Structural identification with physics-informed neural ordinary differential equations

Journal Article

Author(s):
Lai, Zhilu; Mylonas, Charilaos; Nagarajaiah, Satish; Chatzi, Eleni

Publication date:
2021-09-15

Permanent link:
https://doi.org/10.3929/ethz-b-000488395

Originally published in:
Journal of Sound and Vibration 508, https://doi.org/10.1016/j.jsv.2021.116196

Funding acknowledgement:
679843 - Smart Monitoring, Inspection and Life-Cycle Assessment of Wind Turbines (EC)

Structural Identification with Physics-informed Neural Ordinary Differential Equations

Zhilu Lai[1,*], Charilaos Mylonas[1], Satish Nagarajaiah[2], Eleni Chatzi[1]

[1] Chair of Structural Mechanics and Monitoring, Institute of Structural Engineering, Department of Civil, Environmental and Geomatic Engineering, ETH-Zürich, Zürich, Switzerland

[2] Deparment of Civil and Environmental Engineering, Rice University, Houston, USA.

ABSTRACT

This paper exploits a new direction of structural identification by means of Neural Ordinary Differential Equations (Neural ODEs), particularly constrained by domain knowledge, such as structural dynamics, thus forming Physics-informed Neural ODEs, aiming at governing equations discovery/approximation. Structural identification problems often entail complex setups featuring high-dimensionality, or stiff ODEs, which pose difficulties in the training and learning of conventional data-driven algorithms who seek to unveil the governing dynamics of a system of interest. In this work, Neural ODEs are re-casted as a two-level representation involving a physics-informed term, that stems from possible prior knowledge of a dynamical system, and a discrepancy term, captured by means of a feed-forward neural network. The re-casted format is highly adaptive and flexible to structural monitoring problems, such as linear/nonlinear structural identification, model updating, structural damage detection, driving force identification, etc. As an added step, for inferring an explainable model, we propose the adoption of sparse identification as an additional tool to transparentize the trained nets into closed-form expressions that embed a more straightforward engineering interpretation. We demonstrate the framework on a series of numerical and experimental examples, with the latter pertaining to a structural system featuring highly nonlinear behavior, which is successfully learned by the proposed framework. The proposed structural identification with Physics-informed Neural ODEs comes with the benefits of direct approximation of the governing dynamics, and a versatile and flexible framework for discrepancy modeling in structural identification problems.

Keywords: Neural Ordinary Differential Equations; physics-informed machine learning; scientific machine learning; discprepancy modeling; structural identification; structural damage detection; structural health monitoring

*Further author information: (send correspondence to laiz@ethz.ch)

1. INTRODUCTION

The term structural identification or structural-system identification (SI)[1–4] defines a set of methodologies for inverse modeling of structural systems, where information stemming from data is used to inform mathematical models that can be used for estimations and predictions of increased confidence on the condition and residual life of structural systems. In recent years, the rapid growth of sensing techniques and the associated development of machine learning methods,[5] has spawned a broad array of identification tools that are reliant on machine learning with applications across various domains.[6–9]

Research efforts have been extensively carried out to transform measured response, either in the lab or in the field, into either mechanistic (physically interpretable) or non-mechanistic ("black-box") models, which can be used for the purpose of prediction given new inputs. In the field of structural engineering, we attempt to classify these efforts into two main clusters, namely purely data-driven methods and physics-driven methods. The first class includes classical identification tools formulated in the time domain, such as eigensystem realization algorithm,[10] or in the frequency domain, such as frequency domain decomposition,[11] or latent space tools, such as blind source separation[12–15] fall in this class. In most cases, these tools are constrained in the application onto identification of linear systems. The physics-driven class typically assumes a model structure, which can be either a prior parameterized model or a general structure of differential equations for instance. The parameters in a prior parameterized model can be recursively updated by minimizing the error between the prediction from the prior structured model and the corresponding experimental data, by means of either an optimization or a stochastic, e.g., Bayesian updating approach.[16–21] If the model structure is given by a general structure, for instance, the format of differential equations, a new branch that is affiliated to the physics-driven class is comprised of the so-called governing equations discovery/approximation methods.[22–25] These attempt to recover a mathematical expression (governing equations) directly from experimental data, without a prior assignment of a specific model structure. The expressive power of these methods predominately depends on the appropriate structuring of a pre-defined pool of candidate basis functions.

In this work, we explore the feasibility of linear/nonlinear structural identification via adoption of Neural ODEs as a tool for governing equations approximation. As a family of deep neural network models, Neural Ordinary Differential Equations (Neural ODEs),[26] have established a framework and viewpoint bridging neural networks, i.e., a typically data-driven element, with differential equations, i.e., a physics-based element. Neural ODEs have proven their capacity in learning the underlying governing dynamics from measured data of a dynamical system.[26–29]

Instead of abstractly seeking a nonlinear transform from input to output variables, Neural ODEs approximate the derivative of the transform with a neural network, allowing for a flexible description of the system evolution. The fact that neural networks can be used as universal function approximators[30] allows for the description of arbitrarily complicated dynamics, covering most of the typical use-cases in engineering. Owing to this, SI by means of Neural ODEs can be applied to complex nonlinear systems that are not easily modeled via purely physics-based models.

Empirical experiments with Neural ODEs in this paper indicate that the networks might be less trainable (requiring more careful training) and the approximated governing equations might be less capable of extrapolation, when applied to high-dimensional dynamical systems or stiff ODEs, which are though often met in the field of structural health monitoring. The integration of domain knowledge into the architectures of deep learning models[31–39] enhances interpretability and physical consistency of the derived models. This work builds on the concept of universal differential equations, as introduced in the work of Rackauckas et al.[40] The approach is adopted for the particular purpose of structural identification by seeding domain knowledge, in this case from structural dynamics, thus forming Structural Identification with Physics-informed Neural ODEs. The framework is advantageous from the point of view of ease of training, interpretability, and flexibility in error/discrepancy modeling. The key idea lies in splitting the format of Neural ODEs into a physics-informed term (a known portion that stems from possible prior knowledge) and an unknown discrepancy term taken up by a neural network. By training these blended Neural ODEs, the discrepancy between the prior assumption on the driving physics and the true system is learned. This approach to discrepancy modeling[41,42] can find application across a diverse suite of problems, such as *model updating*, *structural damage detection*, *nonlinear component modeling*, *driving force identification*, etc. It is noted that very recently, Roehrl et al.[43] share a similar motivation in their work within the field of automatic control, and adopt neural networks to approximate non-conservative forces in a dynamical system. In this paper, we present an adaptation to the specific challenges of structural identification and health monitoring, especially for nonlinear dynamical systems, described by uncertainty. The latter drives the necessity on inference of the model discrepancy term.

The proposed framework contributes to the field of structural identification and health monitoring in terms of: (1) Physics-informed Neural ODEs serving as learners of the governing dynamics of the systems, rather than a simple mapping between input and output relationship. The learned function is in the format of an ODE, which lends itself for reuse in prediction given new initial conditions or new driving forces that are different to the ones used for training. (2) The scheme is applicable to both linear and nonlinear systems, featuring a versatile approach to discrepancy modeling in structural identification problems. It is demonstrated that in the case

of linear systems, the stiffness and damping matrices can be directly recovered from the weights of the trained networks. In the case of nonlinear systems, sparse identification of nonlinear dynamical systems[23,24] is suggested as an additional tool for turning a trained "black-box" neural network into a model with mechanistic intuition.

2. FRAMEWORK

2.1 Neural Ordinary Differential Equations (Neural ODEs)

Neural Ordinary Differential Equations[26] (Neural ODEs) along with their variants[27,29] have raised significant attention in recent years, offering a new paradigm and insights into the linkage of neural networks with differential equations. As a branch of deep neural networks, Neural ODEs can be interpreted as a continuous representation of Residual Networks (ResNets).[44] In ResNets, a hidden state (or hidden layer in the language of neural networks) $\mathbf{h}(t)$ allows transitioning from a layer t to the next layer $t+1$ through a differentiable function $\mathbf{f}_t(\cdot)$:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \mathbf{f}_t(\mathbf{h}_t) \tag{1}$$

This transformation can be interpreted as a numerical integration via a forward Euler's scheme (explicit) with a time step $\Delta t = 1$ (i.e., $\mathbf{h}_{t+1} = \mathbf{h}_t + \mathbf{f}_t(\mathbf{h}_t)\Delta t$). By letting $\Delta t \to 0$, the following limit is attained:

$$\lim_{\Delta t \to 0} \frac{\mathbf{h}_{t+1} - \mathbf{h}_t}{\Delta t} = \frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \boldsymbol{\theta}) \tag{2}$$

This essentially converts a discretized ODE into a continuous ODE. The function $\mathbf{f}_t(\cdot)$ corresponds to a time derivative function $f(\cdot)$ that is constant throughout the flow. The key idea of Neural ODEs builds on parameterizing a continuous dynamical system using neural networks, in the format of ordinary differential equations for an initial value problem (IVP):

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \boldsymbol{\theta}) \qquad \mathbf{h}(t_0) = \mathbf{h}_0 \tag{3}$$

where $f(\cdot)$ is represented by a neural network parameterized by $\boldsymbol{\theta}$; $\boldsymbol{\theta}$ includes all the weights of the neural network; the initial condition is given as $\mathbf{h}(t_0) = \mathbf{h}_0$. $\mathbf{h}(t)$ can be interpreted as the function governing state evolution over time t (trajectories), and $f(\cdot)$ mathematically defines the derivative of the evolution (or the rate of the change of a system state).

In Neural ODEs, one can map the input as an initial state $\mathbf{h}(t_0)$ to an output $\mathbf{h}(t_p)$ (e.g., a state evolved after p time steps) by solving an ODE numerically — $\text{ODESolve}(\mathbf{h}(t_0), f, t_0, t_p, \boldsymbol{\theta})$. This forward procedure is mathematically encoded in a loss function $L(\cdot)$,[26] evaluating the

distance between output $\mathbf{h}(t_p)$ and the desired measurements $\mathbf{h}(t_p)_{\text{measure}}$

$$L\left(\mathbf{h}(t_p)\right) = L\left(\mathbf{h}(t_0) + \int_{t_0}^{t_p} f(\mathbf{h}(t), t, \boldsymbol{\theta})dt\right) = L\left(\text{ODESolve}(\mathbf{h}(t_0), f, t_0, t_p, \boldsymbol{\theta})\right) \tag{4}$$

where ODESolve can be given by established ODE solvers (such as Runge-Kutta methods). Then, the neural network $f(\cdot)$ is trained/learned by determining the weights $\boldsymbol{\theta}$ that minimize the loss function L. Chen et al.[26] have suggested to implement an adjoint sensitivity method that is applicable to any ODE solver in order to compute the gradients $\frac{\partial L}{\partial \boldsymbol{\theta}}$ with respect to the weights $\boldsymbol{\theta}$ for reducing memory costs. Once the gradient $\frac{\partial L}{\partial \boldsymbol{\theta}}$ is computed, optimization methods such as gradient descent, ADAM[45] can be further implemented for determining the weights $\boldsymbol{\theta}$ that minimize L. Ultimately, the mapping from $\mathbf{h}(t)$ to $\frac{d\mathbf{h}(t)}{dt}$ is represented by the trained neural network $f(\cdot)$, which aims to approximate the time derivative of $\mathbf{h}(t)$.

It is noted that the training of a Neural ODE only exploits the state vector $\mathbf{h}(t)$ as input (the explicit measurements/input of the derivative $\frac{d\mathbf{h}(t)}{dt}$ is not required). This allows to learn the underlying dynamics from measured data directly, regardless of the *prior* knowledge on function $f(\cdot)$, due to the universal approximation theorem.[30] The governing equations are finally approximated by a trained neural network. A neural network is essentially a function, thus the procedure of solving a trained Neural ODE in Eq.(3) may exploit established methods of solving ordinary differential equations.

