NYU | TANDON

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

Eleonora Maria Tronci[1], Austin R. J. Downey[2], Connor Madden[2], Mohsen Gol Zardian[2] and Daniel Coble[3]

[1] New York University, New York, USA
[2] University of South Carolina, Columbia, USA
[3] Duke University, Durham, USA

Problem Statement

Physics-based strategies are the most widely used to characterize complex phenomena, focusing on mathematical models that describe the physical principles governing the behavior of a dynamic system

- interpretable and generalizable
- flexibility to incorporate prior knowledge and constraints into

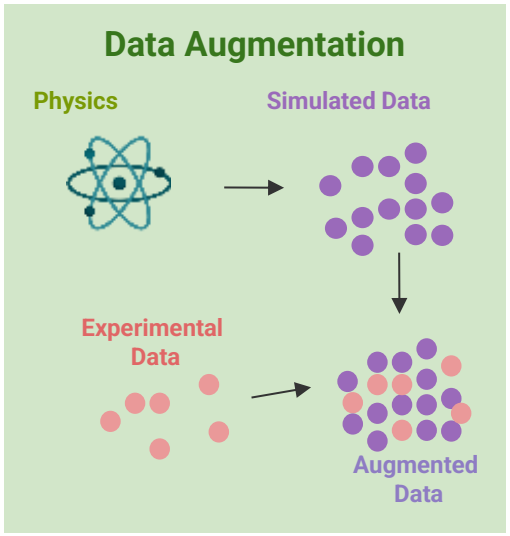
- often have large and time-varying modeling errors
- heavy computational burden
- complex dynamics

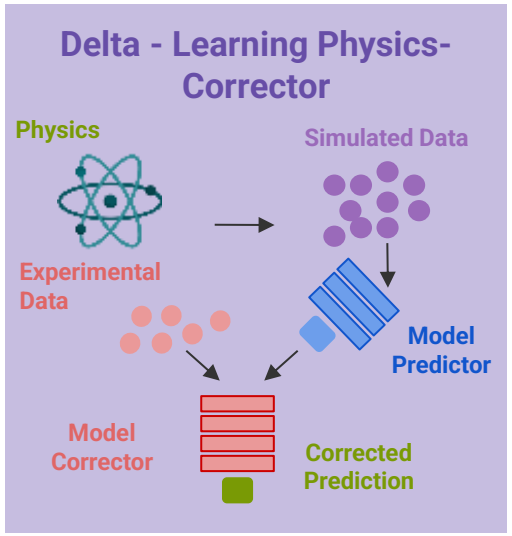
Data-driven strategies have advanced significantly due to the abundance of data and computing resources. These methods utilize data to learn the system dynamics

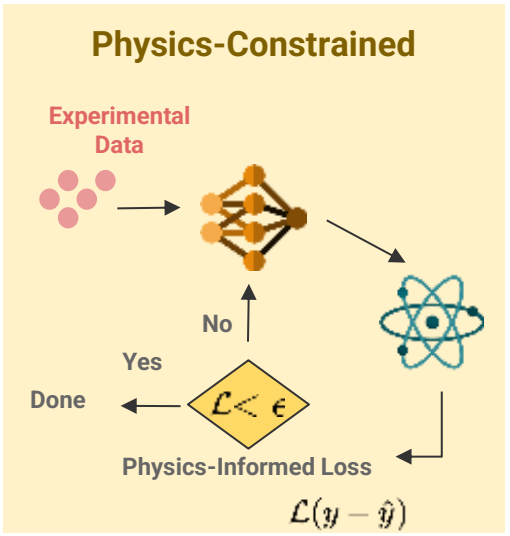
- flexible, adaptive, and scalable
- handle uncertainties and disturbances in the data

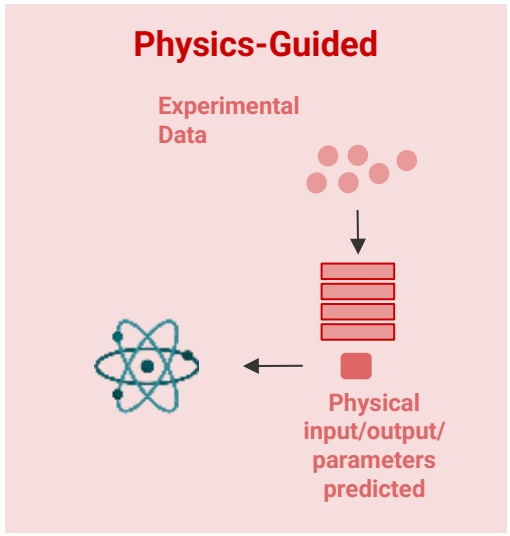
- can represent only the datasets they were trained to learn
- no inference capability for unseen conditions
- low interpretability and explainability

