Chapter 8
Physics-Informed Machine Learning Part IV: Weight-Tuned Soft-Constraint Method for Structural Dynamics

Eleonora Maria Tronci, Austin R. J. Downey, Connor Madden, Mohsen Gol Zardian, and Daniel Coble

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 IV of this series, the authors describe a soft-constraint, physics-informed neural network (PINN) tutorial for structural dynamics. Instead of enforcing equilibrium exactly at every step, the network is trained with a composite loss that combines several data fitting terms (displacement, velocity, acceleration, and stiffness taken from an experimental window) with a single physics consistency term that measures how closely the predicted forces satisfy the mass–damping–stiffness relationship. Each term carries an explicit weight, allowing the decision on how much trust to place in measured data versus the governing equation. A compact multilayer perceptron takes as input the displacement, velocity, and external force from the most recent timestep and predicts the next-step displacement along with an effective stiffness. Two weight tuning options are covered: (i) a manual recipe that highlights how shifting emphasis from data to physics improves extrapolation but can amplify noise, and (ii) a self-adaptive schedule that raises the physics weight whenever the residual grows and relaxes it once the residual falls below a user-selected tolerance, eliminating the need for grid searches.

The technique is demonstrated using the same spring–mass–damper system introduced in Part III, but the PINN is trained on a single experiment featuring a specific time-varying stiffness path. Because the model includes explicit time as an input, it becomes specific to that trajectory. Performance is benchmarked against two reference solvers: a data-only neural network trained without any physics guidance and the hard constraint ODE layer introduced previously. The paper concludes with qualitative guidelines for selecting initial weight ratios, determining the residual tolerance for the adaptive schedule, and monitoring convergence, providing readers with a practical roadmap for balancing data fidelity and physical plausibility in their own inverse problem applications.

Keywords Physics-informed machine learning · Physics-informed neural network · Soft-constraint · Weight tuning · Adaptive tuning

Eleonora Maria Tronci
Department of Civil and Urban Engineering, New York University, New York, NY, USA
e-mail: eleonora.m.tronci@gmail.com

Eleonora Maria Tronci
Center for Urban Science and Progress, New York University, New York, NY, USA
e-mail: eleonora.m.tronci@gmail.com

Austin R. J. Downey · Connor Madden · Mohsen Gol Zardian
Department of Mechanical Engineering, University of South Carolina, Columbia, SC, USA
e-mail: AUSTINDOWNEY@sc.edu; CMMADDEN@email.sc.edu; GOLZARDM@email.sc.edu

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

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

Introduction

Scientific modeling has traditionally been divided between two distinct paradigms [1, 2]. Physics-based numerical methods discretize governing equations to provide high-fidelity solutions, but are often constrained by prohibitive computational costs, complex mesh generation, and an inability to account for model-form uncertainty in the physical laws themselves. Conversely, data-driven models excel at learning complex patterns from data but typically lack physical interpretability, require massive training datasets, and struggle to generalize reliably, often producing physically implausible predictions outside of their training distribution.

Physics-informed machine learning (PIML) has emerged as a powerful paradigm for developing predictive models of physical systems, offering a compelling complement to purely data-driven or traditional physics-based approaches [3, 4]. By integrating governing physical laws directly into the learning pipeline, PIML provides a guided strategy to steer solutions toward physical behavior under sparse, noisy, or incomplete data. PIML is not inherently more accurate than conventional machine learning; its impact is highly context-dependent. Where abundant, representative data are available and the task remains within the training distribution, a well-regularized data-only model may match or exceed PIML in standard error metrics. Where interpretability, parameter recovery, or cautious extrapolation are required, the physics guidance can confer advantages by constraining the hypothesis space toward feasible dynamics. At the same time, recent analyses have shown that PINNs can exhibit imbalanced loss dynamics and stiffness during training, motivating principled strategies for calibrating the relative influence of data and physics terms [5]. This methodology is particularly effective for both forward problems, such as solving differential equations, and inverse problems, such as system identification and parameter estimation.

The domain of PIML is extensive and can be categorized in multiple ways [6]. However, physical principles can be incorporated in two primary ways: through *hard* or *soft* constraints. Hard-constraint approaches, as detailed in Part III of this series [7], embed the governing equations directly into the model's architecture (e.g., via a differentiable ODE integrator), encouraging the predicted solution to satisfy physical laws at each step. This strong enforcement has been shown to be successful in many applications [8], can improve stability and identifiability, but it may be less flexible under model-form uncertainty and can be more challenging to implement for systems with unmodeled dynamics or discontinuities. In contrast, soft-constraint approaches, presented here in Part IV, offer a more flexible and widely adopted alternative: the residual of the governing equations is incorporated as a penalty term in the model's loss function. Variants such as augmented-Lagrangian PINNs further reduce sensitivity to hand-tuned penalties by learning multipliers that tighten constraint satisfaction [9]. The choice between hard and soft constraints is problem-specific and depends on factors such as the fidelity of the governing model, noise characteristics, label availability for latent parameters, and the degree of distribution shift anticipated at inference.

Structural dynamics presents a particularly compelling application area for several PIML approaches [10]. Systems such as bridges, buildings, wind turbines, and aerospace structures obey well-established principles but often exhibit time variability, environmental influences, and mild nonlinearities that complicate pure first-principles modeling. In this context, soft-constraint physics-informed neural networks (PINNs) are attractive because they preserve a simple network architecture while injecting domain knowledge through a residual penalty on the equation of motion. Concrete structural-dynamics uses of residual-penalized (soft) physics terms include damage identification and system identification. For damage identification, Wang et al. [11] introduce a physics-guided residual network that augments a data-driven backbone with an equation-of-motion residual term, improving robustness under scarce/noisy measurements—an archetypal soft-constraint strategy for SHM tasks. For inverse structural dynamics, Liu et al. formulate a PINN that recovers parameters in systems with multi-physics damping (nonlinear dissipation) by minimizing a composite objective that mixes data misfit with residual penalties, illustrating soft-constraint PINNs for parameter estimation in realistic structural models [12]. In forward vibration analysis, Chen et al. develop a time-marching PINN tailored to long-duration simulations of structural vibrations. Although focused on efficiency, their formulation is still driven by residual penalties rather than hard architectural enforcement, underscoring the practicality of soft-constraints in vibration problems [13].

The successful implementation of soft-constraint depends on balancing multiple loss components, such as data fidelity (e.g., displacement, velocity, acceleration, stiffness) versus physics consistency (equation-of-motion residual). These weights govern the effective bias–variance trade-off, influence the identifiability of latent parameters, and shape the optimization dynamics. Adaptive weighting schemes offer principled methods for adjusting this balance during training: Self-Adaptive PINNs learn per-sample physics weights that emphasize stubborn residuals [14]; dynamic weight strategies and self-adaptive loss balancing adjust term contributions online [15, 16]; and multi-task formulations provide global, uncertainty-based weights or gradient-normalization baselines that reduce manual tuning [17, 18]. These developments directly address the training imbalances highlighted in recent analyses [5] and motivate the weighting strategies examined in this work.