2.2 Structural Identification with Physics-informed Neural ODEs

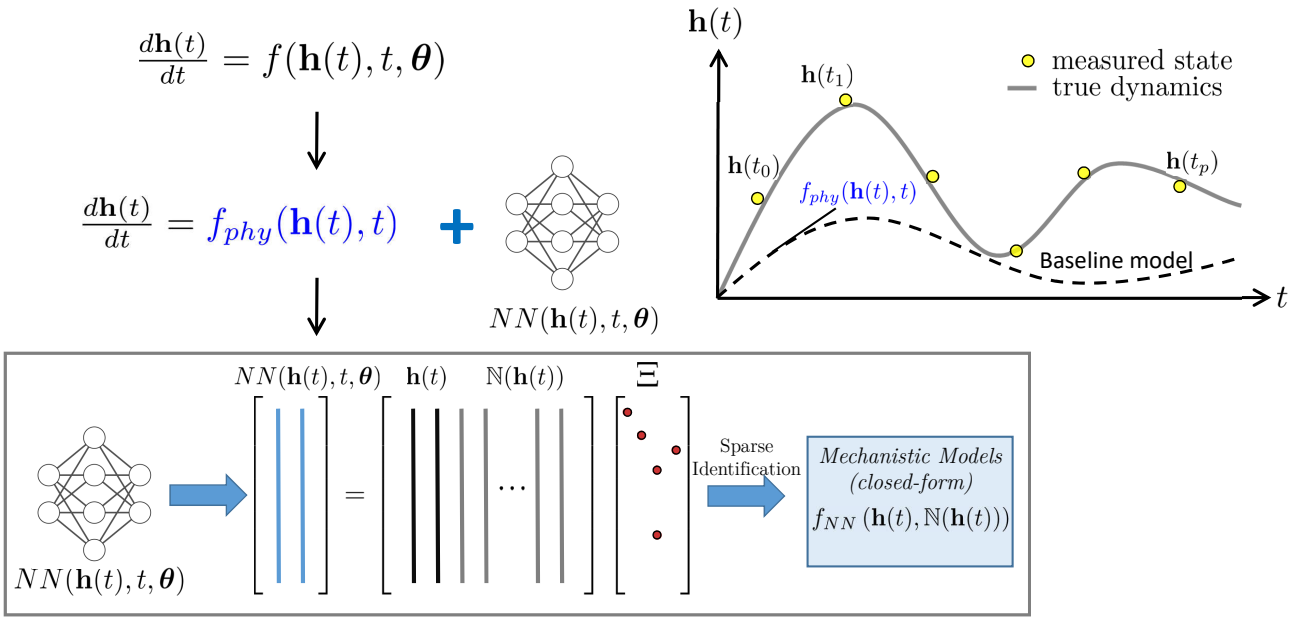


Figure 1: Illustration of Physics-informed Neural ODEs

2.2.1 Physics-informed Neural ODEs

In this section, we will discuss the feasibility of structural identification with Neural ODEs, for dealing with problems of structural identification, damage detection, and health monitoring. One can observe that the format of Neural ODEs aligns with the typical format of a dynamical system, when the hidden state $\mathbf{h}(t)$ is interpreted as a state vector representation: $\mathbf{h}(t) = [\mathbf{x}(t)\ \dot{\mathbf{x}}(t)]^T \in \mathbb{R}^n$, in which $\mathbf{x}(t)$ and $\dot{\mathbf{x}}(t)$ represent the displacement and velocity vectors, respectively.

In consideration of the fact that most real-world structural dynamics problems are forced vibration rather than only IVPs, and with the intention of incorporating domain knowledge into the architecture of Neural ODEs, we reformat these into either Eq.(5a) or Eq.(5b), depending on whether the neural network accounts for the entire or partial entries of $\frac{d\mathbf{h}(t)}{dt}$:

$$\frac{d\mathbf{h}(t)}{dt} = f_{phy}(\mathbf{h}(t), t, \mathbf{u}(t)) + NN(\mathbf{h}(t), t, \boldsymbol{\theta}) \tag{5a}$$

or, for the cases examined as part of this work, where the nonlinear contribution pertains to

specific components of the state vector:

$$\frac{d\mathbf{h}(t)}{dt} = f_{phy}(\mathbf{h}(t), t, \mathbf{u}(t)) + \begin{bmatrix} \mathbf{0}_{p_1\times 1} \\ NN(\mathbf{h}(t), t, \boldsymbol{\theta}) \in \mathbb{R}^{n-p_1-p_2} \\ \mathbf{0}_{p_2\times 1} \end{bmatrix} \tag{5b}$$

in which, n is the dimension of the state vector $\mathbf{h}(t)$; $\mathbf{0}_{p_1\times 1}$ and $\mathbf{0}_{p_2\times 1}$ are zero vectors reflecting that the neural network ($NN(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n-p_1-p_2}$) contributes to only a part of the components of $\frac{d\mathbf{h}(t)}{dt}$ (i.e., the dimension of the output of $NN(\cdot)$ is not necessarily set equal to n); $\mathbf{u}(t)$ is an array of driving forces acting on the system; $f(\cdot)$ in Eq.(3) has been split into a physics-informed term $f_{phy}(\cdot)$, specified by possible *prior* domain knowledge (this can stem *partially* or *entirely* from theoretical models or even finite element models) and a supplemental term $NN(\cdot)$ that is encoded by a feed-forward neural network; $NN(\cdot)$ is parameterized by weights $\boldsymbol{\theta}$ (including $\mathbf{W}^{(j)}$ and $\mathbf{b}^{(j)}$), which is typically given in a chain structure with $l+1$ layers:[46]

$$NN(\cdot) : \begin{cases} \mathbf{h}^{(1)} = \sigma^{(1)}(\mathbf{W}^{(1)T}\mathbf{h}^{(0)} + \mathbf{b}^{(1)}) \\ \mathbf{h}^{(2)} = \sigma^{(2)}(\mathbf{W}^{(2)T}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}) \\ \qquad \vdots \\ \mathbf{h}^{(1)} = \sigma^{(l)}(\mathbf{W}^{(l)T}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \end{cases} \tag{6}$$

where $\mathbf{h}^{(j)}$ denotes a vector of hidden unit at the jth layer, and $\mathbf{h}^{(0)}$ and $\mathbf{h}^{(l)}$ are therefore assigned as input and output vector, respectively; matrix $\mathbf{W}^{(j)}$ provides a linear transform from $(j-1)^{\text{th}}$ layer to j^{th} layer, and vector $\mathbf{b}^{(j)}$ provides the bias vector in j^{th} layer; $\sigma^{(j)}(\cdot)$ denotes a nonlinear function at j^{th} layer usually called an activation function, which can be specified by functions such as $tanh(\cdot)$, $sigmoid(\cdot)$, *ReLU* (rectified linear unit[47]), etc.

It is noted that we incorporate time variable t in $f_{phy}(\cdot)$, as time-dependant factors, such as degradation modeling, environmental effects, can be taken into account. Compared to standard Neural ODEs, as illustrated in Figure 1, the framework basically constrains the search space with a baseline model/function whose derivatives are given by $f_{phy}(\cdot)$. This renders the enhancement of physical consistency for the trained models (i.e., the search space is physically constrained). A similar procedure to Eq.(4) is suggested to define the loss function, with $f(\cdot)$ replaced by $f_{phy}(\cdot) + NN(\cdot)$, and the mean squared error used for defining the loss function.

The Julia libraries[48] termed *DifferentialEquations.jl*[49] and *DiffEqFlux.jl*[50] developed by Rackauckas et al. are herein adopted as tools for implementing Physics-informed Neural ODEs.

It is noted that in Eq.(5), the term $NN(\cdot)$ essentially manages to learn and model the

dependency between the true dynamics and the physics-informed term, which attempts to reflect the existing (but limited) knowledge of the system. This discrepancy modeling approach finds application in different problems within the domain of system identification and SHM, such as

- *model updating*: if $f_{phy}(\cdot)$ is given by an analytical model or finite element model, and $NN(\cdot)$ accounts for the updated information for this model.
- *damage detection*: if $f_{phy}(\cdot)$ is interpreted as the healthy (intact) state of a structural system and $NN(\cdot)$ accounts for an alteration of the system due to damage.
- *modeling nonlinearity*: the direct modeling of a nonlinear dynamical system is non-trivial since the type (function) of underlying nonlinearity is often unknown. $f_{phy}(\cdot)$ can be assigned as the linear portion of the system, while $NN(\cdot)$ is then intended to learn the behavior the nonlinear portion of the system.
- *force identification*: if a system is driven by external forces and $f_{phy}(\cdot)$ is setup to reflect the state evolution of the dynamical system itself (including inertia, damping forces, and conservative forces, etc.), $NN(\cdot)$ can be used for learning the external driving force, which forms a function of time t.

2.2.2 Transparentization of trained nets

In order to further extract closed-form analytical expressions from the trained neural networks $NN(\cdot)$, linking to an expected mechanistic behavior, as an additional (optional) tool, sparse identification of nonlinear dynamical systems (SINDy)[23,24] may be adopted, as illustrated in Figure 1. We refer to this process as transparentization of the Neural ODE. (1) Upon training of the the Neural ODEs, and under availability of information on initial conditions and driving forces, Eq.(5) is solved via numerical integration to obtain the predicted state $\mathbf{h}(t)$. (2) The predicted $\mathbf{h}(t)$ is fed into the trained nets $NN(\cdot)$ to compute the output $NN(\mathbf{h}(t), t, \boldsymbol{\theta})$. (3) SINDy is performed in order to derive closed-form expression of the discrepancy term, through setting up a sparse regression problem to obtain a sparse solution of coefficient matrix $\boldsymbol{\Xi}$:

$$NN(\mathbf{h}(t), t, \boldsymbol{\theta}) = \Theta(\mathbf{h}(t))\boldsymbol{\Xi} \tag{7}$$

where $\Theta(\mathbf{h}(t))$ is an augmented feature matrix of $\mathbf{h}(t)$, which is formed by stacking evaluations of $\mathbf{h}(t_i)$ $(i = 1, 2, 3, ..., N)$ via a series of pre-selected basis functions $\mathbb{N}(\cdot)$, such as polynomial

functions and trigonometric functions, as follows:

$$\Theta(\mathbf{h}(t)) = \begin{bmatrix} \mathbf{h}^T(t_1) & \mathbb{N}(\mathbf{h}^T(t_1)) \\ \mathbf{h}^T(t_2) & \mathbb{N}(\mathbf{h}^T(t_2)) \\ \vdots & \vdots \\ \mathbf{h}^T(t_N) & \mathbb{N}(\mathbf{h}^T(t_N)) \end{bmatrix} \tag{8}$$

where $\mathbf{h}(t_i) = [h_1(t_i), h_2(t_i), ..., h_n(t_i)]^T$ denotes the state vector at time t_i. After a sparse solution of $\boldsymbol{\Xi}$ is determined, the closed-form expressions concatenated with the known portion $f_{phy}(\cdot)$ are given as:

$$\frac{d\mathbf{h}}{dt} = f_{phy}(\mathbf{h}, t, \mathbf{u}) + \boldsymbol{\Xi}^T \left(\Theta(\mathbf{h})\right)^T \tag{9}$$

which is a symbolic expression. This serves as the predictive model of the system, which can be used for new inputs and ICs, beyond those used for training. The fidelity of the model, depends on the richness of the training set, which should be selected so as to activate the full range of underlying nonlinearity. The Python package PySINDy[51] for SINDy is adopted to this end.

