layer is the fully connected without the activation function. Then, its forward formulas can be expressed as

$$\begin{aligned} \boldsymbol{Z}_i &= \boldsymbol{W}_i \boldsymbol{A}_{i-1} + \boldsymbol{B}_i \\ \boldsymbol{A}_i &= \tanh(\boldsymbol{Z}_i), i = 1, 2, \cdots, m-1 \end{aligned} \tag{15}$$

where $\boldsymbol{W}_i$ and $\boldsymbol{B}_i$ are the weights and bias parameters of the ith layer; $\boldsymbol{Z}_i$ and $\boldsymbol{A}_i$ denote the weighted input and activation output of the ith layer, respectively. The following relationship can be obtained From Eq. (9).

$$\begin{aligned} \boldsymbol{A}_0 &= \boldsymbol{x} \\ \tilde{\boldsymbol{x}} &= \boldsymbol{W}_m \boldsymbol{A}_{m-1} + \boldsymbol{B}_m \end{aligned} \tag{16}$$

The automatic differentiation technique derives the output of g from the input to obtain the physical parameters at each time step n.

$$\begin{aligned} \frac{\partial \tilde{\boldsymbol{x}}_n}{\partial \boldsymbol{x}_n} &= \frac{\partial \tilde{\boldsymbol{x}}_n}{\partial \boldsymbol{A}_{m-1}} \cdot \frac{\partial \boldsymbol{A}_{m-1}}{\partial \boldsymbol{Z}_{m-1}} \cdot \frac{\partial \boldsymbol{Z}_{m-1}}{\partial \boldsymbol{A}_{m-2}} \cdots \frac{\partial \boldsymbol{A}_1}{\partial \boldsymbol{Z}_1} \cdot \frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{A}_0} \\ &= \prod_{i=1}^{m-1} \left(\boldsymbol{W}_i^T \cdot \operatorname{diag}\left(1 - \tanh^2(\boldsymbol{Z}_i)\right)\right) \cdot \boldsymbol{W}_m \end{aligned} \tag{17}$$

The time-invariant property requires $\partial \tilde{\boldsymbol{x}}_n / \partial \boldsymbol{x}_n$ to be consistent for different time step n, which creates a tendency for $\boldsymbol{W}_i$ to be close to zero.

The impact of Loss_{phy} is analogous to that of the l_1 and l_2 regularization terms [39], which are used to prevent overfitting by adding a regularization term that penalizes the values of network parameters. During the network training process, sparsity is introduced into the loss function by Loss_{phy} shrinking some weights to zero, which reduces model complexity.

In summary, the weight coefficient for Loss_{phy} can be determined using methods such as grid search, with reference to the weight coefficient for l_1 and l_2 regularization. Generally, the coefficient α for the fitting term $\text{Loss}_{\text{pred}}$ should prevail over β for the regularization term Loss_{phy} to ensure the modeling of specific dynamical systems.

3 Numerical case studies

3.1 Lorenz chaotic system

Lorenz chaotic system, a well-known nonlinear system proposed by Edward Lorenz in 1963 [40], is the beginning of modern chaos research. The Lorenz chaotic system is described by three nonlinear differential equations:

$$\begin{aligned} \dot{x} &= s(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= -qz + xy \end{aligned} \tag{18}$$

where s, r, and q, respectively, represent the Prandtl number, Rayleigh number, and geometric aspect ratio. The Lorenz system exhibits various behaviors for different values of the parameters, including spirals, oscillations, and butterfly-shaped attractors. The most common combination of parameters is to fix $s = 10$ and $q = 8/3$, and then vary r. When r is small, the system exhibits regular oscillations. As r increases, the system undergoes a series of period-doubling bifurcations, leading to chaotic behavior with increasing complexity. For huge values of r, the system exhibits intermittent behavior with periodic bursts of chaotic activity. We investigated Lorenz systems with parameter $r = 28$ for the nonperiodic chaotic state using RK4-PINN.

Currently, the RK4-PINN method is a single-step method. Similar to the teacher-forcing training method for LSTM, the true value of each time step is used as the input to the network instead of the output of the previous step. Therefore, the predicted loss function of RK4-PINN for the Lorenz system is expressed as

$$\text{Loss}_{\text{pred}} = \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{X}_{n+1} - \mathcal{F}(\boldsymbol{X}_n; \theta)\right)^2 \tag{19}$$

with state vector $\boldsymbol{X} = \begin{bmatrix} x & y & z \end{bmatrix}^T$.

According to Eq. (18), the PINN cell for Lorenz chaotic system is depicted in Fig. 3. The PINN cell under consideration features a total of three inputs, corresponding specifically to x, y, z, and three outputs, which are identified as $\tilde{\tilde{x}}$, $\tilde{\tilde{y}}$, $\tilde{\tilde{z}}$. Moreover, it consists of five hidden layers, each comprising 24 neurons. The physical parameters of the Lorenz system in the PINN cell using automatic differentiation is formulated as:

$$\begin{aligned} \tilde{s} &= \frac{\partial \tilde{\tilde{x}}}{\partial y} = -\frac{\partial \tilde{\tilde{x}}}{\partial x} \\ \tilde{r} &= \frac{\partial(\tilde{\tilde{y}} + y + xz)}{\partial x} \\ \tilde{q} &= \frac{\partial(-\tilde{\tilde{z}} + xy)}{\partial z} \end{aligned} \tag{20}$$

Thus, the physical loss can be expressed as

$$\text{Loss}_{\text{phy}} = c_v^2(\tilde{s}) + c_v^2(\tilde{r}) + c_v^2(\tilde{q}) \tag{21}$$

