Chapter 6
Physics-Informed Machine Learning Part III: Hard-Constraint ODE Method for Structural Dynamics

Mohsen Gol Zardian, Austin R.J. Downey, Eleonora Maria Tronci, Conor Madden, Daniel Coble, Sina Navidi, and Chao Hu

Abstract Physics-informed machine learning (PIML) is a methodology that combines principles from physics with machine learning (ML) techniques to enhance the accuracy and interpretability of predictive models. By incorporating physical laws and constraints into the learning process, physics-informed machine learning enables more robust predictions and reduces the need for large amounts of training data. PIML has a wide range of applications in science and engineering, such as modeling physical systems, solving partial differential equations, and performing inverse analysis and optimization.

In Part III of this series, the authors provide an overview of a hard-constraint, indirect-measurement PIML approach for structural dynamics. An ordinary differential equation (ODE) solver implemented within the automatic differentiation engine serves as a differentiable layer that enforces the mass-damping-stiffness equilibrium at every inference step. A compact neural network ingests a window of past acceleration samples and outputs an effective stiffness. The embedded Runge-Kutta-4 integrator then propagates states so that the predicted acceleration satisfies the equation of motion. The technique is demonstrated on a deliberately simple numerical example: a simulated spring, mass, and dashpot whose stiffness drifts over time along several representative paths. A collection of runs is generated, some used for training, the rest held back for validation so that the model must generalize to unseen stiffness trajectories. Performance is benchmarked against two reference solvers. First, a physics-only baseline consisting of a conventional ODE integrator that assumes the stiffness is fixed and therefore cannot reflect its time variation. Second, a data-only baseline consisting of a neural network of identical size trained to minimize next-step acceleration error, without any governing-equation guidance. Results show that the hard-constraint PIML model approaches the data-only network in response accuracy while offering far greater physical interpretability through its explicit stiffness estimates. The paper reports on hyper-parameter tuning for the multi-layer perceptron so readers can readily

Mohsen Gol Zardian · Conor Madden · Daniel Coble
Department of Mechanical Engineering
University of South Carolina, Columbia, SC 29208
e-mail: golzardm@email.sc.edu; CMMADDEN@email.sc.edu; daniel.coble@duke.edu

Austin R.J. Downey
Department of Civil and Environmental Engineering
University of South Carolina, Columbia, SC 29208
e-mail: austindowney@sc.edu

Eleonora Maria Tronci
Department of Civil and Environmental Engineering
New York University
e-mail: e.tronci@northeastern.edu

Daniel Coble
Department of Mechanical Engineering & Materials Science
Duke University
e-mail: daniel.coble@duke.edu

Sina Navidi · Chao Hu
School of Mechanical, Aerospace, and Manufacturing Engineering
University of Connecticut
e-mail: sina.navidi@uconn.edu; chao.hu@uconn.edu

© The Author(s), under exclusive license to River Publishers 2026

Kristin B. Zimmerman, PhD (eds.), *Data Science in Engineering, 2026, Vol. 9*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.13052/rp-9788743814207

adapt the hard-constraint indirect-measurement PIML method to their own inverse-problem applications. Project code are artifacts available are through thorough a public repository.

Keywords Physics-informed · physics-based model · data driven model · machine learning · stiffness identification

Introduction

The rapid growth of sensing technologies and advances in machine learning (ML) have opened up new possibilities for modeling and predicting how engineering systems behave. In structural dynamics and health monitoring, data-driven models have proven highly successful by learning patterns directly from large datasets [1]. However, when it comes to systems such as those encountered in structural dynamics, purely data-driven approaches face two major challenges: they often require far more data than is practical to collect, and their predictions can sometimes violate the relevant physics [2]. These limitations have led to the emergence of physics-informed machine learning (PIML) [3, 4], an approach that blends data-driven modeling with the governing equations [5] of the system to deliver both accurate and physically meaningful predictions of a structures repose on state [6].

Recent work by the authors has explored different ways to integrate physical knowledge into ML models. In Part I of this series, they proposed how embedding domain equations can enhance robustness, improve interpretability, and boost predictive performance in engineering applications [7]. In Part II, they demonstrated the practical application of PIML techniques to structural dynamics, showing how physics-informed architectures can forecast system responses [8]. Their work illustrates the advantages of integrating physical models with neural networks for the inverse challenge of system parameter estimation.

In many fields, data–driven models have achieved remarkable success by capturing patterns directly from large datasets. However, for physical systems such as structural dynamics, purely data–driven approaches face two important limitations: they often require more data than is practical to collect, and they may generate predictions that are inconsistent with known physical laws. This mismatch has motivated the development of physics–informed machine learning (PIML), which integrates governing equations into ML training to achieve both accuracy and interpretability [9].

PIML has already demonstrated its value in applications ranging from fluid mechanics to materials science. In these areas, physical constraints can be incorporated in two main ways. In soft–constraint approaches, the residual of the governing equations is added to the loss function, alongside the mismatch between real and predicted states (As discussed in Part IV of this series [10]). By contrast, hard–constraint approaches embed the equations directly into the model architecture, ensuring that physical laws are satisfied exactly during both training and validation [11]. While harder to design, this strategy is especially important in domains where reliability and physical consistency are critical.

The rapid progress of differentiable solvers has made these ideas more practical. Neural ordinary differential equations (Neural ODEs) [12] allow parameters of dynamical systems to be learned by backpropagating through numerical integrators, while more recent methods have improved stability and efficiency. At the same time, learning frameworks such as DeepONet [13] and Fourier Neural Operators (FNOs) [14] have emerged as powerful tools for approximating families of differential equation solutions. Together, these advances show that combining learning with physics can reduce data requirements, improve generalization, and yield models that are more trustworthy for engineering applications.