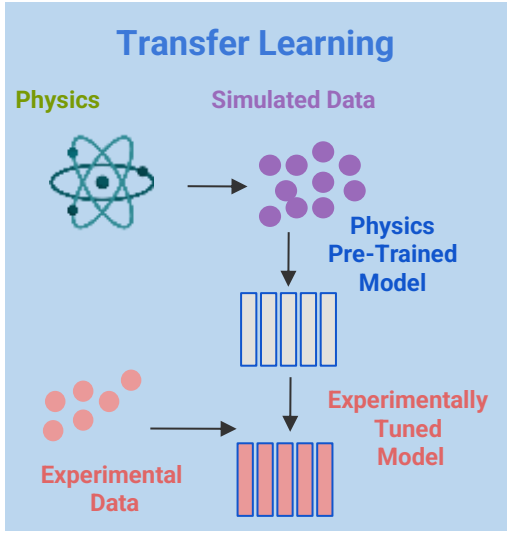
Literature

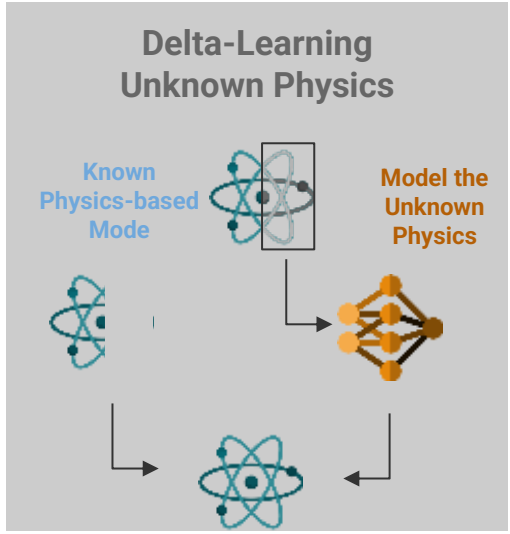


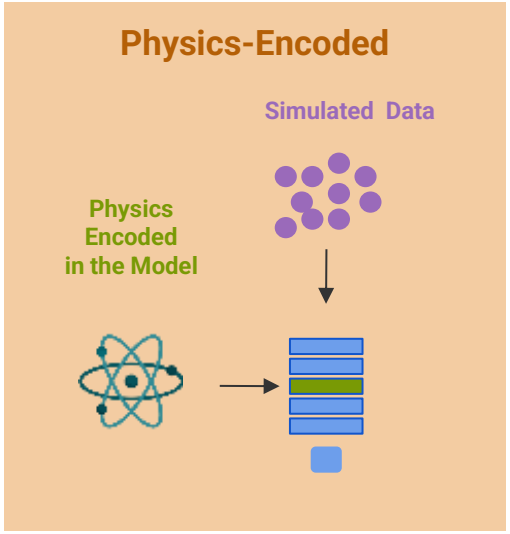












Karniadakis, G.E., Kevrekidis, I.G., Lu, L. *et al.* Physics-informed machine learning. *Nat Rev Phys* **3**, 422–440 (2021).

