INFORMATICA, 2026, Vol. 37, No. 1, 1–24

DOI: https://doi.org/10.15388/25-INFOR607

Novel Loss Function Construction for Neural Network-Based Prediction of Complex High-Dimensional Nonlinear Dynamical Systems

Kun AN[1], Ying SUN[1,*], Minghui YAO[2], Junhua ZHANG[3]

[1] *School of Applied Science, Beijing Information Science and Technology University, Beijing, China*
[2] *School of Aeronautics and Astronautics, Tiangong University, Tianjin, China*
[3] *College of Mechanical Engineering, Beijing Information Science and Technology University, Beijing, China*
e-mail: ankun1906@163.com, sunying0000@126.com, merry_mingming@163.com, hua@bistu.edu.cn

Received: May 2025; accepted: October 2025

Abstract. Traditional loss functions such as mean squared error (MSE) are widely employed, but they often struggle to capture the dynamic characteristics of high-dimensional nonlinear systems. To address this issue, we propose an improved loss function that integrates linear multistep methods, system-consistency constraints, and prediction-phase error control. This construction simultaneously improves training accuracy and long-term stability. Furthermore, the introduction of recursive loss and interpolation strategies brings the model closer to practical prediction scenarios, broadening its applicability. Numerical simulations demonstrate that this construction significantly outperforms both mean square error and existing custom loss functions in terms of performance.

Key words: custom loss function, neural networks, nonlinear dynamical systems, time series prediction.

1. Introduction

Chaotic dynamics are ubiquitous in natural and engineered systems, ranging from weather and climate processes to electronic circuits and structural mechanics. Unlike regular periodic motions, chaotic time series are characterized by sensitive dependence on initial conditions, nonlinearity, and high dimensionality, which make their long-term prediction extremely challenging. Traditional approaches based on analytical modelling or numerical simulation often become intractable when the governing equations are highly nonlinear or the system complexity increases (Zhang *et al.*, 2019). In this context, data-driven methods, particularly neural networks (Sestanovic and Kalinic Milicevic, 2025; Calvo-Rolle *et al.*, 2014), have emerged as effective alternatives for capturing the underlying dynamics and predicting the evolution of chaotic systems.

*Corresponding author:sunying0000@126.com

The success of neural networks in chaotic time series prediction (Fernandes *et al.*, 2020), however, is not determined solely by network architecture; the choice of loss function plays an equally crucial role. A well-designed loss function not only guides the optimization process but also influences the model's ability to capture long-term dynamics rather than merely short-term patterns. Over time, the development of loss functions has become a central factor in improving both predictive accuracy and generalization performance across different applications. Early loss functions mainly include Mean Square Error (MSE), Mean Absolute Error (MAE), and cross-entropy, the latter rooted Information Theory of Shannon (1948). To address robustness issues, Huber proposed a hybrid of MSE and MAE (Huber, 1964). Chopra *et al.* (2005) proposed the contrastive loss to minimize distances between similar samples and maximize those between dissimilar ones, on which the triplet loss (Hadsell *et al.*, 2006) introduced additional constraints through triplet comparison. Customized objectives have also been developed: BLEU-based loss for machine translation (Papineni *et al.*, 2002; Ranzato *et al.*, 2015), BPR loss for recommender systems (Rendle *et al.*, 2009), and focal loss for class-imbalance in object detection (Lin *et al.*, 2017). A principled deep learning approach was proposed for multi-task learning, which weighs multiple loss functions by considering homoskedastic uncertainty (Kendall *et al.*, 2018). In physics-related domains, auxiliary loss functions have been integrated into neural networks to incorporate physical dynamics. Examples include PIDynNet (Liu and Meidani, 2023), a physics-informed neural network for structural systems, and a two-coefficient loss function based on linear multistep methods (Zhang *et al.*, 2023). In addition, Ghazvini *et al.* (2024) combine MSE and LogCosh to mitigate vanishing gradients, while a weight-constrained RNN optimization (Wu *et al.*, 2020) incorporates parameter constraints and regularization to ensure the input-output relationship.

The combination of numerical algorithms and neural networks has advanced deep learning for differential equations. The Runge-Kutta Convolutional Neural Network (RKCNN) builds network models using higher-order Runge-Kutta methods (Zhu *et al.*, 2022). PINNs have been integrated with Runge-Kutta schemes for parameter estimation and dynamic modelling (Zhai *et al.*, 2023; He *et al.*, 2023). Connections between the implicit Euler method and neural networks led to the Adaptive Implicit Network (AIM-NET), allowing flexible convergence for improved parameter estimation (Yuan *et al.*, 2020). Implicit Adams Predictor-Corrector Blocks (IABs) combined with Non-local Sparse Attention (NSA) and Attention Feature Fusion (AFF) form the Implicit Adams Predictor-Corrector Module (IAM) (Yin *et al.*, 2023), while discrete ZNN (DZNN) models of Adams-Bashforth type enhance accuracy for time-varying problems (Yang *et al.*, 2020). A linear multi-step framework based on the implicit Adams-Moulton scheme approximates full-order models in low-dimensional space (Xie *et al.*, 2019). Finally, MultiPINN, a multi-head neural network enriched with PINNs, incorporates RBF interpolation and embeds physics-based priors from differential equations and boundary conditions (Li, 2024). These methods improve efficiency and performance by integrating numerical schemes with physics-informed loss functions.

Building on the integration of numerical algorithms and neural networks, interpolation techniques have become important in deep learning for data augmentation, feature

extraction, and regularization, improving accuracy and robustness. Conditional Encoder-Decoder GANs (CEDGANs) treat interpolation as a conditional generation task, capturing spatial relationships (Zhu *et al.*, 2019). Interpolated Adversarial Training enhances adversarial robustness while preserving generalization (Lamb *et al.*, 2019). D'Ambrosio *et al.* (2021) combine PINNs with a functional interpolation technique, the Theory of Functional Connections (TFC), to learn optimal controls. Interpolation Consistency Training (ICT) ensures predictions for interpolated points align with interpolated predictions, reducing overfitting (Verma *et al.*, 2022). RAKI performs nonlinear interpolation of missing k-space lines, improving noise resilience in MRI reconstruction (Akçakaya *et al.*, 2019), while a CNN-based interpolation method addresses missing projection data in sparse-view CT (Lee *et al.*, 2017).

Previous studies on loss functions remain inadequate for modelling and predicting complex, high-dimensional, nonlinear dynamical systems. Traditional loss functions, such as MSE, are widely adopted for their simplicity, but they often overlook critical dynamic features in long-term predictions. Although several custom loss functions have been proposed, they generally suffer from limited accuracy and poor stability, which restrict their applicability to a broader class of dynamic systems. To address these challenges, this paper introduces an error term in the prediction phase, thereby proposing an improved custom loss function. The improved version further integrates interpolation and recursive strategies to better capture temporal dependencies and enhance long-term prediction accuracy. This paper selects the dynamic modelling of a ring truss antenna as a numerical case study, representing a typical chaotic system within aerospace structural dynamics. Numerical simulations demonstrate that the improved loss function consistently outperforms both the traditional MSE and the original custom loss function in terms of accuracy and stability. This study presents a systematic approach to designing advanced loss functions and demonstrates its effectiveness in predicting complex, high-dimensional nonlinear dynamical systems.

2. Constructing Loss Function

2.1. *Mean Square Error*

MSE is a common metric for assessing the predictive performance of a model (Zhou *et al.*, 2021; Rumelhart *et al.*, 1986). It measures the average squared difference between the predicted and actual values. The smaller the Mean Squared Error, the closer the model's predictions are to the actual values and the better the model performance. The formula for the mean square error is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \tag{1}$$

where n is the number of samples, y_i is the actual value of the i-th sample, $\hat{y}_i$ is the predicted value of the i-th sample.

When using MSE as a loss function for training a neural network, the aim is to predict the state vector of the next moment using the state vector of the previous moment. Therefore, MSE is mainly used to calculate the difference between the predicted state vector of the next moment and the numerically solved state vector of the next moment. The calculation formula for the MSE used as the loss function in neural networks is as follows:

$$L_1 = \frac{1}{N-k} \sum_{i=0}^{N-k} \left| X(t_{i+k}) - Y(t_i) \right|^2, \tag{2}$$

where $X(t_i)$ is the approximate solution at time t_i, $Y(t_i) = \tilde{f}(X(t_i))$ is the output of the neural network at time t_i ($\tilde{f}$ represents the trained neural network), k is the prediction step (i.e. use the i-th point to predict the $(i+k)$-th point). Considering the prediction accuracy, k is taken as 1 in this paper.

2.2. *Custom Loss Function*

From previous work (Sun *et al.*, 2024), we already know that the linear multistep method is applicable to nonlinear dynamical systems. The general form of the linear multistep method can be expressed as:

$$X(t_{i+1}) = \sum_{p=0}^{k-1} \alpha_p X(t_{i-p}) + h \sum_{q=-1}^{k-1} \beta_q \dot{X}(t_{i-q}), \tag{3}$$

here, $X(t_i)$ is the approximate solution at time t_i, h is the step size, α and β are constants, and k-1 is the number of steps. This formula indicates that the current point $X(t_i)$ is computed as a linear combination of the previous k-1 points and a linear combination of the derivative values at these points.