Structural dynamics presents a particularly compelling case for PIML. Systems such as bridges, buildings, and aerospace structures are characterized by time–varying properties, nonlinearities, and environmental influences that are difficult to capture with purely physics–based models. At the same time, safety considerations demand models that remain interpretable and physically consistent. Reviews in this area highlight both the potential and the challenges of PIML, noting that soft–constraint formulations often require delicate tuning and may drift from the governing laws when extrapolating [3, 4]. Hard–constraint approaches, by directly encoding the equations of motion, provide a more principled path that ensures predicted responses always respect the underlying mechanics.

In this paper, we focus on the development of a hard–constraint PIML framework for structural dynamics. The method is demonstrated on a mass, spring, and damper system with time-varying stiffness [15], chosen as a benchmark problem to illustrate the challenges of parameter identification under nonstationary conditions. Our framework combines a neural network that estimates effective stiffness from short windows of measured accelerations with a physics layer that propagates states through a numerical integrator. The training objective is defined solely by the mismatch between predicted and observed accelerations, eliminating the need for additional residual penalties. By construction, the governing equation is satisfied at every time step, yielding predictions that are not only accurate but also physically interpretable. We compare the proposed model against physics baselines, demonstrating its advantages in accuracy and robustness. Finally, we discuss sensitivity to hyperparameters to evaluate performance and highlight the broader implications of this approach for structural

health monitoring and other engineering applications where physical consistency is critical. Project code are artifacts are through thorough a public repository for this paper[16].

Methodology

This work develops a hard-constraint physics-informed machine learning (PIML) framework to identify and predict the dynamic response of a mass, spring, and damper system. The central idea is to combine a data-driven neural network with a physics layer such that the governing equation of motion is strictly enforced at every time step. Unlike soft-constraint approaches, where the residual of the governing ODE is included as a penalty term in the loss function, the present framework encodes the ODE directly into the model architecture. As a result, the learned solution is more likely to satisfy the physical law, and training is guided solely by data loss.

Data preparation

The dataset comprises trajectories of displacement, velocity, acceleration, stiffness, and external forcing, obtained from simulated or experimental measurements. Each trajectory is stored in tabular format, where rows correspond to discrete time steps and columns correspond to the measured physical quantities. For clarity, let time be denoted by t, displacement by $x(t)$, velocity by $v(t)$, acceleration by $a(t)$, stiffness by $k(t)$, and external force by $F(t)$.

To reduce redundancy while preserving the essential dynamics, all data is passed through the reprocessing pipeline detailed in Algorithm 1 (summarized in Fig. 1) and discussed in what follows. Trajectories are downsampled by a factor s. A sliding window of length W is then applied to extract local temporal patterns from the acceleration histories. To ensure that every training sample encodes both short- and long-horizon information, consecutive windows are grouped into sequences of length T. To promote generalization, the resulting sequences are randomly shuffled and divided into training and validation sets using a prescribed split ratio ρ. Finally, data is arranged into mini-batches of size B, yielding tensors with dimensions (B, T, W), where B is the batch size, T the number of windows per segment, and W the number of time steps within each window. This organization facilitates efficient parallel training while maintaining sufficient temporal resolution. The overall data preparation process used in this study is illustrated in Fig. 1, which outlines the preprocessing pipeline employed in Algorithm 1 to transform raw sensor trajectories into mini-batches.

Algorithm 1 preprocessing pipeline from raw data to mini-batches

procedure BUILDBATCHES($\{\mathrm{CSV}_n\}_{n=1}^{N}$, s, T, W, B, ρ)
Require: each CSV_n has L rows and D columns (e.g., t, x, v, a, k, F)
Ensure: iterable of train/validation mini-batches with shape (B, T, W)
    counts/params: $N \to$ number of trajectories; $L \to$ rows/trajectory; $D \to$ columns; $T \to$ segment length; $W \to$ window length; $B \to$ batch size
    shape $N \times L \times D$**:** $X \leftarrow \mathrm{Stack}(\mathrm{CSV}_1, \ldots, \mathrm{CSV}_N)$
    downsample by factor s**:** $\tilde{X} \leftarrow \mathrm{Downsample}(X, s)$; $\tilde{L} \leftarrow \mathrm{ceil}(L/s)$
    for $n = 1$ **to** N **do**
        for $t = W$ **to** $\tilde{L}$ **do**
            window length W**:** $a_{t:W}^{(n)} \leftarrow [a_{t-W+1}^{(n)}, \ldots, a_{t}^{(n)}]$
        end for
        for $\tau = 1$ **to** $\tilde{L} - T + 1$ **do**
            segment size $T \times W$**:** $S_{\tau}^{(n)} \leftarrow (a_{\tau:W}^{(n)}, \ldots, a_{\tau+T-1:W}^{(n)})$
        end for
    end for
    total segments: $N_{\mathrm{seg}} \leftarrow N(\tilde{L} - T + 1)$
    split ratio ρ**:** $\{\mathrm{train}\}, \{\mathrm{val}\} \leftarrow \mathrm{Split}(\{S_{\tau}^{(n)}\}, \rho)$
    tensor shape (B, T, W)**:** batches $\leftarrow \mathrm{Batch}(\{\mathrm{train/val}\}, B)$
end procedure

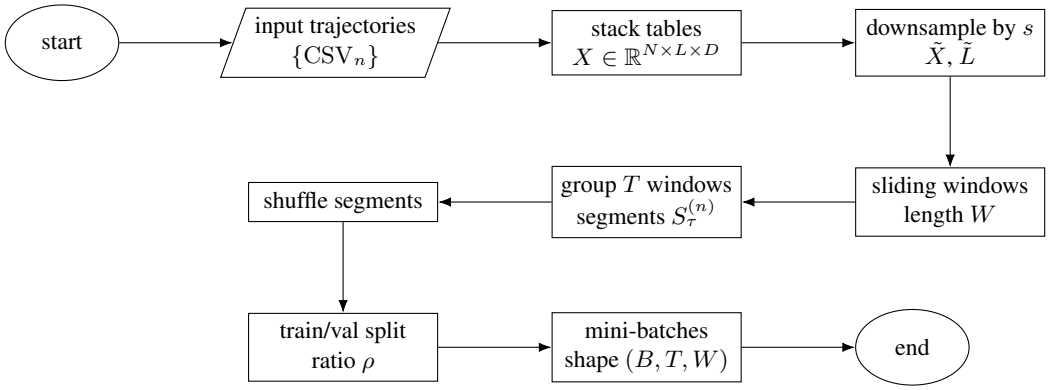


Fig. 1 Preprocessing pipeline from raw trajectories to mini-batches.