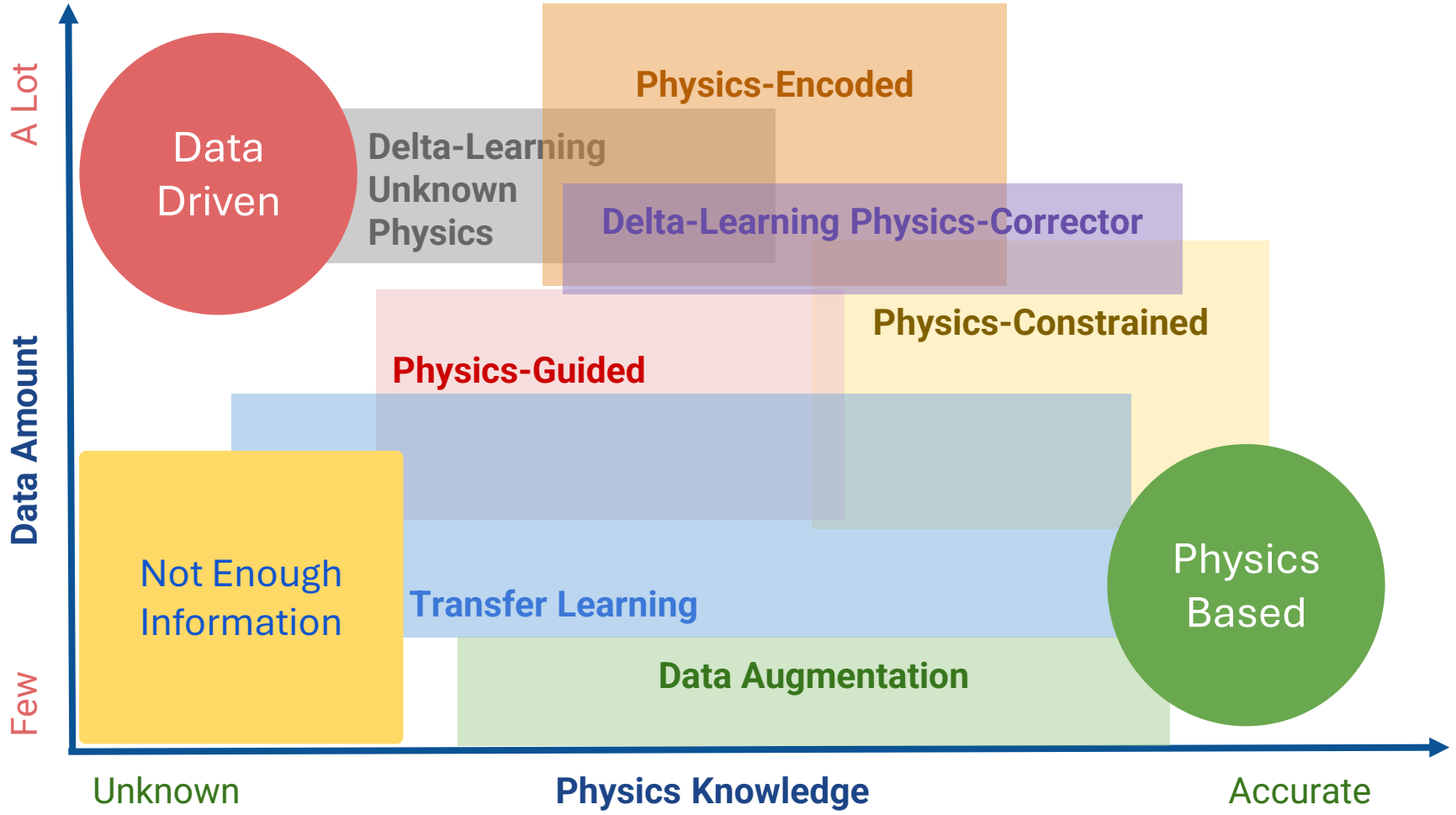
Problem Statement

Variety of strategies classified according to three major characteristics:

1. the **amount** and **quality** of **data** that is utilized to describe a given model

2. the **strategy** chosen to incorporate the physics into the problem

3. the level of **physical knowledge** and understanding representing the phenomena of interest