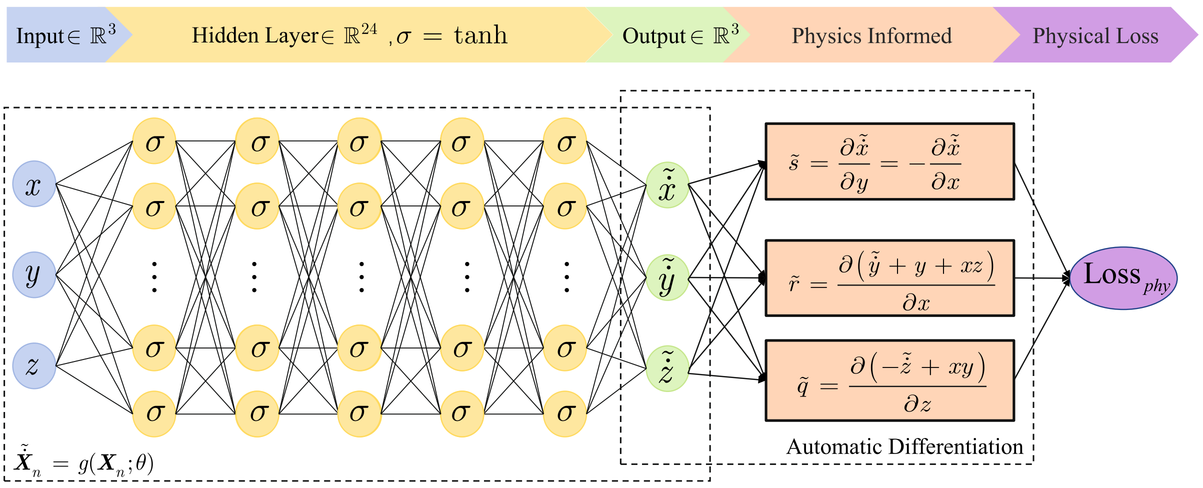
Input $\in \mathbb{R}^3$
Hidden Layer $\in \mathbb{R}^{24}$, $\sigma = \tanh$
Output $\in \mathbb{R}^3$
Physics Informed
Physical Loss
x
y
z
σ
$\tilde{\dot{x}}$
$\tilde{\dot{y}}$
$\tilde{\dot{z}}$
$\tilde{s} = \frac{\partial \tilde{\dot{x}}}{\partial y} = -\frac{\partial \tilde{\dot{x}}}{\partial x}$
$\tilde{r} = \frac{\partial(\tilde{\dot{y}} + y + xz)}{\partial x}$
$\tilde{q} = \frac{\partial(-\tilde{\dot{z}} + xy)}{\partial z}$
Loss_{phy}
Automatic Differentiation
$\tilde{\dot{X}}_n = g(X_n;\theta)$

Fig. 3 PINN cell for the Lorenz chaotic system

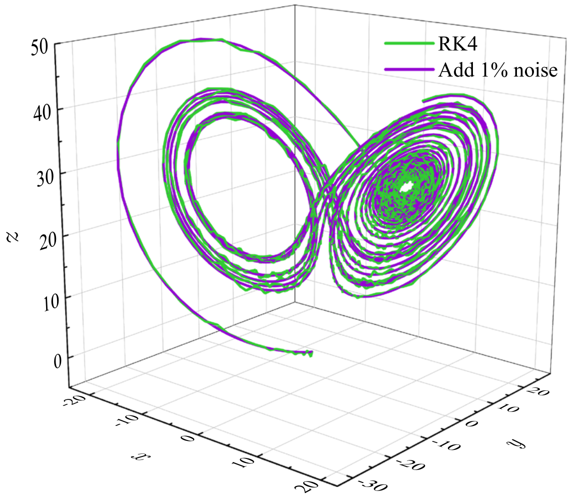

Fig. 4 Lorenz system dataset with 1% noise

The Lorenz dataset used in this work was obtained through the RK4 method, with a time step of 0.01 s over 30 s. For the Lorenz system, the small changes in the initial values will cause significantly alter the trajectory. The initial values of the training and validation set were randomly generated within the interval of $[-2, -2, -2; 2, 2, 2]$. There were only 20 training samples, and the trajectories were randomly divided into a training set (80%) and a validation set (20%). As shown in Fig. 4, to prevent the dataset from being overly idealized by matching the RK4-PINN, 1% noise was added to the dataset.

The batch size and training epochs of RK4-PINN for Lorenz system were 32 and 300, respectively. During the training process, a learning rate adjustment was implemented with an initial learning rate of 0.01, a fixed step size of 10, and a decay rate of 0.9. The RK4-PINN was trained by minimizing the loss through the gradient-based optimizer Adam. The grid search method was utilized to find the values of weights α and β. The results of parameter estimation for the Lorenz system with $s = 10$, $r = 28$, and $q = 8/3$ are shown in Table 1.

When $\beta = 1$ is fixed, the lower the position in Table 1, the larger the value of α, indicating that the weight of $\text{Loss}_{\text{pred}}$ is more dominant than that of Loss_{phy}. It can be observed that when $\alpha = 10$ and $\alpha = 100$, the estimation performances of the Lorenz parameters are better. This result is in line with the analysis presented in Sect. 2.3, where it is shown that the weight coefficient of Loss_{phy}, acting as a regularization term, should be lower than that of $\text{Loss}_{\text{pred}}$. However, as shown in the last line in Table 1, $\beta = 0$ is the most extreme case. And then, the method becomes RKNN where Loss_{phy} is not included in the loss function. The estimation results of the three parameters by RKNN are entirely wrong, and the standard deviations are too large to determine the estimated values. This demonstrates that the Loss_{phy} proposed in this work can effectively guide networks to learn the potential dynamic characteristics.

