

## Highlights

- In our methodology, the main innovation can be seen in the incorporation of deep neural networks (DNN) into a classical numerical integration method by using a hybridized integration time-stepper for nonlinear seismic response prediction;
- We propose to use residual network (ResNet) to learn time-stepping schemes specifically for the nonlinear state variables of the system;
- The proposed time-stepping scheme provides important advantages over current pure data-driven approaches, including (i) a flexible framework incorporating known time-invariant physics information, (ii) requirement of structural seismic response data being circumvented by simple short bursts of trajectories collected from underlying nonlinear components, and (iii) efficiency in training and validation process;
- Our results also indicate that a simple feedforward or convolutional architecture outperforms recurrent networks to fulfill the requirement of prediction accuracy as well as long-range memory in structural dynamics.

# Combination of physics-based and data-driven modeling for nonlinear structural seismic response prediction through deep residual learning

Jia GUO<sup>\*,1</sup>, Ryuta ENOKIDA<sup>2</sup>, Dawei LI<sup>3</sup>, and Kohju IKAGO<sup>2</sup>

<sup>1</sup>Division of Environmental Science and Technology, Kyoto University, Kyoto, Japan

<sup>2</sup>International Research Institute of Disaster Science, Tohoku University, Sendai, Japan

<sup>3</sup>School of Civil Engineering, Lanzhou University of Technology, Lanzhou, China

## Abstract

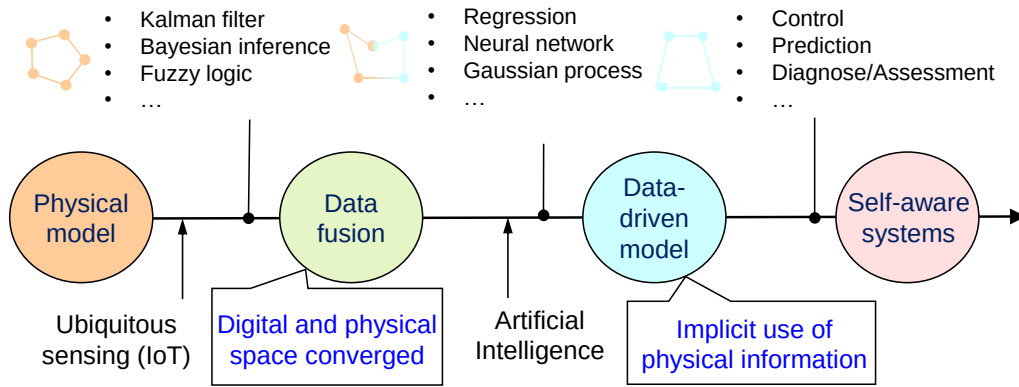
Despite great progress in seeking accurate numerical approximator to nonlinear structural seismic response prediction using deep learning approaches, tedious training process and large volume of structural response data under earthquakes for training and validation are often prohibitively accessible. In our methodology, the main innovation can be seen in the incorporation of deep neural networks (DNN) into a classical numerical integration method by using a hybridized integration time-stepper. In this way, the linear physics information of the structure and the obscure nonlinear dynamics are smoothly combined. We propose to use residual network (ResNet) to learn time-stepping schemes specifically for the nonlinear state variables of the system. Our Physics-DNN hybridized integration (PDHI) time-stepping scheme provides important advantages over current pure data-driven approaches, including (i) a flexible framework incorporating known time-invariant physics information, (ii) requirement of structural seismic response data being circumvented by simple short bursts of trajectories collected from underlying nonlinear components, and (iii) efficiency in training and validation process. Besides, our results indicate that a simple feedforward or convolutional architecture outperforms recurrent networks to fulfill the requirement of prediction accuracy as well as long-range memory in structural dynamic analysis. Several numerical and experimental examples are presented to demonstrate the performance of the method.

**Keywords:** Deep Neural Network, Residual Network, Linear Multistep Method, Eddy Current Damper, Temporal Convolutional Network

## 1 Introduction

Modeling, simulating and identifying nonlinear structural systems enables comprehending and forecasting their complex dynamic characteristics. First principles models based on high-fidelity mathematical representations have been used extensively in seismic analysis of structural systems. However, in the present era of adequate data availability and advanced computing power, we see a growing interest in converging the modeling and identifying processes through data fusion algorithms [1–4], as shown in **Figure 1**. Consistent and comprehensive integration of the dynamic models and physical spaces is a main beneficiary of this convergence and some representative techniques include Kalman filter [5], Bayesian inference [6], Fuzzy theory [7], etc. In similar fashion, current practices on discovering dynamics systems using observational data through deep learning (DL) and artificial intelligence (AI) algorithms alleviate the need of physical model design and have emerged as a new field in structural dynamics [8–10]. The direct relevance to real-time data synchronization between physical spaces and dynamic models in deep learning based algorithms offers the benefit of improved decision supports and automatic planning and maintenance, such as system control, prediction, diagnose/assessment and so forth. Some pioneered previous work includes constructing nonlinear structural response model through long short-term memory (LSTM) for building structures using numerical and experimental data [11] as well as real sensed seismic data [12], approximating governing equations by means of Neural Ordinary Differential Equations (Neural ODEs) [13], modeling structural complex hysteretic behaviors based on Transformer network architecture [14], and employing generalized linear regression approach to uncover the full parametric form of damping behaviors [15]. Specifically, considerable progress has been made in physics-informed learning strategy for modeling and prediction of dynamics. Instead of establishing a purely black-box model, more and more data-driven algorithms intend to embed more physical/engineering interpretations towards an explainable mathematical model with implicit use of physical information, which can be found in [16, 17].

\*Correspondence to: Jia GUO, Email: guo.jia.8x@kyoto-u.ac.jp



**Figure 1** Flow chart showing the discovery of dynamics from physical modeling to data-driven modeling

Despite great progress in seeking accurate numerical approximator to nonlinear structural seismic response prediction using deep learning approaches, tedious training process and large volume of structural response data under earthquakes for training and validation are often prohibitively accessible. In this work, the main innovation can be seen in the incorporation of deep neural networks (DNN) into a classical numerical integration method by using a hybridized integration time-stepper. In particular, we set the nonlinear restoring force vector as augmented states, and hence change the original nonlinear differential equation into a linear one, so that common explicit integration algorithms in control theory for linear time-invariant dynamic system can be applied. We propose to use residual network (ResNet) to learn time-stepping schemes specifically for the nonlinear state variables of the system in the context of linear multistep method. The original explicit integrator is then supplemented with the DNN time-stepper with dimensional expanding. Our Physics-DNN hybridized integration (PDHI) time-stepping scheme provides important advantages over current pure data-driven approaches, including (i) a flexible framework incorporating known time-invariant physics information, (ii) requirement of structural seismic response data being circumvented by simple short bursts of trajectories collected from underlying nonlinear components, and (iii) efficiency in training and validation process. Besides, our results indicate that a simple feedforward or convolutional architecture outperforms recurrent networks to fulfill the requirement of prediction accuracy as well as long-range memory in PDHI. It is also remarkable that the proposed approach is parallelizable and allows relatively large integration time step.

This paper is structured as follows. Section 2 establishes the proposed Physics-DNN hybridized integration time-stepping scheme for nonlinear structural seismic response prediction after the basic problem setup. Some theoretical properties of the PDHI time-stepper, e.g. consistency and stability, are additionally being discussed in this section. Section 3 demonstrates how the residual neural network can be employed for the PDHI time-stepper and the network's corresponding training and prediction strategies are introduced. Section 4 investigates the performance of the proposed method to a numerical benchmark model of a base-isolated seven-story shear building. Experimental identifications of Eddy current damping system are then explored in Section 5. Final conclusions are drawn in Section 6. Data and codes used in this manuscript will be publicly available on GitHub if this manuscript is accepted.

## 2 Problem statement

### 2.1 Mathematical model of nonlinear structural system

In this paper, we consider a general  $n$  degree-of-freedom ( $n$ DOF) nonlinear structural system with time-invariant mass, damping, stiffness, and external force location matrices  $\mathbf{M}, \mathbf{C}, \mathbf{K}, \mathbf{S}_p \in \mathbb{R}^{n \times n}$ , respectively. Let  $\mathbf{q}(t) \in \mathbb{R}^n$  represent the system's displacement vector and  $\mathbf{p}(t)$  represent the local external loading vector. The dynamic equation of motion of the system is given by

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) + \mathbf{R}(t; \boldsymbol{\xi}) = \mathbf{S}_p\mathbf{p}(t), \\ \mathbf{q}(0) = \mathbf{q}_0, \dot{\mathbf{q}}(0) = \dot{\mathbf{q}}_0; \dot{\boldsymbol{\xi}} = 0. \end{cases} \quad (1)$$

where  $\mathbf{R}(t; \boldsymbol{\xi}) \in \mathbb{R}^n$  denotes the nonlinear restoring force vector with parameterization  $\boldsymbol{\xi}$ . The initial conditions at  $t = 0$  are given by initial displacement  $\mathbf{q}_0$  and initial velocity  $\dot{\mathbf{q}}_0$ . In this paper, we focus on the case that the nonlinear contribution of  $\mathbf{R}$  pertains to  $m$  local hysteretic components  $z_i, i = 1, \dots, m$ , and each of the hysteretic component is determined by

$$\dot{z}_i = r_i(z_i, y_i, \dot{y}_i; \boldsymbol{\xi}_i) \quad (2)$$

where  $y_i$  and  $\dot{y}_i$  are the local displacement and velocity of the  $i$ th hysteretic component and  $\xi_i$  is the corresponding nonlinear parameters for the differentiable function  $r_i(\cdot)$ . The relation between the local displacements  $\mathbf{y} = [y_1, y_2, \dots, y_m]^T$  and the global displacements  $\mathbf{u}$  is then defined through

$$\mathbf{y} = \mathbf{T}\mathbf{q} \quad (3)$$

where  $\mathbf{T} \in \mathbb{R}^{m \times n}$  is the connection matrix [18]. Accordingly, the dynamics of the nonlinear system can be determined by using functional analysis with extended Hamilton's principle

$$\int_{t_1}^{t_2} [\delta T - \delta U + \delta W_{nc}] dt = 0, \forall t_1 < t_2 \quad (4)$$

where  $T$  is the kinetic energy,  $U$  is the elastic energy and  $W_{nc}$  is the non-conservative work.  $\delta$  denotes the variation operator. Upon application of the Euler-Lagrange equations, the first two terms in **Eq. (4)** goes to

$$\delta T - \delta U = (\delta \mathbf{q})^T \left( -\frac{d}{dt} \frac{\partial T}{\partial \dot{\mathbf{q}}} - \frac{\partial U}{\partial \mathbf{q}} \right) = -(\delta \mathbf{q})^T \mathbf{M}\ddot{\mathbf{q}} - (\delta \mathbf{q})^T \mathbf{K}\mathbf{q}, \quad (5)$$

where

$$T = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}}, \quad U = \frac{1}{2} \mathbf{q}^T \mathbf{K} \mathbf{q}. \quad (6)$$

The non-conservative forces in the dynamic system are the external loading force  $\mathbf{S}_p \mathbf{p}(t)$ , the damping force  $\mathbf{C}\dot{\mathbf{q}}$  and the nonlinear restoring force  $\mathbf{z} = [z_1, \dots, z_m]^T$ , and thus the change in non-conservative work  $W_{nc}$  is given by

$$\delta W_{nc} = (\delta \mathbf{q})^T \mathbf{P}(t) - (\delta \mathbf{q})^T \mathbf{C}\dot{\mathbf{q}} - (\delta \mathbf{y})^T \mathbf{z} \quad (7)$$

Substituting **Eq. (3)** into the last of **Eq. (7)** and combining **Eq. (2)** with **Eq. (5)** in matrix form gives the complete equations of motion of this nonlinear structural system

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{T}^T \mathbf{z} = \mathbf{S}_p \mathbf{p}(t), \\ \dot{\mathbf{z}} = \mathbf{g}(\mathbf{z}, \mathbf{T}\mathbf{q}, \mathbf{T}\dot{\mathbf{q}}, \boldsymbol{\xi}); \quad \dot{\boldsymbol{\xi}} = 0 \\ \mathbf{q}(0) = \mathbf{q}_0, \quad \dot{\mathbf{q}}(0) = \dot{\mathbf{q}}_0, \quad \mathbf{z}(0) = \mathbf{z}_0. \end{cases} \quad (8)$$

Let  $\mathbf{x} = [\mathbf{q}; \dot{\mathbf{q}}]$  and  $\mathbf{x}_0 = [\mathbf{q}_0; \dot{\mathbf{q}}_0]$ . The main equation in **Eq. (8)** becomes

$$\dot{\mathbf{x}} = \mathbf{A}_c \mathbf{x} + \mathbf{B}_c^z \mathbf{z} + \mathbf{B}_c^p \mathbf{p} \quad (9a)$$

$$\dot{\mathbf{z}} = \hat{\mathbf{g}}(\mathbf{z}, \mathbf{H}\mathbf{x}; \boldsymbol{\xi}) = \hat{\mathbf{g}}(\mathbf{z}, \mathbf{u}; \boldsymbol{\xi}). \quad (9b)$$

where

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}, \quad \mathbf{B}_c^z = \begin{bmatrix} \mathbf{0} \\ -\mathbf{M}^{-1}\mathbf{T}^T \end{bmatrix}, \quad \mathbf{B}_c^p = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{S}_p \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} \end{bmatrix}. \quad (10)$$

$\mathbf{u} \in \mathbb{R}^{2m}$  is obtained from the global state vector  $\mathbf{x}$  using the connection matrix  $\mathbf{T}$ . In **Eq. (9b)**,  $\mathbf{u}$  serves as an external input of the differential equation  $\hat{\mathbf{g}}$ . It is noteworthy that the equations of motion of the nonlinear structural system can be viewed as a combination of two parts: the linear differential part in **Eq. (9a)** and nonlinear differential part in **Eq. (9b)**. Following this, the construction of the nonlinear restoring force  $\mathbf{z}$  (augmented state vector) will be separated from the construction of the state vector  $\mathbf{x}$ .