During the training process of a neural network, all time step information is available because the entire batch of training data is known. Therefore, we use the Adams-Moulton method in the implicit linear multistep method. The Adams-Moulton method requires information about the current time step, but is more numerically stable than the Adams-Bashforth method in the display linear multistep method. The first part of the loss function can then be written as $\varepsilon_{1i} = \sum_{p=0}^{k-1} \alpha_p X(t_{i-p}) + h \sum_{q=-1}^{k-1} \beta_q [Y(t_{i-q}) + \bar{f}(t_{i-q})] - X(t_{i+1})$. Previous work has utilized both the one-step form (improved Euler method) and the two-step form (Simpson's method) of the linear multistep method to construct the loss function for neural networks. For the sake of convenience in computing the loss function, this paper adopts the Second-order linear multistep method.

A typical dynamical system we study can be written as:

$$\frac{dX(t)}{dt} = f\big(X(t)\big) + \bar{f}(t), \tag{4}$$

where $X(t) = [\boldsymbol{x}^T(t)\ \boldsymbol{v}^T(t)]^T \in \mathbb{R}^{2n}$ is the state vector of the system, $f(X(t))$ includes the generalized restoring force, and $\bar{f}(t)$ contains the external excitation force. Based on

the displacement and velocity, it is divided into the first n dimensions and the last n dimensions. The output of the neural network is denoted as $Y = \tilde{f}(X_t)$, which is also divided into Y_1 and Y_2. Since $X(t) = [\boldsymbol{x}^T(t)\ \boldsymbol{v}^T(t)]^T \in \mathbb{R}^{2n}$ is divided into $X_1(t)$ and $X_2(t)$, we can define the second term in the loss function L_2, denoted as $\varepsilon_{2j} = Y_1(t_j) - X_2(t_j)$. The second term ε_{2j} enforces the inherent consistency between displacement and velocity (i.e. the derivative of displacement must equal the velocity component). More generally, this can be interpreted as a system-specific consistency constraint, ensuring that the neural network respects the intrinsic structure of the dynamical system.The loss function is given by Eq. (5):

$$L_2 = \frac{\gamma_1}{N-k+1} \sum_{i=k-1}^{N-1} |\varepsilon_{1i}|^2 + \frac{\gamma_2}{N+1} \sum_{j=0}^{N} |\varepsilon_{2j}|^2, \tag{5}$$

where γ_1 and γ_2 are weight coefficients.

Although L_2 successfully embeds system dynamics into the loss function through the implicit linear multistep method term and system-consistency term, it does not explicitly constrain the errors that arise during the iterative prediction phase. As a result, long-term predicting may still suffer from error accumulation. This limitation motivates the improved framework introduced in Section 2.3.

2.3. *Improved Custom Loss Function*

In Section 2.2, the loss function was designed to constrain only the training residuals. Although this approach improved short-term accuracy, it did not address the error during iterative prediction. To overcome this limitation, we propose an improved framework that simultaneously penalizes both training residuals and prediction-phase discrepancies, thereby enhancing long-term stability.

We next turn our attention to the iterative prediction process, which is fundamentally different from training. During prediction, the true state at the current time step is unknown, so the network must rely on its own previous outputs to generate future states. This inherently necessitates the use of an explicit linear multistep method, as the current state cannot be incorporated implicitly. We denote the abstract form of the explicit linear multistep formula in the prediction process as:

$$X(t_{m+1}) = \sum_{p=0}^{k-1} \alpha_p X(t_{m-p}) + h \sum_{q=0}^{k-1} \beta_q \left[Y(t_{m-q}) + \bar{f}(t_{m-q}) \right], \tag{6}$$

here, $X(t_{m+1})$ is the approximate solution at time t_{m+1}, h is the step size, α and β are constants, and k-1 is the number of steps.

Eq. (6) presents the iterative formula employed during the prediction phase, from which we derive a novel loss term: $\varepsilon_{3m} = \sum_{p=0}^{k-1} \alpha_p X(t_{m-p}) + h \sum_{q=0}^{k-1} \beta_q [Y(t_{m-q}) + \bar{f}(t_{m-q})] - X(t_{m+1})$. This term primarily constrains errors generated during iterative prediction. Incorporating it into the loss function serves a dual purpose: firstly, it explicitly

penalises deviations arising when the model must rely on its own outputs rather than true states during rolling predictions, thereby mitigating long-term error accumulation. Secondly, as the prediction phase relies on explicit numerical methods (which inherently introduce truncation errors), the new term embeds these numerical consistency requirements into the training objective. Unlike the model in Sun *et al.* (2024), which focused solely on training residuals, the proposed framework introduces a dual-constraint mechanism that integrates both training accuracy and prediction-phase error control, thereby mitigating error accumulation in long-term predictions. Following the previous notation, the complete form of the improved loss function can be expressed as:

$$L_3 = \frac{\gamma_1}{N-k+1}\sum_{i=k-1}^{N-1}|\varepsilon_{1i}|^2 + \frac{\gamma_2}{N+1}\sum_{j=0}^{N}|\varepsilon_{2j}|^2 + \frac{\gamma_3}{N-k+1}\sum_{m=k-1}^{N-1}|\varepsilon_{3m}|^2, \quad (7)$$

where

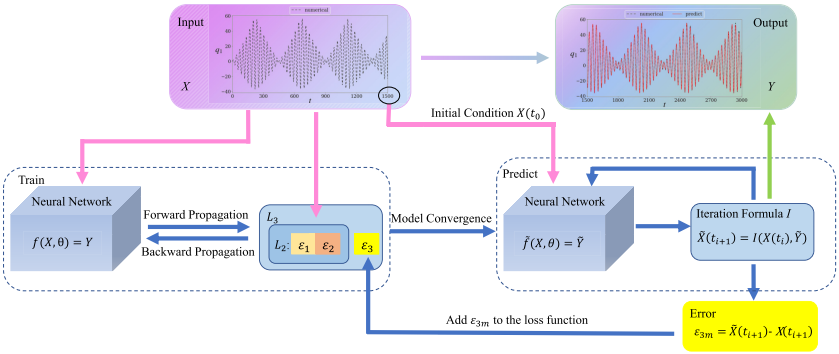
$$\begin{cases} \varepsilon_{1i} = \sum_{p=0}^{k-1}\alpha_p X(t_{i-p}) + h\sum_{q=-1}^{k-1}\beta_q[Y(t_{i-q}) + \bar{f}(t_{i-q})] - X(t_{i+1}), \\ \varepsilon_{2j} = Y_1(t_j) - X_2(t_j), \\ \varepsilon_{3m} = \sum_{p=0}^{k-1}\alpha_p X(t_{m-p}) + h\sum_{q=0}^{k-1}\beta_q[Y(t_{m-q}) + \bar{f}(t_{m-q})] - X(t_{m+1}). \end{cases} \quad (8)$$

γ_1, γ_2 and γ_3 are weight coefficients. In the numerical simulations conducted in this paper, the specific formula employed for Eq. (8) is:

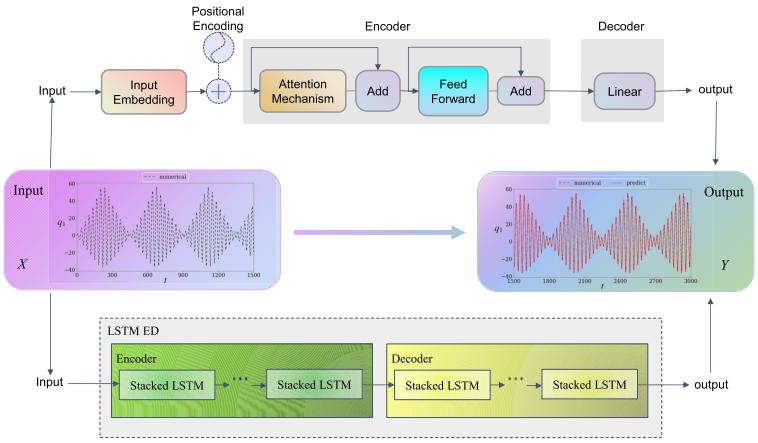
$$\begin{cases} \varepsilon_{1i} = X(t_i) + \frac{h}{2}[Y(t_{i+1}) + \bar{f}(t_{i+1}) + Y(t_i) + \bar{f}(t_i)] - X(t_{i+1}), \\ \varepsilon_{2j} = Y_1(t_j) - X_2(t_j), \\ \varepsilon_{3m} = X(t_m) + h[Y(t_m) + \bar{f}(t_m)] - X(t_{m+1}). \end{cases} \quad (9)$$

It is worth highlighting that the improved custom loss function is constructed independently of specific iterative methods, which ensures its generality and applicability across various prediction methods. This construction strategy incorporates the prediction-phase error into the loss function. Consequently, it can be applied to a wide range of data-driven models that rely on numerical methods during the prediction phase, without being restricted to a specific iterative method.