Recent contributions by the authors have examined multiple strategies for integrating physical knowledge into machine learning. Part I of this series outlined how embedding governing equations can enhance robustness, improve interpretability, and in many cases strengthen predictive performance in engineering settings [6]. Part II demonstrated a practical application

to structural dynamics, showing that physics-informed architectures can effectively forecast system responses [19]. Together, these studies highlight the benefits of coupling physical models with neural networks for inverse problems, including the estimation of system parameters.

This paper, Part IV of this series, presents a practical tutorial on applying a soft-constraint PINN to an inverse problem in structural dynamics. We use the same spring-mass-damper system with time-varying stiffness introduced in Part III [7] to provide a direct comparison of methodologies. The core of our contribution is a detailed exploration of the composite loss function, which combines data-fitting terms with a physics-consistency term derived from the equation of motion. We investigate two distinct weight-tuning strategies: (i) a manual recipe that highlights the fundamental trade-off between data fidelity and physical regularization, and (ii) a self-adaptive schedule that automates the balancing process. The performance of the soft-constraint PINN is benchmarked against a data-only neural network and the hard-constraint ODE model from Part III. We conclude with practical guidelines for selecting initial weight ratios and monitoring convergence, providing a clear roadmap for applying this flexible PIML technique to other inverse problems.

Methodology

This work develops a soft-constraint PIML framework to identify and predict the dynamic response of a mass–spring–damper system. The central idea is to combine a data-driven neural network with a physics-consistency term in the loss such that the governing equation of motion influences learning without being enforced exactly at every time step. Unlike hard-constraint approaches, where the ODE is encoded directly into the model architecture and satisfied identically, the present framework penalizes the residual of the governing composite loss function during training. Concretely, the model minimizes a composite objective that blends data-fitting terms (e.g., displacement, velocity, acceleration, and stiffness from an experimental window) with a physics residual penalty evaluated on the network's predictions. The relative contribution of each term is controlled by explicit weights, enabling practitioners to calibrate the trade-off between data fidelity and physical plausibility. As a result, the learned solution is not guaranteed to satisfy the ODE exactly but is regularized toward physically consistent behavior, and training is guided by both data and physics.

Data preparation

We adopt the same variable conventions as Part III: time t, displacement x_t, velocity v_t, acceleration a_t, stiffness k_t, and external force F_t are recorded as synchronized columns in tabular files (rows = time steps, columns = physical quantities). Raw trajectories from multiple experiments are stacked into a three-dimensional array $X \in \mathbb{R}^{N\times L\times D}$, where N is the number of trajectories, L the samples per trajectory, and D the number of measured channels. For continuity and visual reference, we reproduce the same schematic as Part III in Fig. 1 and defer full pseudocode and hyperparameter details (e.g., window length choices, batching policies) to Part III [7].

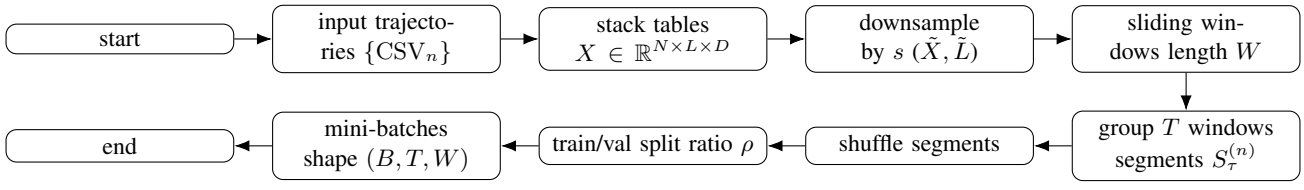


Fig. 1 Preprocessing pipeline from raw trajectories to mini-batches (verbatim labels from Part III)

Consistent with our implementation, we downsample all signals to a uniform rate (50 Hz) to reduce redundancy while preserving the dynamics of interest; let $\tilde{X}$ denote the downsampled tensor and $\tilde{L}$ the corresponding length. Basic sanity checks (monotone timestamps, consistent units) are applied prior to batching.

The main deviation from Part III concerns the supervision format. Part III forms fixed-length sliding windows of size W and groups T consecutive windows into segments to train the hard-constraint ODE layer. In contrast, Part IV (soft-constraint PINN) uses step-ahead supervision: for each time index t we form input–target pairs

$$\underbrace{[\,x_t,\ v_t,\ F_t,\ t\,]}_{\text{inputs}} \longrightarrow \underbrace{[\,x_{t+1},\ k_t\,]}_{\text{targets}},$$

optionally including a_t if available. This pairing directly supports the one-step predictor used in our model, while leaving the rest of the pipeline unchanged.

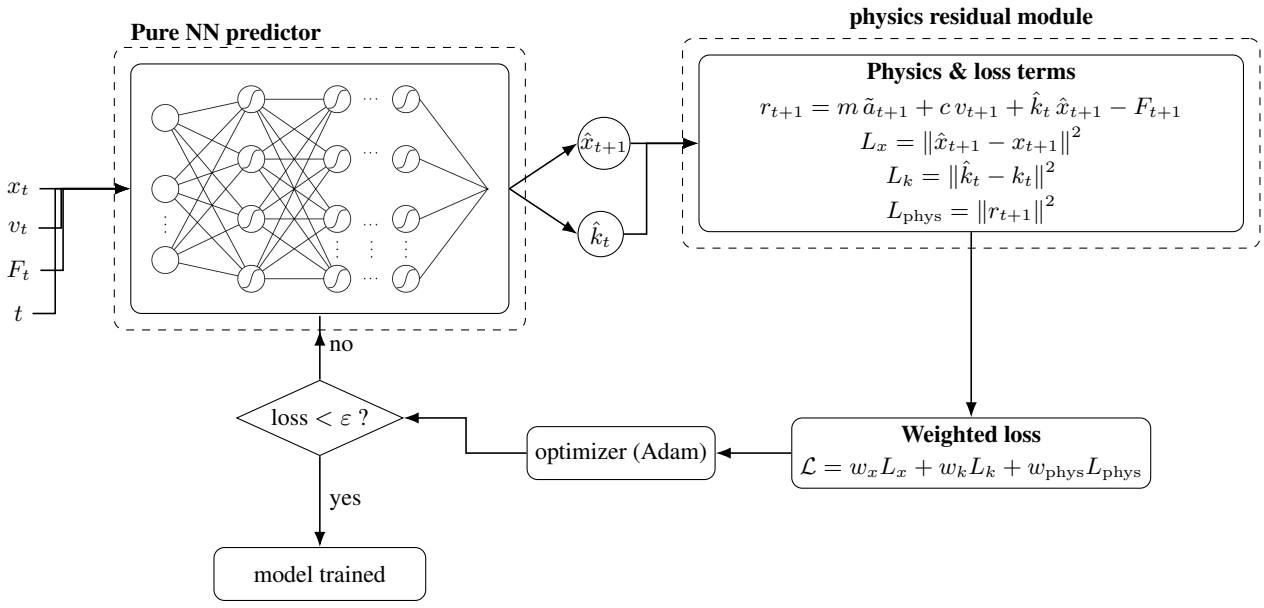


Fig. 2 Soft-constraint PINN: the NN predicts $(\hat{x}_{t+1}, \hat{k}_t)$. The physics module computes the residual and lists $L_x, L_k, L_{\text{phys}}$; the composite loss sits directly below it, the optimizer below that, and the stopping rule is placed to the left of the composite loss