## 2.2 Physics-DNN hybridized integration time-stepper

Considering the exact integrating result of **Eq. (9a)** over a time interval  $[t_{k-1}, t_k = t_{k-1} + h]$  by using Volterra integral equation

$$\mathbf{x}_k = e^{\mathbf{A}_c h} \mathbf{x}_{k-1} + \int_{t_{k-1}}^{t_k} e^{\mathbf{A}_c (h-\tau)} (\mathbf{B}_c^z \mathbf{z}(t_{k-1} + \tau) + \mathbf{B}_c^p \mathbf{p}(t_{k-1} + \tau)) d\tau, \quad (11)$$

where  $t_0 = 0$  and  $k = 1, \dots, N$ . The simplest numerical method for **Eq. (11)** is realized by holding  $\mathbf{z}$  and  $\mathbf{p}$  to be piece-wise constant over the full interval, which gives

$$\mathbf{x}_k = \mathbf{A} \mathbf{x}_{k-1} + \mathbf{B}_z \mathbf{z}_{k-1} + \mathbf{B}_p \mathbf{p}_{k-1} \quad (12)$$

where  $\mathbf{A} = e^{\mathbf{A}_c h}$ ,  $\mathbf{B}_z = [\mathbf{A} - \mathbf{I}] \mathbf{A}_c^{-1} \mathbf{B}_c^z$ ,  $\mathbf{B}_p = [\mathbf{A} - \mathbf{I}] \mathbf{A}_c^{-1} \mathbf{B}_c^p$ . Since the exponential integrator in **Eq. (12)** has been studied extensively with high accuracy in many applications [19], and the effectiveness of the zero order assumption for  $\mathbf{z}$  and  $\mathbf{p}$  in seismic analysis has been validated in [5], we use this classic exponential time-step scheme for the integration of the linear part of the differential equation in **Eq. (9a)**.

As for the nonlinear part in **Eq. (9b)**, there is a wide range of deep learning based techniques developed for identifying nonlinear dynamics. To clarify the underlying stability and accuracy of the deep learning approach, the model of deep neural networks is derived from the general form of linear multistep method [20]:

$$\sum_{s=0}^M \alpha_s \mathbf{z}_{k-s} = h \sum_{s=0}^M \beta_s \hat{g}(\mathbf{z}_{k-s}, \mathbf{u}_{k-s}; \boldsymbol{\xi}), \quad k = M, \dots, N \quad (13)$$

Different choices of the coefficients  $\alpha_s$  and  $\beta_s$  result in different time-stepping schemes. In this research, we use residual networks to approximate two main families of linear multistep method: Adams-Bashforth methods and Adams-Moulton methods.

### 2.2.1 Adams-Bashforth methods

The  $M$ -step Adams-Bashforth(AB) method, where  $M = 1, 2, 3$  respectively, is written as

$$\begin{aligned} \mathbf{z}_k - \mathbf{z}_{k-1} &= h \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}), \text{ Eulers method} \\ \mathbf{z}_k - \mathbf{z}_{k-1} &= h \left( \frac{3}{2} \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}) - \frac{1}{2} \hat{g}(\mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}) \right), \\ \mathbf{z}_k - \mathbf{z}_{k-1} &= h \left( \frac{23}{12} \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}) - \frac{16}{12} \hat{g}(\mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}) + \frac{5}{12} \hat{g}(\mathbf{z}_{k-3}, \mathbf{u}_{k-3}; \boldsymbol{\xi}) \right). \end{aligned} \quad (14)$$

In contrast to replacing the function  $\hat{g}$  by a neural network, we directly construct each flow map  $h \sum_{s=0}^M \beta_s \hat{g}(\mathbf{z}_{k-s}, \mathbf{u}_{k-s}; \boldsymbol{\xi})$  at the right-hand side of **Eq. (14)** using the residual neural network (ResNet):

$$\begin{aligned} \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{ex}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}), \\ \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{ex}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}, \mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}), \\ \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{ex}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}, \mathbf{z}_{k-2}, \mathbf{u}_{k-2}, \mathbf{z}_{k-3}, \mathbf{u}_{k-3}; \boldsymbol{\xi}). \end{aligned} \quad (15)$$

where  $\mathcal{N}_{ex}$  is the residual function in ResNet and we define  $\mathbf{z}_{k-1} + \mathcal{N}_{ex}$  as the explicit DNN time-stepper in this research. For a complete exposition of the ResNet and the residual function, the reader is referred to Section 3.

### 2.2.2 Adams-Moulton methods

The  $M$ -step Adams-Moulton(AM) method, where  $M = 0, 1, 2$  respectively, is written as

$$\begin{aligned} \mathbf{z}_k - \mathbf{z}_{k-1} &= h \hat{g}(\mathbf{z}_k, \mathbf{u}_k; \boldsymbol{\xi}), \text{ backward Euler method} \\ \mathbf{z}_k - \mathbf{z}_{k-1} &= \frac{1}{2} h (\hat{g}(\mathbf{z}_k, \mathbf{u}_k; \boldsymbol{\xi}) + \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi})), \text{ trapezoidal rule} \\ \mathbf{z}_k - \mathbf{z}_{k-1} &= h \left( \frac{5}{12} \hat{g}(\mathbf{z}_k, \mathbf{u}_k; \boldsymbol{\xi}) + \frac{8}{12} \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}) - \frac{1}{12} \hat{g}(\mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}) \right). \end{aligned} \quad (16)$$

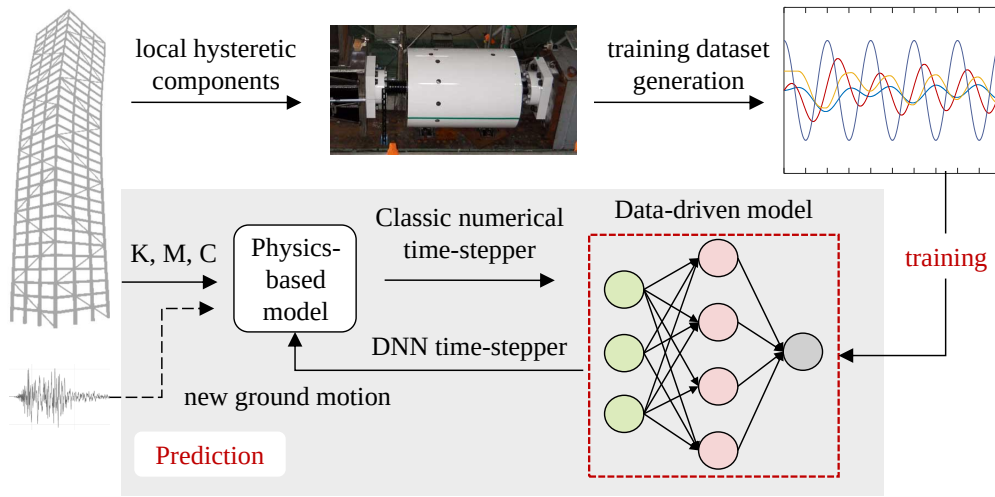
As seen above, Moulton methods are all implicit methods and one needs to refer to the iterative method to solve the implicit formula. This can be done using the fixed-point iteration strategy with an initial guess  $\mathbf{z}_k^{[0]} = \mathbf{z}_{k-1}$ ,

$$\begin{aligned} \mathbf{z}_k^{[j+1]} &= \mathbf{z}_{k-1} + h \hat{g}(\mathbf{z}_k^{[j]}, \mathbf{u}_k; \boldsymbol{\xi}), \quad j = 0, 1, 2, \dots \\ \mathbf{z}_k^{[j+1]} &= \mathbf{z}_{k-1} + \frac{1}{2} h (\hat{g}(\mathbf{z}_k^{[j]}, \mathbf{u}_k; \boldsymbol{\xi}) + \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi})), \\ \mathbf{z}_k^{[j+1]} &= \mathbf{z}_{k-1} + h \left( \frac{5}{12} \hat{g}(\mathbf{z}_k^{[j]}, \mathbf{u}_k; \boldsymbol{\xi}) + \frac{8}{12} \hat{g}(\mathbf{z}_{k-1}, \mathbf{u}_{k-1}; \boldsymbol{\xi}) - \frac{1}{12} \hat{g}(\mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}) \right). \end{aligned} \quad (17)$$

Similar to AB methods, we build the ResNet to approximate this process:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{im}(\mathbf{z}_{k-1}, \mathbf{u}_k; \boldsymbol{\xi}), \\ \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{im}(\mathbf{z}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}; \boldsymbol{\xi}), \\ \mathbf{z}_k &= \mathbf{z}_{k-1} + \mathcal{N}_{im}(\mathbf{z}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}, \mathbf{z}_{k-2}, \mathbf{u}_{k-2}; \boldsymbol{\xi}). \end{aligned} \quad (18)$$

$\mathcal{N}_{im}$  is the residual function of the implicit DNN time-stepper  $\mathbf{z}_{k-1} + \mathcal{N}_{im}$ . The demanding current state  $\mathbf{u}_k$  is achievable by the updating of system state  $\mathbf{x}_k$  based on **Eq. (12)**, where  $\mathbf{u}_k = \mathbf{H}\mathbf{x}_k$ . One can find that unlike the implicit AM methods, extra iterative computation for solving the equation involving both the current state and the previous state is not required in the implicit DNN time-stepper, the process of which can be automatically learned by a successful trained network. This leads to exceptional efficiency for solving the nonlinear problem.



**Figure 2** Physics-DNN hybridized integration time-stepping scheme

### 2.2.3 Consistency and Stability

The basic idea of this method is to generate the future states of the nonlinear dynamic system with two different time-steppers: exponential integration stepper for the displacement and velocity states  $x$  and DNN flow map stepper for the augmented nonlinear restoring force states  $z$ . We call this discrete time-stepping scheme a *Physics-DNN hybridized integration time-stepper* (PDHI time-stepper) in this research, since this time-stepper treats the easily obtained linear physics information of the structure using classical numerical integration method as well as the obscure nonlinear dynamics by leveraging features of the data-driven method simultaneously. The proposed algorithm is

---

**Algorithm 1:** PDHI time-stepper

---

**Requirements:** Provide time-invariant mass, damping, stiffness, connection and location selection matrices  $M, C, K, S_p, T$  of the structural system, and the trained neural network  $\mathcal{N}$  for the nonlinear components

**Input** : Given initial states  $x_0, z_0$  and the external force vector  $p_k = p(t_k) = p(kh)$ , for  $k = 0, 1, \dots, l$

- 1 Construct the state-space matrices  $A, B_z, B_p$  by using  $M, C, K, S_p$  and  $T$ ;
  - 2 **for**  $k \leftarrow 1$  **to**  $l$  **do**
  - 3      $x_k \leftarrow Ax_{k-1} + B_z z_{k-1} + B_p p_{k-1}$ ;
  - 4     **if** *Explicit DNN time-stepper* **then**  $z_k = z_{k-1} + \mathcal{N}_{ex} \leftarrow \text{Eq. (15)}$ ;
  - 5     **else** *Implicit DNN time stepper*,  $z_k = z_{k-1} + \mathcal{N}_{im} \leftarrow \text{Eq. (18)}$ ;
  - 6 **end**
- 

summarized in **Algorithm 1** and **Figure 2**. In contrast to classic numerical methods, the theoretical properties of the PDHI time-stepper, e.g, consistency and stability, are inextricably bound up with twofold equations: the exponential integrator in **Eq. (12)**, and the DNN integrator in **Eq. (15)** or **Eq. (18)**. We discuss several important aspects for accurate and stable approximation when using the PHDI time-stepper in this section:

- (a) *Consistency*: It is known that the  $M$ -step AB methods are  $M$ th-order methods, i.e., the local truncation error attains order  $\mathcal{O}(h^{M+1})$  while the total accumulated error is on the order of  $\mathcal{O}(h^M)$ , and the  $M$ -step AM methods are  $M + 1$ th-order methods [20]. Although the DNN-based time stepper resembles the linear multistep methods, it does not belong to the  $M/M + 1$ th-order methods. With ground truth dataset, the approximation error only comes from the network training algorithm. In an appropriate training process, **Eqs. (15)** and **(18)** can result in the exact future state for the system. For comprehensive study about the error bound for the solution using DNN as time stepper, the reader is referred to [21].
- (b) *Stability*: On one hand, the linear part of the differential equation is integrated exactly through the exponential integration, which can help to mitigate the stiffness phenomenon. On the other hand, according to the second Dahlquist barrier [22], the explicit AB methods have a bounded stability region and can never be A-stable, while the implicit AM methods can only be A-stable if their order is at most two ( $M = 1$ , the trapezoidal rule). We will show that the AM-based DNN time stepper achieves much better stability than the AB-based DNN time

stepper in the following work and among all the AM family, the trapezoidal rule based DNN time stepper works best, which is consistent with the stability properties of the linear multistep methods. It is worth mentioning that the stability characteristics of the DNN time stepper found herein are only considered as an empirical evidence. A systematic mathematical investigation is beyond the scope of this paper.