Network structure and physics layer

The proposed architecture is designed around the principle of hard-constraint PIML, where the governing physics is enforced directly in the computational structure. As illustrated schematically in Fig. 2, the framework consists of two interconnected components, including a data-driven neural network for parameter estimation and a physics layer that integrates the equations of motion. These components present a hybrid system that combines the flexibility of machine learning with the rigor of classical mechanics.

The data-driven component is a multilayer perceptron (MLP) that receives the windowed acceleration histories of length W as input. The output of this MLP is a variable value of stiffness, $\hat{k}(t)$, which can change across time steps and trajectories. This flexibility enables the model to capture both constant and nonstationary stiffness patterns that may arise from material variability, nonlinearities, or environmental influences.

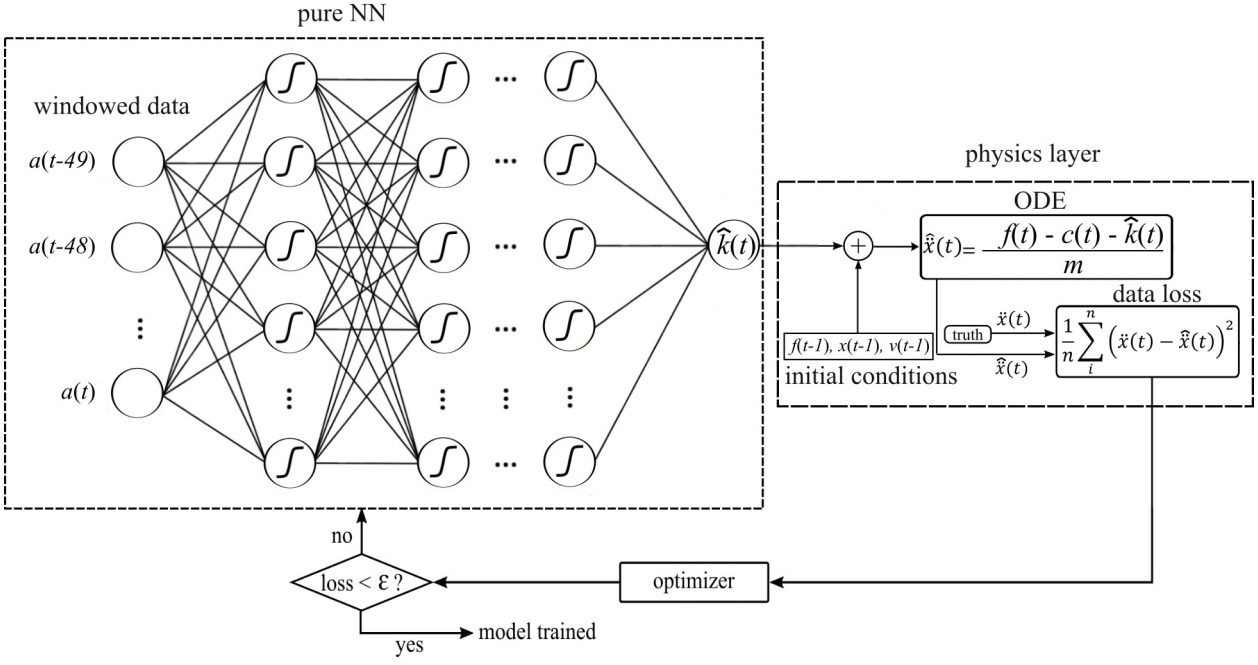


Fig. 2 High-level architecture of the proposed hard-constraint PIML framework. Unlike soft-constraint layouts [9], the governing ODE is directly embedded in the physics layer.

The estimated stiffnesses are then passed to the physics layer, which incorporates the mass, spring, and damper of the motion equation :

$$m\ddot{x} + c\dot{x} + k_{\text{true}}x = F(t) \tag{1}$$

Here, m and c are fixed system parameters, while $\hat{k}(t)$ is provided by the neural network. Given initial conditions and the external force history, the physics layer propagates the system state vector (x, v) forward in time. Numerical integration schemes, such as the fourth-order Runge–Kutta method, are employed to approximate the continuous dynamics with high accuracy. From the updated states, the predicted acceleration $\hat{a}(t)$ is obtained at each time step.

By embedding the ODE solver into the model, the framework ensures that predictions remain consistent with the governing law of motion. Unlike soft-constraint approaches, there is no need to include the residual of the ODE in the loss function, because the residual is automatically satisfied by construction. This design eliminates the need to balance competing loss terms and avoids issues of weighting between data and physics. As a result, the network focuses on learning to map from observed histories to stiffness estimates, while the physics layer guarantees that state updates respect fundamental dynamics. The overall structure achieves a strict hard-constraint formulation that combines physical validity and data-driven adaptability.

To further clarify the inference workflow, Fig. 3 presents the sequential interaction between the neural network and the embedded ODE solver. At each time step t_i, the ML model produces a stiffness estimate $\hat{k}_i$, which is then passed to the ODE solver (here chosen as RK4) together with the external forcing input and the previous system state. The solver propagates displacement and velocity states to the next time step, from which the predicted acceleration is reconstructed. This recursive process enforces the governing dynamics at every step while allowing the network to compute the stiffness evolution from observed acceleration windows.