The workflow of the neural network and the composition of the loss function L_3 are illustrated in Fig. 1. Figure 1a illustrates the composition of the loss function, and Fig. 1b shows the underlying model of the two neural networks used in this paper. Firstly, the data X is input into the neural network f, and its parameters θ are optimized to minimize the loss function. Upon completing the training, the trained neural network $\tilde{f}$ is obtained. Subsequently, the last data point from the training set is fed into $\tilde{f}$ as the initial value, and the network outputs the corresponding prediction. The next moment's data is then iteratively obtained using formula I. Through continuous iterations, the final prediction is produced. In this process, the loss function L_2 consists of ε_{1i} and ε_{2j}. Additionally, the error ε_{3m} from



(a) Loss function L_3



(b) Neural networks

Fig. 1. Loss functions L_3 and neural networks.

the iterative process during the prediction phase is introduced as a new term to L_2, forming the final loss function L_3. Figure 1b illustrates the neural network architectures employed for time-series prediction, including a Transformer-based encoder-decoder model (top) and an LSTM-based encoder-decoder model (bottom). Both models take sequential input data X, representing the system's dynamic evolution, and aim to predict future states Y. The Transformer architecture incorporates input embedding, positional encoding, multi-head attention mechanisms, and feedforward layers to capture long-range dependencies. In contrast, the LSTM architecture relies on stacked LSTM layers in both the encoder and decoder to model temporal correlations. The output results are compared with numerical solutions to assess prediction accuracy.

3. Neural Networks Setting

3.1. *Neural Network*

In this study, we consider three representative neural network models for time series prediction of nonlinear dynamical systems: the LSTM_ED model proposed in Sun *et al.* (2024), the S-Transformer, and the Conv-AT. The S-Transformer (Simplified Transformer) is a lightweight variant of the standard Transformer, designed for efficient time series prediction. Unlike the original Transformer, it employs a simplified decoder composed of linear layers, since iterative prediction in this study does not require autoregressive sequence generation. Despite its streamlined architecture, the S-Transformer retains the advantages of multi-head attention mechanisms, such as capturing long-range dependencies and supporting parallel computation. The Conv-AT (Convolutional Attention Transformer) is another variant of the Transformer architecture, tailored to balance local and global feature extraction. Before entering the attention module, the input sequence is processed by a convolutional layer to capture local dependencies and enrich the query representation. The original sequence, after linear transformation, serves as the key and value, allowing the attention mechanism to model relationships between convolution-enhanced local features and the original global representations. This design enables Conv-AT to simultaneously capture fine-grained local structures and long-range dependencies. In this paper, these three neural network models are used solely as validation platforms to test the effectiveness of the proposed improved loss function. To ensure a fair comparison, all numerical simulations are conducted under identical hyperparameter settings.

3.2. *Network Setting*

The numerical example used in this paper is a beam-ring structure with two degrees of freedom. Based on the framework design, we built three neural networks. The first model is used to predict the linear terms, and the next two are used to predict the nonlinear terms. The parameters of the neural network models using the custom loss function and the improved custom loss function are the same. For different beam-ring structures, different strategies are adopted in setting parameters such as learning rate and number of training times, and we have taken the optimal combination of parameters in the simulation after several numerical simulations. However, to ensure a rigorous comparison of loss functions within the same beam-ring structure, we strictly maintained the same parameter settings throughout.

4. Numerical Simulation

4.1. *Beam-Ring Construction*

In this section, we consider a nonlinear dynamic system that originates from the beam-ring equivalent model of a circular truss antenna subjected to periodic thermal excitation

(Zhang *et al.*, 2019). This system can be proposed to model the truss antenna after it is deployed, which is characterized by light weight, high flexibility, and low stiffness. Under periodic thermal loads, such structures can exhibit complex responses including large deformations, resonances, and even chaotic vibrations, making the model an ideal benchmark for testing prediction frameworks. By applying the improved loss function to this system, we are able to evaluate its effectiveness in handling error accumulation and long-term prediction in nonlinear and chaotic dynamics. It should be emphasised that this antenna model serves merely as a representative case study, and the improved loss function framework is generalisable to a broad class of nonlinear dynamical systems. The following nonlinear dynamics system for the beam-ring equivalent model of a loop truss antenna is considered:

$$\begin{aligned}&\ddot{q}_1 + \varepsilon\mu_1\dot{q}_1 + \omega_1^2 q_1 + \varepsilon^2(f_{11}q_1 + f_{12}q_2)\cos\Omega t + \varepsilon\alpha_1 q_1^2 + \varepsilon\alpha_2 q_2^2 + \varepsilon\alpha_3 q_1 q_2 \\ &\quad + \varepsilon\alpha_4\dot{q}_1\dot{q}_2 + \varepsilon(\alpha_5 q_1 + \alpha_6 q_2)\ddot{q}_1 + \varepsilon(\alpha_7 q_1 + \alpha_8 q_2)\ddot{q}_2 = \varepsilon f_1\cos\Omega t, \qquad (10a)\\ &\ddot{q}_2 + \varepsilon\mu_2\dot{q}_2 + \omega_2^2 q_2 + \varepsilon^2(f_{21}q_1 + f_{22}q_2)\cos\Omega t + \varepsilon\beta_1 q_1^2 + \varepsilon\beta_2 q_2^2 + \varepsilon\beta_3 q_1 q_2 \\ &\quad + \varepsilon\beta_4\dot{q}_1\dot{q}_2 + \varepsilon(\beta_5 q_1 + \beta_6 q_2)\ddot{q}_1 + \varepsilon(\beta_7 q_1 + \beta_8 q_2)\ddot{q}_2 = \varepsilon f_2\cos\Omega t, \qquad (10b)\end{aligned}$$

where q_1 and q_2 represent dimensionless displacements, f_1 and f_2 are parameters of the external excitation, f_{11}, f_{12}, f_{21} and f_{22} are parameters of the parametric excitation.

Transform Eqs. (10a) and (10b) into the following state equations:

$$\begin{aligned}&\dot{q}_1 = p_1, \qquad (11a)\\ &\dot{q}_2 = p_2, \qquad (11b)\\ &\dot{p}_1 = \frac{\varepsilon f_1\cos\Omega t - E_1 - \varepsilon(\alpha_7 q_1 + \alpha_8 q_2)\frac{\varepsilon f_2\cos\Omega t - E_2}{1+\varepsilon(\beta_7 q_1 + \beta_8 q_2)}}{1 + \varepsilon(\alpha_5 q_1 + \alpha_6 q_2) - \frac{\varepsilon^2(\alpha_7 q_1 + \alpha_8 q_2)(\beta_5 q_1 + \beta_6 q_2)}{1+\varepsilon(\beta_7 q_1 + \beta_8 q_2)}}, \qquad (11c)\\ &\dot{p}_2 = \frac{\varepsilon f_2\cos\Omega t - E_2 - \varepsilon(\beta_5 q_1 + \beta_6 q_2)\frac{\varepsilon f_1\cos\Omega t - E_1}{1+\varepsilon(\alpha_5 q_1 + \alpha_6 q_2)}}{1 + \varepsilon(\beta_7 q_1 + \beta_8 q_2) - \frac{\varepsilon^2(\alpha_7 q_1 + \alpha_8 q_2)(\beta_5 q_1 + \beta_6 q_2)}{1+\varepsilon(\alpha_5 q_1 + \alpha_6 q_2)}}. \qquad (11d)\end{aligned}$$

where

$$\begin{aligned}E_1 &= \varepsilon\mu_1 p_1 + \omega_1^2 q_1 + \varepsilon^2(f_{11}q_1 + f_{12}q_2)\cos\Omega t + \varepsilon\alpha_1 q_1^2 + \varepsilon\alpha_2 q_2^2 \\ &\quad + \varepsilon\alpha_3 q_1 q_2 + \varepsilon\alpha_4 p_1 p_2, \qquad (12a)\\ E_2 &= \varepsilon\mu_2 p_2 + \omega_2^2 q_2 + \varepsilon^2(f_{21}q_1 + f_{22}q_2)\cos\Omega t + \varepsilon\beta_1 q_1^2 + \varepsilon\beta_2 q_2^2 \\ &\quad + \varepsilon\beta_3 q_1 q_2 + \varepsilon\beta_4 p_1 p_2. \qquad (12b)\end{aligned}$$

Extract an external excitation term from Eq. (11c) and (11d) to generate a generalized form of Eq. (4), which is suitable for our neural network approach:

$$\dot{q}_1 = p_1, \tag{13a}$$

$$\dot{q}_2 = p_2, \tag{13b}$$

$$\dot{p}_1 = \left[\frac{\varepsilon f_1 \cos\Omega t - E_1 - \varepsilon(\alpha_7 q_1 + \alpha_8 q_2)\frac{\varepsilon f_2 \cos\Omega t - E_2}{1+\varepsilon(\beta_7 q_1 + \beta_8 q_2)}}{1 + \varepsilon(\alpha_5 q_1 + \alpha_6 q_2) - \frac{\varepsilon^2(\alpha_7 q_1 + \alpha_8 q_2)(\beta_5 q_1 + \beta_6 q_2)}{1+\varepsilon(\beta_7 q_1 + \beta_8 q_2)}} - \varepsilon f_1 \cos\Omega t\right] + \varepsilon f_1 \cos\Omega t, \tag{13c}$$

$$\dot{p}_2 = \left[\frac{\varepsilon f_2 \cos\Omega t - E_2 - \varepsilon(\beta_5 q_1 + \beta_6 q_2)\frac{\varepsilon f_1 \cos\Omega t - E_1}{1+\varepsilon(\alpha_5 q_1 + \alpha_6 q_2)}}{1 + \varepsilon(\beta_7 q_1 + \beta_8 q_2) - \frac{\varepsilon^2(\alpha_7 q_1 + \alpha_8 q_2)(\beta_5 q_1 + \beta_6 q_2)}{1+\varepsilon(\alpha_5 q_1 + \alpha_6 q_2)}} - \varepsilon f_2 \cos\Omega t\right] + \varepsilon f_2 \cos\Omega t. \tag{13d}$$