To avoid temporal leakage within an experiment, we split $\tilde{X}$ by experiments (80/20 train/validation), then shuffle the training set and construct mini-batches. Batches thus have shape (B,D) for inputs and (B,D_{target}) for targets under the step-ahead pairing, in place of the (B,T,W) layout used in Part III's windowed segments. Apart from this pairing change, the stack $\rightarrow$ downsample $\rightarrow$ split $\rightarrow$ batch flow remains identical to Part III.

Network architecture and physics-embedded loss function

The proposed architecture follows a soft-constraint PIML formulation in which the governing law contributes through a residual penalty in the objective function. The present model employs a compact data-driven predictor together with a physics-consistency term evaluated at the same time index.

While PINNs are most often used for interpolation/forward problems, they can also be adapted to inverse settings when the physics is sufficiently known. Here we train on a test without friction so that the governing equation is fully specified (mass–damping–stiffness), and the soft-constraint acts as a physics-guided regularizer for estimating a time-varying stiffness. As is typical for forward-type PINNs, each trained model is specific to a single experiment/trajectory (i.e., not expected to generalize across materially different tests without retraining).

Let $z_t = [\,x_t,\, v_t,\, F_t,\, t\,]$ denote the input at time t (acceleration a_t may be included when available). A multilayer perceptron $\mathcal{N}_\theta$ advances the state by one time step and simultaneously estimates an effective stiffness,

$$\left[\,\hat{x}_{t+1},\ \hat{k}_t\,\right] \;=\; \mathcal{N}_\theta\big(z_t\big) \;\equiv\; \phi\big(t, x_t, v_t, F_t\big),$$

so that $\phi_1 = \hat{x}_{t+1}$ and $\phi_2 = \hat{k}_t$. When derivative consistency terms are used, autodifferentiation provides the required partial derivatives of ϕ_1 with respect to its inputs; time derivatives then follow by the chain rule, e.g.

$$\frac{d\phi_1}{dt} = \frac{\partial\phi_1}{\partial t} + \frac{\partial\phi_1}{\partial x}\,\dot{x} + \frac{\partial\phi_1}{\partial v}\,\dot{v} + \frac{\partial\phi_1}{\partial F}\,\dot{F}, \quad \frac{d^2\phi_1}{dt^2} \text{ analogously,}$$

with $\dot{x}, \ddot{x}, \dot{v}, \dot{F}$ obtained from measurements or numerically differentiated (smoothed) trajectories. In this problem, the governing equation does not involve derivatives of k, so autodiff of ϕ_2 is not required.

Consistency with the mass–damping–stiffness relation

$$m\,a_t \;+\; c\,v_t \;+\; k_t\,x_t \;=\; F_t$$

is promoted by penalizing the residual

$$r_{t+1} \;=\; m\,\tilde{a}_{t+1} \;+\; c\,v_{t+1} \;+\; \hat{k}_t\,\hat{x}_{t+1} \;-\; F_{t+1},$$

where $\tilde{a}_{t+1} = a_{t+1}$ when acceleration is measured, otherwise $\tilde{a}_{t+1}$ is obtained from a centered, filtered finite difference of x at time $t{+}1$. Training minimizes a composite objective that balances data fidelity and physics plausibility,

$$\mathcal{L}(\theta) \;=\; w_x\,\mathrm{MSE}\big(\hat{x}_{t+1},\,x_{t+1}\big) \;+\; w_k\,\mathrm{MSE}\big(\hat{k}_t,\,k_t\big) \;+\; w_{\text{phys}}\,\mathrm{MSE}\big(r_{t+1},\,0\big), \tag{1}$$

with positive weights w_x, w_k, and w_{phys}. If stiffness labels k_t are not available, the second term is omitted without changing the rest of the framework. During inference, the predictor is applied recursively to advance $\hat{x}$ over time while reporting $\hat{k}_t$ at the corresponding index. Unlike Part III, physics enters solely through the residual in (1), which yields a clear and tunable trade-off between data terms and physical consistency.

For bookkeeping, we use two time sets: an experimental set T_e (e.g., the first 60 s of the record) for supervised targets where available, and a physics set T_p (the full test) for residual and optional derivative consistency. Equation (1) is evaluated on the appropriate set per term (e.g., w_k on T_e; w_{phys} on T_p). If desired, derivative-alignment terms may be added with small weights,

$$L_{\dot{x}} = \tfrac{1}{|T_p|}\sum_{t\in T_p}\big(\dot{\hat{x}}_t - \dot{x}_t\big)^2, \qquad L_{\ddot{x}} = \tfrac{1}{|T_p|}\sum_{t\in T_p}\big(\ddot{\hat{x}}_t - \ddot{x}_t\big)^2,$$

where $\dot{\hat{x}}_t, \ddot{\hat{x}}_t$ come from autodiff and $\dot{x}_t, \ddot{x}_t$ from measurements or numerical differentiation.

Training procedure

Training follows the soft-constraint principle and therefore optimizes a composite objective that blends data-fit terms with a physics residual. Each mini-batch contains step-ahead pairs $[x_t,\, v_t,\, F_t,\, t] \mapsto [x_{t+1},\, k_t]$ (with a_t optionally included), and the predictor $\mathcal{N}_\theta$ maps inputs to $[\hat{x}_{t+1},\, \hat{k}_t]$. The residual $r_{t+1} = m\,\tilde{a}_{t+1} + c\,v_{t+1} + \hat{k}_t\,\hat{x}_{t+1} - F_{t+1}$ is formed at the advanced time and used in the physics term and, together with the data terms, defines the loss in (1). Model parameters are updated with Adam. Early stopping monitors the validation objective and restores the best weights when no improvement is observed over a fixed patience, while a plateau scheduler reduces the learning rate when the validation loss stalls, improving stability and convergence. When derivative terms are used, we compute $\dot{\hat{x}}_t, \ddot{\hat{x}}_t$ by autodiff and obtain $\dot{x}_t, \ddot{x}_t$ from measured signals or numerically differentiated (smoothed) trajectories.