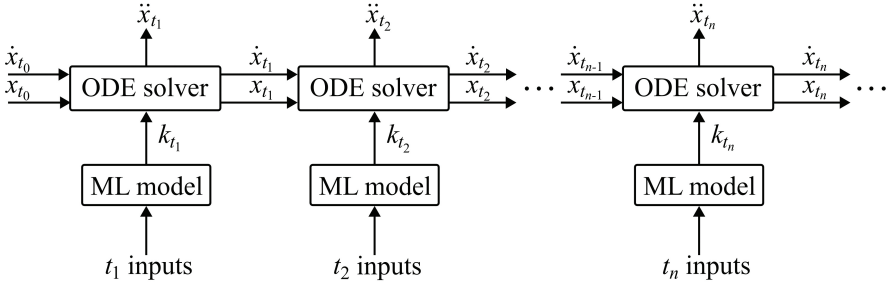


Fig. 3 Detailed workflow of the hard-constraint inference method. The ODE solver is embedded within the automatic differentiation engine, and the ML model generates time-varying stiffness estimates $\hat{k}(t)$ at each step.

Training procedure

The training strategy follows the hard-constraint principle by optimizing only for consistency between predicted and measured accelerations. The training pipeline is summarized in Algorithm 2. At each epoch, the training set is divided into mini-batches, and for each batch the MLP produces variable stiffness $\hat{k}(t)$. These estimates are passed to the physics layer, which integrates the mass, spring, and damper in the equation of motion to propagate displacement and velocity states and compute predicted accelerations $\hat{a}(t)$. The model output is then compared directly with measured accelerations $a(t)$ to compute the loss.

The loss function is defined exclusively as the mean squared error (MSE):

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\ddot{x}(t)_i - \hat{\ddot{x}}(t)_i\right)^2, \tag{2}$$

where N is the number of predicted time steps within a batch. This simple formulation reflects the hard-constraint design, since the governing ODE is embedded in the physics layer, the residual is inherently satisfied and does not need to be included in the loss function. The training objective, therefore, focuses entirely on reducing the difference between observed and predicted accelerations.

Algorithm 2 Physics-informed training with time-distributed MLP

procedure TRAINPINN($\mathcal{B}$, E, η; g_θ, $\mathcal{P}_\phi$, m, c)
Require: batches $\mathcal{B}$ of shape (B, T, W); forces $\{F_t\}$; measured accelerations $\{a_t\}$
Ensure: trained parameters θ
  model: $g_\theta : \mathbb{R}^W \to \mathbb{R}$ predicts $\hat{k}_t$; $\mathcal{P}_\phi$ is a mass–spring–damper step (fixed m, c, e.g., RK4)
  number of batches: $M \to |\mathcal{B}|$
  for $e = 1$ **to** E **do**
    for $j = 1$ **to** M **do**
      select batch: $\mathbf{B} \to \mathcal{B}[j]$
      segment index (parallelizable):
      for $i = 1$ **to** B **do**
        for $t = 1$ **to** T **do**
          stiffness per time step: $\hat{k}_t^{(i)} \leftarrow g_\theta(a_{t:W}^{(i)})$
        end for
        initial state: $s_0^{(i)} \leftarrow (x_0^{(i)}, v_0^{(i)})$
        for $t = 1$ **to** T **do**
          forward physics step: $s_t^{(i)} \leftarrow \mathcal{P}_\phi(s_{t-1}^{(i)}, \hat{k}_t^{(i)}, F_t^{(i)})$
          predicted acceleration: $\hat{a}_t^{(i)} \leftarrow \mathrm{Accel}(s_t^{(i)})$
        end for
        segment loss (MSE): $L_{\mathrm{seg}}^{(i)} \leftarrow \frac{1}{T}\sum_{t=1}^{T}(\hat{a}_t^{(i)} - a_t^{(i)})^2$
      end for
      batch loss: $L_{\mathrm{batch}} \leftarrow \frac{1}{B}\sum_{i=1}^{B} L_{\mathrm{seg}}^{(i)}$
      Adam update: $\theta \leftarrow \mathrm{Adam}(\theta, \nabla_\theta L_{\mathrm{batch}}, \eta)$
    end for
  end for
  return θ
end procedure

Model parameters are updated using the Adam optimizer, which provides adaptive learning rates and robust convergence for nonlinear problems. In practice, training stability is further enhanced by techniques such as early stopping and learning rate scheduling. We applied an early stopping criterion during network training to avoid unnecessary computation once the model had converged. The validation loss was monitored at each epoch, and the training process was terminated when no improvement was detected beyond a specified patience threshold. Learning rate scheduling, such as reducing the rate when validation loss plateaus, so the optimizer explores the parameter space efficiently for converging to a stable solution.

The batch-wise structure of the algorithm ensures efficient use of computational resources and supports parallelization across samples. Each batch loss is computed as the average of segment losses, which allows the model to learn stiffness patterns consistently across multiple sequences. By iterating over all batches and epochs, the model gradually refines its parameters until the predicted accelerations closely align with experimental observations. This design guarantees a balance between data-driven adaptation and strict adherence to the underlying physics.

Case Study

To evaluate the proposed hard-constraint PIML framework, we apply it to a classical single-degree-of-freedom (SDOF) spring, mass and damper oscillator subjected to an external force $F(t)$. This canonical model is representative of a wide range of structural dynamics problems in mechanical and civil engineering [15]. The system dynamics are governed by the equation of motion already introduced in Eq. (1), where the objective is to recover the time-varying stiffness parameter $k(t)$ indirectly from measured acceleration responses. The reference physics-based model was implemented in MATLAB Simulink/Simscape and is available in the public repository for this work [16]. The main physical parameters used in the simulation, including the mass, damping coefficient, stiffness range, and excitation characteristics, are summarized in Table 1.