### 3 Deep Residual Learning

#### 3.1 ResNet

As mentioned in Section 2.2, since there is a term of  $z_{k-1}$  at the right-hand side of every equation in Eqs. (15) and (18), we choose to employ the residual neural network (ResNet) to model the increment between time steps  $z_{k-1}$  and  $z_k$  to speed up learning process by reducing the impact of vanishing gradients. ResNet was originally presented in [23] for image analysis in 2015 and received considerable prominence since. As shown in Figure 3, the idea of ResNet is to introduce a shortcut connection, namely *identity operator*, as well as a residual function  $\mathcal{N}$ , where the subscript *ex* and *im* from Eqs. (15) and (18) respectively are removed to simplify notation. We explicitly let a few stacked layers as the basic building block to approximate the residual mapping  $\mathcal{N}$  to  $z_k - z_{k-1}$ , instead of a full mapping  $\mathcal{F}$  to  $z_k$ . The full mapping  $\mathcal{F}$  can be thus viewed as  $z_{k-1} + \mathcal{N}$ . Although the above two strategies are capable of asymptotically approximating the desired function, it has been proved that the ease of learning is different: the residual process adds neither extra parameter nor computational complexity but serves as a perfect evolution operator of the underlying equation.

For different DNN time-steppers, let  $\mathbf{i}_k$  be the input of the ResNet at discrete time step  $k$ . Considering both of Eqs. (15) and (18), the general expression of  $\mathbf{i}_k$  can be given by,

$$\mathbf{i}_k = \{z_{k-M:k-1}, \mathbf{u}_{k-M:k-r}; \xi\} \quad (19)$$

where  $z_{k-M:k-1} = \{z_{k-M}, \dots, z_{k-1}\}$  and  $\mathbf{u}_{k-M:k-r} = \{\mathbf{u}_{k-M}, \dots, \mathbf{u}_{k-r}\}$ .  $r$  for the external inputs  $\mathbf{u}$  is associated with the type of the DNN time-stepper:  $r = 1$  for  $\mathcal{N}_{ex}$  and  $r = 0$  for  $\mathcal{N}_{im}$ .  $M$  has the same definition as in Eq. (13). For  $z$ ,  $z_{k-M:k-1} = \{z_{k-1}\}$  when  $k - M > k - 1$ . The output of the ResNet  $\mathbf{o}_k$  comes from  $z_{k-1} + \mathcal{N}$  for all different DNN time-steppers.

The form of the residual function  $\mathcal{N}$  is flexible in this research. We investigate the standard multilayer perceptron (MLP) network, the convolutional neural network (CNN) and the recurrent neural network (RNN) as the basic building block for the ResNet, respectively. Each of the network architecture is discussed in detail in the following section and compared in terms of modeling accuracy and computational efficiency in Section 4.1.

#### 3.2 Basic building block

##### 3.2.1 Multilayer perceptron

A *multilayer perceptron* (MLP), a class of feedforward artificial neural network, is the simplest neural network, which applies linear and nonlinear transformations to the inputs recursively. For a basic building block of the residual function  $\mathcal{N}$  that uses a total of  $L$  layers, let  $\mathbf{h}_k^l$  be the intermediate state at  $l$ th layer at time  $t_k$ , the MLP is recursively defined using the following equations:

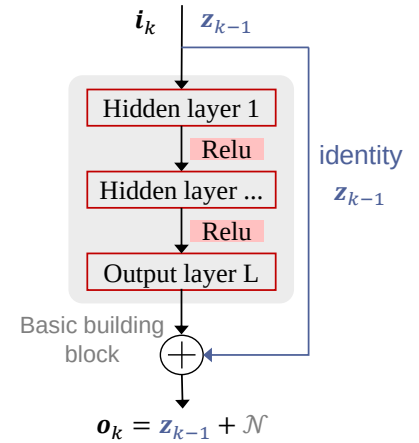
$$\text{hidden layer: } \mathbf{h}_k^l = \sigma(\mathbf{W}^l \mathbf{h}_k^{l-1} + \mathbf{b}^l), \text{ for } 1 \leq l \leq L-1 \quad (20a)$$

$$\text{output layer: } \mathbf{h}_k^L = \mathbf{W}^L \mathbf{h}_k^{L-1} + \mathbf{b}^L = \mathbf{o}_k \quad (20b)$$

where  $\mathbf{h}_k^{l-1} \in \mathbb{R}^{\mathcal{H}_{in}}$  and  $\mathbf{h}_k^l \in \mathbb{R}^{\mathcal{H}_{out}}$ .  $\mathbf{W}^l \in \mathbb{R}^{\mathcal{H}_{out} \times \mathcal{H}_{in}}$  and  $\mathbf{b}^l \in \mathbb{R}^{\mathcal{H}_{out}}$  are the weight matrix and bias vector respectively, and  $\sigma$  is the ReLU activation function [24] in this research. Note that  $\mathbf{h}_k^0$  is equal to  $\mathbf{i}_k$  in Eq. (19) for MLP.

##### 3.2.2 Convolutional neural network

Similar to the MLP, the *convolutional neural network* (CNN) also consists of several hidden layers and an output layer. However, inside the hidden layer, 1D-convolutional layers [25] are included to perform convolutions, which is



**Figure 3** Overview of the residual neural network

described as

$$\text{hidden layer: } \mathbf{h}_k^l = \sigma \left( \mathbf{W}_{K_l}^l \otimes \mathbf{h}_{K_l}^{l-1} + \mathbf{b}^l \right), \text{ for } 1 \leq l \leq L-1 \quad (21a)$$

$$\text{output layer: } \mathbf{h}_k^L = \mathbf{W}^L \mathbf{h}_k^{L-1} + \mathbf{b}^L = \mathbf{o}_k \quad (21b)$$

where  $\otimes$  is the convolution operator,  $\mathbf{W}_{K_l}^l$  is a set of  $\mathcal{H}_{in}$  one dimensional kernels with kernel size  $K_l$  used for extracting a new set of features  $\mathcal{H}_{out}$ , and  $\mathbf{b}^l \in \mathbb{R}^{\mathcal{H}_{out}}$ .  $\mathbf{W}^L$  and  $\mathbf{b}^L$  are identical to the output layer in the above typical MLP-based model. Note that the causal 1D-convolution is able to operate on the input before the current step, the convolutional computation in Eq. (21a) is thus given by,

$$\mathbf{W}_{K_l}^l \otimes \mathbf{h}_{K_l}^{l-1} = \sum_{\tau=0}^{K_l} \mathbf{W}_{\tau}^l \mathbf{h}_{k-\tau}^{l-1}. \quad (22)$$

Since the convolutional filter in Eq. (22) convolves states' history information naturally, when setting the kernel size  $K_l$  to the value of  $M$ ,  $\mathbf{h}_k^0$  for the CNNs only depends on the neighborhood states at one time step  $k$  or  $k-1$ , e.g.  $\mathbf{h}_k^0 = \{z_{k-1}, u_{k-1}\}$ ,  $K_l = 3$  for  $\mathcal{N}_{ex}$  with  $M = 3$ , and other inputs in Eq. (19) is going to be involved into the network by the convolutional filter automatically. This operation can be viewed as the simplest case of temporal convolutional networks (TCN) with zero padding and unit stride. For comprehensive study about the TCN model, the reader is referred to [26]. When the long-range temporal pattern in TCN is not consistent with the input format in  $\mathbf{i}_k$ , we set  $K_l = 1$  and  $\mathbf{h}_k^0 = \mathbf{i}_k$  for CNNs in this research. Note that the CNN-based model always results in a faster learning process than the fully connected MLP-based model given the same  $\mathcal{H}_{in}$  and  $\mathcal{H}_{out}$ , since each neuron in one layer connects to every neuron in the next layer in MLP while shares the same kernel weights in CNN.

### 3.2.3 Recurrent neural network

Among all neural network architectures, *recurrent neural networks* (RNNs) are natural candidates for sequence modeling: an RNN allows to keep track of time dependencies from the input sequences with the help of its hidden states. In many practical applications, the training process of RNN turns to be especially difficult due to the vanishing gradient problem and long short-term memory (LSTM) networks were invented to deal with this issue with the introduction of *gates* [27]. In this work, we employ the LSTM for the RNN-based model. The forms of the equations for the LSTM cell are expressed as:

$$\text{hidden layer: } \mathbf{f}_k^l = \sigma \left( \mathbf{W}_f^l \mathbf{X}_k^l + \mathbf{U}_f^l \mathbf{h}_{k-1}^l + \mathbf{b}_f^l \right) \quad (23a)$$

$$\mathbf{i}_k^l = \sigma \left( \mathbf{W}_i^l \mathbf{X}_k^l + \mathbf{U}_i^l \mathbf{h}_{k-1}^l + \mathbf{b}_i^l \right) \quad (23b)$$

$$\mathbf{o}_k^l = \sigma \left( \mathbf{W}_o^l \mathbf{X}_k^l + \mathbf{U}_o^l \mathbf{h}_{k-1}^l + \mathbf{b}_o^l \right) \quad (23c)$$

$$\tilde{\mathbf{C}}_k^l = \tanh \left( \mathbf{W}_c^l \mathbf{X}_k^l + \mathbf{U}_c^l \mathbf{h}_{k-1}^l + \mathbf{b}_c^l \right) \quad (23d)$$

$$\mathbf{C}_k^l = \mathbf{f}_k^l \circ \mathbf{C}_{k-1}^l + \mathbf{i}_k^l \circ \tilde{\mathbf{C}}_k^l \quad (23e)$$

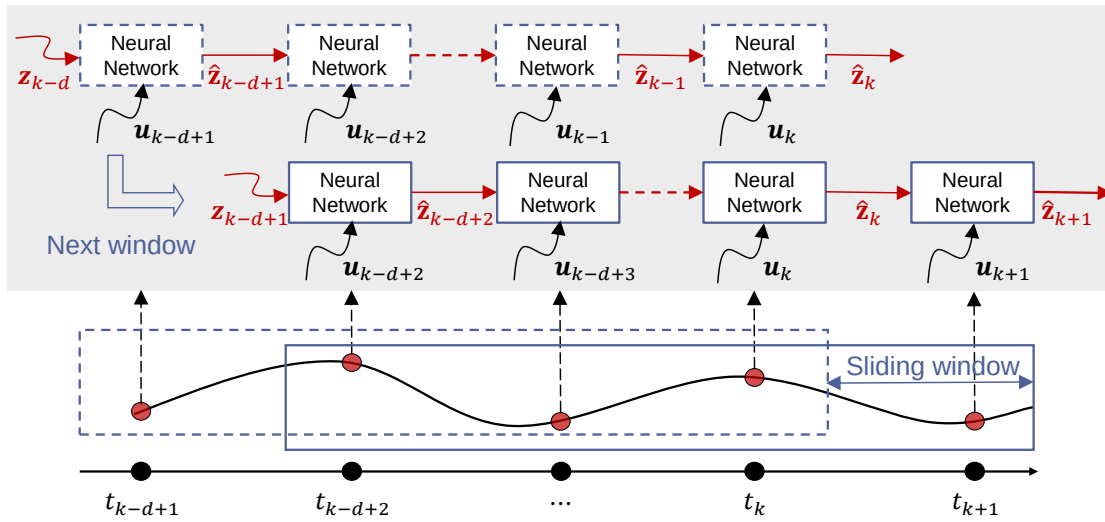
$$\mathbf{h}_k^l = \mathbf{o}_k^l \circ \tanh \left( \mathbf{C}_k^l \right), \quad 1 \leq l \leq L-1 \quad (23f)$$

$$\text{output layer: } \mathbf{h}_k^L = \mathbf{W}^L \mathbf{h}_k^{L-1} + \mathbf{b}^L = \mathbf{o}_k \quad (23g)$$

where  $\mathbf{f}_k^l, \mathbf{i}_k^l, \mathbf{o}_k^l \in \mathbb{R}^{\mathcal{H}_{out}}$  are the forget, input, output gate's activation vectors,  $\mathbf{W}_f^l, \mathbf{W}_i^l, \mathbf{W}_o^l, \mathbf{W}_c^l \in \mathbb{R}^{\mathcal{H}_{out} \times \mathcal{H}_{in}}$  and  $\mathbf{U}_f^l, \mathbf{U}_i^l, \mathbf{U}_o^l, \mathbf{U}_c^l \in \mathbb{R}^{\mathcal{H}_{out} \times \mathcal{H}_{out}}$  are weight matrices, and  $\mathbf{b}_f^l, \mathbf{b}_i^l, \mathbf{b}_o^l, \mathbf{b}_c^l \in \mathbb{R}^{\mathcal{H}_{out}}$  are biases.  $\mathbf{X}_k^l \in \mathbb{R}^{\mathcal{H}_{in}}$  is the input states and  $\mathbf{C}_k^l \in \mathbb{R}^{\mathcal{H}_{out}}$  is the cell states for the LSTM. The input states  $\mathbf{X}_k^0$  is set to  $\mathbf{i}_k$  in Eq. (19) in this research. To clearly understand the history information encoded effects in LSTM, we simplify Eq. (23) into the following expression

$$\mathbf{o}_k = f_{LSTM}(\mathbf{i}_k, \mathbf{h}_{k-1}, \mathbf{C}_{k-1}) \quad (24)$$

where  $f_{LSTM}$  is the LSTM cell function which maps  $\mathbf{i}_k, \mathbf{h}_{k-1}, \mathbf{C}_{k-1}$  into  $\mathbf{o}_k$ , and  $l$  is omitted for simplicity's sake. One can know from Eq. (24) that, unlike MLP-based and CNN-based model, the current output  $\mathbf{o}_k$  from the LSTM cell does not only depend on the the current input states  $\mathbf{i}_k$  but also the previous intermediate/hidden states  $\mathbf{h}_{k-1}$  and cell states  $\mathbf{C}_{k-1}$ , and the entire sequences of  $\mathbf{h}_k$  and  $\mathbf{C}_k$  are calculated and stored by the LSTM inherently, given the initial value  $\mathbf{h}_0 = 0$  and  $\mathbf{C}_0 = 0$ .