Taking the case of $\alpha = 100$ as an example in the following analysis, the training and validation loss curves are shown in Fig. 5. It can be seen that the curve con-

Table 1 Parameter estimation for the Lorenz system

α	β	s_{mean}	s_{std}	s_{error} (%)	r_{mean}	r_{std}	r_{error} (%)	q_{mean}	q_{std}	q_{error} (%)
1	1	9.305	0.193	6.95	28.059	0.322	0.21	2.775	0.024	4.06
10	1	9.750	0.243	2.50	27.945	0.343	0.20	2.781	0.044	4.26
100	1	10.254	0.553	2.54	28.010	0.528	0.03	2.751	0.094	3.15
1000	1	10.438	0.929	4.38	27.979	1.217	0.08	2.739	0.350	2.70
10	0	6.668	123.243	33.32	4.689	235.298	83.25	5.394	63.139	102.24

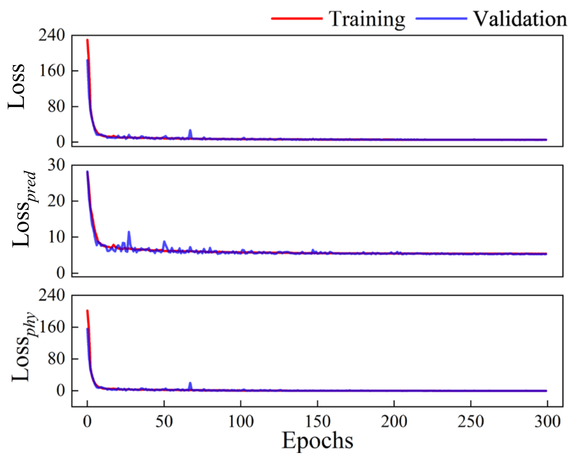

Fig. 5 Training and validation loss curves

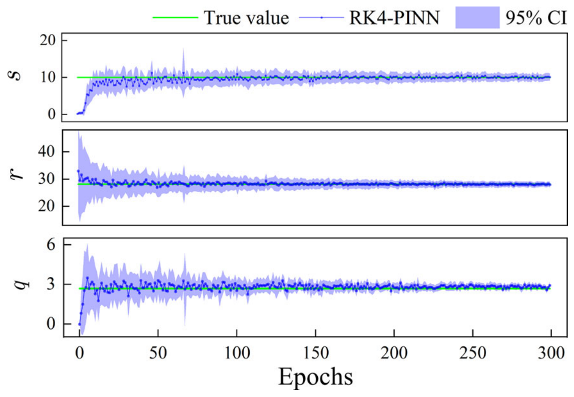

Fig. 6 Estimation process for the parameters of the Lorenz system

verges earlier for Loss$_{\text{phy}}$, while Loss$_{\text{pred}}$ fluctuates more.

The estimation process for the Lorenz system is shown in Fig. 6, where RK4-PINN can use the estimated mean and variance to draw the 95% confidence interval (CI). Figure 6 illustrates that although the RK4-PINN method initially has a larger confidence interval, the uncertainty of the parameters gradually decreases as the embedding of physical time-invariant properties.

To compare the modeling capabilities of the trained RK4-PINN for the Lorenz system, LSTM was introduced to the same small dataset. The LSTM cell had three neurons in both its input and output layers, and 128 neurons in its hidden layer. Teacher-forcing method was applied to train LSTM for 300 epochs, the same as RK4-PINN. In addition, the estimated parameters by RK4-PINN were also used to substitute for Eq. (18) to predict the trajectory.

For the more extensive ranges of randomly initial points outside of the training data set, take a randomly generated point [−3.163, 7.312, 7.158] as an example, the predicted motion trajectories for the next 30 s by different algorithms are shown in Fig. 7. Due to the complexity of the Lorenz chaotic system, even if the estimated parameters are close to the true values, the calculated motion differs significantly from the reference trajectory. Compared with LSTM, RK4-PINN exhibits a better modeling performance in capturing the underlying behaviors on the same small-scale dataset due to the seamless physical information integration. The modeling result of LSTM presents a periodic solution with small oscillations, while RK4-PINN can clearly model the Lorenz attractor.

As shown in Fig. 8, the three modeling methods exhibit a similar butterfly-shaped pattern. While the trajectories exhibit unpredictability and complexity in three-dimensional space, the positions of the butterfly-shaped Lorenz attractors are relatively close to each other.