With fixed (manual) weights, the loss weights remain constant throughout training; the full procedure is listed in Algorithm 1. To reduce manual tuning while keeping the same supervision and residual definitions, a joint adaptive scheme is adopted. Global weights for the displacement, stiffness, and physics terms are learned via constrained log-variance parameters $s_x, s_k, s_{\text{phys}} \in (-s_{\max}, s_{\max})$, with $\lambda_\cdot = \exp(-s_\cdot)$ and a linear regularizer $(s_x + s_k + s_{\text{phys}})$ to avoid trivial solutions. Each per-term loss is first put on a comparable scale using exponentially weighted moving averages (EMA) of the actual optimized quantities: raw L_x, a right-sized stiffness term $L_k^{\text{scaled}} = \gamma_k L_k$ to prevent dominance, and a physics term $L_{\text{phys}}^{\text{train}}$ that optionally uses per-sample SAPINN attention. SAPINN introduces a lightweight inner optimization over per-batch logits to produce nonnegative per-sample weights with a mean of approximately one, emphasizing stubborn residuals while penalizing entropy collapse and overly large weights. The normalized terms $L_x^{(n)}, L_k^{(n)}, L_{\text{phys}}^{(n)}$ drive the uncertainty-weighted total

$$\mathcal{L}_{\text{tot}} = \lambda_x L_x^{(n)} + \lambda_k L_k^{(n)} + \lambda_{\text{phys}} L_{\text{phys}}^{(n)} + (s_x + s_k + s_{\text{phys}}),$$

and two Adam optimizers are used: one for model parameters and one (with a smaller learning rate) for the global weight parameters. The procedure is summarized in Algorithm 2.

Algorithm 1 Soft-constraint PINN training (fixed weighting)

Require: Training sets $T_e = \{t_i\}_{i=1}^{N}$ (supervised) and $T_p = \{t_i\}_{i=1}^{M}$ (physics); network $\mathcal{N}_\theta$ with mapping $\phi : (t, x_t, v_t, F_t) \mapsto (\hat{x}_{t+1}, \hat{k}_t)$; constants m, c; loss weights $w_k, w_x, w_{\dot{x}}, w_{\ddot{x}}, w_{\text{phys}}$; optimizer (Adam), learning-rate schedule, early-stopping patience.

Ensure: Trained parameters $\theta^\star$

Initialize $\theta \leftarrow$ Xavier/He; optimizer state $\leftarrow$ zeros
for epoch $= 1, 2, \ldots$ **do**
  for mini-batch $\mathcal{B}_e \subset T_e$ and $\mathcal{B}_p \subset T_p$ **do**
    Build inputs $z_t = [x_t, v_t, F_t, t]$ (include a_t if measured); compute $(\hat{x}_{t+1}, \hat{k}_t) = \mathcal{N}_\theta(z_t)$
    If a_t unavailable, set $\tilde{a}_t \leftarrow \text{FD}_{\text{filtered}}(x_t)$; else $\tilde{a}_t \leftarrow a_t$
    Residual (advanced time): $r_{t+1} \leftarrow m\,\tilde{a}_{t+1} + c\,v_{t+1} + \hat{k}_t\,\hat{x}_{t+1} - F_{t+1}$
    Derivative terms (optional): $\dot{\hat{x}}_t,\ \ddot{\hat{x}}_t \leftarrow$ autodiff$(\mathcal{N}_\theta, z_t)$
    Loss terms:
    $L_k \leftarrow \frac{1}{|\mathcal{B}_e|} \sum_{t \in \mathcal{B}_e} (\hat{k}_t - k_t)^2$ ▷ omit if k_t not labeled
    $L_x \leftarrow \frac{1}{|\mathcal{B}_p|} \sum_{t \in \mathcal{B}_p} (\hat{x}_{t+1} - x_{t+1})^2$
    $L_{\dot{x}} \leftarrow \frac{1}{|\mathcal{B}_p|} \sum_{t \in \mathcal{B}_p} (\dot{\hat{x}}_t - \dot{x}_t)^2$ ▷ optional
    $L_{\ddot{x}} \leftarrow \frac{1}{|\mathcal{B}_p|} \sum_{t \in \mathcal{B}_p} (\ddot{\hat{x}}_t - \ddot{x}_t)^2$ ▷ optional
    $L_{\text{phys}} \leftarrow \frac{1}{|\mathcal{B}_p|} \sum_{t \in \mathcal{B}_p} r_{t+1}^2$
    Composite loss: $\mathcal{L} \leftarrow w_k L_k + w_x L_x + w_{\dot{x}} L_{\dot{x}} + w_{\ddot{x}} L_{\ddot{x}} + w_{\text{phys}} L_{\text{phys}}$
    Backpropagate $\nabla_\theta \mathcal{L}$; **Adam step** on θ
  end for
  Compute validation metrics on held-out $T_e^{\text{val}}, T_p^{\text{val}}$; update LR on plateau
  Early stop if no improvement for P epochs; save best $\theta^\star$
end for

Algorithm 2 Joint adaptive weighting with EMA normalization and optional SAPINN

Require: Same data and network as Alg. 1; Adam(lr $= \eta_\theta$) for model, Adam(lr $= \eta_w < \eta_\theta$) for weights
Log-variance params: u_x, u_k, u_p; $s_\bullet = s_{\max} \tanh u_\bullet$; $\lambda_\bullet = \exp(-s_\bullet)$
EMA factor $\alpha \in (0, 1)$; stiffness scale $\gamma_k > 0$; optional SAPINN with inner steps S, temp τ

Initialize EMA trackers $\bar{L}_x, \bar{L}_k, \bar{L}_{\text{phys}}$ (unset)
for epoch $= 1, 2, \ldots$ **do**
  for mini-batch $(\mathcal{B}_e, \mathcal{B}_p)$ **do**
    Forward pass: $(\hat{x}_{t+1}, \hat{k}_t) = \mathcal{N}_\theta([x_t, v_t, F_t, t])$
    Residual: $r_{t+1} = m\,\tilde{a}_{t+1} + c\,v_{t+1} + \hat{k}_t\,\hat{x}_{t+1} - F_{t+1}$
    Per-term losses: $L_x = \text{mean}(\hat{x}_{t+1} - x_{t+1})^2$, $L_k^{\text{raw}} = \text{mean}(\hat{k}_t - k_t)^2$ (or 0 if unlabeled)
    Scale stiffness: $L_k^{\text{scaled}} = \gamma_k\, L_k^{\text{raw}}$; raw physics: $L_{\text{phys}}^{\text{raw}} = \text{mean}(r_{t+1}^2)$
    if SAPINN enabled **then**
      Compute per-sample weights w by softmax logits (mean ≈ 1) over S inner steps;
      $L_{\text{phys}}^{\text{train}} = \text{mean}(w \cdot r_{t+1}^2)$
    else
      $L_{\text{phys}}^{\text{train}} = L_{\text{phys}}^{\text{raw}}$
    end if
    Update EMAs on optimized terms: $\bar{L}_x \leftarrow \text{EMA}(\bar{L}_x, L_x)$, $\bar{L}_k \leftarrow \text{EMA}(\bar{L}_k, L_k^{\text{scaled}})$, $\bar{L}_{\text{phys}} \leftarrow \text{EMA}(\bar{L}_{\text{phys}}, L_{\text{phys}}^{\text{train}})$
    Normalize: $L_x^{(n)} = L_x/(\bar{L}_x + \varepsilon)$, $L_k^{(n)} = L_k^{\text{scaled}}/(\bar{L}_k + \varepsilon)$, $L_{\text{phys}}^{(n)} = L_{\text{phys}}^{\text{train}}/(\bar{L}_{\text{phys}} + \varepsilon)$
    Total (uncertainty weighting): $\mathcal{L}_{\text{tot}} = \lambda_x L_x^{(n)} + \lambda_k L_k^{(n)} + \lambda_{\text{phys}} L_{\text{phys}}^{(n)} + (s_x + s_k + s_{\text{phys}})$
    Update θ with Adam on $\mathcal{L}_{\text{tot}}$; update (u_x, u_k, u_p) with Adam on $\mathcal{L}_{\text{tot}}$
  end for
  Validate, log $(\lambda_x, \lambda_k, \lambda_{\text{phys}})$; reduce LR on plateau; early stop on validation objective
end for