**Figure 4** AR process during training using a window with length  $d$  (implicit DNN time-stepper,  $M=0$ )

### 3.3 Training and prediction strategy

To achieve success long-range prediction for sequence analysis in MLP-based and CNN-based models, we introduce the autoregressive (AR) process in *training* by adding an AR sliding window with width  $d$ , as shown in **Figure 4**. Within each window, the output of the nonlinear restoring force  $z$  from the previous time steps is fed back to the input of the current time step sequentially. Only the input of  $z$  at several starting steps in the window is given by the labeled ground truth data. To be more specific, taking the implicit DNN time-stepper ( $M=0$ ) for example, the joint probability of a set output  $\hat{\mathbf{z}} = \{\hat{z}_{k-d+1}, \dots, \hat{z}_{k-1}, \hat{z}_k\}$ , given an initial input  $z_{k-d}$  and an additional external input  $\mathbf{u} = \{u_{k-d+1}, \dots, u_{k-1}, u_k\}$ , is factorized as a product of conditional probabilities as follows:

$$p(\hat{\mathbf{z}}|\mathbf{u}) = \prod_{i=k-d}^{k-1} p(\hat{z}_{i+1}|\hat{z}_i, u_{i+1}), \quad (25)$$

where  $\hat{z}_{k-d} = z_{k-d}$  and the hat symbol is used to denote neural network's estimated output values in this paper. We show that only a small size AR window ( $d = 10 \sim 30$ ) employed can make the network model possible to estimate long-range prediction. Since one time sequence with  $N$  steps can generate  $N + 1 - d$  AR widows/batches, the fact that these batches can be effectively parallelized speeds up operations in the training process.

Since RNNs are specifically designed to deal with sequential data and have the feed back connections inherently, the AR process is not included in LSTM's training. However, a window with limited length  $d$  is still applied, which means that LSTM can only capture dependencies up to  $d$  previous times steps during the training. The efficiency of unfolding the LSTM  $d$  time-steps in the past and ignoring dependencies longer than  $d$  has already been proved in other reference, c.f.[28]. To be more specific, by unfolding the LSTM  $d$  time-steps in the past, **Eq. (24)** changes into

$$\mathbf{o}_k = f_{LSTM}(\mathbf{i}_k, \mathbf{h}_{k-1}, \mathbf{C}_{k-1}) = \mathcal{F}_{LSTM}(\mathbf{i}_k, \mathbf{i}_{k-1}, \dots, \mathbf{i}_{k-d+1}, \mathbf{h}_{k-d}, \mathbf{C}_{k-d}) \quad (26)$$

where  $\mathcal{F}_{LSTM}$  is iteratively constructed via local function  $f_{LSTM}$ . In each window, we ignore the sequence dependency longer than  $d$  and thus assume  $(\mathbf{h}_{k-d}, \mathbf{C}_{k-d}) = 0$ .

As for the network *prediction* task in **Algorithm 1**, MLP-based, CNN-based, and RNN-based models all follow the same pattern. In the implicit DNN time-stepper ( $M=0$ ), for instance, the prediction process takes the following form:

$$p(\hat{\mathbf{z}}|\mathbf{u}) = \prod_{i=0}^{N-1} p(\hat{z}_{i+1}|\hat{z}_i, u_{i+1}). \quad (27)$$

where  $\mathbf{h}_0$  and  $\mathbf{C}_0$  in LSTM are initialized to zero before the prediction. With the initial condition  $z_0$  at  $t = 0$ , the prediction process runs sequentially: after each sample  $z$  is predicted, it is fed back into the network to predict the next sample.

The loss function in a time series prediction problem evaluates the dissimilarity between the predicted values from the PDHI time-stepper which includes the network and the actual measures. The mean square error (MSE) is adopted in this research, i.e.,

$$\mathcal{L}_{MSE}(\theta_j) = \frac{1}{l \times m} \|\hat{\mathbf{z}}(\theta_j) - \mathbf{z}\|_2^2 \quad (28)$$

where  $\theta_j$  stands for all network parameters at epoch  $j$ . It should be noted that practically, a common approach consists in computing the update  $\theta_j \rightarrow \theta_{j+1}$  over  $N_b$  batches of the training data. Suggestions for the size of the batch can be found in [29]. Additionally, we use the ADAM optimizer [30], an extension to stochastic gradient descent (SDG) that computes adaptive learning rates  $\eta$  for different parameters using estimates of first and second moments of the gradients for all networks' training process in this work. The weight updating equation is written as

$$\begin{aligned} m_j &= \beta_1 m_{j-1} + (1 - \beta_1) \nabla \mathcal{L}_{MSE}(\theta_j) \\ v_j &= \beta_2 v_{j-1} + (1 - \beta_2) \nabla \mathcal{L}_{MSE}(\theta_j)^2 \\ \hat{m}_k &= \frac{m_k}{1 - \beta_1^j}, \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^j} \\ \theta_{j+1} &= \theta_j + \frac{\eta}{\sqrt{\hat{v}_k + \epsilon}} \hat{m}_k. \end{aligned} \quad (29)$$

Good default settings for ADAM are  $\eta = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-7}$ . Basically, the training process for neural networks always consists of two steps: (i) the loss function  $\mathcal{L}_{MSE}$  is calculated with weights  $\theta_j$  given the training data in **Eq. (28)** (forward pass), and (ii) its 1st moment  $\nabla \mathcal{L}_{MSE}(\theta_j)$  and 2nd moment  $\nabla \mathcal{L}_{MSE}(\theta_j)^2$  are back-propagated through the network to update the parameters in **Eq. (29)** (backward pass). All of the networks are built and trained using the deep learning open-source library Pytorch [31].

## 4 Numerical investigation

In this section, we examine the proposed PDHI time-stepper for structural seismic response prediction by a numerical example of a base-isolated seven-story shear building as shown in **Figure 5** (a). This benchmark numerical example can also be found in other references (c.f [5, 18]). Properties of this model are: story mass  $m = 2 \times 10^3$  kg, inter-story elastic stiffness  $k = 1 \times 10^6$  N/m, and 1% inherent damping ratio of the primary structure. Nonlinear hysteretic additional damping with Bouc-Wen model (red curve in **Figure 5** (b)) is incorporated into the isolation story, which means that  $\mathbf{T} = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ . The damping force is described by

$$\begin{aligned} \dot{z} &= \frac{1}{\eta} \left[ A\dot{u} - \nu(\beta z |z|^{l-1} |\dot{u}| + \gamma |z|^l \dot{u}) \right] \\ &:= g(z, u, \dot{u}, \boldsymbol{\xi}); \quad \boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4]^T = \left[ \frac{A}{\eta}, \frac{\nu\beta}{\eta}, \frac{\nu\gamma}{\eta}, l \right]^T \\ &= \xi_1 \dot{u} - (\xi_2 z |z|^{\xi_4-1} |\dot{u}| + \xi_3 |z|^{\xi_4} \dot{u}) \end{aligned} \quad (30)$$

where the parameter  $A$  determines the hysteretic amplitude,  $\beta, \gamma$  and  $l$  control the basic hysteretic shape of the nonlinear model and  $\nu, \eta$  are degradation shape functions. The construction of DNN time-stepper is inherently problem dependent. In this example, we consider two common cases encountered in seismic prediction:

- Case 1: The damping displacement-force dataset is constructed from a single set of parameters, where  $\boldsymbol{\xi} = [2.45 \times 10^6, 0.0123, 0.0061, 2]^T$ . In this case,  $\boldsymbol{\xi}$  is same for all dataset and thus not included as a input feature of the neural network.
- Case 2: The damping displacement-force dataset is constructed from multiple sets of parameters, where  $\xi_1$  is allowed to vary between  $1.225 \times 10^6$  and  $2.675 \times 10^6$ , and  $\xi_2$  is allowed to vary within the range  $[6.15 \times 10^{-3}, 0.0185]$ . The values of  $\xi_3$  and  $\xi_4$  are the same as Case 1. This case is relevant to multi-query applications, e.g. seismic optimal design and sensitivity analysis.

### 4.1 Case 1: Training and prediction with single set of parameters

Training and validation data is collected from 20 harmonic displacement trajectories  $u = A \cos(2\pi ft)$  with random frequencies  $f$  and random amplitudes  $A$ . The data is generated using Latin hypercube sampling (LHS) statistical method. The values of amplitude  $A$  are randomly distributed within the interval  $[0, 50]$  mm and the values of frequency  $f$  are randomly distributed with the interval  $[0.3, 1]$  Hz for ten trajectories and  $[1, 5]$  Hz for the other ten trajectories, respectively. The former frequency range is designed to recover damper's seismic behavior in long period and the latter covers the domain where high frequency is more prominent. Corresponding damping forces are then calculated according to **Eq. (30)** through standard Runge-Kutta time-stepping schemes. All the data used for training and validation in this case is depicted in **Figure 6**.

For each trajectory, the time span is two seconds and the sampling rate is 200Hz, i.e., a total of  $2 \times 200 = 400$  time steps. As a result, the sizes of the input and output dataset are (*sequences\_number*, *time\_steps*, *features*)

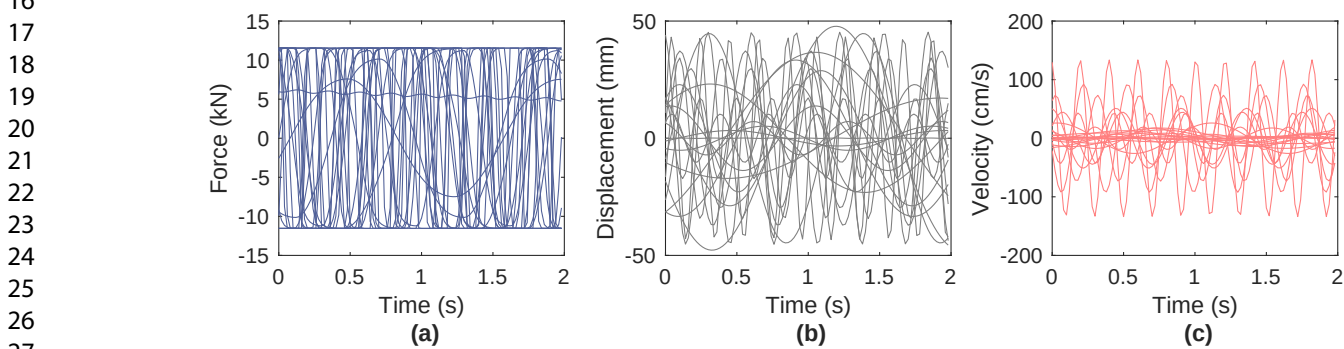
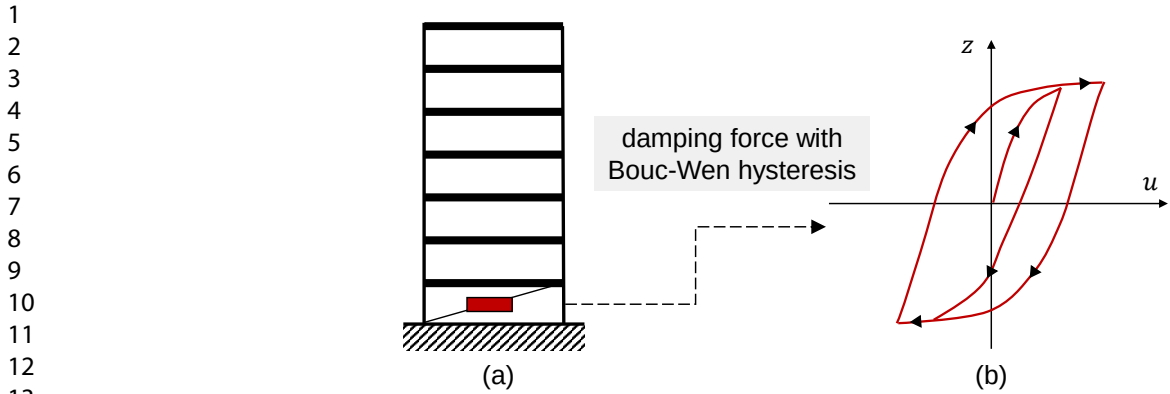


Figure 6 Noise free ground truth data for training and validation (20 trajectories for each feature)

$= (20 \times 400 \times 3)$  and  $(20 \times 400 \times 1)$ , taking the implicit DNN time-stepper ( $M=0$ ) for example, where the input features are  $\mathbf{i}_k = \{z_{k-1}, u_k, \dot{u}_k\}$  and the output feature is  $\mathbf{o}_k = z_k$ . All of the features have been rescaled to the range  $[-1, 1]$  before training. An *Auto-Regressive-Window* with width  $d = 20$  is applied into each sequence in this case. For each trajectory with 400 time steps, 381 AR windows are generated and 20 trajectories create  $381 \times 20 = 7620$  batches in total. The inputs and outputs shapes of the network turn to be  $(batch, window\_width, features) = (7620 \times 20 \times 3)$  and  $(7620 \times 20 \times 1)$  consequently. The dataset is then shuffled along its first dimension and we use a 90%-10% split for training and validation dataset.