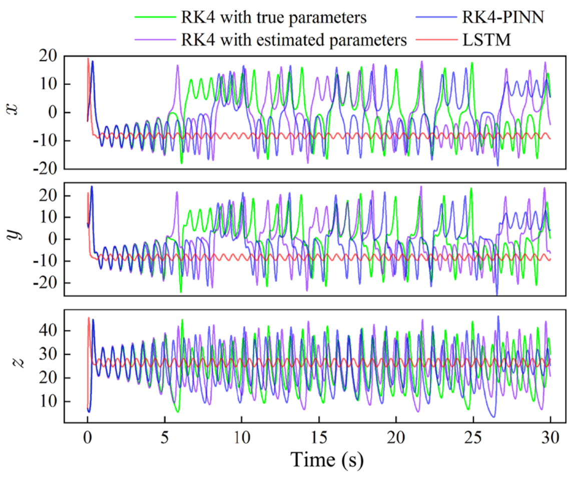

Fig. 7 Comparison of different algorithms for modeling the Lorenz system

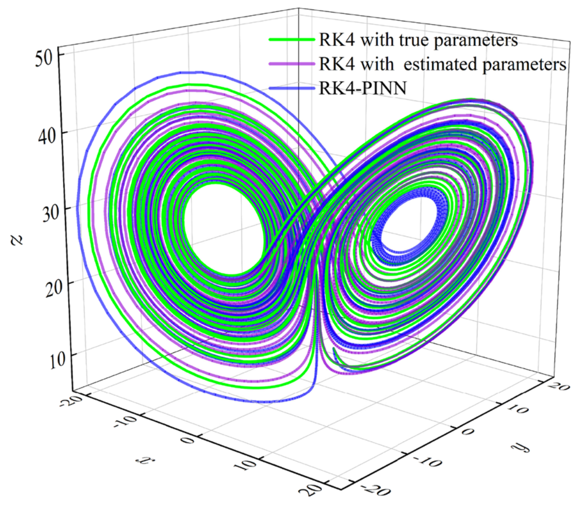

Fig. 8 Similar butterfly-shaped pattern obtained by three modeling methods

3.2 A SDOF simplified model with a nonlinear Duffing spring

The highly simplified model, which views a structure as a series of connected mass systems, is widely used in civil engineering to study the elastic–plastic seismic response of structures. As presented in Fig. 9, a SDOF simplified model with a nonlinear Duffing spring is studied to verify the parameter estimation capability of RK4-PINN. As shown in Fig. 10, the Duffing spring demonstrates no residual displacement and gradually exhibits a third-order polynomial hardening behavior.

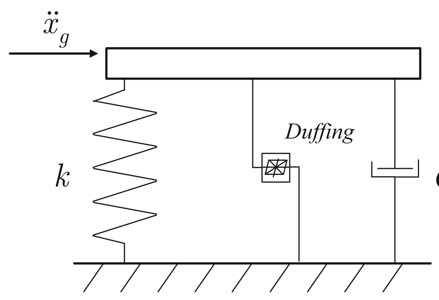

Fig. 9 A SDOF simplified model with a nonlinear Duffing spring

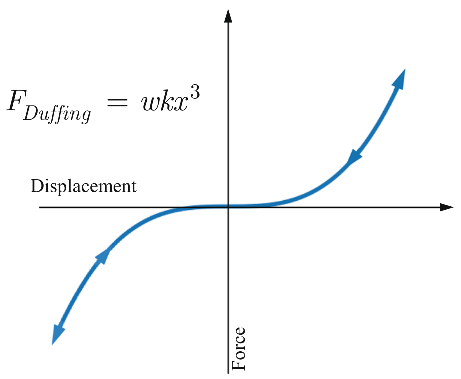

Fig. 10 Force-displacement curve of Duffing spring

The governing equation of motion for the SDOF model can be written as

$$m\ddot{x} + c\dot{x} + kx + wkx^3 = -m\ddot{x}_g \tag{22}$$

where x, $\dot{x}$, and $\ddot{x}$ are the relative displacement, velocity, and acceleration vector to the ground; $\ddot{x}_g$ is the ground acceleration input; w is the nonlinear coefficient of the Duffing spring. In this numerical example, the state vector can be written as $\boldsymbol{X} = \begin{bmatrix} x & \dot{x} \end{bmatrix}^T$. Then, Eq. (22) can be expressed in state space as

$$\begin{aligned} \dot{\boldsymbol{X}} &= \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = f(\boldsymbol{X}) \\ &= \begin{bmatrix} \dot{x} \\ -\frac{c}{m}\dot{x} - \frac{k}{m}x - w\frac{k}{m}x^3 \end{bmatrix} + \begin{bmatrix} 0 \\ -\ddot{x}_g \end{bmatrix} \end{aligned} \tag{23}$$

Notice that the first element in $\dot{\boldsymbol{X}}$ is the velocities in $\boldsymbol{X}$, then it can be directly represented by the identity transformation. In contrast, the mapping of the second element can be represented by a PINN cell, denoted as g_a.

$$\tilde{\ddot{x}} = g_a(x, \dot{x}; \theta) \tag{24}$$

The PINN cell for this SDOF simplified model can also be relatively simple, with only two hidden layers and

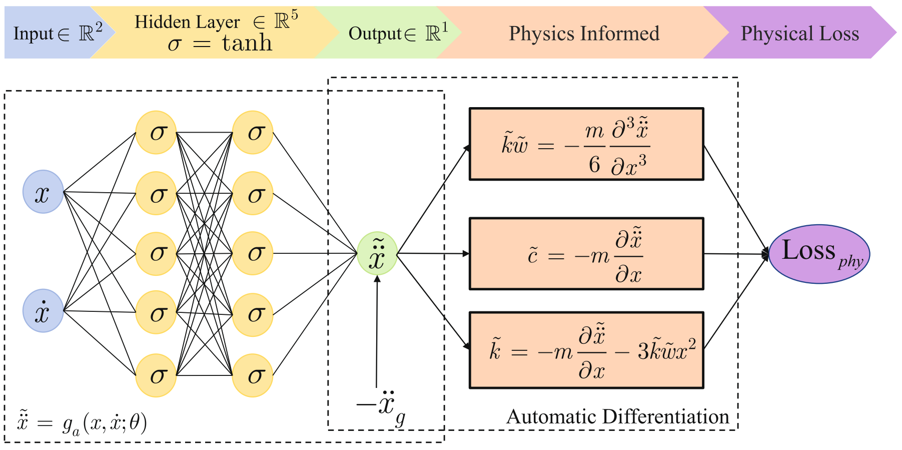
Input $\in \mathbb{R}^2$
Hidden Layer $\in \mathbb{R}^5$ $\sigma = \tanh$
Output $\in \mathbb{R}^1$
Physics Informed
Physical Loss
x
$\dot{x}$
σ
$\tilde{\ddot{x}}$
$-\ddot{x}_g$
$\tilde{x} = g_a(x, \dot{x}; \theta)$
$\tilde{k}\tilde{w} = -\frac{m}{6}\frac{\partial^3 \tilde{\ddot{x}}}{\partial x^3}$
$\tilde{c} = -m\frac{\partial \tilde{\ddot{x}}}{\partial \dot{x}}$
$\tilde{k} = -m\frac{\partial \tilde{\ddot{x}}}{\partial x} - 3\tilde{k}\tilde{w}x^2$
Loss_{phy}
Automatic Differentiation