Table 1 System parameters used in the SDOF spring–mass–damper simulation.

Parameter	Symbol	Value
Mass	m	$3.8\ \mathrm{kg}$
Stiffness (maximum)	$k_{\max}$	$1500\ \mathrm{N/m}$
Stiffness (minimum)	$k_{\min}$	$500\ \mathrm{N/m}$
Damping coefficient	c	$0.48\ \mathrm{N/(m \cdot s)}$
External force amplitude	F_0	$1\ \mathrm{kN}$
Excitation frequency	f	$100\ \mathrm{Hz}$

Figure 4 illustrates the two configurations considered in this study. In Fig. 4(a), the assumed oscillator for the system (i.e. the one used in the ODE in Figure 2) is modeled with a conventional linear mass, spring, and damper arrangement in which stiffness $k(t)$ is permitted to vary with time, thereby simulating gradual stiffness degradation or environmental variability. Fig. 4(b) shows the system that is used to develop the dataset; an additional nonlinear restoring force f is introduced in parallel with the damper, enabling the system to capture the more complex friction behavior.

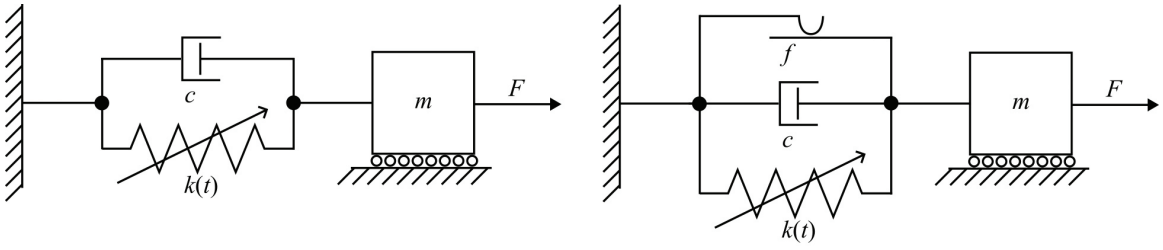


Fig. 4 Schematic of the case study system: (a) classical spring–mass–damper with time-varying stiffness $k(t)$ used as the assumed model for the system; (b) extended configuration with an additional nonlinear restoring force f used for developing the training dataset.

The input data for training are obtained by simulating multiple realizations of the system response under various forcing conditions and time-varying stiffness. Displacement, velocity, acceleration, and external force trajectories are collected at discrete time steps and organized following the preprocessing pipeline described in Algorithm 1. These trajectories form the training and validation datasets used to drive the learning process outlined in Algorithm 2. By embedding the physics directly through Eq. (1), the framework ensures that all predicted accelerations remain consistent with the governing dynamics, while the neural network focuses exclusively on estimating the hidden stiffness evolution.

Model performance is assessed by comparing predicted and measured accelerations through the mean squared error (MSE) of the data. Additionally, the reconstructed stiffness trajectories $\hat{k}(t)$ are benchmarked against ground truth values to evaluate how well the model captures the underlying parameter. The results presented in the following section demonstrate that the framework not only reproduces accurate dynamic responses but also yields interpretable stiffness estimates.

Simulated Stiffness Trajectories

Figure 5 presents the simulated stiffness evolutions used for training and validation across the 100 trajectories. As shown, stiffness values decrease nonlinearly. This diversity ensures that the training set captures a wide distribution of possible system behaviors. As highlighted in earlier sections, these stiffness histories are not directly available to the neural network during training but are used to benchmark the predictions.

Simulated Dynamic Response

Figure 6 shows how the response passes through resonance as stiffness decreases over time: the decay in stiffness (k) lowers the natural frequency (ω_{n}), bringing it into alignment with the frequency of the forcing function (F), producing the visible amplitude build-up near mid-simulation. As k continues to drop, ω_{n} moves past the forcing band and the oscillation amplitude subsides, consistent with the expected resonance-crossing behavior.

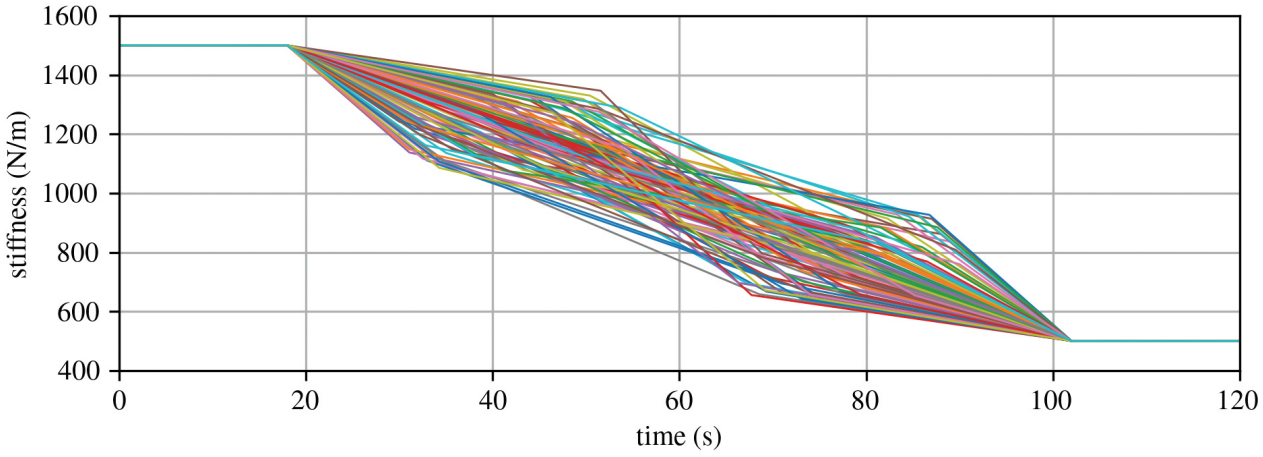


Fig. 5 True stiffness profiles $k(t)$ across 100 simulated trajectories of 120 s each. The variability in decay paths provide a diverse training and testing environment.