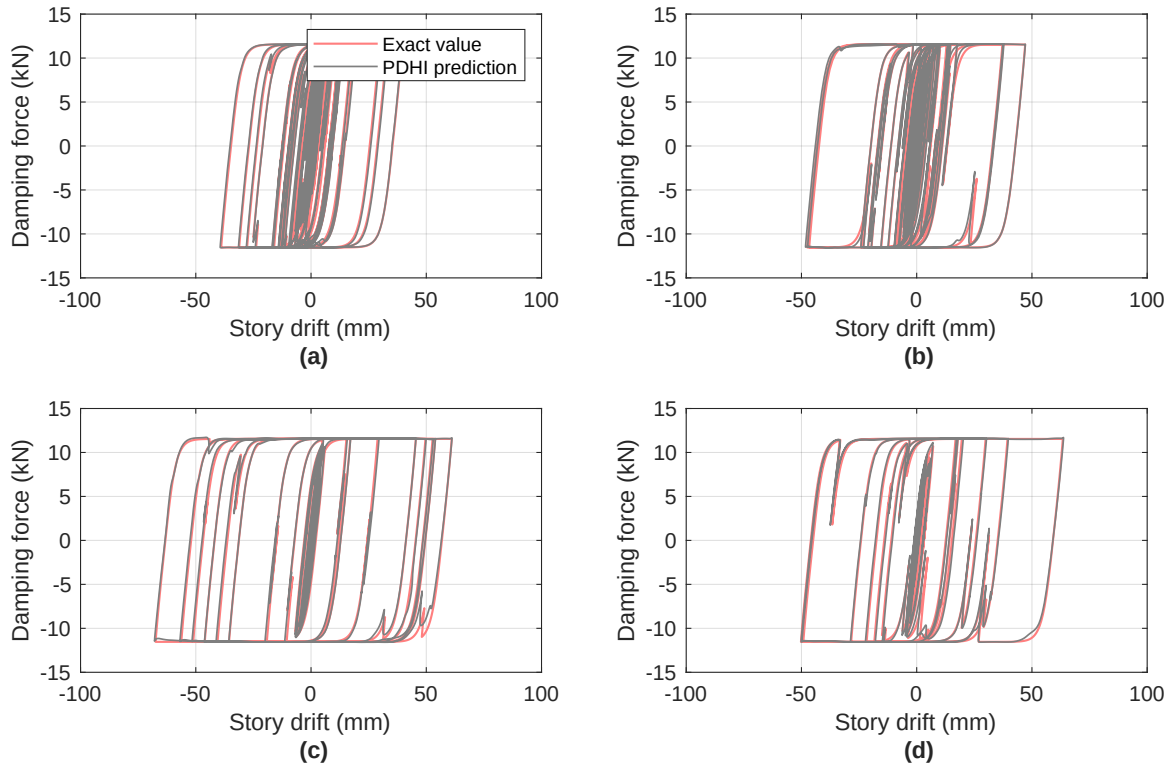
The DNN time-stepper based on the MLP model consists of two fully connected hidden layers of size 32-32 and one identity operator. The training dataset is divided into min-batches of size 32 and we train the DNN model for 200 epochs to ensure the convergence of the loss function. After the training finished, we march the trained network further for seismic prediction according to **Algorithm 1**. The earthquake records used for prediction are listed in **Table 1**. The relative damping force MSE between the network predicted data and ground truth for different member families of the DNN time-stepper are summarized in **Table 2**. Clearly, it can be found that although the ResNet is only trained using the data from short-term harmonic behaviors (total time  $t_N = 2s$ ), the proposed PDHI time-stepper achieves long-term stability and accuracy in seismic prediction (total time  $t_N \approx 30s$ ) with all different families of DNN time-stepper, except for the case of Explicit ( $M=1$ ). It can be seen that implicit DNN time-steppers return more accurate results compared to explicit DNN time-steppers. This might due to the different stability properties of each family, as illustrated in Section 2.2.3. Among all of the members, the implicit ( $M=1$ ) DNN time-stepper performs best, probably because it approximates the trapezoidal rule, which has superior stability properties among all of the

Table 1 Descriptions of the excitation for seismic prediction

| No. | Excitation   | Duration | Original PGV (Rescaled factor) | Use        |
|-----|--------------|----------|--------------------------------|------------|
| 1   | El Centro-NS | 31.18s   | 34.91cm/s (1.43)               | Prediction |
| 2   | Taft-EW      | 27.19s   | 17.16cm/s (2.91)               | Prediction |
| 3   | Kobe-NS      | 20s      | 90.62cm/s (0.55)               | Prediction |
| 4   | Kumamoto-EW  | 30s      | 90.57cm/s (0.55)               | Prediction |

**Table 2** Relative MSE between the predicted and the ground truth for the damping force of Bouc-Wen model, for different member families of the DNN time-stepper, and different input ground motions

| Record             | El Centro-NS          | Taft-EW               | Kobe-NS               | Kumamoto-EW           | Mean                  |
|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Implicit ( $M=0$ ) | $1.50 \times 10^{-3}$ | $2.10 \times 10^{-3}$ | $8.60 \times 10^{-3}$ | $8.90 \times 10^{-3}$ | $5.28 \times 10^{-3}$ |
| Implicit ( $M=1$ ) | $1.20 \times 10^{-3}$ | $1.70 \times 10^{-3}$ | $1.30 \times 10^{-3}$ | $2.00 \times 10^{-3}$ | $1.55 \times 10^{-3}$ |
| Implicit ( $M=2$ ) | $1.10 \times 10^{-3}$ | $2.30 \times 10^{-3}$ | $4.30 \times 10^{-3}$ | $3.60 \times 10^{-3}$ | $2.83 \times 10^{-3}$ |
| Explicit ( $M=1$ ) | $7.00 \times 10^{-2}$ | $1.65 \times 10^{-1}$ | $2.11 \times 10^{-1}$ | $1.61 \times 10^{-1}$ | $1.52 \times 10^{-1}$ |
| Explicit ( $M=2$ ) | $1.30 \times 10^{-3}$ | $2.90 \times 10^{-3}$ | $2.50 \times 10^{-3}$ | $8.70 \times 10^{-3}$ | $3.85 \times 10^{-3}$ |
| Explicit ( $M=3$ ) | $2.20 \times 10^{-3}$ | $6.20 \times 10^{-3}$ | $4.00 \times 10^{-3}$ | $1.89 \times 10^{-2}$ | $7.83 \times 10^{-3}$ |

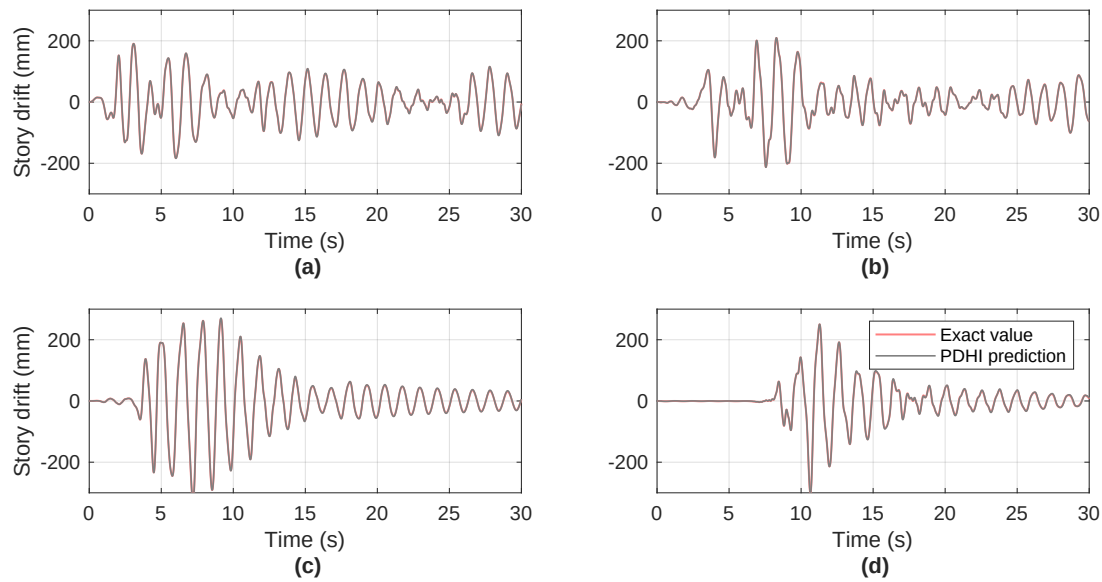


**Figure 7** Damping hysteretic behaviors for (a) El Centro-NS, (b) Taft-EW, (c) Kobe-KJM000, and (d) Kumamoto-EW (noise free)

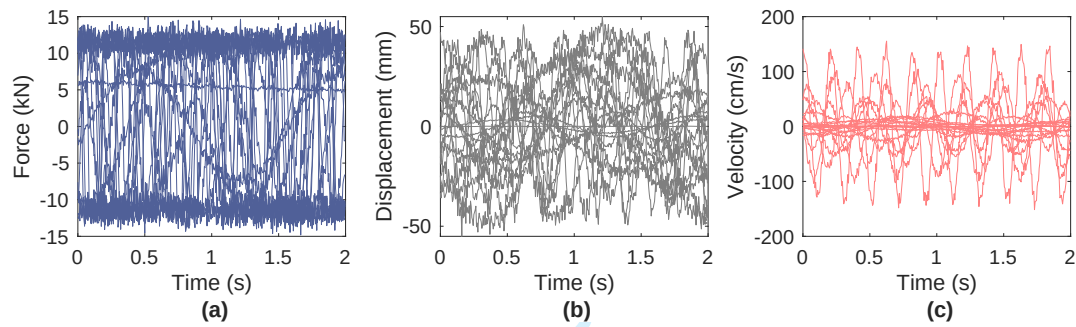
linear multistep methods. Moreover, the relatively poor performance of implicit ( $M=0$ ) or Explicit ( $M=1$ ) may imply underfitting due to fewer data to build an accurate model, while implicit ( $M=2$ ) or Explicit ( $M=3$ ) makes the model more prone to overfitting. Possible ways to improve the prediction accuracy for small  $M$  is to increase complexity of the neural network to make it sufficiently descriptive to approximate the flow map function for relatively large time step, while introduce regularization terms to simplified the model for large  $M$ . It turns out that the model built by implicit ( $M=1$ ) or Explicit ( $M=2$ ) results in a good fit. A typical predicted hysteretic loop for the isolation story, the displacement time history for the top story using implicit ( $M=1$ ) DNN time-stepper are additionally plotted in **Figures 7** and **8**, to further reveal its favorable performance.

To verify the robustness of the proposed algorithm, the original training and validation data is then polluted by a standard normally distributed random noise of 10% level, as shown in **Figure 9**. According to **Table 3**, it can be seen that the implicit family members could still lead to superior accuracy for the cases where window width  $d = 20 \sim 30$ , while the prediction results from the explicit family members are of inferior quality. Typical damping hysteretic behavior prediction results using PDHI time-stepper (Implicit ( $M=1$ )) with respect to 10% noise are reported in **Figure 10**.

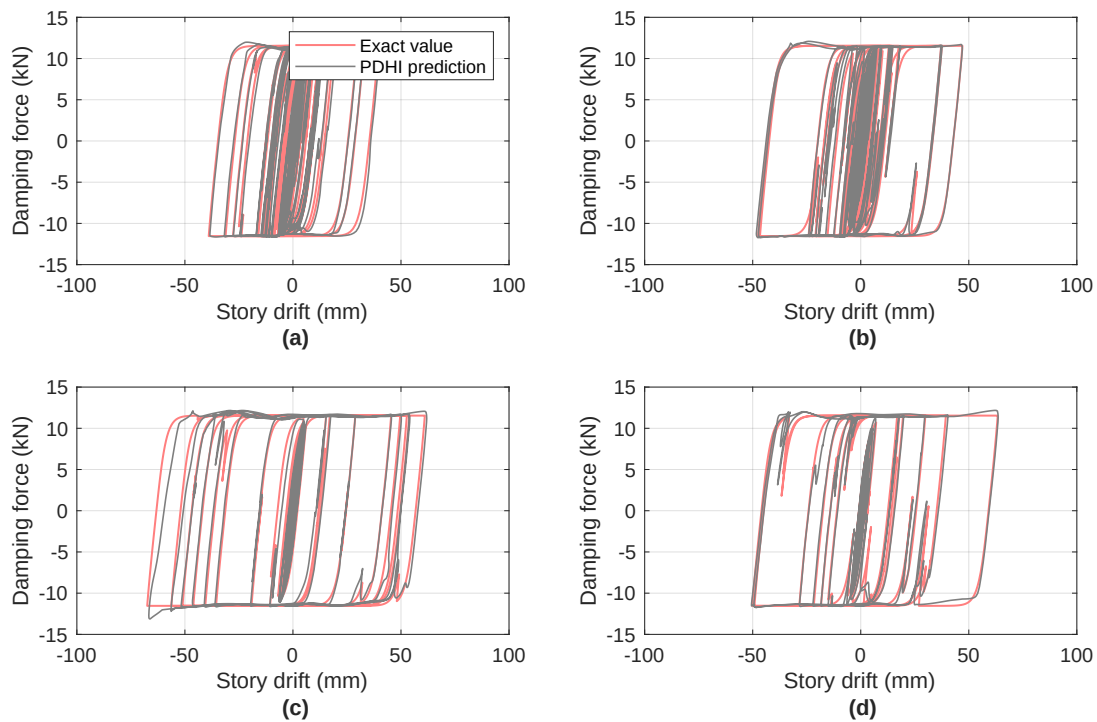
In table **Table 3**, we also study the efficiency of the autoregressive window presented in Section 3.3, with the width  $d$  ranging from 2 to 30 being investigated. It can be found that the performance of the proposed time-stepper is greatly affected by the existence of the autoregressive process, since large window width results in higher predictive accuracy. However, when the AR window is too long ( $d > 30$ ), the contribution from the AR window seems trivial.