3. NUMERICAL EXAMPLES

3.1 A 4-degree-of-freedom (4-DOF) Dynamical System with Cubic Nonlinearity

3.1.1 Free vibration case

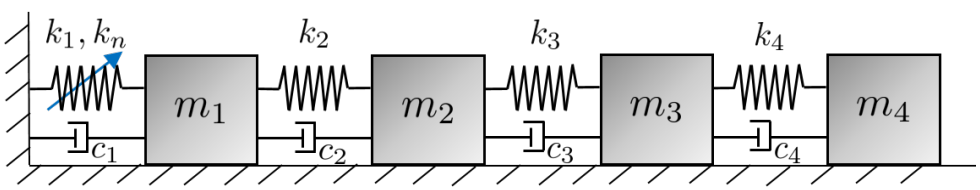


Figure 2: Illustration of a 4-DOF nonlinear structural dynamical system

To illustrate the proposed framework, we first present a numerical example of a 4-DOF structural system, which is schematically illustrated in Figure 2. The system represents a typical spring-mass model often adopted in structural dynamics simulations, with the first DOF featuring an additional nonlinear stiffness element, of cubic nonlinearity $k_n x_1^3$. The equations of motion of

the free vibration are given as:

$$\begin{cases} m_1\ddot{x}_1(t) + (k_1+k_2)x_1(t) - k_2x_2(t) + c_1\dot{x}_1(t) + k_nx_1^3(t) = 0 \\ m_2\ddot{x}_2(t) - k_2x_1(t) + (k_2+k_3)x_2(t) - k_3x_3(t) + c_2\dot{x}_2(t) = 0 \\ m_3\ddot{x}_3(t) - k_3x_2(t) + (k_3+k_4)x_3(t) - k_4x_4(t) + c_3\dot{x}_3(t) = 0 \\ m_4\ddot{x}_4(t) - k_4x_3(t) + k_4x_4(t) + c_4\dot{x}_4(t) = 0 \end{cases} \tag{10}$$

where $x_i(t)$, $\dot{x}_i(t)$ and $\ddot{x}_i(t)$ $(i = 1, 2, 3, 4)$ designate the vectors of displacement, velocity and acceleration, respectively; the stiffness coefficients are selected as $k_1 = k_2 = k_3 = k_4 = 10$; viscous damping is assumed, with the damping coefficients $c_1 = c_2 = c_3 = c_4 = 0.5$; while the coefficient associated with the cublic nonlinearity is selected as $k_n = 2$; the mass for each DOF are set to be equal to 1 $(m_1 = m_2 = m_3 = m_4 = 1)$ reflecting a mass normalized system. The state vector $\mathbf{h}(t) \in \mathbb{R}^{8\times1}$ is formulated as::

$$\mathbf{h}(t) = [x_1(t)\ x_2(t)\ x_3(t)\ x_4(t)\ \dot{x}_1(t)\ \dot{x}_2(t)\ \dot{x}_3(t)\ \dot{x}_4(t)]^T \tag{11}$$

Eq.(10) can be therefore rewritten in state-space form, comprising superposition of a linear term $d\mathbf{h}_L$ and a nonlinear vector $d\mathbf{h}_{NL}$, which includes the cubic nonlinearity $k_nx_1^3(t)$:

$$\frac{d\mathbf{h}(t)}{dt} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K} & -\mathbf{C} \end{bmatrix} \mathbf{h}(t) + \begin{bmatrix} 0 & 0 & 0 & 0 & -k_nx_1^3(t) & 0 & 0 & 0 \end{bmatrix}^T}_{f_{true}(\cdot)} \tag{12a}$$

or

$$\frac{d\mathbf{h}(t)}{dt} = d\mathbf{h}_L + d\mathbf{h}_{NL} \tag{12b}$$

where $f_{true}(\cdot)$ denotes the complete dynamical system, that is the entire expression on the right-hand-side, and will be later used as a reference for evaluating the learning performance of the proposed Physics-informed Neural ODEs; $\mathbf{0} \in \mathbb{R}^{4\times4}$ is a zero matrix; $\mathbf{I} \in \mathbb{R}^{4\times4}$ is the identity matrix; $\mathbf{K}$ and $\mathbf{C}$ designate the stiffness and damping matrices corresponding to the linear portion of the dynamics, respectively:

$$\mathbf{K} = \begin{bmatrix} k_1+k_2 & -k_2 & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & 0 \\ 0 & -k_3 & k_3+k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix}; \quad \mathbf{C} = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \end{bmatrix}; \tag{12c}$$

Eq.(12) is considered as the reference ODE that will be learnt and approximated. In Table 1, we list three schemes that reflect three different degrees of prior knowledge of the system: (1) scheme 1 (*no physics informed*): this corresponds to adoption of Neural ODEs for directly learning the entire dynamics; (2) scheme 2 (*physics weakly informed*): this reflects incorporation of knowledge with respect to existence of a linear transfer matrix, albeit at (severe) error, corresponding to 30% reduction of matrices $\mathbf{K}$ and $\mathbf{C}$; (3) scheme 3 (*physics strongly informed*): the true linear transfer matrix is assumed to be available.

In addition, Table 1 lists the ground-truth of the model discrepancy term $f_{true}(\cdot) - f_{phy}(\cdot)$ for each scheme.

Table 1: Three implementation schemes of Neural ODEs

Schemes	$f_{phy}(\cdot)$	$NN(\cdot)$	$f_{true}(\cdot) - f_{phy}(\cdot)$
scheme 1 (*no physics informed*)	$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{h}(\mathrm{t})$		$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\mathbf{K} & -\mathbf{C} \end{bmatrix} \mathbf{h}(t) + d\mathbf{h}_{NL}$
scheme 2 (*physics weakly informed*)	$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -0.7\mathbf{K} & -0.7\mathbf{C} \end{bmatrix} \mathbf{h}(\mathrm{t})$	NN: $\mathbb{R}^8 \rightarrow \mathbb{R}^4$	$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -0.3\mathbf{K} & -0.3\mathbf{C} \end{bmatrix} \mathbf{h}(t) + d\mathbf{h}_{NL}$
scheme 3 (*physics strongly informed*)	$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{K} & -\mathbf{C} \end{bmatrix} \mathbf{h}(\mathrm{t})$		$d\mathbf{h}_{NL}$

According to Eq.(5), Eq.(12) can be split as:

$$\frac{d\mathbf{h}(t)}{dt} = f_{phy}(\mathbf{h}(t)) + \begin{bmatrix} \mathbf{0} \\ NN(\mathbf{h}(t)) \end{bmatrix} \tag{13}$$

where $f_{phy}(\mathbf{h}(t))$ is the physics-informed term, each time described by one of the three schemes in Table 1; $\mathbf{0}$ is a zero vector $\in \mathbb{R}^4$; the unknown (discrepancy) portion of the dynamical system is every time learned by a neural network (NN: $\mathbb{R}^8 \rightarrow \mathbb{R}^4$). The input for the Neural ODEs is the state variable $\mathbf{h}(t)$, while the output is assumed to be limited to the 4-dimensional acceleration vector $[\ddot{x}_1(t), \ddot{x}_2(t), \ddot{x}_3(t), \ddot{x}_4(t)]^T$, since the first four equations in Eq.(12) are always $\dot{x}_1(t) = \dot{x}_1(t)$, $\dot{x}_2(t) = \dot{x}_2(t)$, $\dot{x}_3(t) = \dot{x}_3(t)$ and $\dot{x}_4(t) = \dot{x}_4(t)$. It is noted that, in this example, the architecture of the $NN(\cdot)$ is designed identically for the three schemes featuring only one hidden layer, with a hyperbolic tangent function *tanh* assigned as activation function $\sigma(\cdot)$. According to the expression in Eq.(6), the architecture is given by:

$$NN(\mathbf{h}) : \mathbf{W}^{(2)T}[\sigma(\mathbf{W}^{(1)T}\mathbf{h} + \mathbf{b}^{(1)})] + \mathbf{b}^{(2)} \tag{14}$$

The number of neurons assumed in the hidden layer is equal to 30, and therefore the dimension of the weights are assigned as: $\mathbf{W}^{(1)} \in \mathbb{R}^{8\times 30}$; $\mathbf{b}^{(1)} \in \mathbb{R}^{30}$; $\mathbf{W}^{(2)} \in \mathbb{R}^{30\times 4}$; $\mathbf{b}^{(2)} \in \mathbb{R}^{4}$.

A free vibration problem is considered here under two cases of initial conditions $\mathbf{h}_1(t_0) = [2, 0, 0, 0, 0, -2, 0, 0]^T$ and $\mathbf{h}_2(t_0) = [-2, 0, 0, 3, -2, 0, 0, 0]^T$. Simulated state data $x_i(t)$ and $\dot{x}_i(t)$ $(i = 1, 2, 3, 4)$ over a duration of 6 seconds are used for training the Neural ODEs.

Figure 3 compares the history of loss evaluations under consideration of the three schemes, overviewed in Table 1, which reflect a different level of knowledge of the underlying physics. In the first 3000 iterations, the response for initial condition $\mathbf{h}_1(t_0)$ is used to perform a training, followed by a training for response under initial condition $\mathbf{h}_2(t_0)$. It is evident that the schemes that are informed by prior knowledge of the system (scheme 2 and 3) display faster convergence than scheme 1, which assumes no prior knowledge on the underlying physics. In addition, schemes 2 and 3 correspond to lower loss values in the range of 10^{-2}, while scheme 1 corresponds to higher loss values for the same number of iterations. The integration of prior knowledge of the system into the neural network architecture, Physics-informed Neural ODEs exhibit faster convergence and a higher precision than the direct implementation of Neural ODEs.

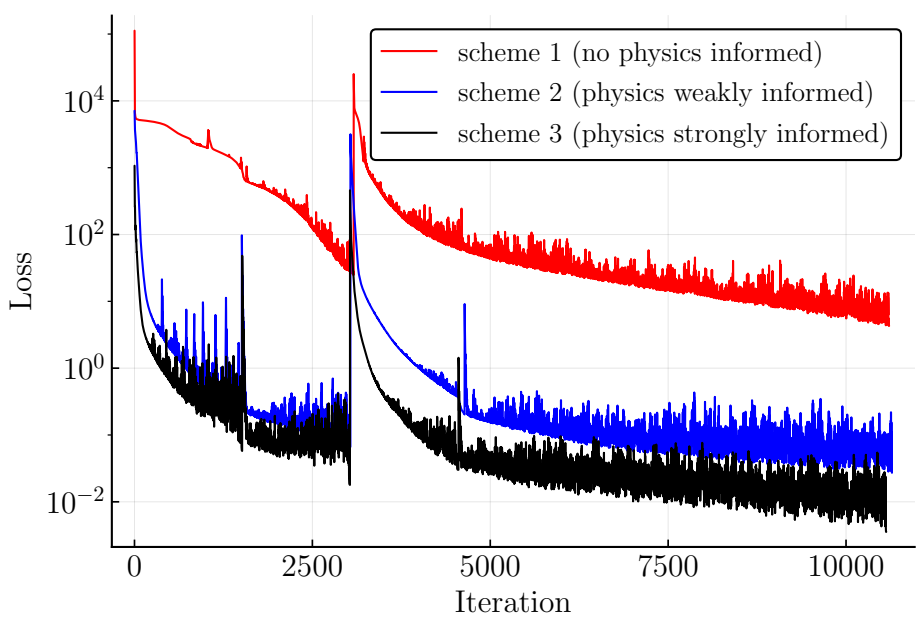
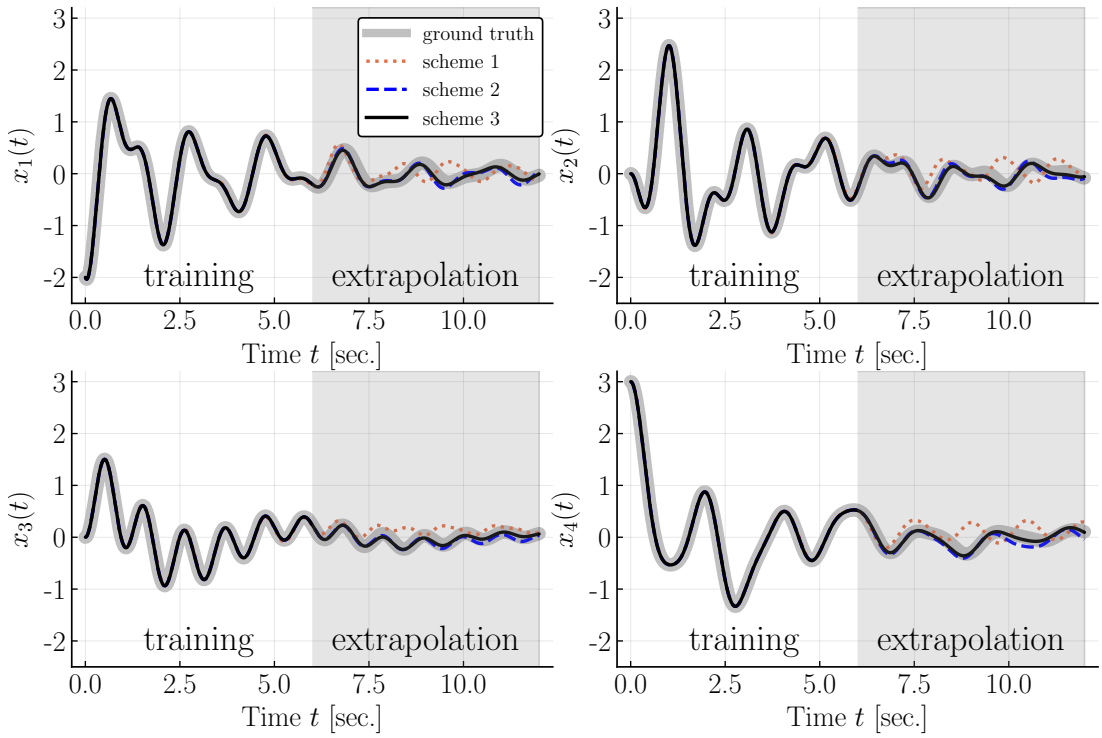