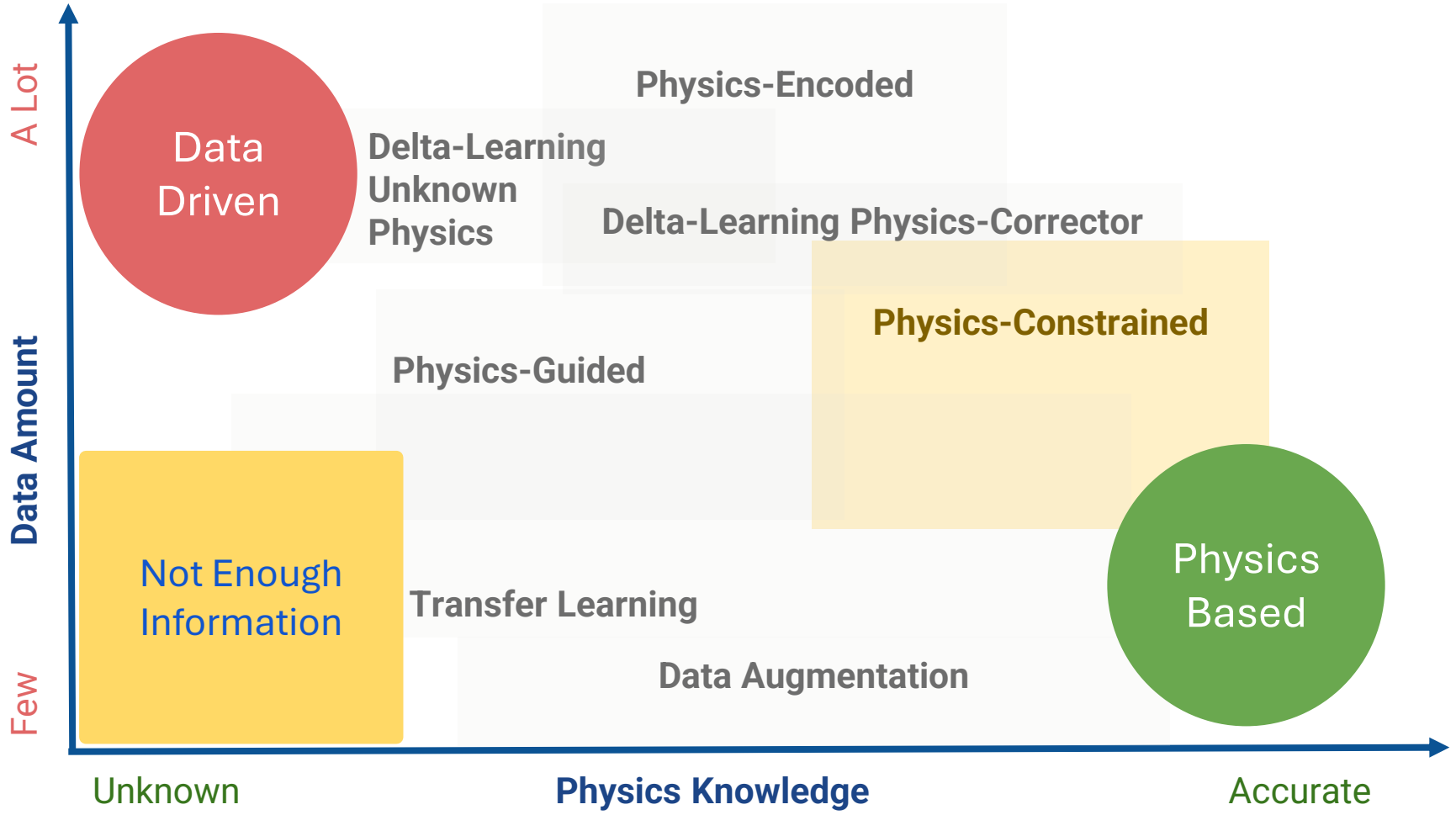
Problem Statement

Variety of strategies classified according to three major characteristics:

1. the **amount** and **quality** of **data** that is utilized to describe a given model

2. the **strategy** chosen to incorporate the physics into the problem

3. the level of **physical knowledge** and understanding representing the phenomena of interest



Soft vs. Hard Constraints: The Constrained PIML Spectrum

Hard Constraint

Soft Constraint

Soft vs. Hard Constraints: The Constrained PIML Spectrum

Hard Constraint

Physics is embedded in the architecture

satisfies equations exactly

Naturally suppress noise that does not align with physics

The only hyperparameters are the architecture

Is less flexible with unmodeled dynamics

Soft Constraint

Soft vs. Hard Constraints: The Constrained PIML Spectrum

Hard Constraint

Physics is embedded in the architecture satisfies equations exactly

Naturally suppress noise that does not align with physics

The only hyperparameters are the architecture

Is less flexible with unmodeled dynamics

Soft Constraint

Physics is added as a residual penalty to the loss function acting as a regulator

Weighting allows a decision on how much to trust data versus the physics

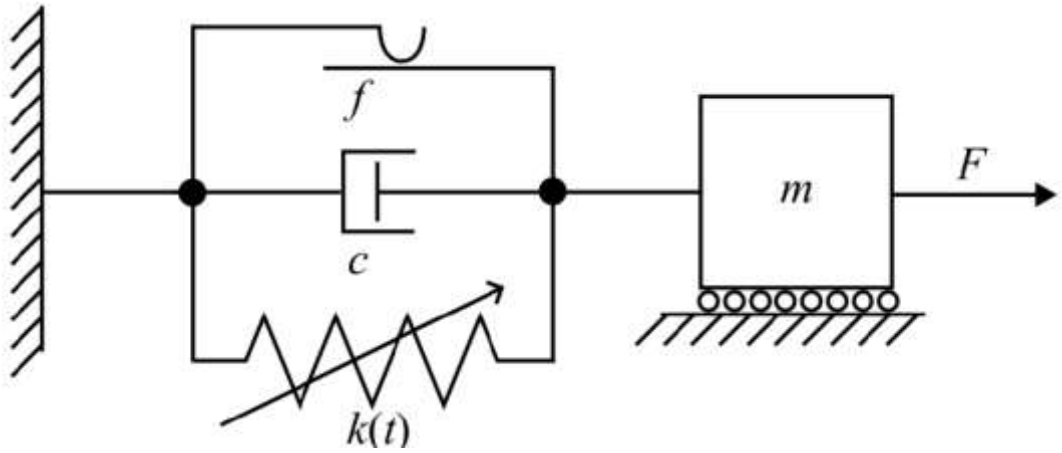
Architecture and weighting hyperparameters

Suited for systems with unmodeled dynamics or model-form uncertainty

Study Case: Simulated Spring-Mass-Damper

System and data generation

- SDOF oscillator with external forcing F(t)
- Time-varying stiffness k(t) (monotonic degradation)
- Reference model built in MATLAB Simulink/Simscape



Two regimes studied

- a) Linear (time-varying k(t))
- b) Extended: add nonlinear restoring force f

Why this benchmark is instructive

Time-varying stiffness creates a natural inverse problem. The soft-constraint residual supplies structure even when k labels are removed, but weight choice controls whether the solution stays physically plausible without sacrificing response amplitude.

