PHDME: Physics-Informed Diffusion Models without Explicit Governing Equations

Anonymous authors
Paper under double-blind review

Abstract

Diffusion models show expressive results for forecasting complex dynamics, but purely data-driven variants can be unreliable when observations are scarce. Physics-informed methods address this limitation by enforcing structure through governing equations, however, those explicit underlying equations are often unknown or too complex to model accurately. Hence, we introduce **PHDME** (**P**ort-**H**amiltonian **D**iffusion **M**odel), a structured diffusion framework that learns physical constraints from sparse observations. PHDME first learns the underlying dynamics using a data-efficient Gaussian process, under the distributed Hamiltonian structure. Samples from this learned posterior of the Gaussian process are then used to construct a diverse, physically structured training dataset, enabling the diffusion model to learn representations that preserve shared dynamical structure under limited observations. Then the learned deep physics prior guides the diffusion training based on Bayesian uncertainty. Experiments on a synthetic string, real-world spring data, and a two-dimensional nonlinear membrane show that PHDME provides predictions with better accuracy and physical correctness. Besides, PHDME does not require access to analytic governing equations during training or sampling. These results demonstrate that reliable dynamical predictions can be obtained even when both data and knowledge of the underlying dynamics are limited.

1 Introduction

Partial differential equation (PDE) systems arise across a wide range of real-world domains, including soft robotics (Zheng & Lin, 2022), heat transfer processes (Zobeiry & Humfeld, 2021), and electrical systems (Cieślik, 2021). Accurate prediction of these PDE systems is critical for system control (Beckers & Colombo, 2026), collision avoidance (Missura & Bennewitz, 2019), and long-horizon planning (Li et al., 2025). However, forecasting remains challenging in practice because the underlying dynamics are often spatiotemporally complex and observational data are limited. A common example arises in robotics, where fully instrumenting a soft robot with sensors is often expensive and physically difficult. Moreover, computing PDE solutions with traditional numerical solvers are computationally expensive, which is often impractical for real-time control or long-horizon forecasting. To address these challenges, various deep learning frameworks have been proposed to learn the underlying dynamics directly from collected data. However, methods such as neural ODEs (Chen et al., 2018) and neural PDEs (Zubov et al., 2021) still incur substantial computational costs. Training requires repeated forward-time integrations together with backward sensitivity computations through potentially stiff, multiscale solvers. These costs scale with the prediction horizon, state dimension, and solver stiffness, leading to high runtime and memory usage that force compromises in model fidelity and spatial grid resolution. Although alternative frameworks, such as discrete-time autoregressive models, avoid integration costs, they suffer from error accumulation over long rollouts (McCabe et al., 2023).

We therefore seek an efficient approach that predicts all future states jointly without autoregressive rollout. Learning a conditional distribution over these trajectories would further allow us to sample multiple plausible system evolutions and estimate predictive uncertainty. Diffusion models (Sohl-Dickstein et al., 2015) offer flexible generative priors for forecasting in dynamical systems. Unlike traditional deterministic predictors, diffusion models learn distributions over possible dynamics rather than producing a single deterministic output. Denoising diffusion defines a forward Markov corruption process with Gaussian perturbations and

trains a reverse process (Ho et al., 2020; Karras et al., 2022) to estimate the score of the data distribution. This approach has achieved state-of-the-art synthesis in images (Xia et al., 2023; Xu et al., 2023), videos (Ho et al., 2022; Liang et al., 2024), and audio (Guo et al., 2024). In scientific machine learning, diffusion models support inverse problems (Chung et al., 2023) and planning under uncertainty (Römer et al., 2025). System trajectories can be represented as images or volumes whose axes index space and time and whose entries encode physical state variables, seeFig. 1. Conditioned on initial observations, a diffusion model can then jointly generate future states across this grid. Nevertheless, standard diffusion models are purely data-driven, so their samples may align with dataset statistics while violating the physical laws that govern the real world, especially under distribution shift. This lack of explicit physics limits performance and reliability and weakens guarantees in applications such as safety-critical systems (Tan et al., 2023).