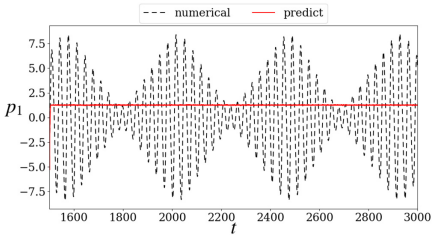
The chaotic time series can be obtained by setting the initial values of the state vectors and the parameters in Eq. (13) as follows:

$$\begin{aligned}
&q_{10} = 0.0414, && q_{20} = -0.0699, && p_{10} = 0.0288, && p_{20} = -0.0304,\\
&\mu_1 = 0.0812, && \mu_2 = 0.0789, && \alpha_1 = -0.466, && \alpha_2 = -0.495,\\
&\alpha_3 = 0.469, && \alpha_4 = -0.4, && \alpha_5 = -0.759, && \alpha_6 = -0.124,\\
&\alpha_7 = 0.12, && \alpha_8 = 0.742, && \beta_1 = -0.0875, && \beta_2 = 0.0466,\\
&\beta_3 = -0.0634, && \beta_4 = -0.0769, && \beta_5 = -0.058, && \beta_6 = 0.01,\\
&\beta_7 = 0.0217, && \beta_8 = -0.0302, && f_{11} = 10, && f_{12} = 10,\\
&f_{21} = 10, && f_{22} = 10, && f_1 = 83, && f_2 = 15,\\
&\varepsilon = 0.001, && \omega_1 = 0.2, && \omega_2 = 2\omega_1 + 0.319\varepsilon, && \Omega = \omega_1 - 0.001\varepsilon.
\end{aligned}$$

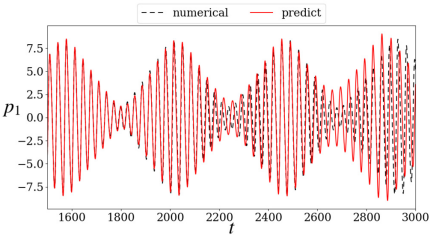
The dataset is a time series generated by solving Eq. (13) with the ordinary differential equation (ODE) solver. The sampling time is 0–1500s, the time step is 0.0125, so the total number of training data is 4.8×10^5. The predicted time range is 1500–3000s, the time step is 0.0125 and the total number of predicted data is 4.8×10^5. This is due to the fact that we add the error generated by the iterative prediction method to the loss function as well, and therefore need to ensure the consistency of the time step to get a good prediction.

4.2. *Results and Discussion*

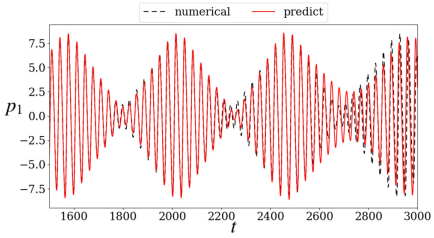
After identifying the equations, neural networks and loss functions, the focus of this section shifts to a comparative analysis of the predictive performance of the three selected loss functions. In this study, we employed three different neural networks to perform numerical simulations for the beam-ring structure, and evaluated the predictive performance using three different loss functions. Due to space limitations, we only display the prediction results on plane (t, p_1) only. While the other images are not shown here, they consistently support the conclusions drawn in this paper. Figures 2–4 illustrate the results of three neural networks' predictions for the four planes of the beam-ring structure. We plot the significance of each subplot of the image as follows: (a) Prediction results using the loss



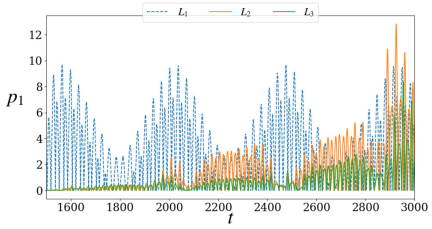
(a) Loss function L_1



(b) Loss function L_2

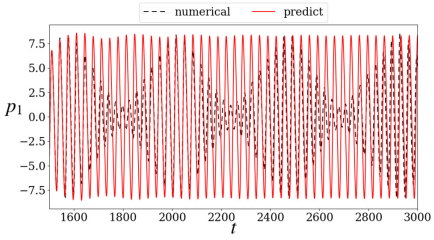


(c) Loss function L_3

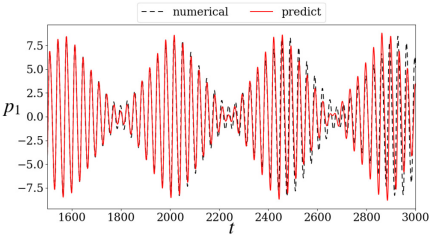


(d) Comparison of point-wise errors

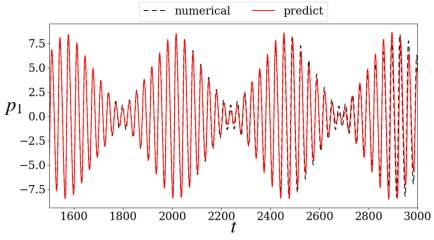
Fig. 2. Prediction performance of LSTM_ED for the plane (t, p_1) using three different loss functions, and error comparison.



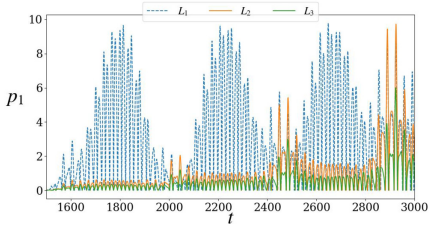
(a) Loss function L_1



(b) Loss function L_2

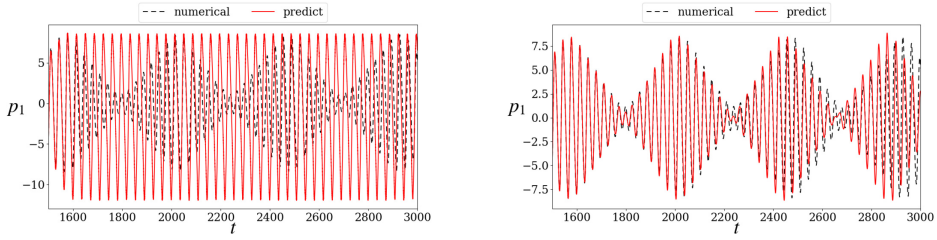


(c) Loss function L_3



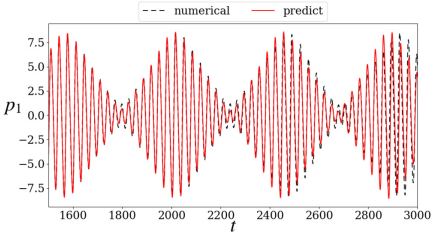
(d) Comparison of point-wise errors

Fig. 3. Prediction performance of S-Transformer for the plane (t, p_1) using three different loss functions, and error comparison.

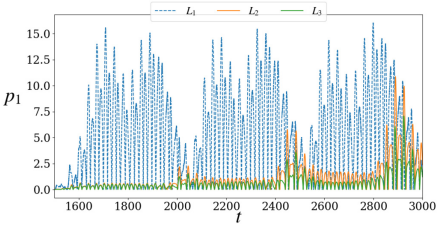


(a) Loss function L_1

(b) Loss function L_2



(c) Loss function L_3



(d) Comparison of point-wise errors

Fig. 4. Prediction performance of Conv-AT for the plane (t, p_1) using three different loss functions, and error comparison.

function L_1, (b) Prediction results using the loss function L_2, (c) Prediction results using the loss function L_3, and (d) Comparison of point-wise errors for different loss functions.

When we use L_1 as the loss function for training, the aim is to predict the state vector of the next moment using the state vector of the previous moment. As can be seen in Fig. 2, the prediction result of LSTM_ED is almost a straight line, which means that constant iterations of the prediction results will eventually converge to the same value. This is due to the model's inability to capture the complex relationship between the moments before and after the data and the improper choice of the loss function, which prevents the model from being optimized correctly. Similarly, when other models are trained using L_1 as the loss function, the displacement time series curves of S-Transformer and Conv-AT exhibit regular oscillations at high frequencies, which do not match the numerical solutions solved by the ode solver. In summary, none of the three neural networks can effectively predict Eq. (13) when using the MSE loss function.