Goal: estimate k(t) while predicting x(t) accurately

Study Case: Simulated Spring-Mass-Damper

$$ma_t + cv_t + k_t x_t = F_t$$

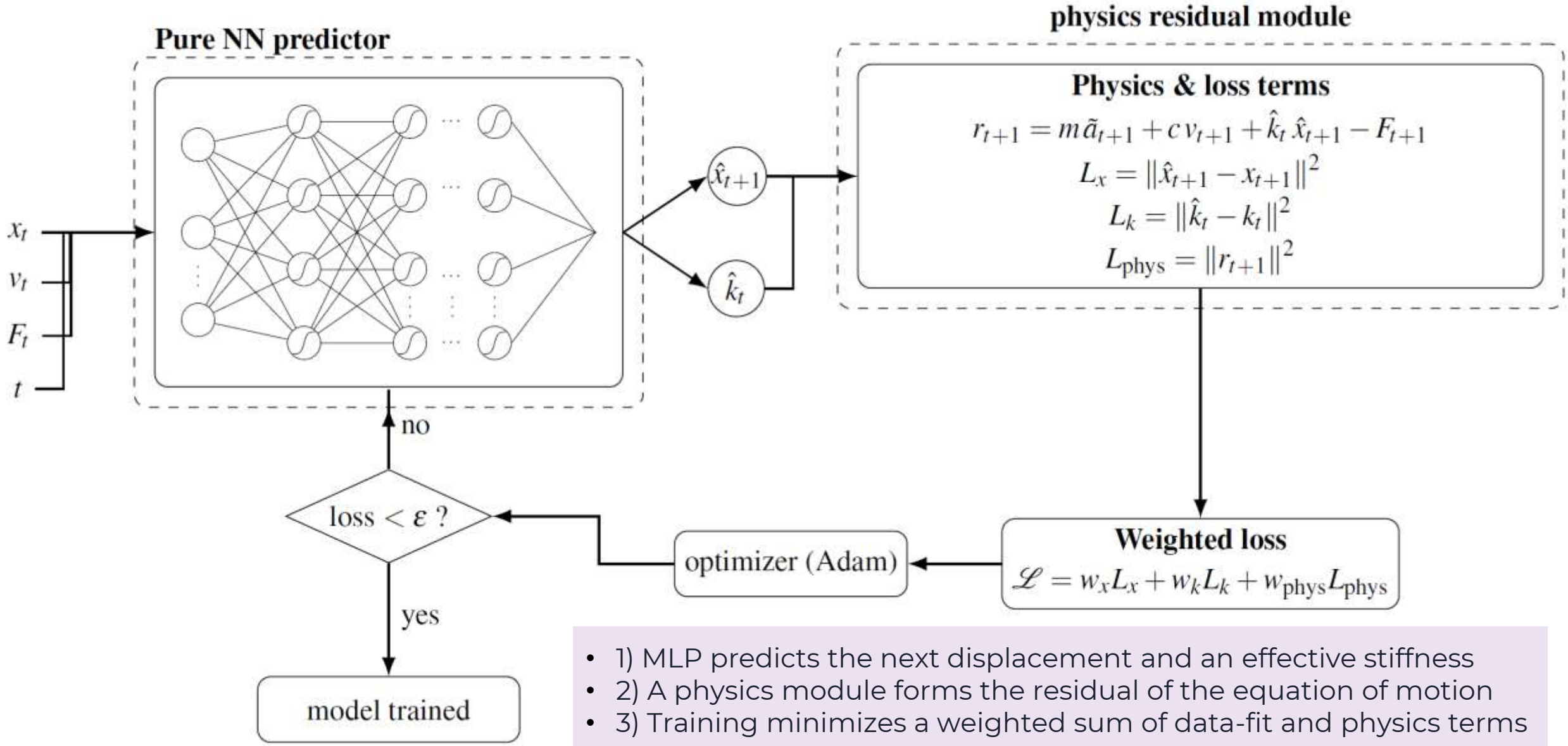
Step-ahead supervision

$$[x_t,\ v_t,\ F_t,\ t] \ \rightarrow\ [\hat{x}_{t+1},\ \hat{k}_t]$$

- Network input includes time → model becomes trajectory-specific
- Stiffness labels k_t optional (if missing, omit that loss term)
- Acceleration a_t may be measured or estimated via filtered finite differences

Feature	Part III (Hard Constraint)	Part IV (Soft Constraint)
Supervision Style	Sliding windows/segments	Step-ahead (one-step) predictor
Input Vector	Windowed segments $\{x, v, F\}$.	Instantaneous state: $[x_t, v_t, F_t, t]$.
Target Output	Integrated ODE state.	Next displacement $\hat{x}_{t+1}$ and stiffness $\hat{k}_t$
Batch Shape	3D: (Batch, Segments, Window).	2D: (B, D) (Batch, Dimensions)
Role of Time	Implicit in the ODE integration.	Explicit input: makes the model trajectory-specific

Physics-Loss Function



- 1) MLP predicts the next displacement and an effective stiffness
- 2) A physics module forms the residual of the equation of motion
- 3) Training minimizes a weighted sum of data-fit and physics terms