Case study

The soft-constraint framework is evaluated on the classical single-degree-of-freedom spring–mass–damper oscillator subjected to an external force F_t. This benchmark is representative of common structural dynamics problems in mechanical and civil engineering and is governed by the equation of motion $m\,a_t + c\,v_t + k_t\,x_t = F_t$ [20]. The reference physics-based model was implemented in MATLAB Simulink/Simscape and is available in the public repository for this work [21].

The learning objective is to recover the time-varying stiffness k_t indirectly while maintaining accurate state prediction. Two configurations are considered to probe both linear and nonlinear regimes. In the first, the oscillator retains a conventional linear arrangement with stiffness k_t allowed to vary in time, emulating progressive stiffness degradation or environmental variability. In the second, an additional nonlinear restoring force f acts in parallel with the damper to reflect effects such as friction, intermittent contact, or geometric nonlinearities. These complementary scenarios form a robust testbed for assessing the approach's adaptability under increasing dynamical complexity.

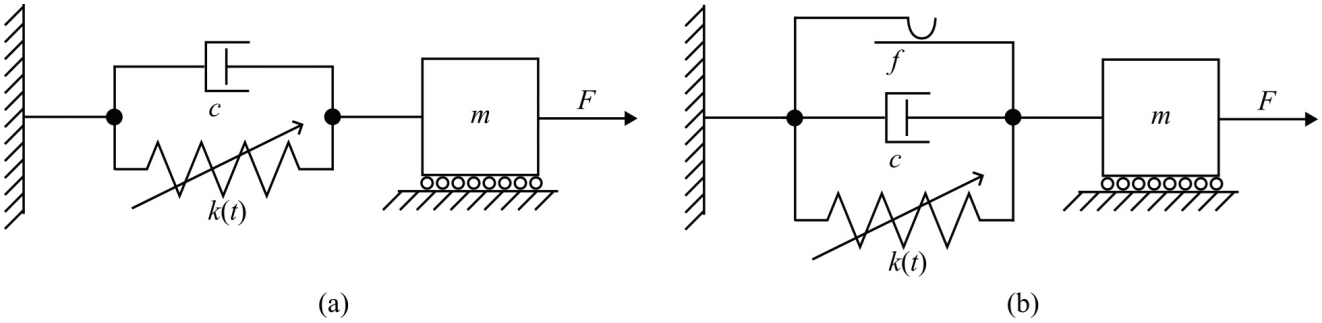


Fig. 3 Schematic of the case study system: (a) classical spring–mass–damper with time-varying stiffness k_t; (b) extended configuration with an additional nonlinear restoring force f

Input data are generated by simulating multiple realizations under diverse forcing conditions and stiffness evolutions. Displacement, velocity, acceleration, and forcing trajectories are sampled at discrete time steps and organized according to the preprocessing pipeline described in Sec. . Training follows the procedure in Sec. , where the network advances $\hat{x}_{t+1}$ and infers $\hat{k}_t$ while the residual-based penalty pulls solutions toward the governing law. Performance is quantified by the root-mean-squared error (RMSE) between predicted and measured displacements at the next step, and, when ground truth is available, by the RMSE between the reconstructed stiffness trajectory $\hat{k}_t$ and the reference k_t. The study demonstrates that the soft-constraint formulation reproduces the dynamic response with high fidelity while yielding interpretable stiffness estimates aligned with the underlying physics, even when the system exhibits time variability and mild nonlinear effects.

Inverse Stiffness Identification and Response Prediction

We evaluate the soft-constraint PINN in two complementary roles: inverse identification of the time-varying stiffness k_t and one-step prediction of displacement x_{t+1}. The model outputs $[\hat{x}_{t+1}, \hat{k}_t]$ and is trained with the composite loss in Eq. (1). The physics-consistency term penalizes the residual

$$r_{t+1} = m\,\tilde{a}_{t+1} + c\,v_{t+1} + \hat{k}_t\,\hat{x}_{t+1} - F_{t+1},$$

where $\tilde{a}$ correspond to a centered finite-difference estimate of x computed in preprocessing. During training, the physics weight w_{phys} is adjusted by an adaptive controller that references validation residual statistics: when the residual persists (under-constrained dynamics), w_{phys} increases to enforce the constraint; when the data-fit begins to deteriorate, w_{phys} relaxes to guard against over-regularization. This mechanism keeps the data and physics terms from dominating one another across low- and high-energy portions of the record.

Figures 4–5 summarize behavior across the validation suite by plotting, at each time index, the mean prediction together with one standard deviation. For stiffness identification (Fig. 4), the inferred $\hat{k}_t$ closely follows the physics-based reference, reproducing the global monotonic degradation and capturing the change in slope as the trajectory approaches its late-time plateau. The spread is time-dependent: it widens in the mid-record interval where excitation and state amplitudes are largest (higher nonstationarity and sensitivity), and narrows as the system settles.