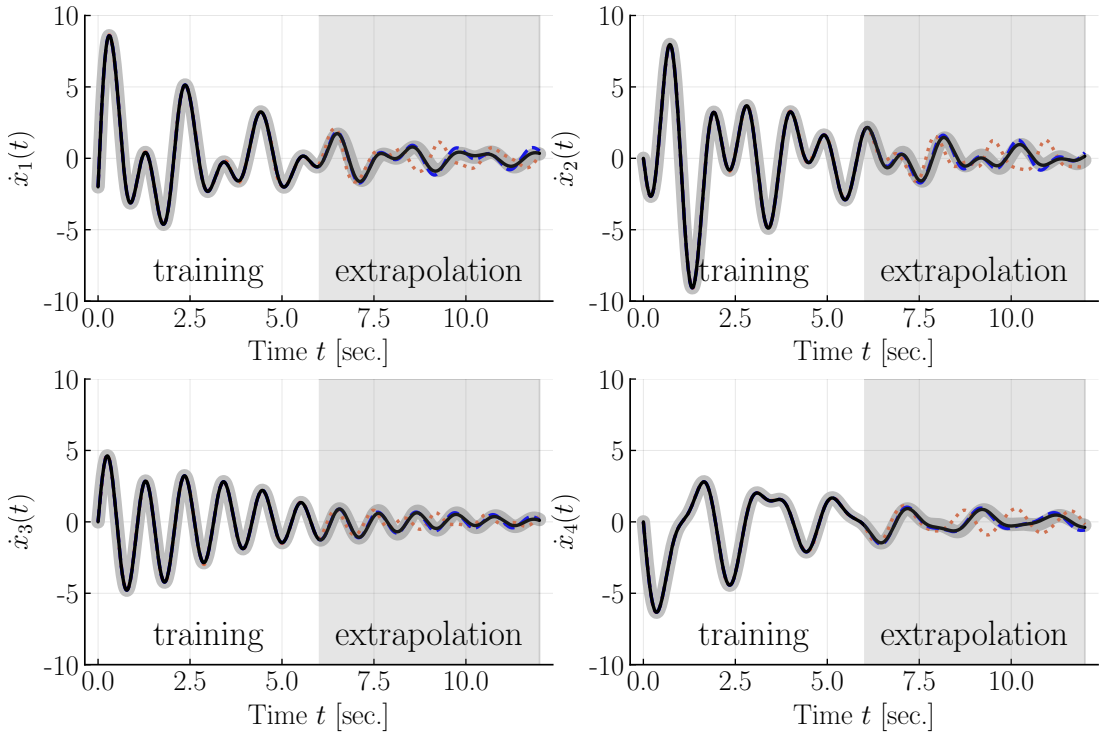
Figure 3: History of evaluations of loss function for the three schemes listed in Table 1.

After the completion of training, Figure 4 shows the prediction of displacement and velocity response $x_i(t)$ and $\dot{x}_i(t)$ $(i = 1, 2, 3, 4)$ for each one of the trained models according to the three schemes listed in Table 1, under the initial conditions $\mathbf{h}_2(t_0)$. The results from 0 second to 6 seconds (the period of the data used for training) display identical predictions by all three schemes

as expected, closely approximating the ground truth. When extrapolating the prediction to 12 seconds, it is observed that the prediction by scheme 1 starts to slightly deviate from the ground truth. This indicates that the extrapolation ability of the trained scheme 1 is weaker than predictive ability of schemes 2 and 3.



(a) displacement response estimation



(b) velocity response estimation

Figure 4: Response estimation for each candidate model, under the initial conditions $\mathbf{h}_2(t_0)$ used for training.

In the testing stage, a new initial condition $\mathbf{h}_3(t_0) = [0, 4, 0, 0, 0, 0, 0, 0]^T$ that is different to those used in training, i.e., $\mathbf{h}_1(t_0)$ and $\mathbf{h}_2(t_0)$, is adopted. Figure 5 indicatively illustrates the prediction of velocity response. It is observed that scheme 1 under-performs with respect to schemes 2 and 3. It is noted that magnitude of the response lies beyond the range used in training (if we compare Figure 5 to Figure 4b), which implies that Physics-informed ODEs are capable of extrapolation and generalization to some extent, provided they have captured the degree of nonlinearity that describes the physics.

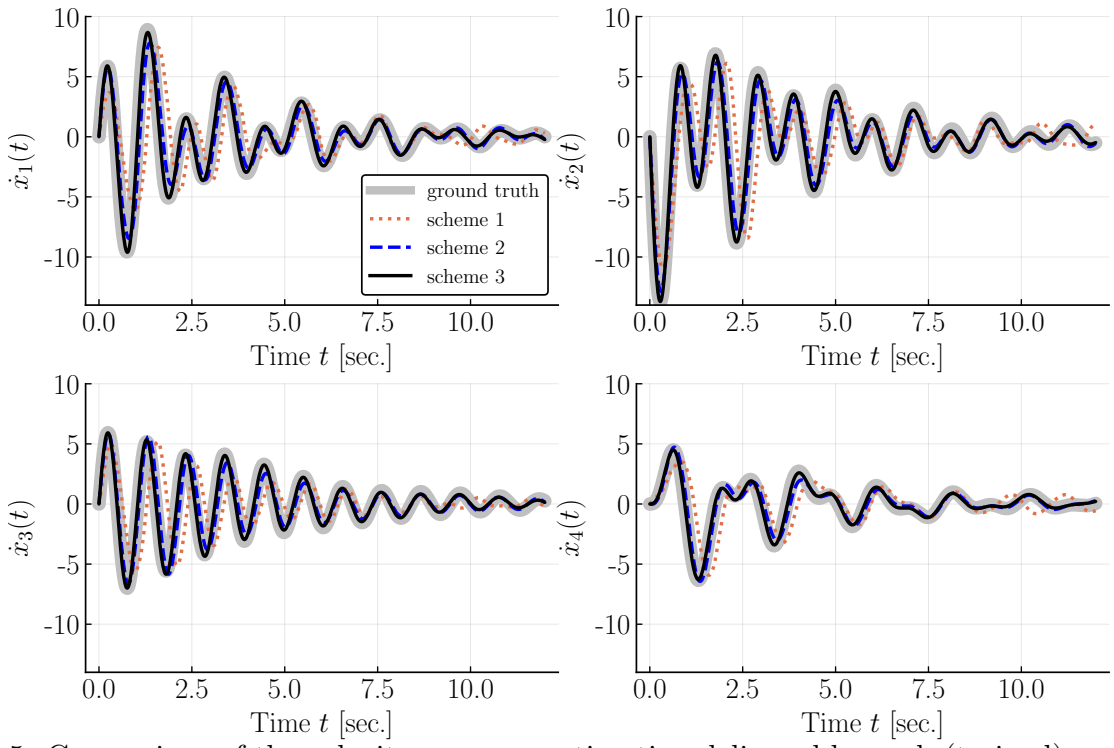
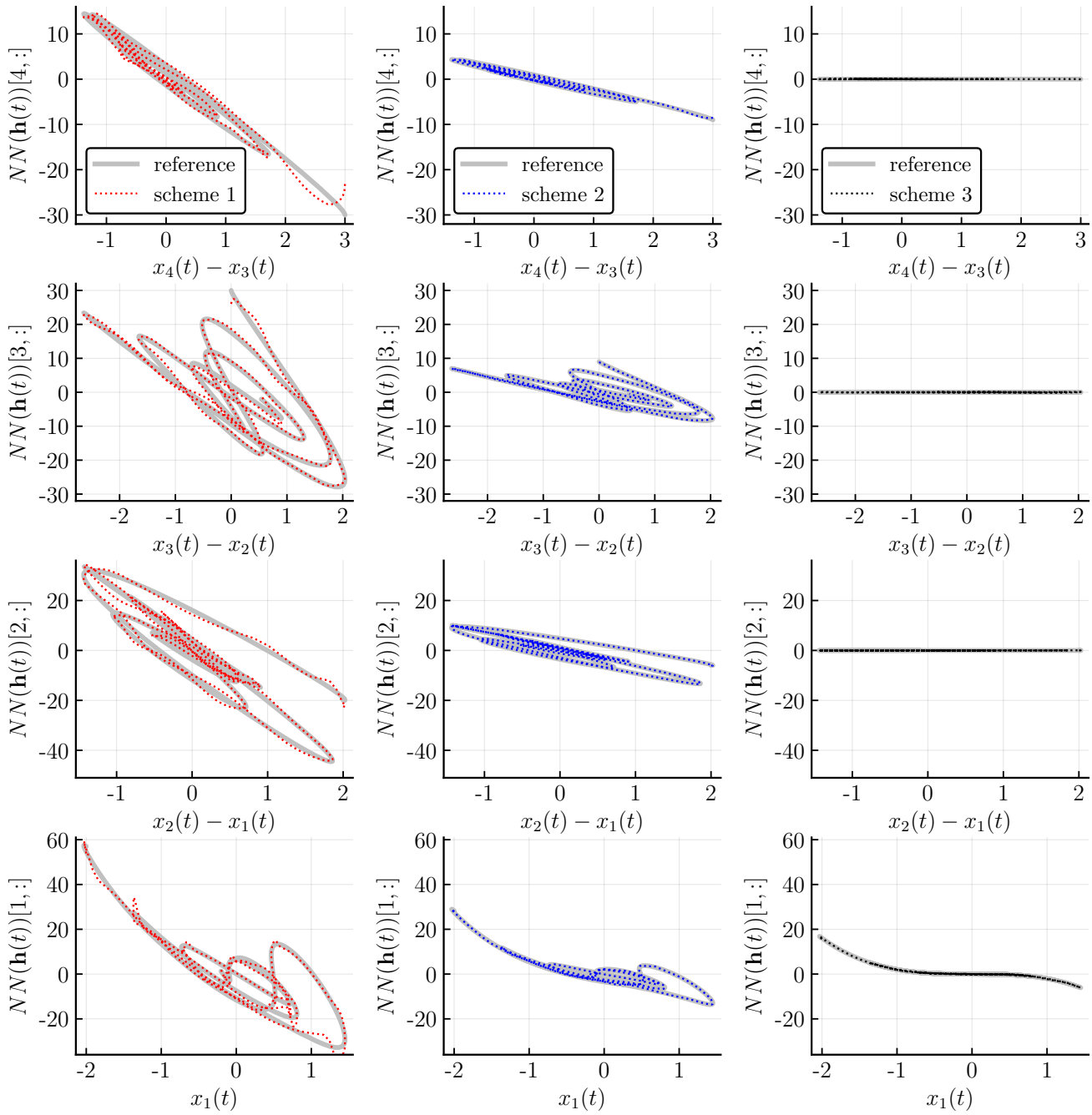


Figure 5: Comparison of the velocity response estimation delivered by each (trained) candidate models for an initial condition that is different to the ones used for training.

The neural network $NN(\cdot)$ used by each scheme essentially accounts for the discrepancy terms between $f_{true}(\cdot)$ and $f_{phy}(\cdot)$, corresponding to the acceleration vector. In Figure 6, we visualize the outputs of trained $NN(\mathbf{h}(t))$ *versus* each inter-DOF displacement $(x_i(t) - x_{i-1}(t))$ for each scheme, in comparison to their corresponding references. The references stem from the $f_{true}(\cdot) - f_{phy}(\cdot)$ listed in Table 1 with respect to the acceleration vector. Notation $NN(\mathbf{h}(t))[:, j]$ $(j = 1, 2, 3, 4)$ on the y-axis in Figure 6 represents the jth element of the output of the trained nets $NN(\cdot)$ fed by the predicted state $\mathbf{h}(t)$.