Fig. 11 PINN cell for the SDOF simplified model with a nonlinear Duffing spring

five neurons per layer. Corresponding to Eq. (24), g_a has two inputs and one output. The architecture of the PINN cell is shown in Fig. 11. It is noteworthy that the output layer bias of the PINN cell is set to $-\ddot{x}_g$.

The predicted loss function of RK4-PINN for the SDOF model is expressed as

$$\mathrm{Loss}_{\mathrm{pred}} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{X}_{n+1} - \mathcal{F}\left(\boldsymbol{X}_n; \theta\right)\right)^2 \tag{25}$$

where N is the amount of sampled data.

The physical parameters in g_a using automatic differentiation is formulated as

$$\begin{aligned}
\tilde{k}\tilde{w} &= -\frac{m}{6}\frac{\partial^3 \tilde{\ddot{x}}}{\partial x^3} \\
\tilde{c} &= -m\frac{\partial \tilde{\ddot{x}}}{\partial \dot{x}} \\
\tilde{k} &= -m\frac{\partial \tilde{\ddot{x}}}{\partial x} - 3\tilde{k}\tilde{w}x^2
\end{aligned} \tag{26}$$

Then, the physical loss function can be expressed as

$$\mathrm{Loss}_{\mathrm{phy}} = c_v^2(\tilde{c}) + c_v^2(\tilde{k}) + c_v^2(\tilde{w}) \tag{27}$$

Furthermore, the stiffness and damping of the actual structure are typically positive. The ReLU activation function can be used to add this part of the loss function by

$$\mathrm{Loss}_{\mathrm{pos}} = \mathrm{ReLU}\left(-\tilde{k}\right) + \mathrm{ReLU}\left(-\tilde{c}\right) \tag{28}$$

Finally, the entire loss function of RK4-PINN for the SDOF simplified model can be expressed as

$$\mathrm{Loss} = \alpha\,\mathrm{Loss}_{\mathrm{pred}} + \beta\,\mathrm{Loss}_{\mathrm{phy}} + \gamma\,\mathrm{Loss}_{\mathrm{pos}} \tag{29}$$

For this numerical example, the values of the physical parameters are shown in Table 2. The SDOF model response dataset used in this work was generated using the RK4 method. The parameter estimation results of the proposed method were evaluated under three levels of Gaussian noise: 1%, 5%, and 10%. As shown in Fig. 12, the ground motions of El Centro (I-ELC180) during the Imperial Valley earthquake of May 18, 1940, and Chi-Chi (TCU052-W) during the Chichi earthquake of September 20, 1999, were chosen as the input ground motions to form the datasets. The EL Centro record was sampled for 10 s at a frequency of 100 Hz. The peak ground acceleration (PGA) was randomly adjusted between 0.5g and 3g to generate 20 response samples. These 20 response samples were randomly divided into a training set (80%) and a validation set (20%). To distinguish it from the excitation used in the training and validation set, the Chi-Chi record, sampled for 20 s at a frequency of 100 Hz, was used to generate a test set with 20 samples using the same tuning amplitude operation.

The experiment employed a batch size of 32 and was trained for 300 epochs. A learning rate adjustment was implemented throughout the training process, with an initial learning rate of 0.01, a fixed step size of 10, and a decay rate of 0.9. The optimization algorithm

Table 2 Physical parameters for the SDOF simplified model

Physical parameters	Values	Units
m	50	kg
c	100	Ns/m
k	8000	N/m
w	10	–

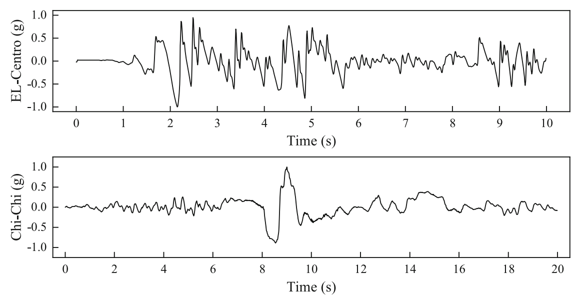

Fig. 12 Seismic excitation curves of EL Centro and Chi-Chi with PGA $= 1g$

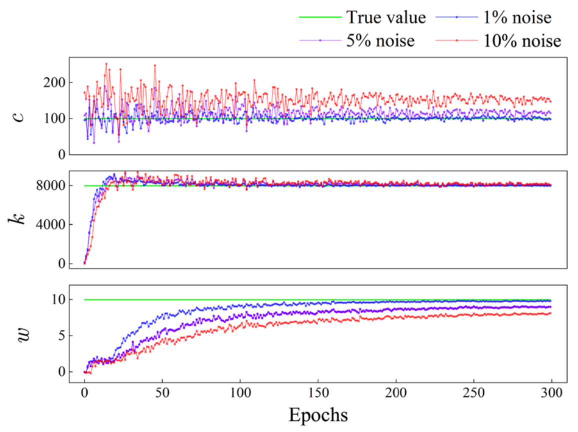

Fig. 13 Estimation process for the parameters of the SDOF simplified model

used in this study is Adam, a commonly used optimization algorithm in deep learning. The weight coefficients were tuned using the grid search method, and it was finally determined that α was set to 1000, β was set to 1, and γ was set to 10. To prevent gradient explosion, a technique called gradient clipping was used in the training process, which limits the magnitude of the gradients to no more than 2 during backpropagation.