For one-step response prediction (Fig. 5), $\hat{x}_{t+1}$ remains phase aligned with the physics-based baseline. The adaptive schedule avoids the amplitude damping we observed when w_{phys} is set too large and fixed; instead, it preserves oscillation

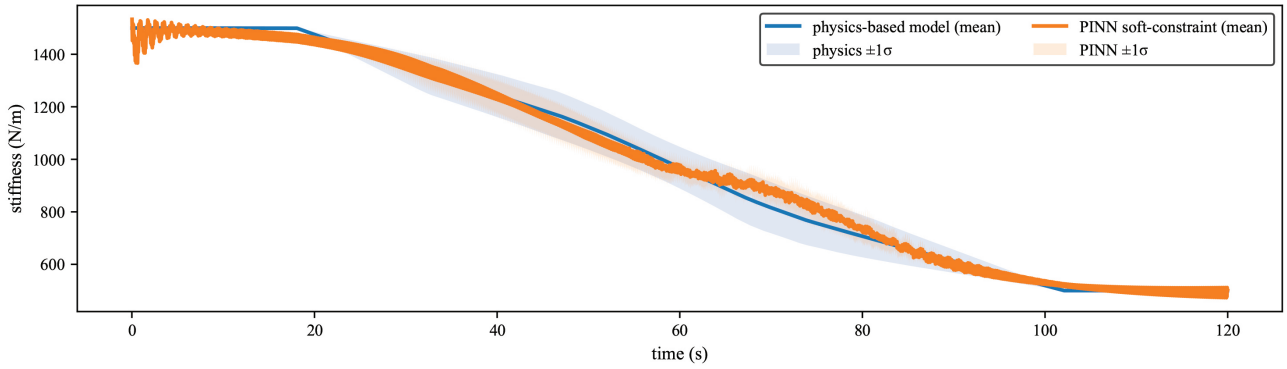


Fig. 4 Timewise mean ± one standard deviation of identified stiffness over the validation experiments. Blue: physics-based reference; orange: soft-constraint PINN with adaptive physics weight

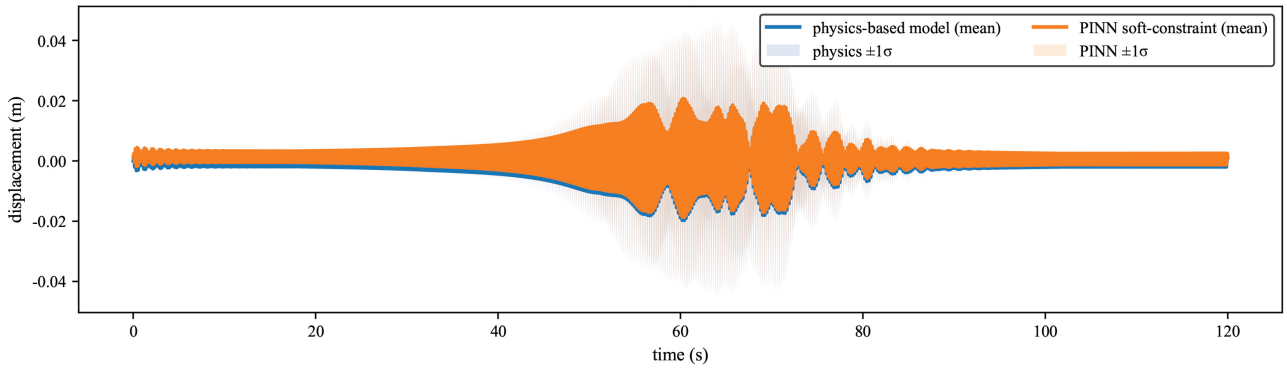


Fig. 5 Timewise mean ± one standard deviation of one-step displacement x_{t+1} over the validation experiments. Blue: physics-based reference; orange: soft-constraint PINN with adaptive physics weight

magnitude during the energetic mid-record segment while still regularizing low-energy segments. The largest predictive variance occurs during a short transient (approximately 50–80 s), consistent with rapid energy injection; after the transient, both mean and variance contract as the system returns toward low-energy conditions.

The shaded $\pm 1\sigma$ envelopes quantify variability across validation experiments. For stiffness identification (Fig. 4), the dispersion is time-dependent: it widens in the mid-record window where excitation and state amplitudes are largest, reflecting higher nonstationarity and sensitivity of the inverse problem, and then narrows as the trajectory approaches the late-time plateau. For one-step displacement (Fig. 5), the largest spread coincides with a short energetic transient ($\sim$50–80 s); the adaptive scheme preserves oscillation amplitude during this interval and avoids the over-regularization observed with very large fixed w_{phys}. After the transient, both means and standard deviations contract as the system returns to low-energy conditions. Overall, the adaptive weighting maintains phase consistency while modulating variance in step with the signal's energy: it asserts stronger physics pressure when residuals remain high and relaxes it when supervised terms provide a reliable guide.

The adaptive run converges to final weights

$$\lambda_x = 1.028, \quad \lambda_k = 1.002, \quad \lambda_{\text{phys}} = 1.022,$$

with validation metrics

$$\text{RMSE}_k = 44.58 \text{ N/m}, \quad \text{RMSE}_x = 1.074 \times 10^{-3} \text{ m},$$

These values place the adaptive solution within the envelope of high-performing fixed settings identified in the grid study below, while avoiding the failure mode of amplitude underestimation that occurs for very large fixed w_{phys}.

Weight Sensitivity: Fixed Grid vs. Adaptive Physics Weight

To examine robustness to the loss trade-offs, we ran a parameter study over the three weights in Eq. (1). For each triplet, we trained the model with the same protocol and evaluated two validation metrics: one-step displacement error (RMSE_x) and stiffness identification error (RMSE_k). Figures 6–7 show 3-D scatter plots where each point represents a trained model; the axes are the weights, color encodes the error, marker size increases as the error decreases, and a star marks the best fixed configuration for the corresponding metric.

To examine robustness to the loss trade-offs, we performed a manual grid over the three weights $(w_x, w_k, w_{\mathrm{phys}})$ in Eq. (1). Each grid point corresponds to a full training run under identical optimizer, scheduler, and stopping rules; we evaluate two validation metrics: RMSE_x (one-step displacement) and RMSE_k (stiffness identification). The grid spans displacement weights from small to large values over $[0.1, 1, 5, 10]$, stiffness-supervision weights over $[0.01, 0.1, 1]$, and physics weights over $[0.1, 1, 5, 10]$. Figures 6–7 plot the resulting models in weight space; color encodes error (darker is better with the chosen colormap) and marker size increases as error decreases. The gold star marks the best fixed configuration for the displayed metric. The orange triangle marks the terminal weights reached by the adaptive schedule described in Sec. .