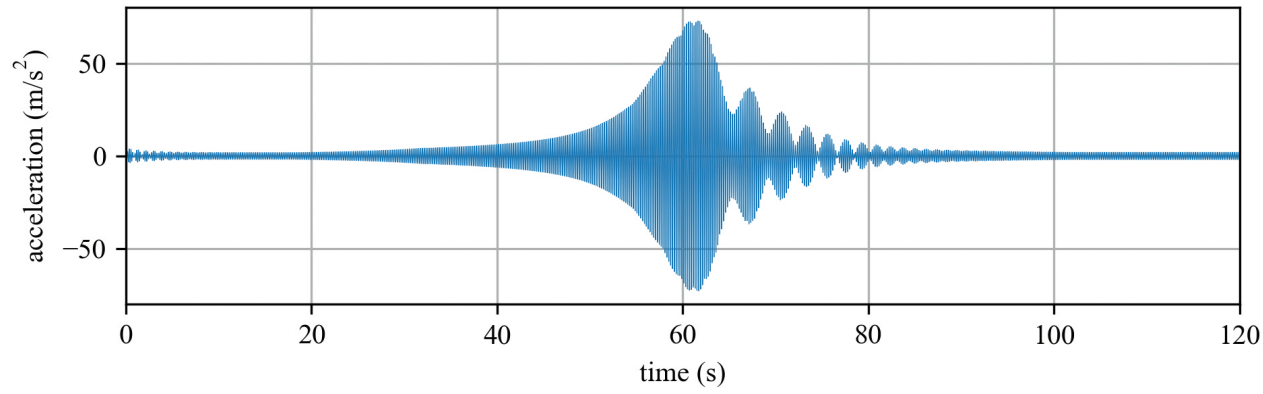


Fig. 6 Measured acceleration response for one test trajectory, showing how the system moves through resonance during a reduction in stiffness.

Inverse Stiffness Identification

Figure 7 compares three stiffness trajectories for a representative test case, the solely data-driven neural network, the physics-informed machine learning (PIML) that includes stiffness indirectly by minimizing the acceleration residual of the governing dynamics, and the physics-based ground truth. Both learning approaches reproduce the global trend of stiffness degradation and closely track the reference over the entire time span. The solely data-driven model yields a smooth estimate that aligns well with the ground truth, demonstrating strong interpolation capability within the training distribution. The physics-informed model, which embeds the spring, mass, and damper equations in the learning loop, sometimes exhibits fluctuations in the estimated stiffness. These arise from sensitivity to transient dynamics and reflect the model's insistence on satisfying the equations of motion at each step rather than fitting a smooth regression.

The close agreement between the PIML learned stiffness and the physics-based ground truth shown in Figure 7 suggests that the measured responses are well captured by the framework. While incorporating physics helps guide the solution space toward behavior consistent with the assumed dynamics, it does not guarantee correctness. The approach holds promise for extending inference beyond the systems and conditions represented in the training data, including under noise, parameter drift, or unseen loading. However, its success remains dependent on model adequacy, identifiability, and optimization stability.

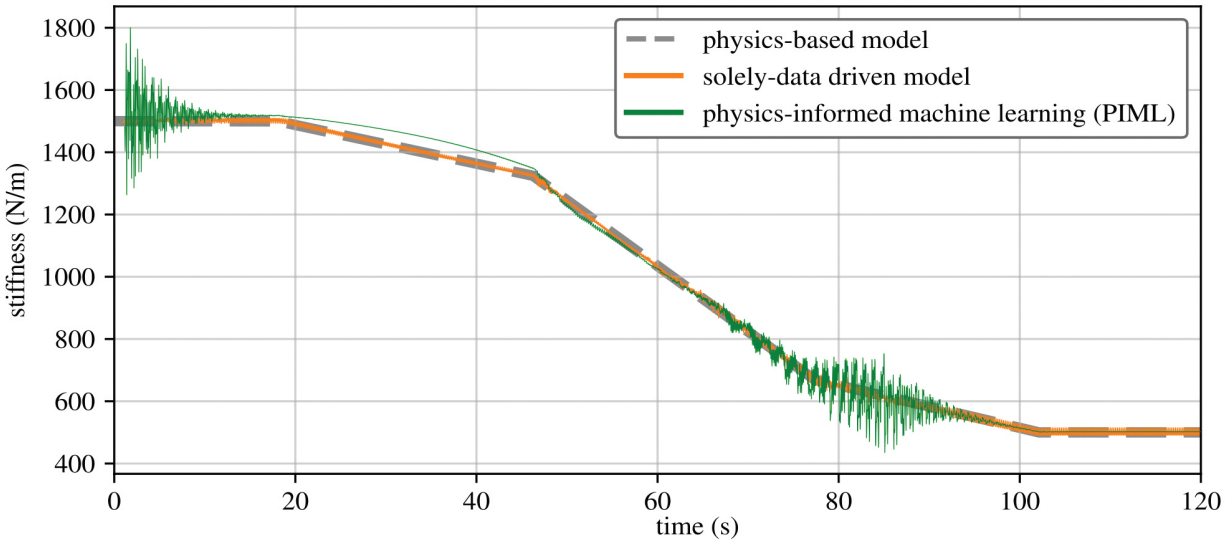


Fig. 7 Comparison of stiffness estimates for the considered test case.