The Weighting Scheme

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

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

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

Strategy	Mechanism	Benefit
Manual Recipe	Fixed emphasis on data vs. physics	Improves extrapolation but can amplify noise
Self-Adaptive	Learns per-sample weights for "stubborn" residuals	Emphasizes difficult regions of the trajectory
Dynamic/Multi-task	Uncertainty-based or gradient-normalized balancing	Eliminates exhaustive grid searches and manual tuning

Fixed Weights Training Process

Stage 1
The Forward Pass

Input: Batch Data
$[x_t, v_t, F_t, t]$

↓

Model: Neural Network (MLP)

Next State $[x_{t+1}]$

Stiffness $[k_t]$

Outputs

Stage 2
The Loss Calculation

Data Fidelity

Compare
Prediction vs. Ground Truth

Data Loss (L_x, L_k)

Physics Consistency

Equation
$ma + cv + kx - F = 0$

Physics Residual (L_{phys})

Stage 3
The Weighted Sum

Weights

w_x w_k w_{phys}

Equation Block
Total Loss = $w_x L_x + w_k L_k + w_{phys} L_{phys}$

Action: Backpropagation

The Adaptive Weighting

Stage 1
The Forward Pass

Input: Batch Data
$[x_t, v_t, F_t, t]$

↓

Model: Neural Network (MLP)

Next State $[x_{t+1}]$

Stiffness $[k_t]$

Outputs

Stage 2
The Loss Calculation

Data Fidelity

Compare
Prediction vs. Ground Truth

Data Loss (L_x, L_k)

Physics Consistency

Equation
$ma + cv + kx - F = 0$

Physics Residual (L_{phys})

Stage 3
The Weighted Sum

Weights

λ_x λ_k λ_{phys}

Equation Block
Total Loss = $\lambda_x L_x + \lambda_k L_k + \lambda_{phys} L_{phys}$

Action: Backpropagation

The Adaptive Weighting

Stage 1
The Forward Pass

Input: Batch Data
$[x_t, v_t, F_t, t]$

↓

Model: Neural Network (MLP)

Next State
$[x_{t+1}]$

Stiffness
$[k_t]$

Outputs

Stage 2
The Loss Calculation

Data Fidelity

Compare
Prediction vs. Ground Truth

Data Loss
(L_x, L_k)

Physics Consistency

Equation
$ma + cv + kx - F = 0$

Physics Residual
(L_{phys})

Adapter

- Takes raw losses and divides them by their running average
- Scales mismatched units to a unified range

Stage 3
The Weighted Sum

Weights

λ_x λ_k λ_{phys}

Equation Block
Total Loss = $\lambda_x L_x + \lambda_k L_k + \lambda_{phys} L_{phys}$

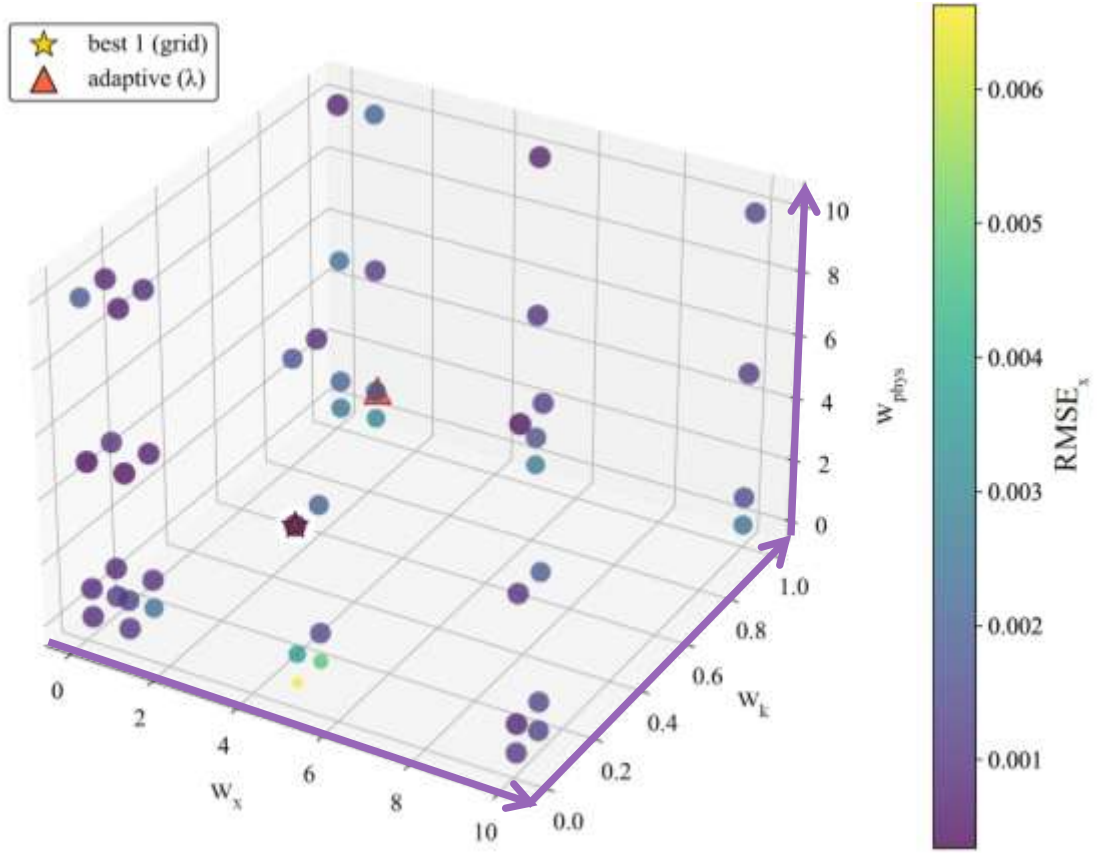
Action: Backpropagation with dual optimization