One can see that the discrepancy terms in scheme 1 (Figure 6a) are captured by the neural network, while mild "mismatch" is still observed. This can be used to explain the sub-par

predictive capability of scheme 1, as shown in Figure 4 and 5. It also demonstrates that if the governing dynamics (or the discrepancy dynamics) are not fully captured, the trained model only serves for interpolation of the training data, with less potential in its exploitation for prediction given new initial or forcing conditions. Schemes 2 and 3 (Figure 6b and 6c) yield a higher precision, almost perfectly capturing the behavior of the reference (measurements) with no evident "mismatch". Hence, the proposed framework achieves effective discrepancy modeling given different levels of prior knowledge of the system, and the Physics-informed neural ODEs can be sufficiently trained to adaptively represent this discrepancy. In addition, it is noted that the architecture of the neural network $NN(\cdot)$ only features one hidden layer and the size of the training data set is plausibly small, which demonstrates the possibilities of using shallow neural networks and small data set to train the Physics-informed Neural ODEs.



(a) scheme 1 (no physics informed)

(b) scheme 2 (physics weakly informed)

(c) scheme 3 (physics strongly informed)

Figure 6: Outputs of the trained $NN(\mathbf{h}(t))$ *versus* each inter-DOF displacement $(x_i(t) - x_{i-1}(t))$ for the three schemes listed in Table 1, compared against their corresponding references.

In this example, following the proposed framework in Section 2.2.2, we undertake a further step for schemes 2 and 3 with the purpose of translating the model inferred by the trained neural networks into closed-form expressions via sparse identification. The reference, or true,

discrepancy term listed in Table 2 is obtained as the difference between true model $f_{true}(\cdot)$ in Eq.(12) and prior models $f_{phy}(\cdot)$ with the correspondence to the acceleration vector $x_i(t)$ ($i = 1, 2, 3, 4$). Table 2 compares the true and recovered discrepancy terms, as obtained from the sparse identification for schemes 2 and 3. It is evident that the recovered closed-form expressions comprise the appropriate terms, with the respective identified coefficients lying close to the correct ones. This transparentization step serves to offer a more mechanistic insight as opposed to a "black-box" neural network.

Table 2: Recovered discrepancy models via sparse identification

		True discrepancy model	Recovered discrepancy model
scheme 2	$\ddot{x}_1$	$-6x_1 + 3x_2 - 0.15\dot{x}_1 - 2x_1^3$	$-6.002x_1 + 2.998x_2 - 0.149\dot{x}_1 - 1.994x_1^3$
	$\ddot{x}_2$	$3x_1 - 6x_2 + 3x_3 - 0.15\dot{x}_2$	$3.008x_1 - 6.000x_2 + 2.999x_3 - 0.151\dot{x}_2$
	$\ddot{x}_3$	$3x_2 - 6x_3 + 3x_4 - 0.15\dot{x}_3$	$3.006x_2 - 5.975x_3 + 2.969x_4 - 0.151\dot{x}_3$
	$\ddot{x}_4$	$3x_3 - 3x_4 - 0.15\dot{x}_4$	$2.993x_3 - 2.982x_4 - 0.149\dot{x}_4$
scheme 3	$\ddot{x}_1$	$-2x_1^3$	$-1.996x_1^3$
	$\ddot{x}_2$	0	0
	$\ddot{x}_3$	0	0
	$\ddot{x}_4$	0	0

3.1.2 Forced vibration case

The proposed framework is here demonstrated for the forced vibration case with known input excitation. We implement this for the same system presented in Section 3.1.1, where an external force is applied on the first DOF. In the training stage for this case, a white-noise excitation is used for a duration of 4 seconds at a 100 Hz sampling rate. We scale the excitation across four different levels of amplitudes from low to high for training, resulting in an adequate training dataset in order to successfully train the neural network in capturing the cubic nonlinearity. Scheme 3, listed in Table 1, is adopted for implementing the Physics-informed Neural ODEs, where the true linear transform matrix is assumed to be available. Upon completion of the training, the trained Neural ODEs are first used to predict the structural response (displacement, velocity, and acceleration) of the system under a white noise excitation, which is used for training, as shown in Figure 7. As observed, the response of the first mass ($x_1(t)$, $\dot{x}_1(t)$, and $\ddot{x}_1(t)$) closely approximates the reference ground truth. We also present the force-displacement behavior of the first mass m_1 derived from the trained Physics-informed Neural ODEs (we sum up the products of acceleration at each DOF and its mass, and plot the summation versus the displacement x_1),

in comparison to the ground truth derived from the analytical model. It is observed that the recovery is capable of capturing the major behavior of the cubic nonlinearity of the system, though the two tips of the loops are not perfectly captured.

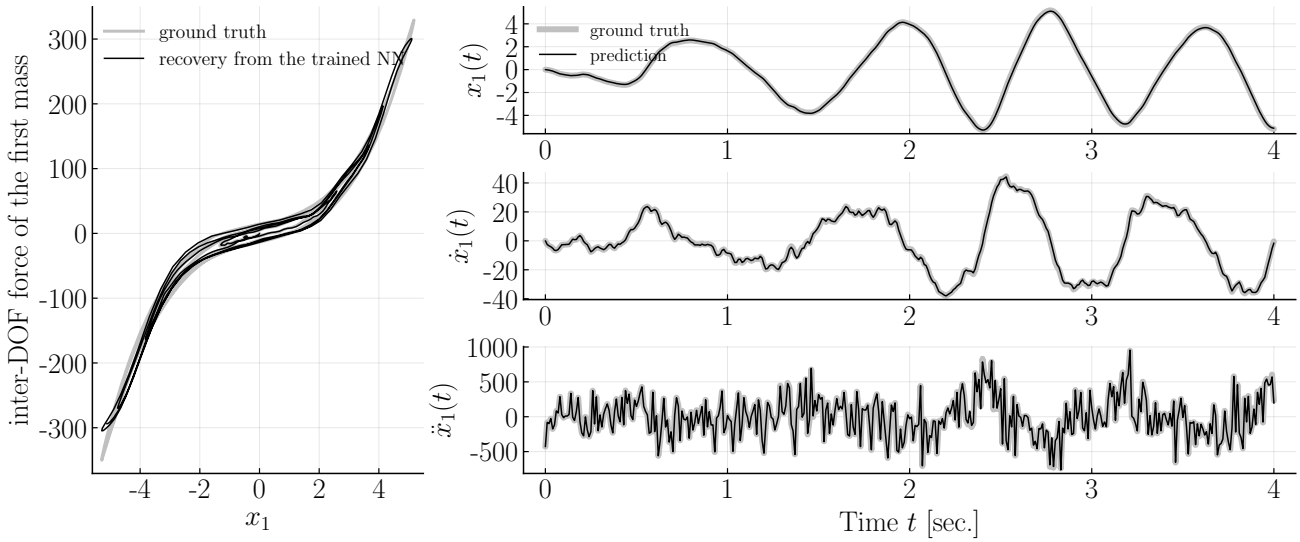


Figure 7: (Left): the force-displacement loops of the first DOF; (right): displacement, velocity and acceleration response estimation for the first degree of freedom, using the excitation applied for training.

In the testing (validation) stage, we apply i) a white-noise excitation that is different from the one used for training, and ii) a record of the El-centro earthquake, i.e., a non-white excitation to the first DOF of the system, respectively. The predicted time history of $x_1(t)$, $\dot{x}_1(t)$, and $\ddot{x}_1(t)$ are now illustrated in Figures 8 and 9, respectively. As observed, the estimated response does follow the dynamics of the system as the governing dynamics are closely approximated by the neural network embedded in the ODEs, albeit the generalization ability of the trained model in this forced random vibration case proves weaker than the one in the free vibration case. Let's revisit Eq.(4), and for a forced vibration case, one also integrate the excitation to predict the state, potentially introducing added numerical error to the state prediction.

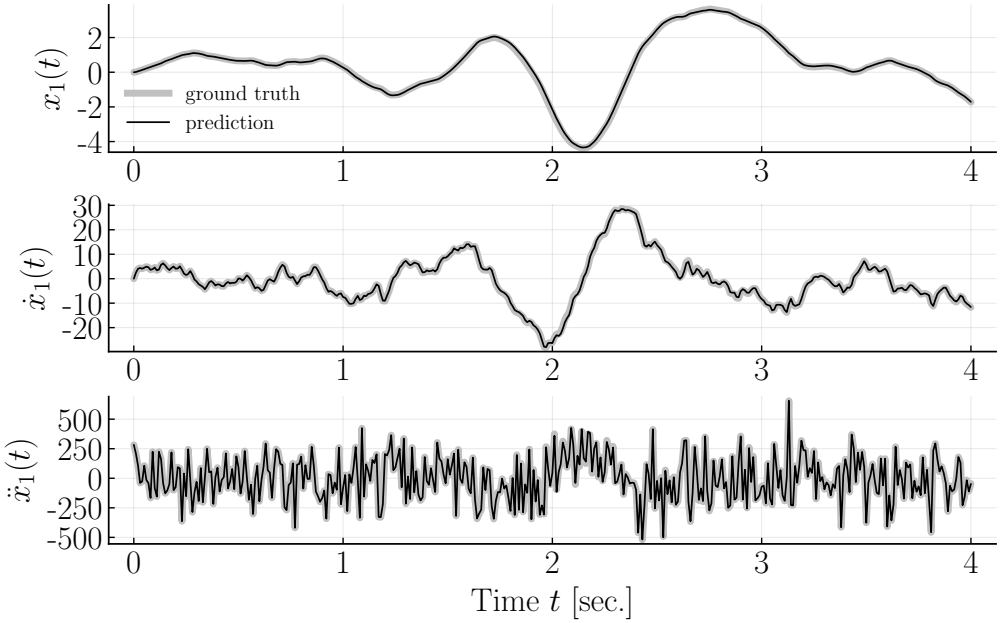


Figure 8: Validation I: displacement, velocity and acceleration response for the first degree of freedom using a white-noise excitation that is different to the ones adopted for training.

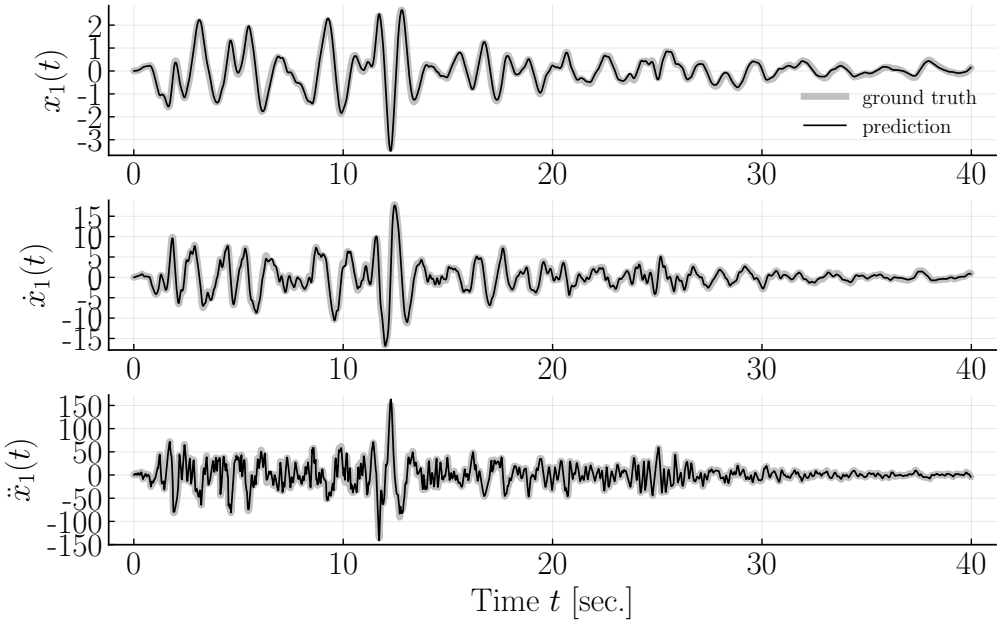


Figure 9: Validation II: displacement, velocity and acceleration response for the first degree of freedom using a record of the El-centro earthquake that is different to the one adopted for training.