Table 3 presents the estimation results of model parameters by RK4-PINN under three different noise levels. The proposed method delivers highly accurate estimation results for parameter k across all three noise levels, with errors consistently remaining below 3%. The accuracy of estimation results for parameter c varies across different noise levels, exhibiting higher accuracy at 1% and 5% noise levels but encountering significant error increases at the 10% noise level. Equation (26) underscores the coupling between parameter k and the nonlinear parameter w. Estimation errors can significantly impact w, particularly in noisy environments. w has the best recognition result at 1% noise level with only 1.99% error, but reaches 19.51% error at 10% noise level. In conclusion, the parameter estimation results of the RK4-PINN method are affected by noise, especially the nonlinear term parameters may exhibit estimation errors in high noise levels. This is due to the fact that the RK4 method, utilizing deterministic integrator, lacks the modeling of uncertain noise. Small noise integrators [41] may help the proposed method extend to a stochastic setting, which has the potential to enhance robustness. The parameter estimation process for the SDOF simplified model during the network training is shown in Fig. 13.

As presented in Table 3, the RKNN method was introduced for comparison with the parameter estimation results of RK4-PINN. When considering the linearly related parameters such as c and k, the estimated mean results obtained by RKNN are close to the actual values. However, the corresponding standard deviations are considerably large, indicating a higher level of uncertainty and less convincing results. In the case of the nonlinear parameters w, RKNN fails to accurately estimate the value, resulting in unreliable and imprecise results. On the other hand, incorporating third-order information in RK4-PINN enables highly accurate parameter estimation with a significantly lower error rate of only 1.99% and a higher confidence level than other methods. Based on the findings, it can be concluded that the parameter estimation capability of RK4-PINN is superior to that of RKNN due to the embedded priori physical information.

Table 3 Parameter estimation for the SDOF simplified model

Method	Noise(%)	c_{meam}	c_{std}	c_{error} (%)	k_{mean}	k_{std}	k_{error} (%)	w_{meam}	w_{std}	w_{error} (%)
RK4-PINN	1	100.62	0.62	0.62	8015.40	23.03	0.19	9.80	0.10	1.99
RK4-PINN	5	116.22	0.94	16.22	8104.56	82.13	1.31	8.96	0.21	10.35
RK4-PINN	10	151.65	2.68	51.65	8177.18	104.77	2.21	8.05	0.25	19.51
RKNN	1	101.73	2.61	1.73	7769.70	11918.36	2.88	7.09	14.87	29.08

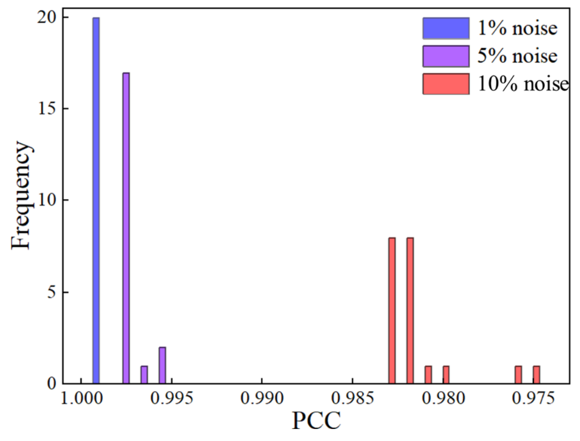

Fig. 14 PCC histogram of the 20 test samples under the three noise levels

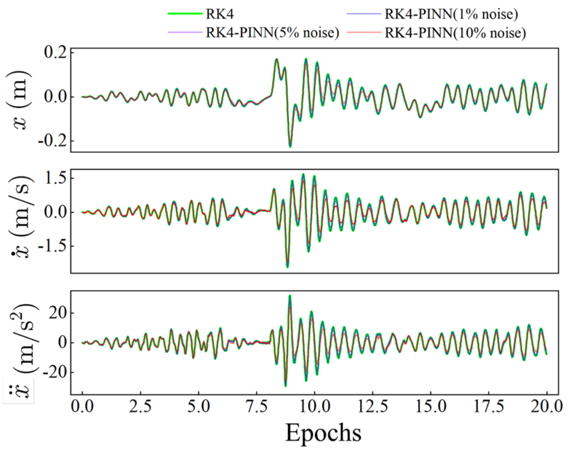

Fig. 15 Response and predictions of test sample with the minimum PPC

The test dataset was generated from Chi-Chi seismic record that was not included in the training dataset. Therefore, using the test dataset allows for a more comprehensive and objective evaluation of RK4-PINN in modeling the SDOF simplified model. The Pearson correlation coefficient (PCC) is a statistical measure to evaluate the relationship between two variables. In this study, the PCC is used to assess the similarity between the prediction of the model response by RK4-PINN and the reference response calculated by the RK4 method, which is calculated as

$$PCC(\boldsymbol{X},\tilde{\boldsymbol{X}})=\frac{\sum_{i=1}^{n}\left(\boldsymbol{X}_i-\overline{\boldsymbol{X}}\right)\left(\tilde{\boldsymbol{X}}_i-\overline{\tilde{\boldsymbol{X}}}\right)}{\sqrt{\sum_{i=1}^{n}\left(\boldsymbol{X}_i-\overline{\boldsymbol{X}}\right)^2}\sqrt{\sum_{i=1}^{n}\left(\tilde{\boldsymbol{X}}_i-\overline{\tilde{\boldsymbol{X}}}\right)^2}} \tag{30}$$

Figure 14 shows the PCCs of all 20 test set samples under three noise levels. It can be seen that the values of PCCs are very close to 1, indicating a strong correlation between the predictions and reference responses. Despite the increasing noise levels, all Pearson correlation coefficients remained high, with values above 0.97. This finding suggests that the RK4-PINN method is highly robust in the ability to model the potential dynamic behavior of the system.