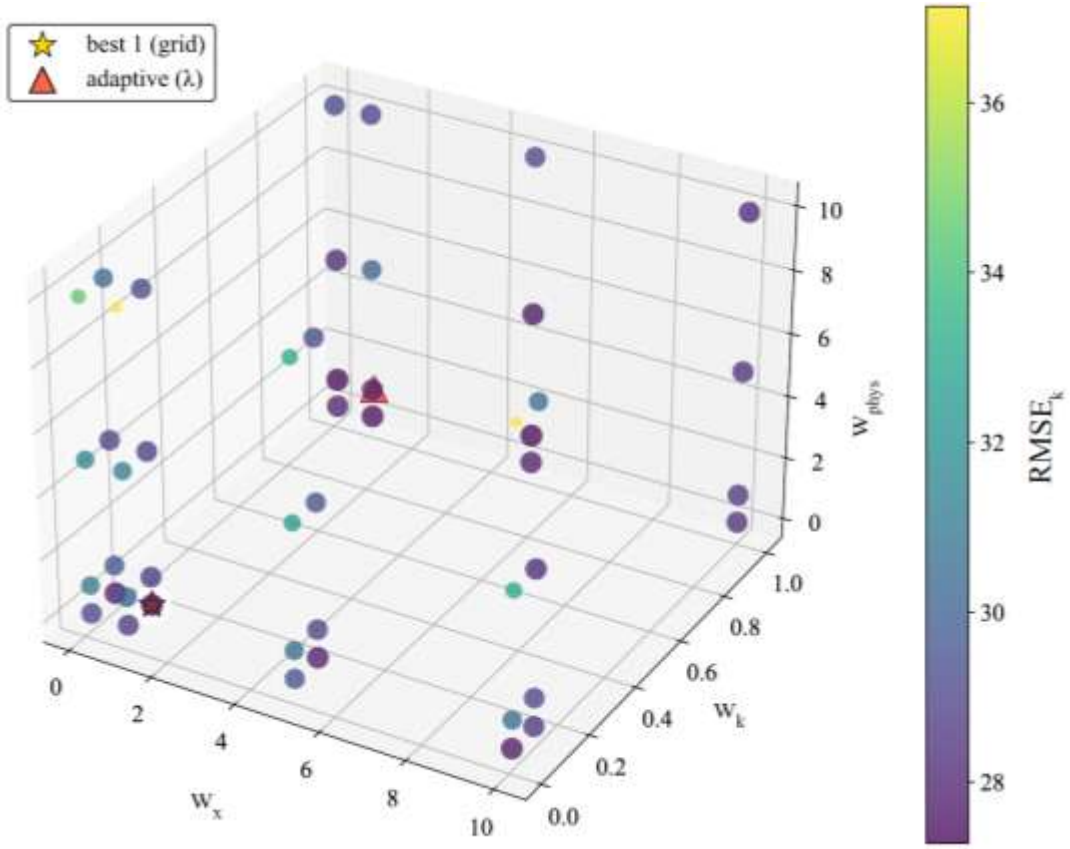
Updates Model Parameters
Updates Weight Parameters using a separate Adam optimizer with a smaller learning rate[4]

Classical training vs. Adaptive

Represent **low error** (Better performance)
Represent **high error** (Worse performance)