4. EXPERIMENTAL EXAMPLES

4.1 A Structural System Equipped with a Negative Stiffness Device

To further validate the proposed framework via use of experimental data, we employ the proposed scheme to learn the governing dynamics of a structure equipped with a negative stiffness device (NSD)[52–57] serving for vibration mitigation, which results in a highly nonlinear dynamical system. As shown in Figure 10, the structure is a three-story frame that has been tested on a single-axis shake table. The structure is subjected to ground motion, thus following a forced vibration response. In between the first floor and the shake table, a negative stiffness device was installed introducing a modification to the primary force-displacement behavior of the first floor. Figure 11b schematically presents the mechanical behavior of the NSD in terms of the delivered force-displacement loops; the negative stiffness effect is activated when the displacement (or inter-story drift) is larger than a pre-defined value (note that the value of 0.2 inch is adopted as the pre-defined value in this experimental testing). As shown in Figure 11c, a linear primary structure (illustrated in Figure 11a) equipped with a NSD delivers an almost bi-linear behavior. A further description of the model can be found in the corresponding report[53] of the Multidisciplinary Center for Earthquake Engineering Research (MCEER), established at University at Buffalo.

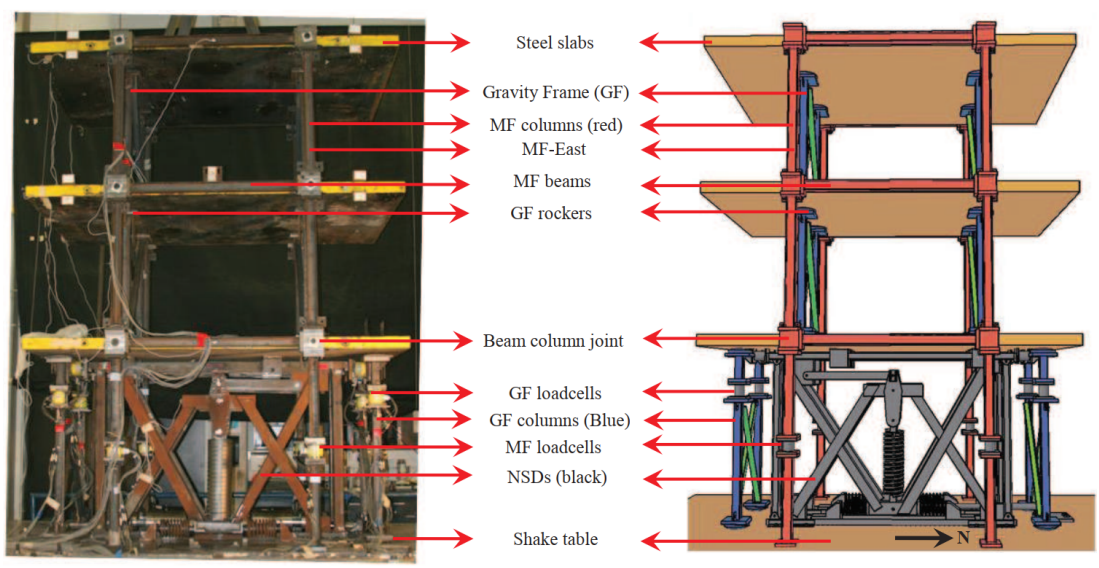
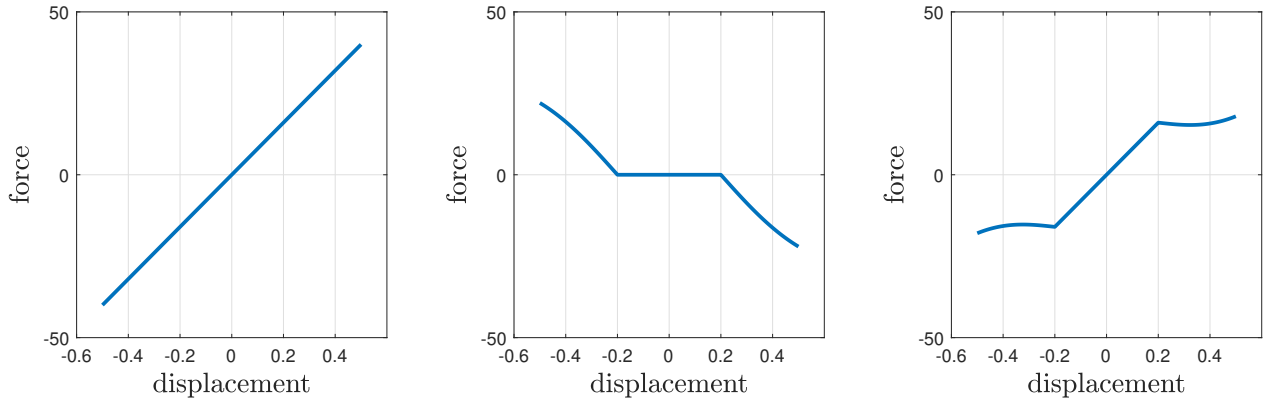


Figure 10: The structural system equipped with a negative stiffness device in between the first floor and the shake table.



(a) primary structure (b) NSD (c) primary structure + NSD

Figure 11: Schematic illustration of the force-displacement behaviors of (a) a linear primary structure; (b) NSD; and (c) primary structure installed with NSD

This structure is heavily instrumented with linear variable differential transformers (measuring displacements), strain gauges, and accelerometers. The measured ground acceleration provided by the shake table and the time histories of the displacement response at each slab are used to train a Physics-informed Neural ODE. The dynamical system is simplified into a 3-DOF spring-mass model. The velocities for each slab are computed through numerical differentiation followed by a low-pass filtering.

4.2 Implementation of Physics-informed Neural ODEs

In the training stage, two phases are distinguished for learning the dynamics of this system: (1) Phase 1 (model updating); (2) Phase 2 (nonlinear component (the NSD) modeling). These are described in detail in what follows.

4.2.1 Phase 1: the structure without NSD installed

In Phase 1, we consider the structure *without* NSD installed. A 3-DOF spring-mass dynamical system is considered to represent the system. The equations of motion are established by defining a state vector $\mathbf{h}(t) = [x_1(t)\ x_2(t)\ x_3(t)\ \dot{x}_1(t)\ \dot{x}_2(t)\ \dot{x}_3(t)]^T \in \mathbb{R}^{6\times 1}$, where $x_i(t)$ and $\dot{x}_i(t)$ $(i = 1, 2, 3)$ are the relative displacement and velocity of the ith slab, respectively:

$$\frac{d\mathbf{h}(t)}{dt} = f_{phy}(\mathbf{h}(t)) + \begin{bmatrix} \mathbf{0} \\ NN_1(\mathbf{h}(t)) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{1}\ddot{x}_g(t) \end{bmatrix} \quad \text{(Phase 1)} \tag{15}$$

or explicitly:

$$\frac{d\mathbf{h}(t)}{dt} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \mathbf{h}(t) + \begin{bmatrix} \mathbf{0} \\ NN_1(\mathbf{h}(t)) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{1}\ddot{x}_g(t) \end{bmatrix} \quad \text{(Phase 1)} \tag{16a}$$

in which, $\mathbf{0} \in \mathbb{R}^{3\times 1}$ is a zero vector; $\mathbf{1} \in \mathbb{R}^{3\times 1}$ denotes a vector of all ones; $\ddot{x}_g(t)$ is the ground acceleration provided by the shake table; the mass matrix $\mathbf{M}$, stiffness matrix $\mathbf{K}$, and damping matrix $\mathbf{C}$ are given by:

$$\mathbf{M} = \frac{1}{386}\begin{bmatrix} 9.2 & 0 & 0 \\ 0 & 9.2 & 0 \\ 0 & 0 & 9.0 \end{bmatrix} \text{ (unit: lbs); } \mathbf{K} = \begin{bmatrix} 20.26 & -11.16 & 0 \\ -11.16 & 20.33 & -9.17 \\ 0 & -9.17 & 9.17 \end{bmatrix} \text{ (unit: kips/inch);}$$
$$\mathbf{C} = \begin{bmatrix} 0.015 & 0 & 0 \\ 0 & 0.015 & 0 \\ 0 & 0 & 0.015 \end{bmatrix} \text{ (unit: inch/sec.}^2\text{)} \tag{16b}$$

The physics-informed term $f_{phy}(\cdot)$ retains the *prior* knowledge of mass, stiffness, and damping coefficients that are reported in the literature,[53] illustrated in Eq.(16). It is noted that a highly accurate representation of $\mathbf{M}$, $\mathbf{K}$, and $\mathbf{C}$ is not necessary, since the supplemental term $NN_1(\cdot)$ is able to learn the discrepancy by training to update the initial model. The architecture of the neural network $NN_1(\cdot)$ features only one hidden layer comprising 10 neurons, and no nonlinear activation functions are used since we only consider a liner dynamical system in this case, as illustrated in Table 3. Similar to the numerical example, the input for $NN_1(\cdot)$ is the state variable $\mathbf{h}(t) \in \mathbb{R}^{6\times 1}$, and the output is only three-dimensional, contributing to the last three elements of $\frac{d\mathbf{h}(t)}{dt}$ (e.g., acceleration vector $[\ddot{x}_1(t), \ddot{x}_2(t), \ddot{x}_3(t)]^T$), since the first three equations in Eq.(16a) are always by default identities (in the continuous domain) where no updating is required.

Table 3: Physics-informed Neural ODEs for the NSD experiment.

initial model $f_{phy}(\mathbf{h}(t))$	$\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \mathbf{h}(t)$
$NN_1(\mathbf{h}(t)) : \mathbb{R}^6 \to \mathbb{R}^3$	$\mathbf{W}^{(2)T}(\mathbf{W}^{(1)T}\mathbf{h} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$ number of neurons in hidden layers: 10
$NN_2(\mathbf{h}(t)) : \mathbb{R}^6 \to \mathbb{R}^1$	$\bar{\mathbf{W}}^{(3)T}\sigma[\bar{\mathbf{W}}^{(2)T}\sigma(\bar{\mathbf{W}}^{(1)T}\mathbf{h} + \bar{\mathbf{b}}^{(1)}) + \bar{\mathbf{b}}^{(2)}] + \bar{\mathbf{b}}^{(3)}$ number of neurons in hidden layers: 20

A scaled record of the Kobe earthquake and the Pacoima earthquake was used as the ground motion for the shake table. Under this scaled low-amplitude earthquake excitation, the structure remains in the linear regime. Thus, a linear neural network is sufficient, which is also easier to interpret. Additionally to ground motion measurements, the measured displacement data $x_1(t), x_2(t), x_3(t)$ at a 256 Hz sampling rate and their corresponding computed velocities $\dot{x}_1(t), \dot{x}_2(t), \dot{x}_3(t)$ are used for training the Physics-informed Neural ODEs.

Figure 12 compares the measured time history of the displacement and velocity of the first degree of freedom when the structure is subjected to the scaled Kobe earthquake, with the computed predictions from the initial model and the ones from the initial model supplemented by $NN_1(\cdot)$. The initial model is capable of capturing the major dynamics of the system while discrepancy can still be observed. As observed, the amplitudes of the response decay faster than the measured ones, indicating that the damping matrix is not appropriately estimated in the initial model. It shows that the prediction of the response (the black dashed line) via the trained Neural ODEs of Eq.(15) agrees highly with the corresponding measured data. Only a limited data set is available for this linear regime testing. Thus, in order to validate the generalization ability of the trained model, we extrapolate the prediction for a duration of 10 seconds beyond the training data set. The obtained results indicate that the prediction via the trained Neural ODEs still follows the dynamics of the measured data with no evident inconsistent. This demonstrates that the updated model (initial model supplemented by the trained discrepancy term NN_1) is capable of representing the structural system in Phase 1.