**Figure 8** Top story drift prediction results for (a) El Centro-NS, (b) Taft-EW, (c) Kobe-KJM000, and (d) Kumamoto-EW



**Figure 9** Training and validation data with 10% noise (20 trajectories for each feature)



**Figure 10** Damping hysteretic behaviors for (a) El Centro-NS, (b) Taft-EW, (c) Kobe-NS, and (d) Kumamoto-EW (10% noise)

**Table 3** Relative MSE between the predicted and the ground truth for the damping force of Bouc-Wen model, for different noise levels and different window widths

| Window width | Implicit ( $M=1$ )    |                       | Explicit ( $M=2$ )    |                       |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|
|              | noise=0.0%            | noise=10.0%           | noise=0.0%            | noise=10.0%           |
| $d=2$        | $5.19 \times 10^{-2}$ | $5.93 \times 10^{-2}$ | $6.08 \times 10^{-2}$ | $2.46 \times 10^{-1}$ |
| $d=5$        | $2.60 \times 10^{-3}$ | $8.37 \times 10^{-2}$ | $1.59 \times 10^{-2}$ | $1.55 \times 10^{-1}$ |
| $d=10$       | $1.80 \times 10^{-3}$ | $3.18 \times 10^{-2}$ | $3.40 \times 10^{-2}$ | $1.97 \times 10^{-1}$ |
| $d=20$       | $1.55 \times 10^{-3}$ | $4.00 \times 10^{-3}$ | $3.85 \times 10^{-3}$ | $1.33 \times 10^{-1}$ |
| $d=30$       | $2.40 \times 10^{-3}$ | $7.20 \times 10^{-3}$ | $1.19 \times 10^{-2}$ | $8.44 \times 10^{-2}$ |

**Table 4** Descriptions of different neural network architectures

| Name  | 1st layer                         | Activation | 2nd layer            | Total parameters |
|-------|-----------------------------------|------------|----------------------|------------------|
| MLP   | Dense: 32 FC neurons              | ReLU       | Dense: 32 FC neurons | 3,041            |
| CNN   | Conv with $1 \times 2$ kernel: 32 | ReLU       | Dense: 32 FC neurons | 1,313            |
| LSTM1 | LSTM: 32 LSTM cells               | ReLU       | LSTM: 32 LSTM cells  | 13,217           |
| LSTM2 | LSTM: 16 LSTM cells               | ReLU       | LSTM: 16 LSTM cells  | 3,537            |

\* All of the networks have a FC output layer that transfer the dimension of the output of the 2nd layer to the desired number of output features, which is not mentioned in the above Table.

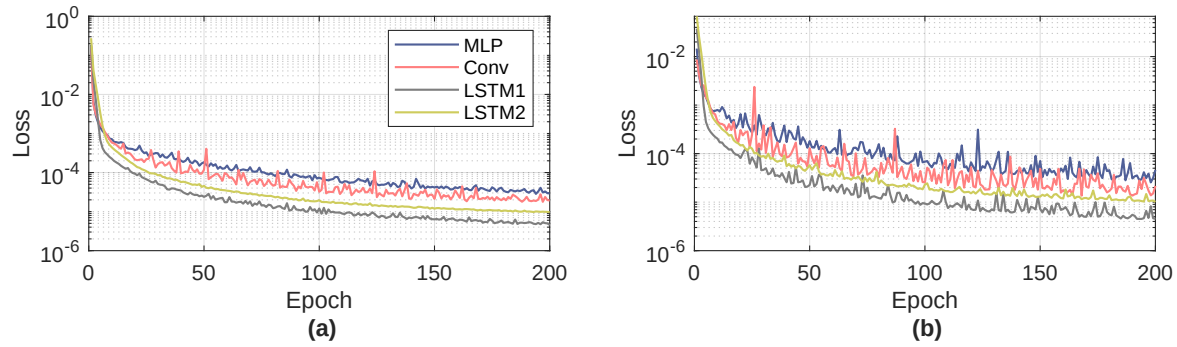
A too long AR window additionally degrades the training efficiency and might cause overfitting. As a result, a limited width of the AR window ( $d = 10 \sim 30$ ) is enough to secure the best long-term prediction capability for the DNN time-stepper. Finally, an empirical evaluation of different neural network architectures is conducted herein. Basic information of each architecture is referred to **Table 4**. The previously used multilayer perceptron (MLP) consists of two hidden layers and each layer has 32 fully connected (FC) neurons. It serves as a baseline model to be compared. To demonstrate the efficiency of CNN-based model, the first dense layer in MLP is replaced by a 1-D convolutional layer in the newly established CNN block. This makes the total number of parameters to be trained decrease from 3041 to 1313, which shortens the training process. Furthermore, two long short-term memory (LSTM) based recurrent neural networks (RNNs) are built: LSTM1 has 32 LSTM cells in each layer, and LSTM2 has a total number of parameters similar to MLP. **Figure 11** illustrates typical training and validation loss curves for different neural network architectures. Results are included in **Table 5**: by using the Implicit ( $M=1$ ) strategy, all models promptly fulfill the prediction accuracy. But obviously, the performance of LSTM is poorer than MLP and CNN, especially when noise level is high. Besides, LSTM has larger number of model parameters and requires more GPU memory for training. Our results indicate that both MLP and CNN with the autoregressive process are enough to be used and capable of capturing features of long-range patterns for hysteretic behaviors. These architectures tend to outperform the LSTM baseline in prediction accuracy and are over a magnitude faster to train.

## 4.2 Case 2: Training and prediction with multiple sets of parameters

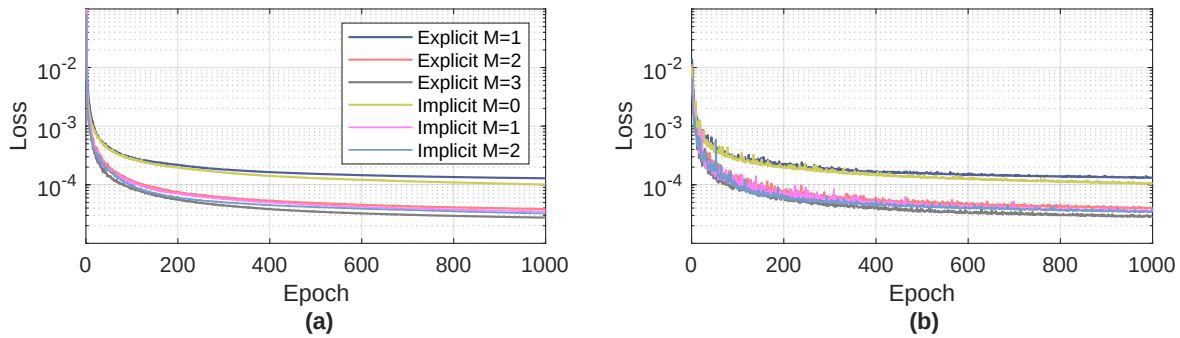
The dataset in Case 2 contains 60 harmonic displacement and damping force pairs for training and validation, the volume of which is larger than Case 1 since physical parameters are additionally taken as input features for neural networks. A 90%-10% split for training and validation dataset is applied in this case. Harmonic displacement data is also generated using LHS statistical method, where the values of  $\xi_1$  are randomly distributed within the interval  $[1.225 \times 10^6, 2.675 \times 10^6]$ ,  $\xi_2$  within the interval  $[6.15 \times 10^{-3}, 0.0185]$ , amplitude  $A$  within the interval  $[0, 50]$ mm, frequency  $f$  within the interval  $[0.3, 1]$ Hz for thirty trajectories and  $[1, 5]$ Hz for the other thirty trajectories, respectively. The time span is still two seconds with a sampling rate of 200Hz for each trajectory. The sizes of the input and output dataset are (*sequences\_number, time\_steps, features*) =  $(60 \times 400 \times 5)$  and  $(60 \times 400 \times 1)$ , taking the implicit DNN

**Table 5** Relative MSE between the predicted and the ground truth for the damping force of Bouc-Wen model, for different neural network architectures

| Implicit ( $M=1$ ) | MLP                   | CNN                   | LSTM1                 | LSTM2                 |
|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| noise=0.0%         | $1.55 \times 10^{-3}$ | $1.60 \times 10^{-3}$ | $1.70 \times 10^{-3}$ | $4.3 \times 10^{-3}$  |
| noise=10.0%        | $4.00 \times 10^{-3}$ | $3.10 \times 10^{-3}$ | $7.82 \times 10^{-2}$ | $3.54 \times 10^{-2}$ |



**Figure 11** Training and validation losses incurred during different neural networks training: (a) training, and (b) validation



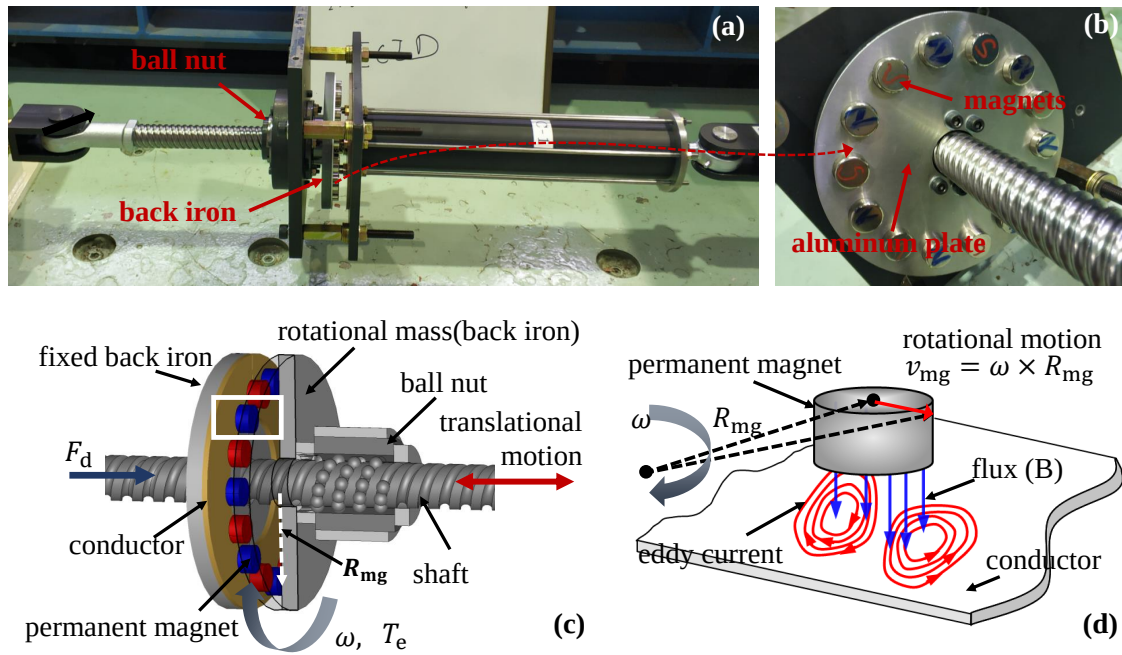
**Figure 12** Training and validation losses incurred during neural network training for Case 2: (a) training, and (b) validation

time-stepper ( $M=0$ ) for example, where the input features are  $\mathbf{i}_k = \{z_{k-1}, u_k, \dot{u}_k, \xi_1, \xi_2\}$  and the output feature is  $\mathbf{o}_k = z_k$ . AR window is of width  $d = 20$  and thus the inputs and outputs shapes of the network turn to be  $(batch, window\_width, features) = (22860 \times 20 \times 5)$  and  $(22860 \times 20 \times 1)$ .

The ResNet with the same MPL building block as the one used in Case 1 (Dense: 32 FC neurons in each layer, 2 layers in total) is trained and validated. Convergence results run over 1000 epochs are shown in **Figure 12**. The relative MSE between the predicted and the ground truth damping force is listed in **Table 6**. For each seismic prediction, different parameters for the Bouc-wen model are tested:  $[\xi_1, \xi_2] = [1.47 \times 10^6, 0.016]$  for El Centro-NS,  $[1.96 \times 10^6, 0.0185]$  for Taft-EW,  $[2.45 \times 10^6, 0.0086]$  for Kobe-KJM000, and  $[2.94 \times 10^6, 0.0135]$  for Kumamoto-EW. As it is presented, considering physical parameters and system states simultaneously as network's input requires more epochs/datasets for training and might result in relatively large predictive errors, compared to Case 1. However, since the errors are still relatively small (1%-3%) and the long-term dynamic behaviors are correctly captured, all of the DNN time-stepper members, except Explicit ( $M=1$ ), are deemed suitable for prediction. Again, the Implicit ( $M=1$ ) DNN time-stepper achieves the best performance among all the methods.