For the loss functions L_2 and L_3 based on the linear multistep method, the three neural networks can roughly predict the displacement trend. Specifically, on the plane (t, p_1), the neural network using the original loss function L_2 can predict the approximate trend of the displacement time series data, but there is still a considerable error in the details. As can be seen from the figure, it starts to be noticeable around 2200 seconds and the error starts to gradually increase after 2500 seconds. The neural network with the improved loss function L_3 can not only predict the overall trend of the displacement time series, but also control the detail error within a small range. This shows that the improved loss function L_3 has a great advantage in prediction accuracy. The error plots in Figs. 3d–4d also show that the error of the loss function L_3 is relatively small and does not increase further.

Table 1
Comparison of errors for loss functions L_2 and L_3.

Neural networks	Comparison	Planes			
		(t, q_1)	(t, q_2)	(t, p_1)	(t, p_2)
LSTM_ED	L_2	161.9232	4.7384	5.7993	0.7185
	L_3	46.1518	2.9058	1.5152	0.4531
	%	**71.50%**	**38.68%**	**73.87%**	**36.94%**
S-Transformer	L_2	81.5512	3.4842	2.5207	0.5361
	L_3	22.9896	3.1256	0.6811	0.4922
	%	**71.81%**	**10.29%**	**72.98%**	**8.18%**
Conv-AT	L_2	102.8211	4.4109	3.1926	0.6818
	L_3	36.2087	3.5679	1.0740	0.5601
	%	**64.78%**	**19.11%**	**66.36%**	**17.85%**

To summarize, it can be seen from the Figs. 2–4 that the improved loss function L_3 not only has a smaller error, but also predicts the trend more accurately. Detailed results of all the above numerical simulations (including mean square error and improvements for all numerical simulations) are given in Table 1. Rows L_2 and L_3 report the mean squared error between the predicted and numerical solutions obtained using the two respective loss functions. The "%" row indicates the percentage improvement achieved by loss function L_3 over L_2.

Through the numerical simulations, it is evident that the improved loss function proposed herein more effectively controls the accumulation of prediction errors when predicting the response of this chaotic system. Compared to traditional MSE or the original custom loss function, this approach not only maintains high accuracy in short-term predictions but also significantly reduces the error amplification effect during long-term iterative prediction. This advantage stems from the explicit constraint on prediction-phase errors within the loss function. Consequently, the framework demonstrates enhanced applicability and reliability in practical chaotic vibration scenarios.

5. Advancing the Design of the Loss Function

5.1. *Loss Function Design for Recursive Prediction*

As can be seen from the previous presentation or Fig. 1, the approach employed in the prediction phase is referred to as Recursive Prediction with Numerical Integration. This method iteratively advances the system state by first predicting the temporal derivative using a trained neural network and then updating the state through a numerical integration scheme. Specifically, starting from an initial state at t_0, the model predicts the corresponding derivative, which is subsequently used to compute the next time step via a numerical integration method (in this study, the Forward Euler method is adopted). The newly obtained time step is then fed back into the network to generate the next prediction, and this process continues recursively until the desired prediction time horizon is reached.

The loss function is computed over an entire batch by evaluating the discrepancy between the numerically integrated predictions and the corresponding ground truth values. Specifically, the computation at each time step relies on the ground truth values rather than the numerically integrated values from the previous step. Unlike the prediction phase, which recursively updates the system state step by step, the loss computation directly assesses the deviations across the entire batch in a non-recursive manner. The following provides an example using the ε_{3m} part of the loss function L_3, which is also the aspect of this section that has been improved. The ε_{3m} part is computed using the Forward Euler method, which can be expressed as:

$$X(t_i) + h\tilde{f}\big(X(t_i)\big) = \tilde{X}(t_{i+1}). \tag{14}$$

Here, $\tilde{f}(X(t_i))$ represents the prediction of the neural network at time t_i, while $\tilde{X}(t_{i+1})$ denotes the computed result at time t_{i+1} (For clarity in presenting the improvements in this section, we use ~ to denote predicted values rather than ground truth values.). Thus the ε_{3m} part of the loss function L_3 can be written more clearly as: $\tilde{X}(t_{i+1}) - X(t_{i+1}) = \varepsilon_{3m}$, which is equivalent to $X(t_i) + h\tilde{f}(X(t_i)) - X(t_{i+1}) = \varepsilon_{3m}$. Based on the preceding derivation, the ε_{3m} part of the loss function L_3 for the entire batch (with a batch size of 20) can be expressed as Eq. (15):

$$\begin{bmatrix} X(t_i) \\ X(t_{i+1}) \\ \cdots \\ X(t_{i+19}) \end{bmatrix} + \begin{bmatrix} h\tilde{f}(X(t_i)) \\ h\tilde{f}(X(t_{i+1})) \\ \cdots \\ h\tilde{f}(X(t_{i+19})) \end{bmatrix} - \begin{bmatrix} X(t_{i+1}) \\ X(t_{i+2}) \\ \cdots \\ X(t_{i+20}) \end{bmatrix} = \varepsilon_{3m}. \tag{15}$$

Since the loss function L_3 directly evaluates the deviation over the entire batch in a non-recursive manner, it is computed in parallel. Therefore, we can express it in a vectorized form.

5.1.1. *RecursiveLoss*

As discussed earlier, the loss function is computed in a non-recursive manner, whereas the prediction phase employs a recursive approach. Therefore, to ensure consistency between the loss computation and the prediction process, we reformulate the loss function into a recursive form in this section, allowing it to better capture the error propagation characteristics of the model during prediction. Still considering the entire batch as an example, in Eq. (15), the predictions at subsequent time steps are based on ground truth values. Therefore, we only need to replace these ground truth values with the computed results from the previous time step. After this modification, Eq. (15) takes the following form:

$$\begin{bmatrix} X(t_i) \\ \tilde{X}(t_{i+1}) \\ \cdots \\ \tilde{X}(t_{i+19}) \end{bmatrix} + \begin{bmatrix} h\tilde{f}(X(t_i)) \\ h\tilde{f}(X(t_{i+1})) \\ \cdots \\ h\tilde{f}(X(t_{i+19})) \end{bmatrix} - \begin{bmatrix} X(t_{i+1}) \\ X(t_{i+2}) \\ \cdots \\ X(t_{i+20}) \end{bmatrix} = \varepsilon_{3m}. \tag{16}$$

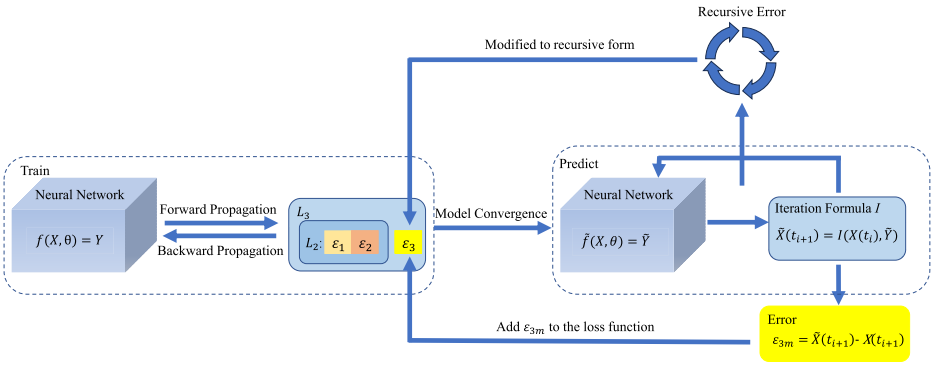


Fig. 5. Loss function L_{3a}.

For the sake of clarity, this formulation is presented in a vectorized form. However, Eq. (16) is inherently computed in a recursive manner rather than in parallel. We write the loss function modified in this way as L_{3a}. The recursive loss is schematized in Fig. 5.

5.1.2. *Results and Discussion*

We still train the model using data from 0 to 1500 seconds and predict time series data from 1500 to 3000 seconds. We plot the displacement time series curve predicted by the Conv-AT and S-Transformer using the new loss function L_{3a}, as shown in Figs. 6–7.