Taking the test sample with a minimum PCC of 0.9741 as an example, the reference response and predictions under three noise levels are plotted in Fig. 15. The curves almost overlap, indicating that the predictions are highly accurate. This finding demonstrates the ability of the RK4-PINN method in accurately predicting responses even under 10% noise level.

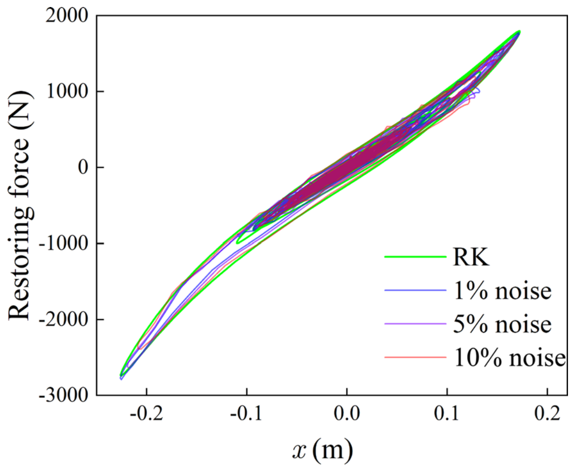

Fig. 16 Hysteresis curves of restoring force versus displacement

To evaluate the capacity of RK4-PINN in predicting the behavior of the nonlinear Duffing spring, hysteresis curves depicting restoring force versus displacement are plotted, as depicted in Fig. 16. For the reference hysteresis curve, the restoring force is calculated as

$$F_\mathrm{r}=c\dot{x}+kx+wkx^3 \tag{31}$$

On the contrary, in prediction tasks, since the parameters c, k, and w need to be estimated, the formula for calculating the restoring force of predictions is as

$$F_\mathrm{r}=-m\ddot{x}-m\ddot{x}_g \tag{32}$$

As shown in Fig. 16, the third-order characteristics of the Duffing spring with a small region of energy dissipation caused by the damping are correctly predicted.

4 Conclusion

This paper proposes a method named RK4-PINN for parameter estimation and modeling of nonlinear dynamical systems, based on the combination of the Runge–Kutta method and PINN. PINN is utilized as the basic integral unit for the RK4 method, allowing for efficient learning of complex dynamic systems. Unlike black-box techniques such as RNN and LSTM, the RK4-PINN method provides a more transparent and interpretable framework for system modeling. The parameters such as stiffness and damping to be estimated are extracted from PINN using automatic differentiation. To ensure that the estimated parameters have a clear physical meaning, time-invariant properties are innovatively added to the loss function as a soft constraint. After RK4-PINN is trained, the system parameters can be obtained automatically. The trained RK4-PINN can also serve as alternative models for the dynamical system, enabling rapid prediction of the response under other excitation.

The RK4-PINN method was tested in two nonlinear case studies: the Lorenz system and a SDOF simplified model with a nonlinear Duffing spring to verify the effectiveness. In each case study, RK4-PINN was trained to learn the corresponding system properties with a small training set of just 20 samples. For the Lorenz system, we analyzed the impact of predicted and physical weight coefficients on parameter estimation results. The results supported the theoretical analysis about the regularization role played by physical soft constraints. Therefore, the predicted weights should prevail over the physical weights when using the RK4-PINN method. Despite the complexity of chaotic systems causing the predictions to differ from the reference, the predicted trajectories from RK4-PINN accurately displayed identical butterfly-shaped Lorenz attractors and similar parameters. For SDOF simplified model, three noise levels of 1%, 5%, and 10% were introduced to evaluate the parameter estimation capability of the proposed method. Although the estimation accuracy of parameters was affected by the noise, the prediction responses were still robust. The hysteresis curves of the restoring force demonstrated that RK4-PINN could effectively learn the dynamic properties of the nonlinear Duffing spring and damping dissipation.

In addition, the RK4-PINN method offers an interpretable framework for parameter estimation and modeling, which can be flexibly integrated with well-researched systems having formulation expressions. Unfortunately, explicitly formulating dynamical systems for practical engineering with ultra-high degrees of freedom is impractical. Thus, future research will focus on methods for embedding physical information in high-dimensional parameter spaces. The availability of complete response data is often restricted to sparse measurements. High-fidelity models can be used for offline training of algorithms, followed by online measurement data assimilation [42] . In further research, finite element models will become an important tool for generating training data. The RK4-PINN method offers a fundamental exploration for further research, with potential applications in civil, aerospace, and mechanical engineering.

Funding This research was supported by grants from the National Natural Science Foundation of China (Grant No. 51978216, 52192664) and Natural Science Foundation of Heilongjiang Province (Grant No. LH2022E119).