**Table 6** Case 2: relative MSE between the predicted and the ground truth for the damping force of Bouc-Wen model, for different member families of the DNN time-stepper, and different input ground motion

| Record             | El Centro-NS          | Taft-EW               | Kobe-NS               | Kumamoto-EW           | Mean                  |
|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Implicit ( $M=0$ ) | $4.70 \times 10^{-3}$ | $3.03 \times 10^{-2}$ | $5.88 \times 10^{-2}$ | $4.49 \times 10^{-2}$ | $3.47 \times 10^{-2}$ |
| Implicit ( $M=1$ ) | $1.02 \times 10^{-2}$ | $1.14 \times 10^{-2}$ | $1.64 \times 10^{-2}$ | $1.79 \times 10^{-2}$ | $1.40 \times 10^{-2}$ |
| Implicit ( $M=2$ ) | $1.36 \times 10^{-2}$ | $1.48 \times 10^{-2}$ | $4.76 \times 10^{-2}$ | $2.01 \times 10^{-2}$ | $2.40 \times 10^{-2}$ |
| Explicit ( $M=1$ ) | $9.10 \times 10^{-3}$ | $3.75 \times 10^{-2}$ | $4.03 \times 10^{-1}$ | $3.05 \times 10^{-1}$ | $1.89 \times 10^{-1}$ |
| Explicit ( $M=2$ ) | $2.30 \times 10^{-3}$ | $3.38 \times 10^{-2}$ | $2.00 \times 10^{-2}$ | $5.27 \times 10^{-2}$ | $2.72 \times 10^{-2}$ |
| Explicit ( $M=3$ ) | $6.80 \times 10^{-3}$ | $8.60 \times 10^{-3}$ | $1.93 \times 10^{-2}$ | $2.64 \times 10^{-2}$ | $1.53 \times 10^{-2}$ |



**Figure 13** (a) Photograph of IECD, (b) photograph of permanent magnets, (c) schematic of IECD, and (d) eddy current effect

**Table 7** Descriptions of earthquake records for training and prediction

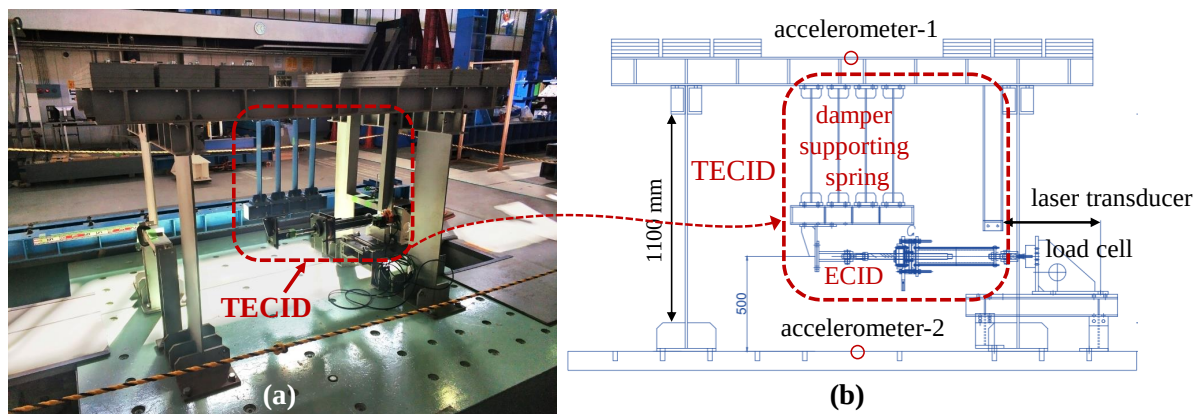
| No. | Excitation   | Duration | Original PGA (Rescaled factor) | Use                   |
|-----|--------------|----------|--------------------------------|-----------------------|
| 1   | Tohoku-NS    | 94.91s   | 3.33m/s <sup>2</sup> (0.39)    | Training & Prediction |
| 2   | Taft-EW      | 27.19s   | 1.76m/s <sup>2</sup> (1.03)    | Prediction            |
| 3   | Kobe-NS      | 20s      | 8.18m/s <sup>2</sup> (0.19)    | Prediction            |
| 4   | Hachinohe-NS | 25.19s   | 2.30m/s <sup>2</sup> (0.51)    | Prediction            |

## 5 Experimental demonstration

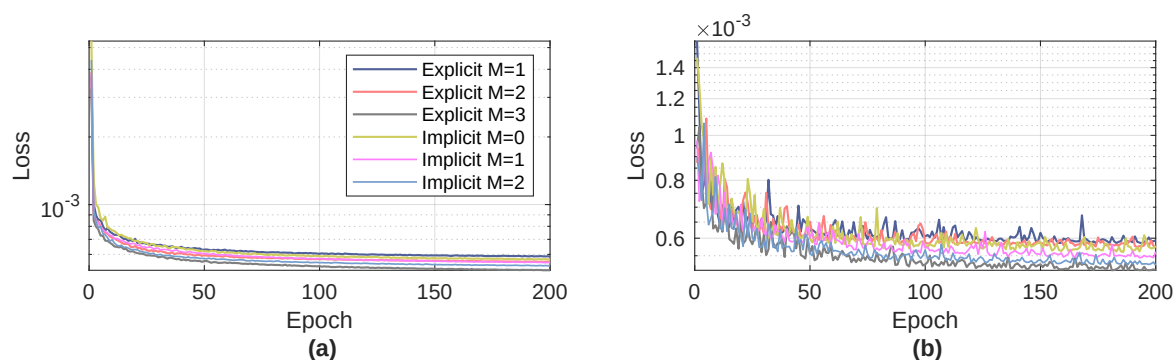
Eddy current damper is getting popular for reducing seismic and wind-induced vibrations in structural control due to its desirable non-contact and non-degradation characteristics [32]. In the damper, eddy currents are generated by the relative movement of the magnets against conductor. The eddy currents then induce a magnetic field, which results in the repulsive electromotive damping force. To enhance the energy dissipation of the damper, a flywheel attached to the ball screw, which translates linear motion to high-speed rotational motion, is additionally incorporated into the damper. By doing this, substantial energy dissipation can be achieved by the damping magnification as well as the mass amplification effects of the damping device. Detailed configuration of the inerter eddy current damper (IECD) used in this experiment is illustrated in **Figure 13**. To address the nonlinear behavior of damper, series of harmonic excitation tests, finite-element analysis of electromagnetic problems, and parametric study are always involved in previous research, while the nonlinear dynamics of eddy current damper in this research is reproduced by the proposed DNN time-stepper only, using limited damping response dataset measured from earthquake excitation directly.

Dynamic tests of a single-degree-of-freedom (SDOF) benchmark structure equipped with a tuned inerter eddy current damper (TIECD) were conducted at the structural laboratory in Tohoku University, as shown in **Figure 14**. The IECD is tuned according to the fixed-point method with optimum mass ratio, additional damping ratio and supporting spring stiffness. The stiffness of the SDOF structure  $k_s$  is 277kN/m and the mass  $m_s$  is 2200kg. Damping ratio  $h_s$  of the primary structure is set to be 0.1%, where  $c_s = 2h_s\sqrt{k_s m_s}$ . For more details on this SDOF model, the reader is referred to [33, 34]. The fundamental period of the SDOF system is 0.54s, and we compressed the time step interval of input ground motion to 0.5 times of the original data to simulate a structure with period of 1.08s. Four seismic records for the shaking table test were employed, as listed in **Table 7**. It should be pointed out that the Tohoku-NS record is split into two dataset: half for training and half for prediction. All of the other three earthquake records: Taft-EW, Kobe-NS and Hachinohe-NS are used for new predictions only.

The network structure of the DNN time stepper is same as the one employed in Section 4.1: a MLP-based



**Figure 14** (a) Photograph of the small-scale specimen with TIECD, and (b) schematic representation of the small-scale specimen with experimental measurements



**Figure 15** Training and validation losses incurred during neural network training: (a) training, and (b) validation

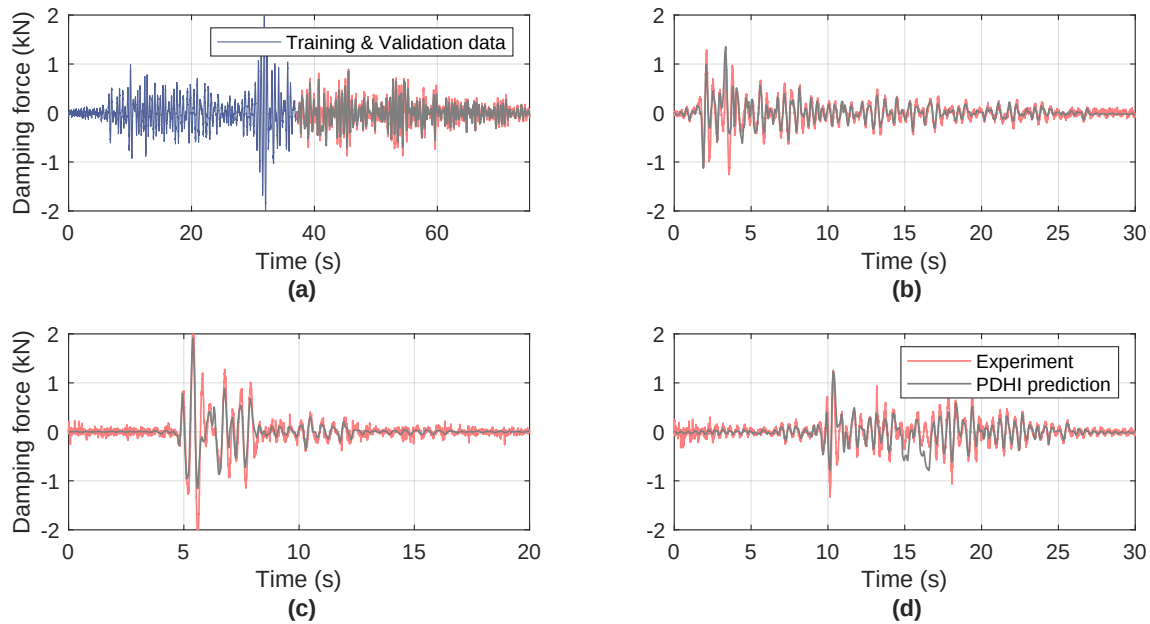
building block with two fully connected hidden layers of size 32-32 and one identity operator, as described in the first line in **Table 4**. Only the previous half time period of Thohoku-NS is used for training and its duration is about 37.11 seconds with a sampling rate of 1000Hz. The training and validation data thus has a total  $N = 37111$  time steps. The input and output dataset are of the shape (*sequences\_number*, *time\_steps*, *features*) =  $(1 \times 37111 \times 3)$  and  $(1 \times 37111 \times 1)$  for the implicit DNN time-stepper ( $M=0$ ), where the input features are  $\mathbf{i}_k = \{z_{k-1}, u_k, \dot{u}_k\}$  and output feature is  $\mathbf{o}_k = z_k$ . Note that  $z_k$  is TIECD's damping force measured by the load cell and  $u_k$  is the damper's dynamic displacement response measured by the laser sensor at time  $t_k$ , the experimental setup of which is depicted in detail in **Figure 14** (b). The velocity of the damper  $\dot{u}_k$  is attained by the numerical differentiation algorithm based on total-variation regularization in [35] from the sequence of  $u_k$  ( $k = 1, \dots, N$ ). Since one time sequence with  $N$  steps can generate  $N + 1 - d$  batches, with an AR window ( $d = 20$ ), the input and output shapes for the ResNet change into (*batch*, *window\_width*, *features*) =  $(37092 \times 20 \times 3)$  and  $(37092 \times 20 \times 1)$ , where the batch number is obtained by  $N + 1 - d = 37111 + 1 - 20 = 37092$ , according to Section 3.3. The training data is divided into mini-batches of size 32 and we typically train the model for 200 epochs and reshuffle the training data using a 90%-10% split in each epoch. Since the initial weights are randomly generated from Gaussian distributions, each network is trained 10 times with different initial weights to fairly evaluate the performance of the DNN model.

Typical training and validation losses incurred during neural network training are displayed in **Figure 15**. To investigate the performance of the data driven time-stepping schemes with respect to different linear multi-step methods, the relative mean square error between the predicted and real measured damping force is reported in **Table 8**. It is worthwhile to note that although the time-step size  $\Delta t = 0.001$ s is rather small in this example, the Adams-Moulton based implicit DNN time stepper with  $M=1$ , which is exactly referring to the trapezoidal rule, still seems to be the best choice. This might be due to the superior stability properties of the trapezoidal rule, as mentioned in Section 2.2.3.