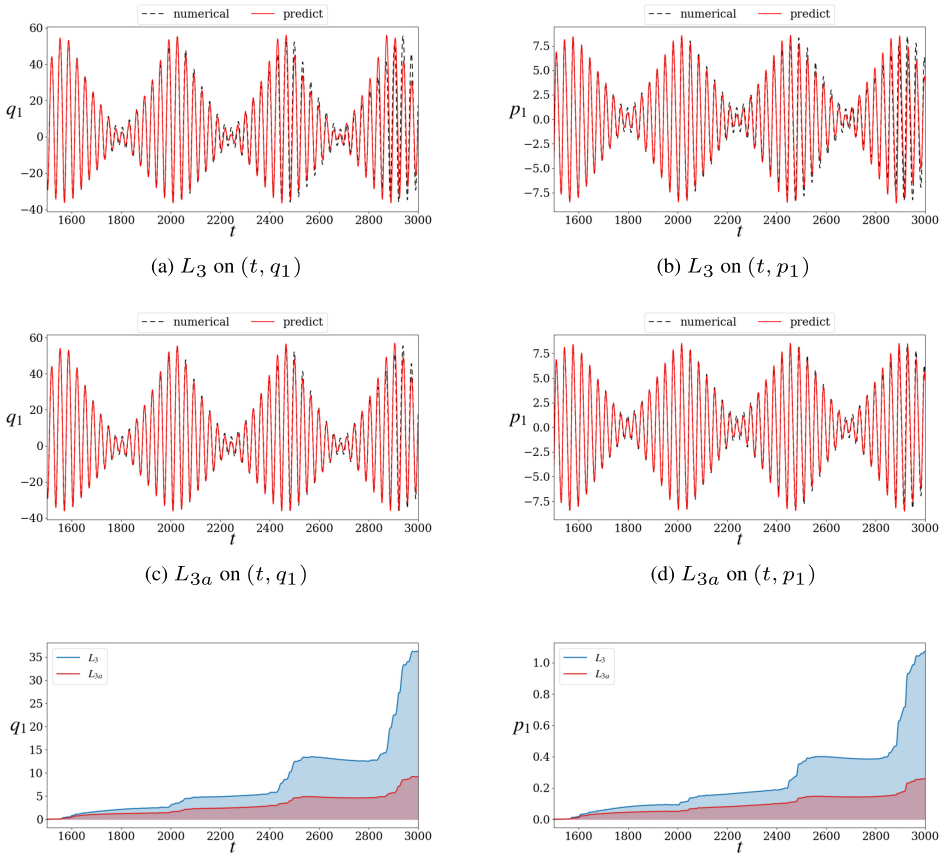
For both neural networks, the displacement time series curves predicted using RecursiveLoss are much better compared to loss function L_3, and are in better agreement with the numerical solution. The better results of RecursiveLoss can also be seen from the cumulative mean square error.

We also draw numerical and predictive solutions for the three-dimensional phase diagram of the beam-ring structure. Figures 8–11 compare the three-dimensional phase portraits of the numerical solution and the predicted solution. The phase portrait illustrates the system's trajectory in phase space, capturing the interplay between key state variables. The strong agreement between the predicted and numerical solutions demonstrates that the neural network effectively captures the system's underlying dynamics. The predicted phase portrait closely aligns with the numerical solution, preserving both the trajectory and essential geometric structures. This consistency highlights the neural network's ability to learn the system's governing dynamics.

Detailed results of all the above numerical simulations (including mean square error and improvements for all numerical simulations) are given in Table 2.

5.2. *Variable Step Loss Function*

As can be seen from the previous sections, the prediction of the loss functions L_3 and L_{3a} is significantly improved, but there is still some error. In addition, the loss functions L_3 and L_{3a} have the limitation that they require the step size of the prediction to be the



(a) L_3 on (t, q_1) (b) L_3 on (t, p_1)

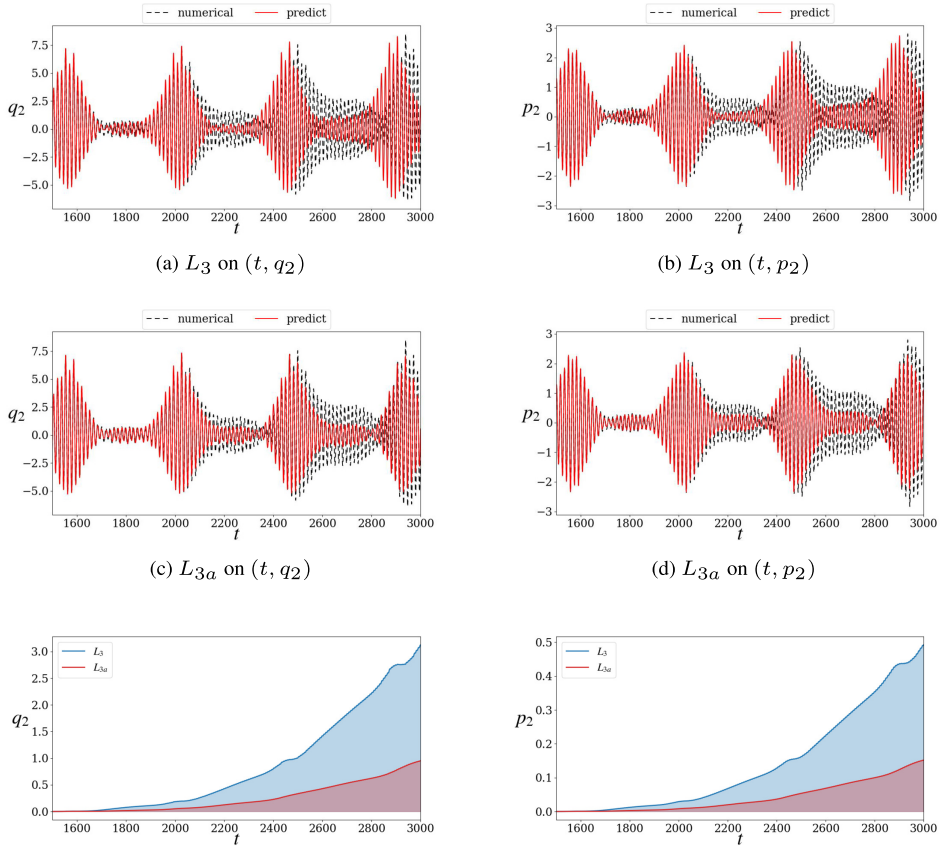
(c) L_{3a} on (t, q_1) (d) L_{3a} on (t, p_1)

(e) Cumulative MSE on (t, q_1) (f) Cumulative MSE on (t, p_1)

Fig. 6. Comparison of displacement time series curves and errors predicted by Conv-AT using two loss functions.

Table 2
Comparison of errors for loss functions L_3 and L_{3a}.

Neural networks	Comparison	Planes			
		(t, q_1)	(t, q_2)	(t, p_1)	(t, p_2)
LSTM_ED	L_3	46.1518	2.9058	1.5152	0.4531
	L_{3a}	10.8199	3.0217	0.3825	0.4776
	%	**76.56%**	**−3.99%**	**74.75%**	**−5.41%**
S-Transformer	L_3	22.9896	3.1256	0.6811	0.4922
	L_{3a}	1.3374	0.9517	0.0448	0.1517
	%	**94.18%**	**69.55%**	**93.43%**	**69.18%**
Conv-AT	L_3	36.2087	3.5679	1.0740	0.5601
	L_{3a}	9.1624	2.6030	0.2589	0.4136
	%	**74.70%**	**27.04%**	**75.90%**	**26.16%**



(a) L_3 on (t, q_2)

(b) L_3 on (t, p_2)

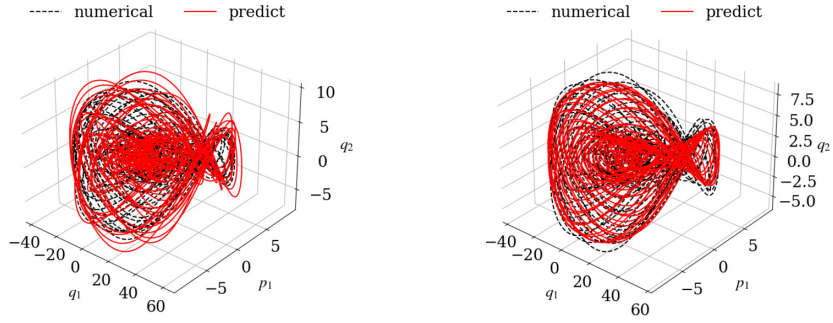
(c) L_{3a} on (t, q_2)

(d) L_{3a} on (t, p_2)

(e) Cumulative MSE on (t, q_1)

(f) Cumulative MSE on (t, p_1)

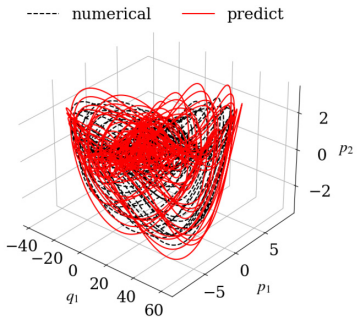
Fig. 7. Comparison of displacement time series curves and errors predicted by S-Transformer using two loss functions.



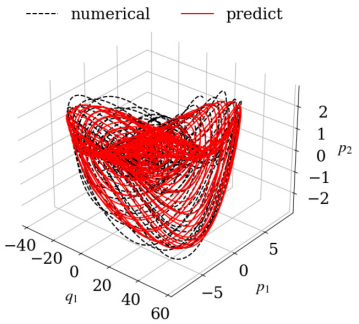
(a) Three-dimensional phase portrait predicted using the loss function L_2

(b) Three-dimensional phase portrait predicted using the loss function L_{3a}

Fig. 8. Three-dimensional phase trajectories of numerical and predicted solutions in the (q_1, p_1, q_2) space.