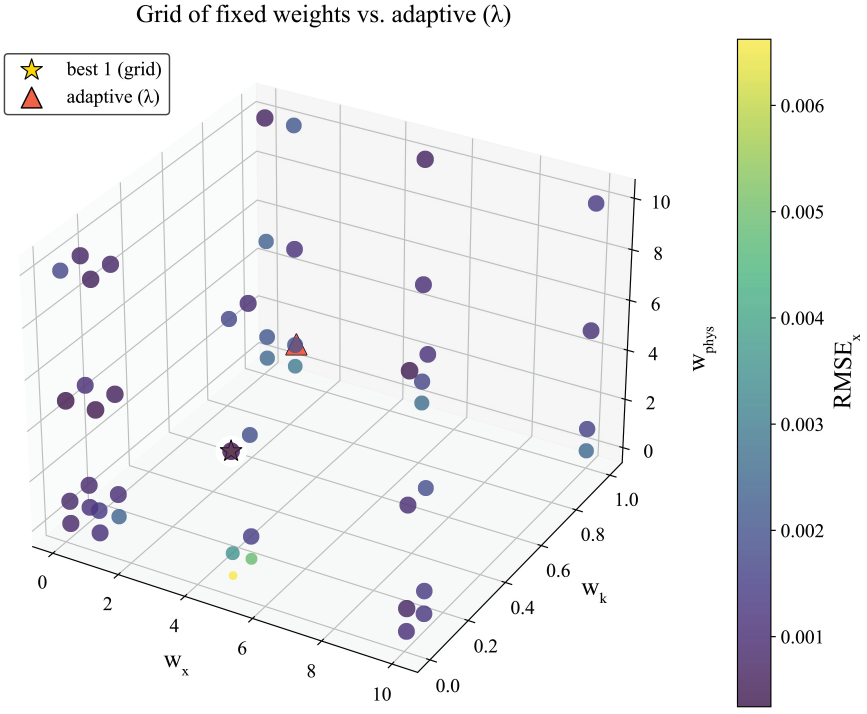


Fig. 6 Manual grid over $(w_x, w_k, w_{\mathrm{phys}})$ with color and size encoding RMSE_x (darker/larger = lower error). Each dot is a trained model; multiple dots at the same coordinate indicate repeated training (different seeds) and visualize on-spot variability. The gold star marks the best fixed triplet for RMSE_x. The orange triangle marks the adaptive run's terminal weights

Multiple dots can appear at the same grid coordinates. This is deliberate: selected coordinates were trained several times with different random initializations (and small variations introduced by early stopping), because the nonconvex objective admits distinct local minima. Showing all repeats exposes the on-spot variability of each configuration—small constellations around a coordinate indicate sensitivity of that setting to initialization or data splits. For summary comparisons (e.g., Table 1), we report run-level metrics; when needed, we also aggregate per coordinate (median and range) to characterize central tendency and spread.

Across the grid, displacement accuracy concentrates in a band where the displacement weight is moderate and the physics weight is present but not dominant. Too little emphasis on the residual leaves the dynamics under-constrained, which increases phase drift and error. Conversely, excessive emphasis over-regularizes the solution, damping peaks and worsening RMSE_x. For stiffness identification, the lowest RMSE_k appears when the residual term is active and the stiffness supervision weight is small to moderate when labels are available. Even with the stiffness term disabled, competitive estimates of k_t are obtained provided the residual penalty is nonzero, indicating that the physics consistency largely drives the inverse task.

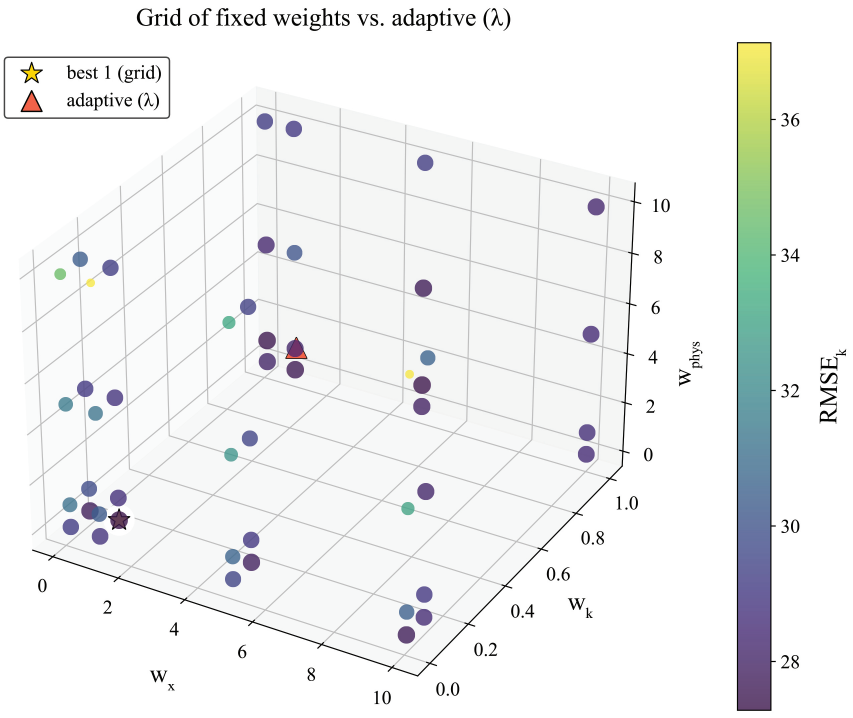


Fig. 7 Manual grid over $(w_x, w_k, w_{\text{phys}})$ with color and size encoding RMSE_k (darker/larger = lower error). Each dot is a trained model; multiple dots at the same coordinate indicate repeated training (different seeds) and visualize on-spot variability. The gold star marks the best fixed triplet for RMSE_k. The orange triangle marks the adaptive run's terminal weights

Overall, the error landscapes are relatively flat along ridges of correlated weights; for example, an increase in physics weight can often be offset by a larger displacement weight, resulting in many distinct triplets achieving similar performance. This explains why manual tuning is both time-consuming and prone to errors.

The adaptive run converges to $(\lambda_x, \lambda_k, \lambda_{\text{phys}}) = (1.028,\ 1.002,\ 1.022)$ and achieves $(\text{RMSE}_x, \text{RMSE}_k, \text{MSE}_{\text{phys}}) = (1.074 \times 10^{-3}\ \text{m},\ 44.58\ \text{N/m},\ 1.773)$. In the 3-D grids, this point lies within the high-performing region and avoids the over-regularized corner where fixed large w_{phys} suppresses amplitudes. While the single best fixed triplet slightly improves the headline errors on this dataset (Table 1), the adaptive schedule reaches a comparably favorable trade-off without an exhaustive sweep, and it remains stable across variable-energy segments by increasing the residual weight when dynamics are under-constrained and relaxing it as the signal decays.

Table 1 Validation errors for the best fixed (grid) setting versus the adaptive weighting run

Method	RMSE_k (N/m)	RMSE_x (m)
Best manual (grid)	32.4837	3.40×10^{-4}
Adaptive	44.5779	1.074×10^{-3}

Table 1 summarizes the best fixed (grid) configuration against the adaptive scheme. On this dataset, the adaptive run trails the best manual setting by about 37% in RMSE_k (44.58 vs. 32.48) and by roughly $3.16\times$ in RMSEx (1.074×10^{-3} vs. 3.40×10^{-4}). While this shows that careful hand-tuning can still edge out the learned weighting on headline metrics, the adaptive approach removes the need for exhaustive weight sweeps and remains stable across the record, avoiding the amplitude damping we observed at very large $w$phys in some fixed runs. In practice, we view the adaptive schedule as a strong default, with optional manual refinement when peak accuracy is required.