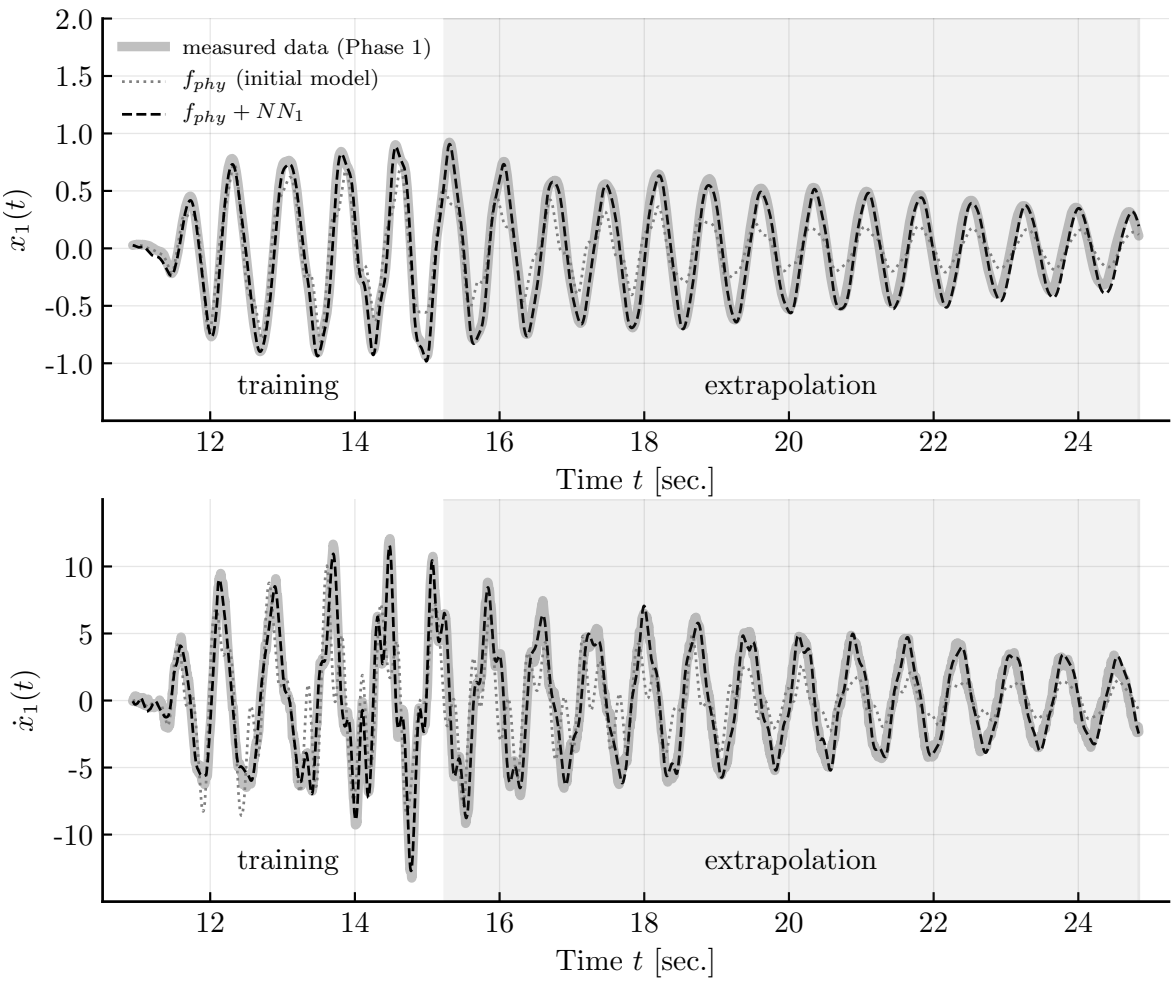


Figure 12: Comparison of time history of the response for displacement $x_1(t)$ and velocity $\dot{x}_1(t)$ for the NSD experiment (Phase 1).

Let's take a further step to interpret the trained $NN_1(\cdot)$ term. Being aware that $NN_1(\cdot)$ merely comprises a linear transformation (no activation functions are used in $NN_1(\cdot)$), Eq.(16) can be expanded as:

$$\frac{d\mathbf{h}(t)}{dt} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \mathbf{h}(t) + \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbb{W}_1 & \mathbb{W}_2 \end{bmatrix} \mathbf{h}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{W}^{(2)T}\mathbf{b}^{(1)} + \mathbf{b}^{(2)} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{1}\ddot{x}_g(t) \end{bmatrix} \tag{17a}$$

where $\mathbb{W}_1 \in \mathbb{R}^{3\times3}$ and $\mathbb{W}_2 \in \mathbb{R}^{3\times3}$ are the first three columns and the last three columns of $\mathbf{W}^{(2)T}\mathbf{W}^{(1)T}$, respectively:

$$\mathbf{W}^{(2)T}\mathbf{W}^{(1)T} = \begin{bmatrix} \mathbb{W}_1 & \mathbb{W}_2 \end{bmatrix} \tag{17b}$$

As is evident, an updated stiffness matrix is conveniently delivered as $\mathbf{K} - \mathbf{M}\mathbb{W}_1$, with a similar procedure being applicable for updating the damping matrix as $\mathbf{C} - \mathbf{M}\mathbb{W}_2$. Table 4 lists the

updated stiffness and damping matrices. Note that the updated stiffness matrix does not deliver a typical symmetric banded matrix as assumed in Eq.(16), and as observed, the updated stiffness is slightly higher than the prior assumption. This also implies that when the actual system is simplified into a 3-DOF spring-mass model, the format of the stiffness matrix might differ from the one derived from the theoretical model. The updated damping matrix delivers non-zero off-diagonal elements that better represent the actual damping of this system, as evidenced in Figure 12 (the decay rate shown in the time history matches the actual measurements).

Table 4: Updated stiffness and damping matrices

Updated stiffness matrix	Updated damping matrix
$\mathbf{K}_{\text{updated}} = \begin{bmatrix} 26.179 & -11.646 & -3.227 \\ -7.987 & 20.510 & -10.838 \\ -7.532 & -9.150 & 13.020 \end{bmatrix}$	$\mathbf{C}_{\text{updated}} = \begin{bmatrix} 0.047 & -0.040 & 0.007 \\ -0.101 & 0.149 & -0.086 \\ 0.273 & -0.343 & 0.169 \end{bmatrix}$

4.2.2 Phase 2: the structural system with NSD installed

After the linear term, $NN_1(\cdot)$, is derived via training in Phase 1, in Phase 2, a new training set is established, this time using data measured from the structure *with* the NSD installed between the first floor slab and the shake table. This results in the first story exhibiting nonlinear behavior. Compared to Phase 1, the dynamical system is now strongly modified and admits a nonlinear constitutive term. In capturing the latter, an additional neural network NN_2, listed in Table 3, that accounts for the mechanical behavior of the installed NSD is supplemented into to Eq.(15), while the learned weights of NN_1 (in Phase 1) are assumed as fixed (no updating is required for the learned weights of NN_1 in Phase 2). It is noted that the output of $NN_2(\cdot)$ is set to be only one-dimensional, since the NSD is installed in the first floor and it is assumed that the introduced modification to the force-displacement behavior only occurs in the first floor (the output of NN_2 only contributes to $\ddot{x}_1(t)$). The Leaky rectified linear unit (Leaky ReLU)[58] scheme is adopted as the activation function in this example.

$$\frac{d\mathbf{h}(t)}{dt} = f_{phy}(\mathbf{h}(t)) + \begin{bmatrix} \mathbf{0}_{3\times1} \\ NN_1(\mathbf{h}(t)) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times1} \\ NN_2(\mathbf{h}(t)) \\ \mathbf{0}_{2\times1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times1} \\ -\mathbf{1}\ddot{x}_g(t) \end{bmatrix} \quad \text{(Phase 2)} \tag{18}$$

In Phase 2, the training data set is derived from the sensor measurement where the ground motion $\ddot{x}_g(t)$ remains the same as the one used earlier the scaled Kobe earthquake and the Pacoima earthquake records. Figure 13 demonstrates the prediction of the first floor displacement and

velocity via use of i) the trained Eq.(15) and ii) the trained Eq.(18) when the ground motion is the Kobe earthquake, compared against the measured data, which serve as reference. As observed, the estimation delivered by the trained Eq.(15) (black dashed line) can no longer follow the reference (nonlinear) behavior, which is strongly modified due to presence of the NSD in Phase 2. By supplementing NN_2, the delivered prediction, which is denoted by the solid blue line indicates a close agreement to the actual measurements, thereby demonstrating the efficacy of the Neural ODE scheme in tracking nonlinear behavior. In addition, similar to Phase 1, the prediction is extrapolated for a duration that is longer to 10 seconds and the trained model ("$f_{phy} + NN_1 + NN_2$") still closely approximates the dynamics of the measured data, while the model derived from Phase 1 ("$f_{phy} + NN_1$") fails to track the behavior. In addition, it is noted that the period of extrapolation is longer than the one of training, further demonstrating the feasibility of using small training data set to successfully train the Physics-informed Neural ODEs. This is of course only valid as long as the training is sufficient to allow capturing the full extend of the nonlinear behavior of the model. In the event of a model featuring by deterioration effects, for example, the required training and underlying state space description would need to be modified. Naturally, what is further of interest from an identification point of view, is whether the NN_2 structure can be deciphered the major dynamics of this highly nonlinear structure.

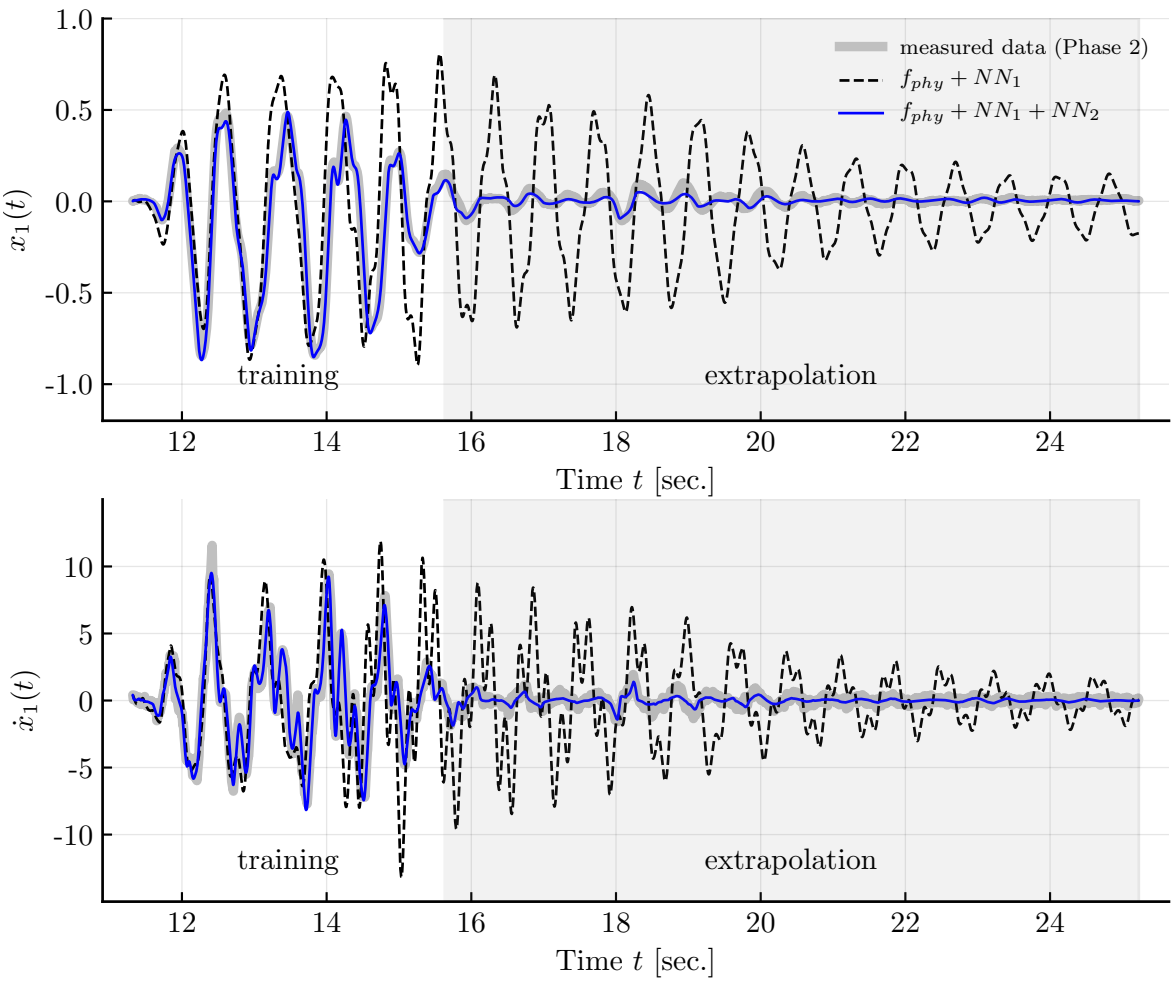


Figure 13: Comparison of time history of the response for displacement $x_1(t)$ and velocity $\dot{x}_1(t)$ for the NSD experiment (Phase 2).