Parameter Study

This section investigates the effect of hyperparameters over two different activation functions to predict the accuracy of the proposed PIML framework. Here, sigmoid and tanh are utilized as two widely used activation functions in the artificial neural network diagrammed in Figure 2. Table 2 reports the RMSE values for different network configurations where depths (number of hidden layers) and widths (neurons per layer) are the main tuning hyperparameters in evaluating the network performance.

For sigmoid, the linear-scale plot (Fig. 8a) highlights the steep error reduction when moving from shallow to deeper networks, while the logarithmic scale (Fig. 8b) makes it easier to distinguish fine differences within the low-error region. For tanh, the linear-scale surface (Fig. 8c) shows overall lower errors compared to sigmoid in shallow configurations, while the logarithmic view (Fig. 8d) emphasizes the robustness of deeper and wider models, where multiple settings converge to similar small RMSE values. By considering the values in Table 2, these plots demonstrate that while tanh offers advantages for shallow networks, both activation functions reach a stable low-error region in deeper and wider architectures.

In general, Figure 8 shows that depth beyond two layers yielded diminishing returns in our hyper-parameter search. Across both activations, error drops sharply when moving from one to two hidden layers, then flattens into a broad low-RMSE basin for 2–4 layers. For shallow networks, tanh achieves consistently lower errors than sigmoid. Increasing width (nodes per layer) helps, but with gentler, nearly monotonic improvements; most of the gain arrives by 50 nodes. The logarithmic views highlight a shallow gains along 2–4 layers and 50–100 nodes. The best accuracy is obtained with four layers and 100 neurons, though several nearby configurations perform nearly as well at lower computational cost.

Table 2 RMSE for sigmoid vs. tanh activation function across different network configuration

Neurons per layer	Number of hidden layers							
	Sigmoid activation				Tanh activation			
	1	2	3	4	1	2	3	4
10	6.20	1.20	2.00	0.90	3.36	0.68	0.37	0.35
25	3.60	0.18	0.30	0.19	2.42	0.14	0.09	0.08
50	1.42	0.13	0.09	0.05	1.83	0.06	0.05	0.04
75	1.23	0.07	0.05	0.04	1.56	0.08	0.04	0.05
100	1.07	0.06	0.04	0.04	1.40	0.05	0.05	0.04

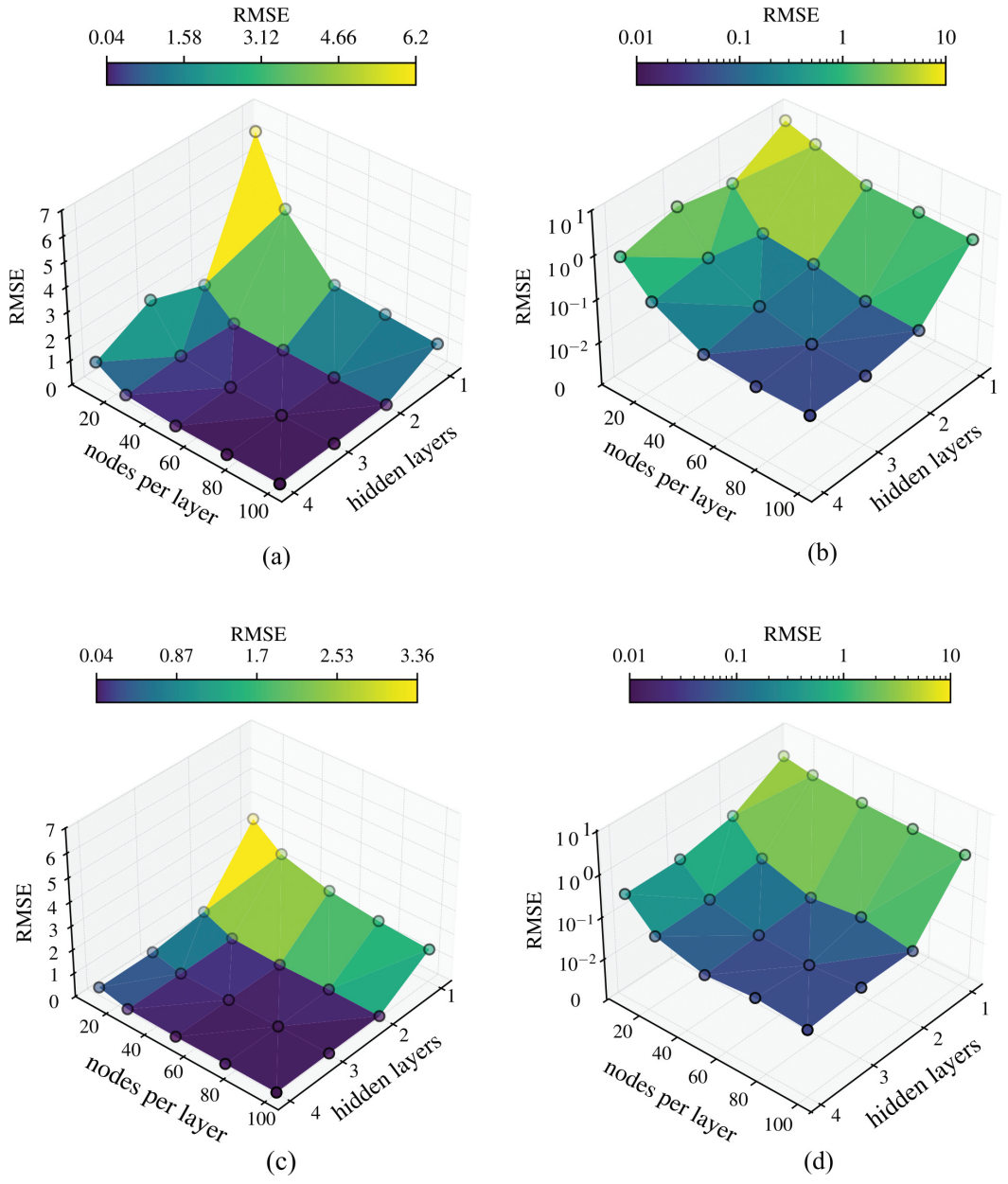


Fig. 8 RMSE surface as a function of hidden layers and neurons per layer for two activation functions. (a) Sigmoid with linear scale, highlighting the sharp error reduction. (b) Sigmoid with logarithmic scale, making small variations more visible. (c) Tanh with linear scale, showing generally lower RMSE than sigmoid in shallow networks. (d) Tanh with logarithmic scale, revealing stable low-error regions across deeper and wider architectures.