Data availability The datasets generated or analyzed during the current study are available on GitHub at https://github.com/VVeida/RK4_PINN or from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Wanner, G., Hairer, E.: Solving Ordinary Differential Equations II, vol. 375. Springer, Berlin Heidelberg New York (1996)
2. Zhang, L., Sun, Y., Wang, A., Zhang, J.: Neural network modeling and dynamic behavior prediction of nonlinear dynamic systems. Nonlinear Dyn. 1–22 (2023)
3. Runge, C.: Über die numerische auflösung von differentialgleichungen. Math. Ann. **46**(2), 167–178 (1895)
4. Kutta, W.: Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Teubner (1901)
5. Dormand, J.R., Prince, P.J.: A family of embedded runge-kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
6. Aniszewska, D.: Multiplicative runge-kutta methods. Nonlinear Dyn. **50**(1–2), 265–272 (2007)
7. Schober, M., Duvenaud, D.K., Hennig, P.: Probabilistic ode solvers with runge-kutta means. Adv. Neural Inf. Process. Syst. **27** (2014)
8. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
9. Weinan, E.: A proposal on machine learning via dynamical systems. Commun. Math. Stat. **1**(5), 1–11 (2017)

10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: bridging deep architectures and numerical differential equations. In: International Conference on Machine Learning, pp. 3276–3285. PMLR (2018)
12. Wang, Y.J., Lin, C.T.: Runge-kutta neural network for identification of dynamical systems in high accuracy. IEEE Trans. Neural Netw. **9**(2), 294–307 (1998)
13. Zhu, M., Chang, B., Fu, C.: Convolutional neural networks combined with runge-kutta methods. Neural Comput. Appl. **35**(2), 1629–1643 (2023)
14. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Adv. Neural Inf. Process. Syst. **31** (2018)
15. Goyal, P., Benner, P.: Discovery of nonlinear dynamical systems using a runge-kutta inspired dictionary-based sparse regression approach. Proc. R. Soc. A **478**(2262), 20210883 (2022)
16. Foroutannia, A., Ghasemi, M.: Predicting cortical oscillations with bidirectional lstm network: a simulation study. Nonlinear Dyn. **111**(9), 8713–8736 (2023)
17. Liu, H., Zhao, C., Huang, X., Yao, G.: Data-driven modeling for the dynamic behavior of nonlinear vibratory systems. Nonlinear Dyn. 1–26 (2023)
18. Tan, Y., Hu, C., Zhang, K., Zheng, K., Davis, E.A., Park, J.S.: Lstm-based anomaly detection for non-linear dynamical system. IEEE Access **8**, 103301–103308 (2020)
19. Salmela, L., Tsipinakis, N., Foi, A., Billet, C., Dudley, J.M., Genty, G.: Predicting ultrafast nonlinear dynamics in fibre optics with a recurrent neural network. Nat. Mach. Intell. **3**(4), 344–354 (2021)
20. Li, S., Yang, Y.: A recurrent neural network framework with an adaptive training strategy for long-time predictive modeling of nonlinear dynamical systems. J. Sound Vib. **506**, 116167 (2021)
21. Alemany, S., Beltran, J., Perez, A., Ganzfried, S.: Predicting hurricane trajectories using a recurrent neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 468–475 (2019)
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
23. Kim, Y.H., Lewis, F.L., Abdallah, C.T.: A dynamic recurrent neural-network-based adaptive observer for a class of nonlinear systems. Automatica **33**(8), 1539–1543 (1997)
24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
25. Chen, R., Jin, X., Laima, S., Huang, Y., Li, H.: Intelligent modeling of nonlinear dynamical systems by machine learning. Int. J. Non-Linear Mech. **142**, 103984 (2022)
26. Wang, R., Kalnay, E., Balachandran, B.: Neural machine-based forecasting of chaotic dynamics. Nonlinear Dyn. **98**(4), 2903–2917 (2019)
27. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
28. Jazwinski, A.H.: Stochastic processes and filtering theory. Courier Corporation (2007)
29. Wan, E.A., Van Der Merwe, R.: The unscented kalman filter for nonlinear estimation. In: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373), pp. 153–158. IEEE (2000)
30. Bisaillon, P., Sandhu, R., Khalil, M., Pettit, C., Poirel, D., Sarkar, A.: Bayesian parameter estimation and model selection for strongly nonlinear dynamical systems. Nonlinear Dyn. **82**, 1061–1080 (2015)
31. Lindley, D.: The use of prior probability distributions in statistical inference and decisions. In: Proc. 4th Berkeley Symp. on Math. Stat. and Prob, pp. 453–468 (1960)
32. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
33. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021)
34. Raissi, M.: Deep hidden physics models: deep learning of nonlinear partial differential equations. J. Mach. Learn. Res. **19**(1), 932–955 (2018)
35. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
36. Zhang, R., Liu, Y., Sun, H.: Physics-informed multi-lstm networks for metamodeling of nonlinear structures. Comput. Methods Appl. Mech. Eng. **369**, 113226 (2020)
37. Li, S., Yang, Y.: Data-driven identification of nonlinear normal modes via physics-integrated deep learning. Nonlinear Dyn. **106**, 3231–3246 (2021)
38. Agarwal, V., Wang, R., Balachandran, B.: Data driven forecasting of aperiodic motions of non-autonomous systems. Chaos Interdiscip. J. Nonlinear Sci. **31**(2) (2021)
39. Demir-Kavuk, O., Kamada, M., Akutsu, T., Knapp, E.W.: Prediction using step-wise l1, l2 regularization and feature selection for small data sets with large number of features. BMC Bioinform. **12**, 1–10 (2011)
40. Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963)
41. Breunung, Thomas, Balachandran, Balakumar: Computationally efficient simulations of stochastically perturbed nonlinear dynamical systems. J. Comput. Nonlinear Dyn. **17**(9), 091008 (2022)
42. Zhao, Xiangxue, Azarm, Shapour, Balachandran, Balakumar: Online data-driven prediction of spatio-temporal system behavior using high-fidelity simulations and sparse sensor measurements. J. Mech. Des. **143**(2), 021701 (2021)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.