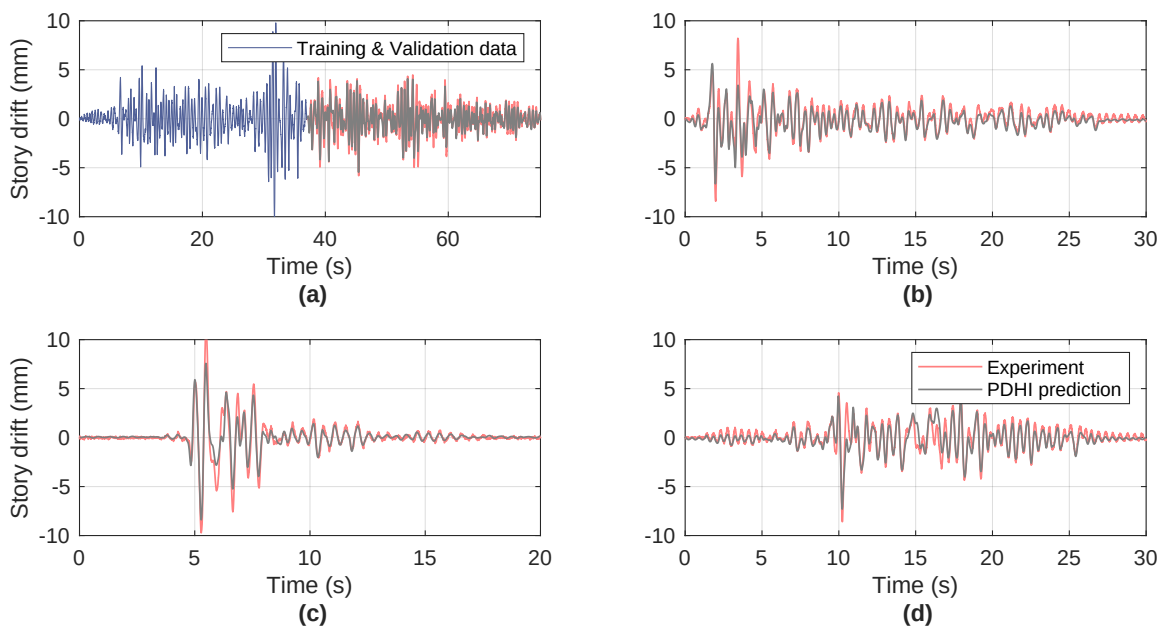
**Figures 16 - 18** depict the comparison between the experimental measured data and the PDHI time-stepper prediction. As expected, although the experiment data has a high level of noise, the PDHI time-stepper is still able to capture satisfied dynamic response of the system for new input excitation. The reason for the robustness of the

**Table 8** Relative MSE between the predicted and the real measurement data for the damping force of TIECD, for different member families of the DNN time-stepper, and different input ground motion

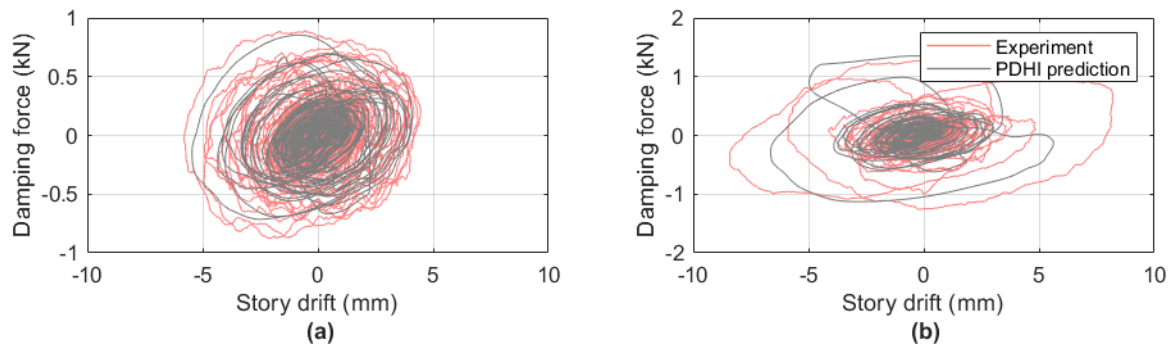
| Record             | Tohoku-NS (half)      | Taft-EW               | Kobe-NS               | Hachinohe-NS          | Mean                  |
|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Implicit ( $M=0$ ) | $1.19 \times 10^{-2}$ | $2.13 \times 10^{-2}$ | $1.16 \times 10^{-2}$ | $8.90 \times 10^{-3}$ | $1.34 \times 10^{-2}$ |
| Implicit ( $M=1$ ) | $8.10 \times 10^{-3}$ | $2.33 \times 10^{-2}$ | $1.04 \times 10^{-2}$ | $1.12 \times 10^{-2}$ | $1.33 \times 10^{-2}$ |
| Implicit ( $M=2$ ) | $1.99 \times 10^{-2}$ | $1.29 \times 10^{-1}$ | $7.31 \times 10^{-2}$ | $3.47 \times 10^{-2}$ | $6.42 \times 10^{-2}$ |
| Explicit ( $M=1$ ) | $1.32 \times 10^{-2}$ | $1.88 \times 10^{-2}$ | $8.30 \times 10^{-3}$ | $1.80 \times 10^{-1}$ | $5.50 \times 10^{-2}$ |
| Explicit ( $M=2$ ) | $5.90 \times 10^{-3}$ | $1.58 \times 10^{-1}$ | $8.90 \times 10^{-3}$ | $3.80 \times 10^{-1}$ | $1.38 \times 10^{-1}$ |
| Explicit ( $M=3$ ) | $8.50 \times 10^{-3}$ | $7.92 \times 10^{-2}$ | $9.70 \times 10^{-3}$ | $9.50 \times 10^{-3}$ | $2.67 \times 10^{-2}$ |



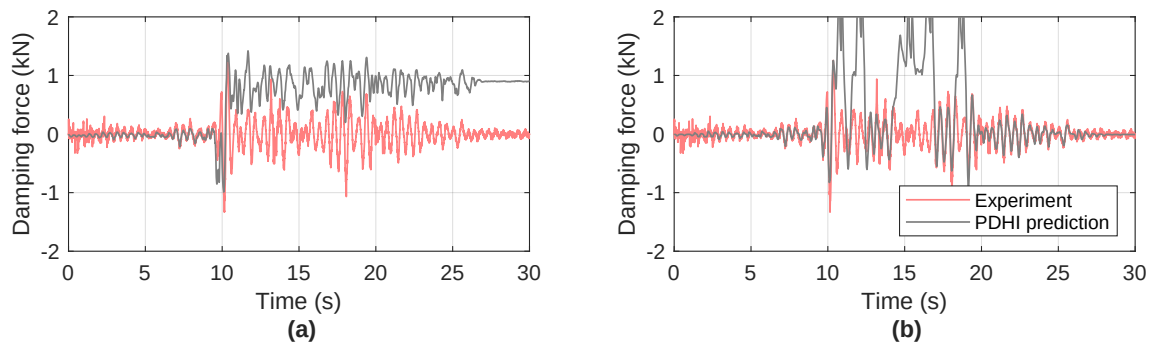
**Figure 16** Damping force prediction results for (a) half time period of Tohoku-NS, (b) entire time period of Taft-EW, (c) entire time period of Kobe-NS, and (d) entire time period of Hachinohe-NS



**Figure 17** Story drift prediction results for (a) half time period of Tohoku-NS, (b) entire time period of Taft-EW, (c) entire time period of Kobe-NS, and (d) entire time period of Hachinohe-NS



**Figure 18** Damping hysteretic behaviors for (a) half time period of Tohoku-NS, and (b) entire time period of Hachinohe-NS



**Figure 19** Damping force prediction results of the explicit DNN time-stepper ( $M = 1$ ) with large errors for (a) Taft-EW, and (b) Hachinohe-NS

proposed algorithm is mainly twofold. On one hand, we use the integral form of the dynamic system and thus require no knowledge of the time derivatives of the state equations, the process of which might introduce higher level of noise corrupting data. On the other hand, the AR window makes the algorithm not sensitive to temporal noise and helps regularize the neural network so as to achieve a best weighted solution of all possible outcomes during training. However, results with large errors (e.g. relative  $MSE > 5e-02$ ) might still happen among different randomly generated initial weights, especially for the scenarios utilizing explicit DNN time-stepper. A typical predicted result of this case is shown in **Figure 19**. Training the network using implicit DNN time-stepper ( $M \leq 1$ ) by multiple times with different initial weights and choosing the best predictive model accordingly is strongly suggested for the application of the proposed method where noise level is high.

## 6 Conclusion

In this study, a discrete Physics-DNN hybridized integration time-stepper is presented to compute the time evolution of nonlinear structural dynamic systems under seismic excitation. The basic idea of this method is to generate the future states of the nonlinear dynamic system with two different time-steppers: exponential integration stepper for the system's displacement and velocity states and DNN flow map stepper for the augmented nonlinear restoring force states. The DNN flow map stepper employs the ResNet to model the nonlinear dynamics and its input-output relation is derived from the general form of linear multistep method, where two different time-stepping schemes are proposed: the implicit DNN time-stepper corresponding to the Adams-Moulton methods and the explicit DNN time stepper corresponding to the Adams-Bashforth methods. Due to the generalized learning capability of neural networks, extra iterative computation for solving the traditional implicit methods can be circumvented. In most of the numerical and experimental examples, it is shown that the implicit DNN time-stepper returns more accurate results compared to explicit DNN time-steppers, which is consistent with the stability properties of the linear multistep methods empirically.

We additionally demonstrate that a simple feedforward/convolutional architecture for the ResNet with autoregressive process outperforms recurrent neural networks to fulfill the requirement of prediction accuracy and long-range prediction. The introduction of the autoregressive window with limited length plays an important role in regularizing the neural network to achieve a best weighted solution, and the direct input of the previous variables for the networks makes the whole training process more effective than the indirect input of hidden states in recurrent networks. Besides,

at training time, the conditional predictions for all different autoregressive windows do not have recurrent connections and thus can be made in parallel, which greatly speeds up the training process.

Note that the work in this manuscript considers the specific task where the linear properties of the system should be given and the spatial distribution of the nonlinearity is known a priori. Considering the system's linear variables trainable and the nonlinearities appear at any coordinate is possible, but addressing both of the identifiability and observability problems as well as maintaining enough physical interpretability for nonlinear systems is challenging. Despite these limitations, the current framework in this paper is intended to show that the proposed approach is an efficient and effective numerical solution option for structural dynamic analysis with unknown nonlinearities and may be worthy of further consideration.

More open questions motivate further investigations, e.g. systematic mathematical proofs for the stability of the proposed method, bringing the idea of adaptive step size into the DNN time-stepping schemes to result in a more efficient numerical integration, making the framework more flexible to incorporate partial physical knowledge considering more generalized cases, and using stochastic components in the neural network to improve the prediction performance.

## Acknowledgement

The authors acknowledge the support of the Japan Society for the Promotion of Science (JSPS). This research is funded by JSPS KAKENHI Grant Number 21K14284. The earthquake record from K-NET and KiK-net of Japanese NIED strong motion seismograph networks was used.

## References

- [1] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.
- [2] Hao Sun and Oral Büyüköztürk. Identification of traffic-induced nodal excitations of truss bridges through heterogeneous data fusion. *Smart Materials and Structures*, 24(7):075032, 2015.
- [3] Shunichi Koshimura. Fusion of real-time disaster simulation and big data assimilation—recent progress. *Journal of Disaster Research*, 12(2):226–232, 2017.
- [4] Rih-Teng Wu and Mohammad Reza Jahanshahi. Data fusion approaches for structural health monitoring and system identification: past, present, and future. *Structural Health Monitoring*, 19(2):552–586, 2020.
- [5] Li Wang, Jia Guo, and Izuru Takewaki. Real-time hysteresis identification in structures based on restoring force reconstruction and Kalman filter. *Mechanical Systems and Signal Processing*, 150:107297, 2021.
- [6] James L Beck and Siu-Kui Au. Bayesian updating of structural models and reliability using markov chain monte carlo simulation. *Journal of engineering mechanics*, 128(4):380–391, 2002.
- [7] Ellen Simoen, Guido De Roeck, and Geert Lombaert. Dealing with uncertainty in model updating for damage assessment: A review. *Mechanical Systems and Signal Processing*, 56:123–149, 2015.
- [8] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.
- [9] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [10] Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 369:113226, 2020.
- [11] Ruiyang Zhang, Zhao Chen, Su Chen, Jingwei Zheng, Oral Büyüköztürk, and Hao Sun. Deep long short-term memory networks for nonlinear structural seismic response prediction. *Computers & Structures*, 220:55–68, 2019.
- [12] Ahmed A Torky and Susumu Ohno. Deep learning techniques for predicting nonlinear multi-component seismic responses of structural buildings. *Computers & Structures*, 252:106570, 2021.
- [13] Zhilu Lai, Charilaos Mylonas, Satish Nagarajaiah, and Eleni Chatzi. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508:116196, 2021.
- [14] Yongjia Xu, Yifan Fei, Yuli Huang, Yuan Tian, and Xinzhen Lu. Advanced corrective training strategy for surrogating complex hysteretic behavior. In *Structures*, volume 41, pages 1792–1803. Elsevier, 2022.

- [15] Jia Guo, Li Wang, Iori Fukuda, and Kohju Ikago. Data-driven modeling of general damping systems by k-means clustering and two-stage regression. *Mechanical Systems and Signal Processing*, 167:108572, 2022.
- [16] Soheil Sadeghi Eshkevari, Martin Takáč, Shamim N Pakzad, and Majid Jahani. Dynnet: Physics-based neural architecture design for nonlinear structural response modeling and prediction. *Engineering Structures*, 229:111582, 2021.
- [17] Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, 215:110704, 2020.
- [18] Zhong-Rong Lu, Renzhi Yao, Li Wang, and Jike Liu. Identification of nonlinear hysteretic parameters by enhanced response sensitivity approach. *International Journal of Non-Linear Mechanics*, 96:1–11, 2017.
- [19] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.
- [20] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.
- [21] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.
- [22] Germund G Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- [25] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.
- [26] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.
- [27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [28] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.
- [29] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.
- [30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [32] Henry A Sodano and Jae-Sung Bae. Eddy current damping in structures. *Shock and Vibration Digest*, 36(6):469, 2004.
- [33] Kohju Ikago, Kenji Saito, and Norio Inoue. Seismic control of single-degree-of-freedom structure using tuned viscous mass damper. *Earthquake Engineering & Structural Dynamics*, 41(3):453–474, 2012.
- [34] Dawei Li, Kohju Ikago, and Ao Yin. Structural dynamic vibration absorber using a tuned inerter eddy current damper. *Mechanical Systems and Signal Processing*, 186:109915, 2023.
- [35] Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011, 2011.