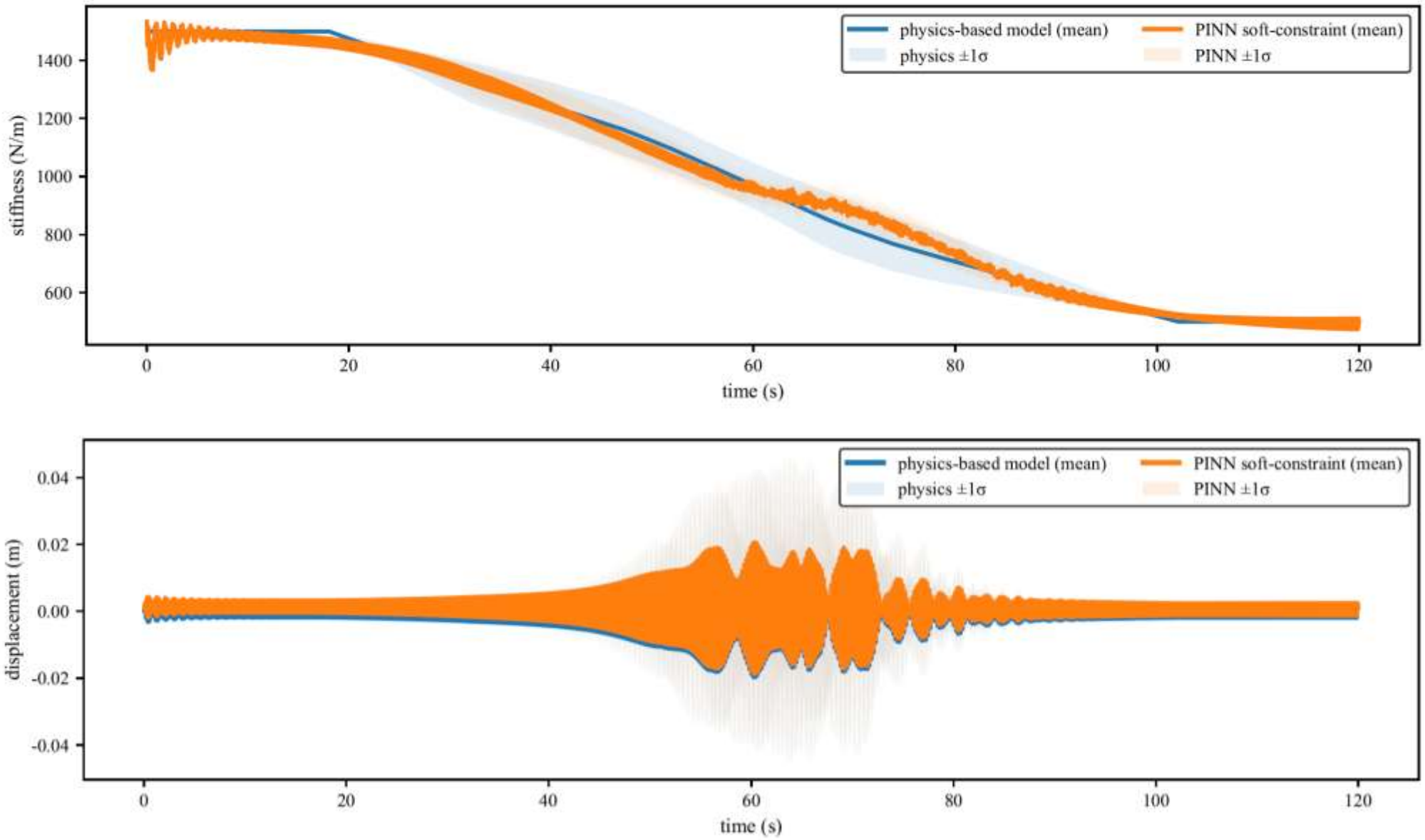
The grid spans displacement weights [0.1,1,5,10], stiffness-supervision weights over [0.01,0.1,1], and physics weights over [0.1,1,5,10].





- Each axis corresponds to one of the weights in our loss function
- Each dot in this cube represents a single, fully trained PINN model with that specific combination of fixed weights
- We visualize the error (RMSE) using both color and size

Classical training vs. Adaptive



Conclusions, Limitations and How to Use it

Main takeaways

- Soft-constraint PINN is a clean, implementable baseline for inverse structural dynamics.
- Residual penalty supplies structure: can identify k(t) with minimal or even no direct k supervision.
- Weight choice strongly affects behavior; adaptive scheduling is a robust default that avoids exhaustive sweeps.
- Uncertainty can be read from ensemble/validation spread ± 1σ curves).

Practical recipe (starting point)

- 1) Start with moderate w_{phys} (not dominant) and verify no amplitude damping
- 2) If $k(t)$ is unstable or non-physical → raise w_{phys} (or add k supervision if available)
- 3) If $x(t)$ amplitude collapses → increase w_x or relax w_{phys}
- 4) Prefer an adaptive schedule when operating across variable-energy segments.

Conclusions, Limitations and How to Use it

Limitations (important for real structures)

- Trajectory-specific: time is an input → limited generalization across tests.
- Depends on residual-model correctness; unmodeled physics can bias k(t).
- Acceleration quality matters (measured vs differentiated).

Future directions suggested by the authors

- Multi-experiment training to improve generalization
- Explore derivative-alignment terms for longer rollouts
- Broader adaptive strategies (reduce dependence on manual tolerances)

Thank You

Questions?