Conclusion

This paper presented a hard constraint physics-informed machine learning (PIML) framework for structural dynamics and demonstrated its effectiveness on a spring, mass, and damper system with time-varying stiffness. By embedding the governing equation of motion directly into the learning process, the proposed approach biases the solution space toward physically consistent behavior while retaining the flexibility and adaptability of data-driven models.The framework successfully recovers hidden stiffness trajectories that closely match the reference dynamics.

Overall, the hard-constraint PIML framework successfully recovered time-varying stiffness and reproduced key response features on the studied spring–mass–damper system. A simple hyperparameter sweep showed that modest depth (2–4 layers)

with moderate width is sufficient to drive test error to roughly $\mathrm{RMSE} \approx 0.04$, indicating practical accuracy without excessive model complexity.

There is a trade-off between the PIML model and a solely data-driven model. While a neural network trained solely on data can mimic stiffness trajectories with high accuracy, it lacks a true understanding of the underlying physics and may fail when exposed to conditions beyond its training distribution. In contrast, the PIML approach, though sometimes yielding responses that deviate from the reference trajectory, actively enforces the governing ODE constraints and therefore reflects a physically guided attempt to capture the system behavior. This enables PIML to generalize more reliably to unseen scenarios, operate effectively with less data, and produce predictions that are physically interpretable.

For the hard-constraint PIML considered here, there is clear promise for inverse problems: coupling measurements with the governing equation of motion to regularize otherwise ill-posed parameter estimates (e.g., time-varying stiffness). While promising, the main challenge observed in this study is the appearance of ripple artifacts in the inferred stiffness, arising from unsteady solutions when the equation of motion is enforced at every step within the learning loop.

Acknowledgments This material is based upon work supported by the National Science Foundation grant numbers CCF - 1956071, CCF-2234921, and CPS - 2237696. Additional support from the Air Force Office of Scientific Research (AFOSR) through award no. FA9550-21-1-0083. Any opinions, findings conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Air Force,

References

1. Fuh-Gwo Yuan, Sakib Ashraf Zargar, Qiuyi Chen, and Shaohan Wang. "Machine learning for structural health monitoring: challenges and opportunities". In Daniele Zonta, Hoon Sohn, and Haiying Huang, editors, *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2020*. SPIE, April 2020.
2. Elizabeth J. Cross, S. J. Gibson, M. R. Jones, D. J. Pitchforth, S. Zhang, and T. J. Rogers. *Physics-Informed Machine Learning for Structural Health Monitoring*, pages 347–367. Springer International Publishing, October 2021.
3. Marcus Haywood-Alexander, Wei Liu, Kiran Bacsa, Zhilu Lai, and Eleni Chatzi. "Discussing the spectrum of physics-enhanced machine learning: a survey on structural mechanics applications". *Data-Centric Engineering*, 5, 2024.
4. X. Liang, Y. Zhang, and J. Chen. "Physics-informed machine learning for structural dynamics: A review and outlook". *Mechanical Systems and Signal Processing*, 215:111246, 2024.
5. Daniel Coble, Liang Cao, Austin R.J. Downey, and James M. Ricles. "Physics-informed machine learning for dry friction and backlash modeling in structural control systems". *Mechanical Systems and Signal Processing*, 218:111522, September 2024.
6. Ying Zhou, Shiqiao Meng, Yujie Lou, and Qingzhao Kong. "Physics-informed deep learning-based real-time structural response prediction method". *Engineering*, October 2023.
7. Eleonora Maria Tronci, Austin R. J. Downey, Azin Mehrjoo, Puja Chowdhury, and Daniel Coble. *Physics-Informed Machine Learning Part I: Different Strategies to Incorporate Physics into Engineering Problems*, pages 1–6. River Publishers, August 2025.
8. Austin R. J. Downey, Eleonora Maria Tronci, Puja Chowdhury, and Daniel Coble. *Physics-Informed Machine Learning Part II: Applications in Structural Response Forecasting*, pages 63–66. River Publishers, August 2025.
9. M. Raissi, P. Perdikaris, and G.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. Eleonora Maria Tronci, Austin Downey, Conor Madden, Mohsen Gol Zardian, and Daniel Coble. *Physics-Informed Machine Learning Part IV: Weight Tuned Soft Constraint Method for Structural Dynamics*. August 2026.
11. Z. Lai, L. Lu, M. Dao, and G.E. Karniadakis. "Hard enforcement of interface and boundary conditions in physics-informed neural networks". *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.
12. R.T.Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. "Neural ordinary differential equations". In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.
13. L. Lu, P. Jin, G. Pang, Z. Zhang, and G.E. Karniadakis. "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators". *arXiv preprint arXiv:1910.03193*, 2019.
14. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Fourier neural operator for parametric partial differential equations". *arXiv preprint arXiv:2010.08895*, 2020.
15. Austin Downey and Laura Micheli. "Vibration mechanics: A practical introduction for mechanical, civil, and aerospace engineers". 2024.
16. ARTS-Lab. "Paper-2026-PIML-part-III-hard-constraint-ODE-method-for-structural-dynamics". GitHub, 2026. https://github.com/ARTS-Laboratory/Paper-2026-PIML-Part-III-Hard-Constraint-ODE-Method-for-Structural-Dynamics.