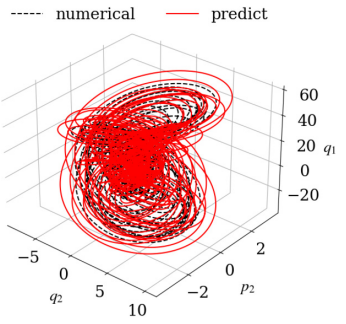


(a) Three-dimensional phase portrait predicted using the loss function L_2

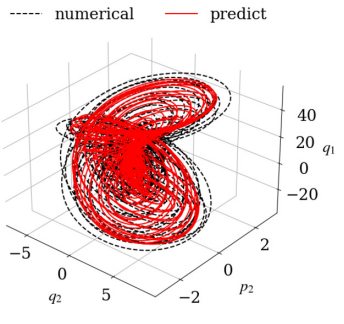


(b) Three-dimensional phase portrait predicted using the loss function L_{3a}

Fig. 9. Three-dimensional phase trajectories of numerical and predicted solutions in the (q_1, p_1, p_2) space.

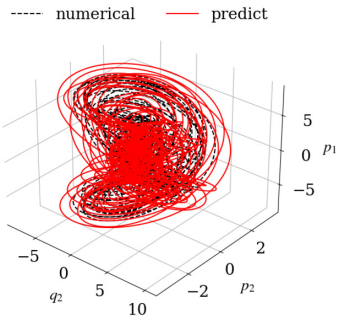


(a) Three-dimensional phase portrait predicted using the loss function L_2

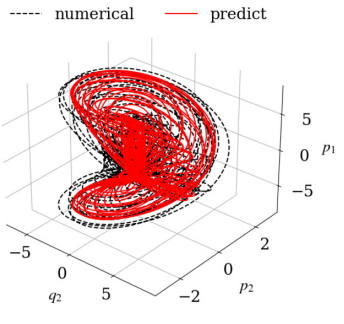


(b) Three-dimensional phase portrait predicted using the loss function L_{3a}

Fig. 10. Three-dimensional phase trajectories of numerical and predicted solutions in the (q_2, p_2, q_1) space.



(a) Three-dimensional phase portrait predicted using the loss function L_2



(b) Three-dimensional phase portrait predicted using the loss function L_{3a}

Fig. 11. Three-dimensional phase trajectories of numerical and predicted solutions in the (q_2, p_2, p_1) space.

same as the step size of the training data. These two drawbacks motivate us to improve it further.

In numerical methods for solving initial value problems of ordinary differential equations, smaller step sizes provide key advantages. They enhance solution accuracy by reducing local truncation errors, particularly in regions with rapid changes. Additionally, smaller step sizes improve the stability of the numerical solution by minimizing the risk of error accumulation.

Therefore, our next goal is to continue to improve the loss function L_{3a} so that the loss function can adapt to different prediction step sizes. The main goal is to adapt the loss function to smaller prediction steps with a fixed step size of the training data, thus taking advantage of higher accuracy and stability in the numerical computation.

5.2.1. *Interpolation*

The ε_{3m} in the loss function L_{3a} is mainly to control the error of the iterative formula in the prediction process, so the step size of the prediction process must be consistent with the step size of the ε_{3m}. This limits the step size at prediction time, i.e. we cannot use smaller step sizes in the prediction phase to improve numerical stability. In fact, in some cases small step sizes are very important for the accuracy of the prediction results. Therefore, if the training data is given, we introduce the interpolation method to use a smaller step size in the prediction phase.

From the perspective of computational time and memory usage, this study employs simple linear interpolation for numerical simulations. However, this choice does not preclude the effectiveness of other interpolation methods, and readers are encouraged to explore alternative approaches.

Linear interpolation is a simple and widely used numerical interpolation method for estimating unknown values between known data points. It is based on two known points and interpolation is performed by drawing a straight line between these two points. Given two known points (x_0, y_0) and (x_1, y_1), linear interpolation is computed arbitrarily by the following equation x values (satisfying $x_0 \leqslant x \leqslant x_1$) corresponds to the y values:

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0} \cdot (x - x_0). \tag{17}$$

The number of insertion points is determined by the desired step size, and the x_k values are then selected to be evenly spaced accordingly. These x_k values can be generated using linear intervals: $x_k = x_0 + k \cdot \frac{x_1 - x_0}{n+1}$, $k = 1, 2, \ldots, n$.

Since the interpolated data actually has some error, we only use the interpolated data to train for part ε_{3m}. Parts ε_{1i} and ε_{2j} are still trained with the original data. We denote the improved loss function as L_{3b}, and the schematic is shown in Fig. 12. As can be seen from the Fig. 12, the step size of the ε_{1i} and ε_{2j} parts is notated as h, which is consistent with the sampling step size of the training data. The ε_{3m} part is closely related to the prediction phase from the previous introduction. Therefore, when the step size of the formula of the iterative formula of the prediction phase is set to h/n, we interpolate the data input to the ε_{3m} part so that it has a step size of h/n. This realizes the use of training data with larger

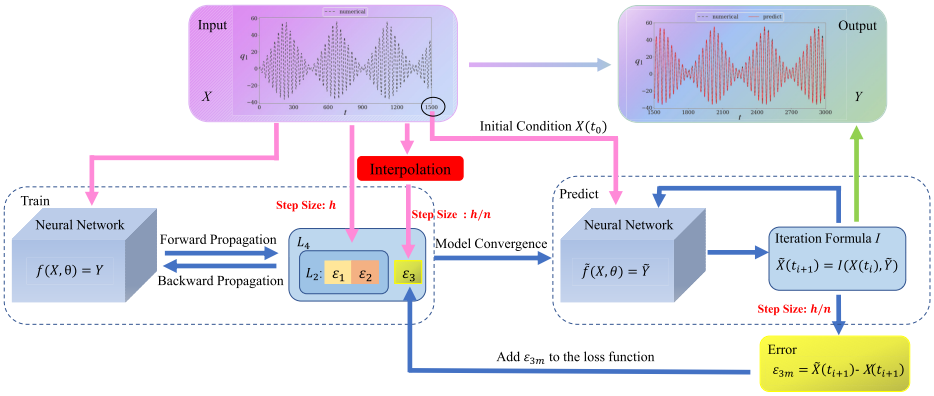


Fig. 12. Loss function L_{3b}.

step sizes, but smaller step sizes during the prediction process to improve the accuracy of the prediction.

Taking the Conv-AT neural network as an example, we refine the step size in the loss function from 1500/120000 to 1500/480000 through interpolation. In each batch, since the number of interpolated data points is nearly four times that of the original data, directly applying recursion to the entire interpolated dataset would result in excessive errors, making it difficult for the model to effectively perform backpropagation and update parameters. To address this, we divide the interpolated data into four groups and apply recursive computation independently within each group. This way the error is not too large thus ensuring that the neural network can back propagate and update the parameters effectively.

5.2.2. *Results and Discussion*

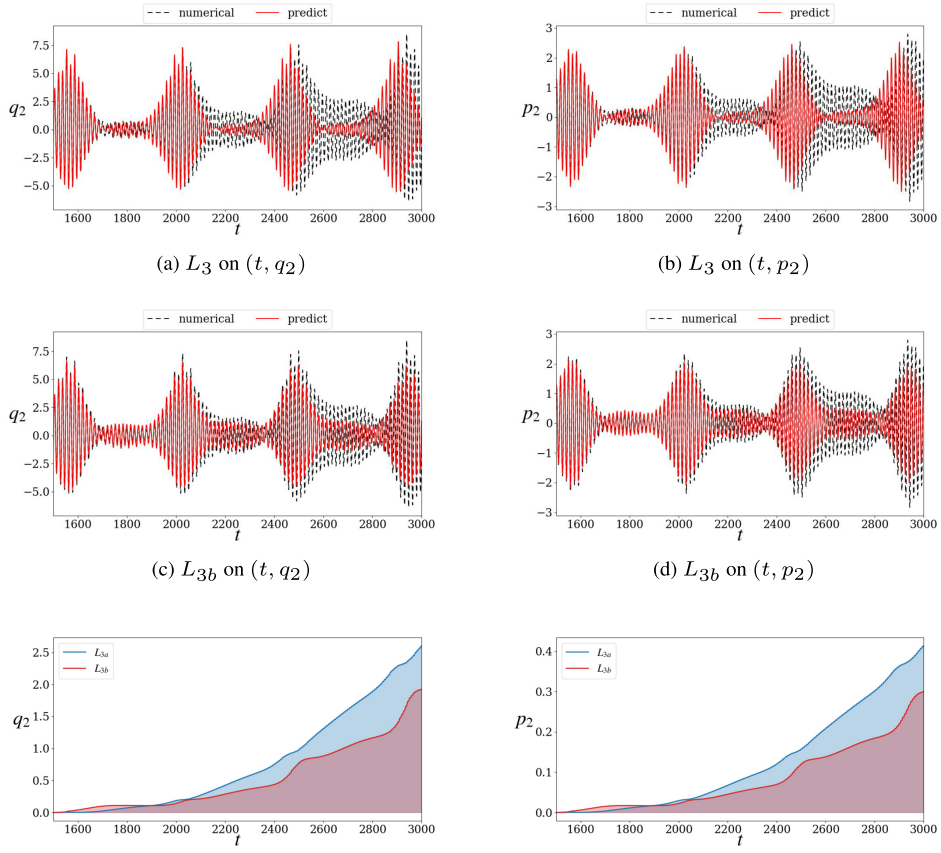
The training and prediction ranges are kept consistent with those in the previous sections. The displacement time series predicted by Conv-AT using the new loss function L_{3b} is shown in Fig. 13.