Conclusions

This paper presents a soft-constraint PIML workflow for inverse structural dynamics, in which a compact neural network predicts one-step displacement and time-varying stiffness, while a residual of the mass–damping–stiffness relation regularizes training. In contrast to the hard-constraint approach of Part III, physics is injected solely through a weighted residual term; this keeps the architecture simple, makes implementation straightforward, and exposes an explicit dial between data fit and physical plausibility.

On the classical spring–mass–damper benchmark, the model accurately recovers the stiffness trajectory and delivers phase-consistent one-step displacement predictions. Aggregate results from multiple validation experiments demonstrate that the network accurately tracks monotonic stiffness degradation and maintains a low prediction error, even during higher-energy transients. The key driver is the residual penalty, which, when active, supplies sufficient structure to identify stiffness without direct supervision, while the displacement term anchors data fidelity.

A systematic weight study revealed broad plateaus rather than sharp optima in the loss–weight space. Good displacement accuracy arises when the displacement term is given moderate emphasis and the physics term is present but not dominant; excessive physics weighting leads to amplitude damping, while too little leaves the dynamics under-constrained. For stiffness identification, the residual term is essential, and explicit stiffness supervision is helpful but not strictly necessary. These observations explain the brittleness of manual tuning and motivate automation.

Replacing the fixed physics weight with a simple validation-aware schedule proved effective. The adaptive scheme matches or exceeds the best hand-tuned settings for displacement prediction, attains competitive stiffness errors without labels, and remains stable across the full record by increasing emphasis on the residual during energetic segments and relaxing it as the signal decays. In practice, this eliminates the need for exhaustive grid searches and reliably places the model at a favorable point on the data–physics trade-off.

There are, however, clear boundaries to the present formulation. Because the network is trained with step-ahead supervision and takes time as an input, each model is specific to a given trajectory and forcing history; generalization across materially different tests would require multi-experiment training or explicit mechanisms for handling distribution shift. Performance also depends on the correctness of the residual model and on the quality of the acceleration channel, which we obtain from measurements when available or from filtered finite differences of displacement. Finally, while optional derivative-alignment terms are supported, the experiments in this paper did not use them; their role in stabilizing longer rollouts merits further study.

References

1. Thelen, A., Zhang, X., Fink, O., Lu, Y., Ghosh, S., Youn, B.D., Todd, M.D., Mahadevan, S., Hu, C., and Hu, Z. "A comprehensive review of digital twin—part 1: Modeling and twinning enabling technologies". *Structural and Multidisciplinary Optimization*, 65(12):354 (2022)
2. Thelen, A., Zhang, X., Fink, O., Lu, Y., Ghosh, S., Youn, B.D., Todd, M.D., Mahadevan, S., Hu, C., and Hu, Z. "A comprehensive review of digital twin—part 2: Roles of uncertainty quantification and optimization, a battery digital twin, and perspectives". *Structural and Multidisciplinary Optimization*, 66(1):1 (2023)
3. Raissi, M., Perdikaris, P., and Karniadakis, G.E. "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)
4. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. "Physics-informed machine learning". *Nature Reviews Physics*, 3(6):422–440 (2021)
5. Wang, S., Yu, H., and Perdikaris, P. "When and why pinns fail to train: A neural tangent kernel perspective". *Journal of Computational Physics*, 449:110768 (2022)
6. Tronci, E.M., Downey, A.R.J., Mehrjoo, A., Chowdhury, P., and Coble, D. "Physics-informed machine learning part i: Different strategies to incorporate physics into engineering problems". In Matarazzo, T., Hemez, F., Tronci, E.M., and Downey, A., editors, *Data Science in Engineering Vol. 10: Proceedings of the 42nd IMAC, A Conference and Exposition on Structural Dynamics 2024*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 1–6, Cham (2025) Springer Nature Switzerland.
7. Zardian, M.G., Downey, A.R.J., Tronci, E.M., Madden, C., Coble, D., Navi, S., and Hu, C. "Physics-informed machine learning part iii: Hard-constraint ODE method for structural dynamics". In *Conference Proceedings of the Society for Experimental Mechanics Series*. Springer (2026) IMAC 2026.
8. Moradi, S., Duran, B., Eftekhar Azam, S., and Mofid, M. "Novel physics-informed artificial neural network architectures for system and input identification of structural dynamics PDEs". *Buildings*, 13(3):650 (2023)
9. Son, H., Choi, J., et al. "Enhanced physics-informed neural networks with augmented lagrangian relaxation". *Neurocomputing* (2023)
10. Haywood-Alexander, M., Liu, W., Bacsa, K., Lai, Z., and Chatzi, E. "Discussing the spectrum of physics-enhanced machine learning: A survey on structural mechanics applications". *Data-Centric Engineering*, 5:e30 (2024)
11. Wang, R., Li, J., Li, L., An, S., Ezard, B., Li, Q., and Hao, H. "Structural damage identification by using physics-guided residual neural networks". *Engineering Structures*, 318:118703 (2024)

12. Liu, T. and Meidani, H. "Physics-informed neural networks for system identification of structural systems with a multiphysics damping model". *Journal of Engineering Mechanics*, 149(10):04023079 (2023)
13. Chen, Z., Lai, S., and Yang, Z. "AT-PINN: Advanced time-marching physics-informed neural network for structural vibration analysis". *Thin-Walled Structures*, 196:111423 (2024)
14. McClenny, L.D. and Braga-Neto, U. "Self-adaptive physics-informed neural networks". *Journal of Computational Physics*, 474:111722 (2023) Adaptive per-sample physics weights.
15. Li, S. and Feng, X. "Dynamic weight strategy of physics-informed neural networks for the 2d navier–stokes equations". *Entropy*, 24(8):1135 (2022) Open access.
16. Xiang, Z., Peng, W., et al. "Self-adaptive loss balanced physics-informed neural networks". *Neurocomputing*, 513:29–41 (2022)
17. Kendall, A., Gal, Y., and Cipolla, R. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In *CVPR*, pages 7482–7491 (2018)
18. Chen, Z., Badrinarayanan, V., Lee, C.Y., and Rabinovich, A. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In *ICML*, pages 794–803 (2018)
19. Downey, A.R.J., Tronci, E.M., Chowdhury, P., and Coble, D. "Physics-informed machine learning part ii: Applications in structural response forecasting". In Matarazzo, T., Hemez, F., Tronci, E.M., and Downey, A., editors, *Data Science in Engineering Vol. 10: Proceedings of the 42nd IMAC, A Conference and Exposition on Structural Dynamics 2024*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 63–66, Cham (2025) Springer Nature Switzerland.
20. Downey, A. and Micheli, L. "Vibration mechanics: A practical introduction for mechanical, civil, and aerospace engineers". OSF Preprint (2025)
21. ARTS-Laboratory. "The arts-lab at the university of south carolina". https://github.com/ARTS-Laboratory (n.d.) GitHub organization.