To gain an insight into the trained NN_2 ($\mathbb{R}^6 \to \mathbb{R}^1$), one can feed the measured $\mathbf{h}(t)$ through the network, and obtain the one-dimensional output which is corresponding to the acceleration at the first slab. We multiply this output by the mass of the first slab to recover the force solely exerted by the NSD, and plot the recovery against the inter-story drift $x_1(t)$ in the first floor, as shown in Figure 14. When comparing against the real measurement, as derived from a load cell measuring the forces exerted by the NSD, it occurs that the signal recovered via the NN_2 term follows the major mechanical behavior of the NSD (close to a bilinear behavior). This indicates that the trained nonlinear term is capable of representing the mechanical behavior of the system. However, one can see the slope (the stiffness) in the region of negative displacement is slightly lower than the actual measurements. In the loss function of Neural ODEs shown in Eq.(4), only displacement and velocity data are taken into account. This leads to information that is possibly imprinted in the acceleration, such as high frequency components of the response, to

not be adequately captured by the neural networks. As manifested at around the 12.5 seconds of the time history of $\dot{x}_1(t)$ in Figure 13, it is observed that the sharp peak is not fully captured by the trained model.

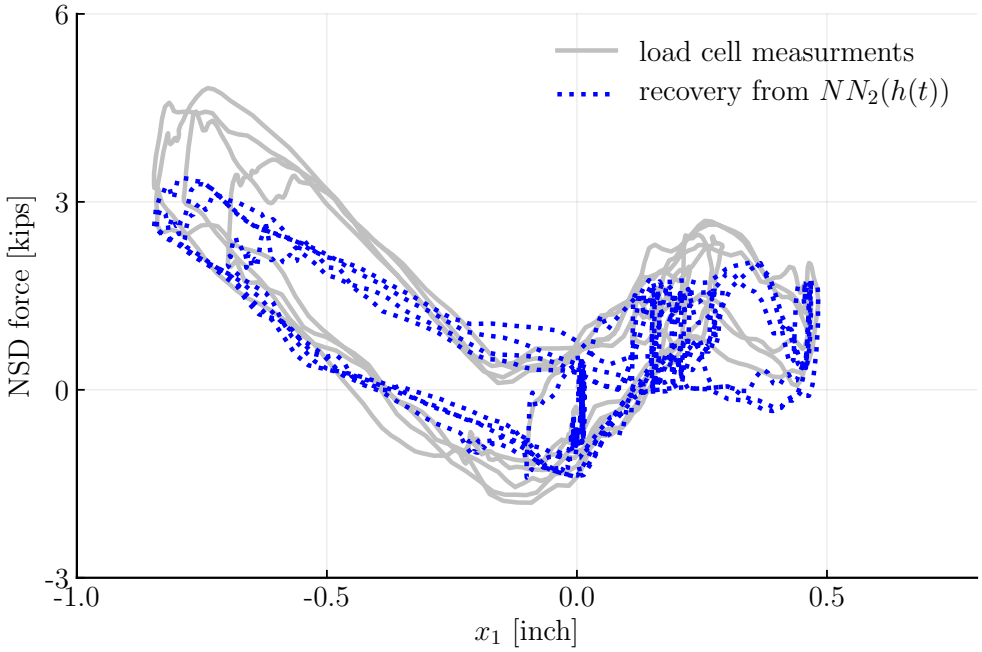


Figure 14: Comparison of the NSD forces estimated via the nonlinear term NN_2, and the measurement delivered by the load cell.

5. CONCLUSIONS

This paper investigates and further demonstrates a promising direction of linear/nonlinear structural identification via use of Physics-informed Neural ODEs. The ODE delivers a mathematical description of the state-space function of a dynamical system. Prior domain knowledge of the dynamical system can be conveniently embedded in such a formulation, as demonstrated in this work, while remaining terms in the inferred representation are captured by a feed-forward neural network, which yields hybrid ODE formulation, herein termed Physics-informed Neural ODEs. The presented numerical and experimental examples in this paper demonstrate the benefits of the proposed framework in the direct approximation of the governing dynamics, offering a versatile framework for discrepancy modeling. A transparentization of the trained neural network can be achieved by coupled implementation of a sparse identification scheme, which distills closed-form expressions from the derived nonlinear neural network terms, thus enhancing engineering insight in the workings of the model as opposed to the more blind approximation delivered by a purely "black-box" model. This paper presents a proof-of-concept for application of the framework in the domain of system identification. Next steps include work towards a hybrid

modeling technique, concatenating high-fidelity finite element models with Neural ODEs. A key limitation in structural identification via Neural ODEs lies in the requirement that the full state (displacement and velocity) should be available (measured).

REFERENCES

[1] Roger Ghanem and Masanobu Shinozuka. Structural-system identification. i: Theory. *Journal of Engineering Mechanics*, 121(2):255–264, 1995.

[2] Lennart Ljung. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pages 1–19, 1999.

[3] KF Alvin, AN Robertson, GW Reich, and KC Park. Structural system identification: from reality to models. *Computers & structures*, 81(12):1149–1176, 2003.

[4] Gaetan Kerschen, Keith Worden, Alexander F Vakakis, and Jean-Claude Golinval. Past, present and future of nonlinear system identification in structural dynamics. *Mechanical systems and signal processing*, 20(3):505–592, 2006.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[6] Charles R Farrar and Keith Worden. *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012.

[7] Osama Abdeljaber, Onur Avci, Serkan Kiranyaz, Moncef Gabbouj, and Daniel J Inman. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, 388:154–170, 2017.

[8] Chathurdara Sri Nadith Pathirage, Jun Li, Ling Li, Hong Hao, Wanquan Liu, and Pinghe Ni. Structural damage identification based on autoencoder neural networks and deep learning. *Engineering Structures*, 172:13–28, 2018.

[9] Byung Kwan Oh, Branko Glisic, Sang Wook Park, and Hyo Seon Park. Neural network-based seismic response prediction model for building structures using artificial earthquakes. *Journal of Sound and Vibration*, 468:115109, 2020.

[10] Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5):620–627, 1985.

[11] Rune Brincker, Lingmi Zhang, and Palle Andersen. Modal identification of output-only systems using frequency domain decomposition. *Smart materials and structures*, 10(3):441, 2001.

[12] Adel Belouchrani, Karim Abed-Meraim, J-F Cardoso, and Eric Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on signal processing*, 45(2):434–444, 1997.

[13] Gaëtan Kerschen, Fabien Poncelet, and J-C Golinval. Physical interpretation of independent component analysis in structural dynamics. *Mechanical Systems and Signal Processing*, 21(4):1561–1575, 2007.

[14] Yongchao Yang and Satish Nagarajaiah. Time-frequency blind source separation using independent component analysis for output-only modal identification of highly damped structures. *Journal of Structural Engineering*, 139(10):1780–1793, 2013.

[15] Satish Nagarajaiah and Yongchao Yang. Modeling and harnessing sparse and low-rank data structure: a new paradigm for structural dynamics, identification, damage detection, and health monitoring. *Structural Control and Health Monitoring*, 24(1):e1851, 2017.

[16] Jann N Yang, Silian Lin, Hongwei Huang, and Li Zhou. An adaptive extended kalman filter for structural damage identification. *Structural Control and Health Monitoring: The Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures*, 13(4):849–867, 2006.

[17] Eleni N Chatzi and Andrew W Smyth. The unscented kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. *Structural Control and Health Monitoring: The Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures*, 16(1):99–123, 2009.

[18] AE Charalampakis and CK Dimou. Identification of bouc–wen hysteretic systems using particle swarm optimization. *Computers & structures*, 88(21-22):1197–1205, 2010.

[19] Saeed Eftekhar Azam, Eleni Chatzi, and Costas Papadimitriou. A dual kalman filter approach for state estimation via output-only acceleration measurements. *Mechanical Systems and Signal Processing*, 60:866–886, 2015.

[20] Zhilu Lai, Ying Lei, Songye Zhu, You-Lin Xu, Xiao-Hua Zhang, and Sridhar Krishnaswamy. Moving-window extended kalman filter for structural damage detection with unknown process and measurement noises. *Measurement*, 88:428–440, 2016.

[21] Ying Lei, Dandan Xia, Kalil Erazo, and Satish Nagarajaiah. A novel unscented kalman filter for recursive state-input-system identification of nonlinear systems. *Mechanical Systems and Signal Processing*, 127:120–135, 2019.

[22] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[23] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, page 201517384, 2016.

[24] Zhilu Lai and Satish Nagarajaiah. Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior. *Mechanical Systems and Signal Processing*, 117:813–842, 2019.

[25] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[26] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[27] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Advances in Neural Information Processing Systems*, pages 3134–3144, 2019.

[28] Mattias Fält and Pontus Giselsson. System identification for hybrid systems using neural networks. *arXiv preprint arXiv:1911.12663*, 2019.

[29] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *arXiv preprint arXiv:2002.08071*, 2020.

[30] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[31] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

[32] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[33] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[34] 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.

[35] Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. *arXiv preprint arXiv:2002.10253*, 2020.

[36] Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, 215:110704, 2020.

[37] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *IEEE Access*, 8:42200–42216, 2020.

[38] Zhiming Zhang and Chao Sun. Structural damage identification via physics-guided machine learning: a methodology integrating pattern recognition with finite element model updating. *Structural Health Monitoring*, page 1475921720927488, 2020.

[39] Soheil Sadeghi Eshkevari, Martin Takáč, Shamim N Pakzad, and Majid Jahani. Dynnet: Physics-based neural architecture design for linear and nonlinear structural response modeling and prediction. *arXiv preprint arXiv:2007.01814*, 2020.

[40] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[41] Zhilu Lai and Satish Nagarajaiah. Semi-supervised structural linear/nonlinear damage detection and characterization using sparse identification. *Structural Control and Health Monitoring*, 26(3):e2306, 2019.

[42] Brian De Silva, David M Higdon, Steven L Brunton, and J Nathan Kutz. Discovery of physics from data: Universal laws and discrepancies. *Frontiers in Artificial Intelligence*, 3:25, 2020.

[43] Manuel A. Roehrl, Thomas A. Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. *arXiv preprint arXiv:2005.14617*, 2020.

[44] 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.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[47] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[48] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[49] Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[50] Chris Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux. jl-a julia library for neural differential equations. *arXiv preprint arXiv:1902.02376*, 2019.

[51] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020.

[52] Apostolos A Sarlis, Dharma Theja R Pasala, MC Constantinou, AM Reinhorn, Satish Nagarajaiah, and DP Taylor. Negative stiffness device for seismic protection of structures. *Journal of Structural Engineering*, 139(7):1124–1133, 2013.

[53] DTR Pasala, AA Sarlis, Satish Nagarajaiah, AM Reinhorn, MC Constantinou, and Douglas Taylor. Seismic response control of structures using a novel adaptive passive negative stiffness device. *MCEER Report 13-0004*. available at http://www.buffalo.edu/mceer/catalog.host.html/content/shared/www/mceer/publications/MCEER-13-0004.detail.html.

[54] DTR Pasala, AA Sarlis, S Nagarajaiah, AM Reinhorn, MC Constantinou, and D Taylor. Adaptive negative stiffness: New structural modification approach for seismic protection. *Journal of structural Engineering*, 139(7):1112–1123, 2013.

[55] Dharma Theja Reddy Pasala, Apostolos Aristotelis Sarlis, Andrei M Reinhorn, Satish Nagarajaiah, MC Constantinou, and Douglas Taylor. Simulated bilinear-elastic behavior in a sdof elastic structure using negative stiffness device: Experimental and analytical study. *Journal of Structural Engineering*, 140(2):04013049, 2014.

[56] DTR Pasala, AA Sarlis, AM Reinhorn, S Nagarajaiah, MC Constantinou, and D Taylor. Apparent weakening in sdof yielding structures using a negative stiffness device: Experimental and analytical study. *Journal of Structural Engineering*, 141(4):04014130, 2015.

[57] Zhilu Lai, Tong Sun, and Satish Nagarajaiah. Adjustable template stiffness device and sdof nonlinear frequency response. *Nonlinear Dynamics*, 96(2):1559–1573, 2019.

[58] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.