As can be seen in Fig. 13, the predicted values of the neural network using the loss function L_{3b} are closer to the numerical solution than L_{3a}, and the cumulative mean-square error plot also shows that L_{3b} accumulates less error. It can be seen that our two improved strategies still give better results when combined.

In fact, we also made the same improvement for LSTM_ED, S-Transformer, but their prediction errors were instead improved. Therefore, we conjecture that the interpolation method is more effective for neural network models that require high numerical stability. Further investigations are needed to quantify its impact on different types of neural networks.

6. Conclusions

In this paper, based on the original linear multistep loss function, a new loss function strategy is proposed under the condition that the time step is guaranteed to be consistent.



(a) L_3 on (t, q_2) (b) L_3 on (t, p_2)

(c) L_{3b} on (t, q_2) (d) L_{3b} on (t, p_2)

(e) Cumulative MSE on (t, q_2) (f) Cumulative MSE on (t, p_2)

Fig. 13. Comparison of displacement time series curves and errors predicted by Conv-AT using two loss functions.

In order to test the effect of the traditional loss function (MSE), custom loss function and the improved custom loss function on the network prediction, different neural networks are used to predict the time series of the beam ring system. Based on the analysis, the key conclusions are summarized as follows:

1. For neural networks of the same size, the use of traditional MSE loss functions is very ineffective and cannot accurately capture the complex dynamic behaviour of the system. However, a custom loss function based on three types of errors can more effectively capture the complex dynamics of the system, thereby significantly enhancing prediction accuracy.
2. Numerical simulations show that the improved loss function has higher prediction accuracy with the same neural network size. In addition, the time increase of using the improved loss function is negligible, and the strategy of improving the loss function is generalisable.

3. To further optimise the loss function, we have introduced RecursiveLoss. By calculating the loss function in a recursive form, the loss function can simulate the error that occurs in the prediction phase, which effectively improves the prediction accuracy. Interpolation methods are then introduced on this basis to reduce the step size of the prediction phase under the same input conditions. The addition of interpolation to the prediction error term of the loss function aims to overcome the dependence of the adapted loss function on the step size and broaden its scope of application. Finally, the applicability of the interpolation method is discussed.

It is worth mentioning that our improvement of the loss function does not involve the addition of specific formulas, but rather the incorporation of iterative prediction methods into the loss function in order to control the errors generated during the iterative process. This strategy is applicable to any iterative method and can therefore provide some assistance in interdisciplinary research in applied mechanics and machine learning.

References

Akçakaya, M., Moeller, S., Weingärtner, S., Uğurbil, K. (2019). Scan-specific robust artificial-neural-networks for k-space interpolation (RAKI) reconstruction: database-free deep learning for fast imaging. *Magnetic Resonance in Medicine*, 81, 439–453.

Calvo-Rolle, J.L., Fontenla-Romero, O., Pérez-Sánchez, B., Guijarro-Berdiñas, B. (2014). Adaptive inverse control using an online learning algorithm for neural networks. *Informatica*, 25(3), 401–414.

Chopra, S., Hadsell, R., LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 539–546.

D'Ambrosio, A., Schiassi, E., Curti, F., Furfaro, R. (2021). Pontryagin neural networks with functional interpolation for optimal intercept problems. *Mathematics*, 9(9), 996.

Fernandes, B., Silva, F., Alaiz-Moreton, H., Novais, P., Neves, J., Analide, C. (2020). Long short-term memory networks for traffic flow forecasting: exploring input variables, time frames and multi-step approaches. *Informatica*, 31(4), 723–749.

Ghazvini, A., Sharef, N.M., Sidi, F.B. (2024). Prediction of course grades in computer science higher education program via a combination of loss functions in LSTM model. *IEEE Access*, 12, 30220–30241.

Hadsell, R., Chopra, S., LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 1735–1742.

He, J., Li, X.X., Zhu, H.Q. (2023). An adaptive discrete physics-informed neural network method for solving the Cahn-Hilliard equation. *Engineering Analysis with Boundary Elements*, 155, 826–838.

Huber, P.J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1), 73–101.

Kendall, A., Gal, Y., Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491.

Lamb, A., Verma, V., Kannala, J., Bengio, Y. (2019). Interpolated adversarial training: achieving robust neural networks without sacrificing too much accuracy. In: *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISec'19)*, pp. 95–103.

Lee, H., Lee, J., Cho, S. (2017). View-interpolation of sparsely sampled sinogram using convolutional neural network. In: *Medical Imaging 2017: Image Processing*, Vol. 10133, p. 1013328.

Li, K.J. (2024). MultiPINN: multi-head enriched physics-informed neural networks for differential equations solving. *Neural Computing and Applications*, 36, 11371–11395.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P. (2017). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 318–327.

Liu, T., Meidani, H. (2023). Physics-informed neural networks for system identification of structural systems with a multiphysics damping model. *Journal of Engineering Mechanics*, 149(10), 04023079.

Papineni, K., Roukos, S., Ward, T., Zhu, W.J. (2002). BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318.

Ranzato, M., Chopra, S., Auli, M., Zaremba, W. (2015). Sequence Level Training with Recurrent Neural Networks. arXiv preprint: arXiv:1511.06732.

Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pp. 452–461.

Rumelhart, D., Hinton, G., Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Sestanovic, T., Kalinic Milicevic, T. (2025). Identification of the optimal neural network architecture for prediction of Bitcoin return. *Informatica*, 36(1), 175–196.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423.

Sun, Y., Zhang, L.Y., Yao, M.H., Zhang, J.H. (2024). Neural network models and Shapley additive explanations for a beam-ring structure. *Chaos, Solitons & Fractals*, 185, 115114.

Verma, V., Kawaguchi, K., Lamb, A., Kannala, K., Solin, A., Bengio, Y., Lopez-Paz, D. (2022). Interpolation consistency training for semi-supervised learning. *Neural Networks*, 145, 90–106.

Wu, Z., Rincon, D., Christofides, P.D. (2020). Incorporating structural process knowledge in recurrent neural network modeling of nonlinear processes. In: *Proceedings of the American Control Conference (ACC)*, pp. 2413–2418.

Xie, X.P., Zhang, G.N., Webster, C.G. (2019). Non-intrusive inference reduced order model for fluids using deep multistep neural network. *Mathematics*, 7(8), 757.

Yang, M., Zhang, Y.N., Hu, H.F. (2020). Discrete ZNN models of Adams-Bashforth (AB) type solving various future problems with motion control of mobile manipulator. *Neurocomputing*, 384, 84–93.

Yin, S.B., Hu, S.H., Wang, Y.B., Yang, Y.H. (2023). High-order Adams Network (HIAN) for image dehazing. *Applied Soft Computing*, 139, 110204.

Yuan, Q.W., He, J.W., Yu, L., Zheng, G. (2020). Aim-Net: bring implicit Euler to network design. In: *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 1926–1930.

Zhai, W.D., Tao, D.W., Bao, Y.Q. (2023). Parameter estimation and modeling of nonlinear dynamical systems based on Runge-Kutta physics-informed neural network. *Nonlinear Dynamics*, 111, 21117–21130.

Zhang, L.Y., Sun, Y., Wang, A.W., Zhang, J.H. (2023). Neural network modeling and dynamic behavior prediction of nonlinear dynamic systems. *Nonlinear Dynamics*, 111, 11335–11356.

Zhang, W., Wu, R.Q., Behdinan, K. (2019). Nonlinear dynamic analysis near resonance of a beam-ring structure for modeling circular truss antenna under time-dependent thermal excitation. *Aerospace Science and Technology*, 86, 296–311.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W. (2021). Informer: beyond efficient transformer for long sequence time-series forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, pp. 11106–11115.

Zhu, D., Cheng, X., Zhang, F., Yao, X., Gao, Y., Liu, Y. (2019). Spatial interpolation using conditional generative adversarial neural networks. *International Journal of Geographical Information Science*, 34(4), 735–758.

Zhu, M., Chang, B., Fu, C. (2022). Convolutional neural networks combined with Runge-Kutta methods. *Neural Computing and Applications*, 35(2), 1629–1643.

K. An is a postgraduate student in the School of Applied Science at Beijing Information Science and Technology University, Beijing, China. He completed a bachelor of science degree in Applied Statistics at Xuzhou University of Technology. His research program encompasses data-driven modelling and deep learning.

Y. Sun was awarded a PhD by Beijing University of Technology. She currently holds the position of associate professor at the School of Applied Science, Beijing Information Science and Technology University. Her research program encompasses data-driven modelling and the analysis of nonlinear dynamical systems, including chaos and bifurcation theory.

M. Yao is currently a professor at School of Aeronautics and Astronautics, Tiangong University, China. Her research interests mainly focus on nonlinear vibrations of mechanical structures, energy harvesting, self-power sensors and systems and artificial intelligence.

J. Zhang received the PhD degree in engineering mechanics from Beijing University of Technology in 2009, and the MSc degree in applied mathematics from Hebei University of Technology in 2005. She is currently a professor with the College of Mechanical and Electrical Engineering, Beijing Information Science and Technology University of China. Her research interests include dynamics and control of nonlinear mechanical systems, design of lightweight metamaterial structures.