Physics-informed training addresses this gap by constraining learning with governing equations. Classic approaches such as physics-informed neural networks (PINNs) (Raissi et al., 2019) encourage learned solutions to satisfy the underlying physics, and recent work has begun to embed such constraints into generative modeling (Shu et al., 2023; Bastek et al., 2024). However, these approaches typically require the governing equations to be known and enforceable during training. In many real systems, the exact governing equations are unknown or prohibitively complex to model, and observations are limited. For example, deriving first-principles equations for soft robots from only a few video frames is particularly challenging because of their highly nonlinear and unstructured dynamics. Under these nontrivial conditions, standard physics-informed pipelines are difficult to deploy. Port-Hamiltonian systems (PHS) provide a physical inductive bias by describing dynamics through stored energy, represented by a Hamiltonian, together with dissipation and energy exchange with the environment (van der Schaft & Jeltsema, 2014). Distributed port-Hamiltonian systems (dPHS) extend this formulation to fields evolving over space and time, accounting for energy transport within the domain and across its boundary. Given a general differential operator, unknown energy functions and operator coefficients can be learned from observations, enabling structured dynamics learning without requiring the complete governing equations in advance (Tan et al., 2024).

Contribution: We provide rapid, physically reliable, multistep forecasting in settings with limited observations. PHDME learns a structured physics prior from observations using a Gaussian process distributed port-Hamiltonian system (GP-dPHS) (Tan et al., 2024). The learned deep physics prior helps to deal with the limited data dilemma by drawing posterior samples to perform data enhancement, and contributes to dynamic representation learning by serving as a deep physics prior. This coupling enables forecasting that respects learned physical structure without requiring explicit governing equations under limited data settings.

Our contributions can be summarized as: **1. Learning with physics from limited observations.** We introduce PHDME, a diffusion framework that uses a learned physics prior both to expand the training data and to guide learning. Starting from a limited set of observations, we draw samples from the GP-dPHS posterior to generate additional, artificial training data. The diffusion model then learns to predict future states jointly from initial observations, using the learned physics *without access* to complete analytic governing equations. **2. Structure-preserving diffusion learning with an uncertainty-weighted loss.** PHDME incorporates the dPHS structure to promote physical consistency. The Bayesian nature of the GP prior provides predictive uncertainty estimates, which adaptively weight the physics loss. This weighting encourages stronger adherence to the physics prior in high confidence regions, while reducing its influence in regions of high uncertainty.

2 Preliminaries

This section provides a brief overview of denoising diffusion models and Gaussian process distributed Port-Hamiltonian systems.

2.1 Denoising Diffusion Models

Diffusion models have demonstrated considerable potential across various domains (Ho et al., 2020; Song & Ermon, 2019; Dhariwal & Nichol, 2021). Recent efforts have extended them to time-series forecasting (Rasul et al., 2021), super-resolution dynamical prediction (Rühling Cachay et al., 2023), and time-invariant physics-

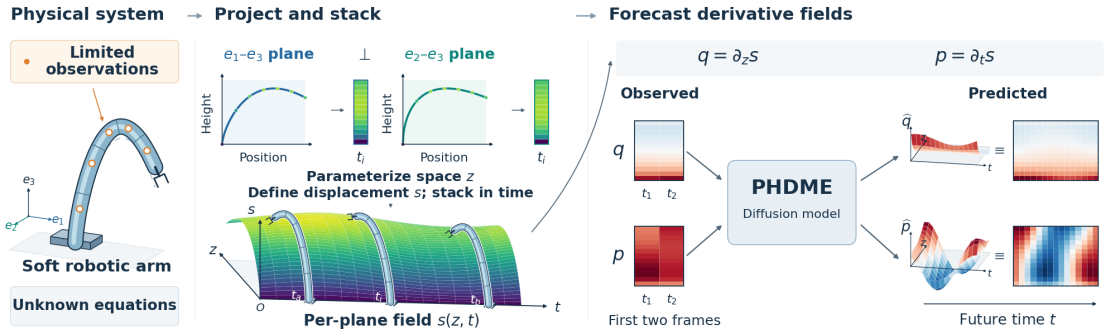

Figure 1: The left panel depicts a soft-robot scenario in which a flexible continuum manipulator has difficult-to-specify dynamics. The middle panel uses a top-down parameterization in which the z-axis is the spatial projection along the arm, pixel value represents displacement, and the t-axis is physical time. The resulting image representation allows iterative reverse diffusion to generate the full spatiotemporal field without an autoregressive physical-time rollout.

informed generation (Bastek et al., 2024). However, spatiotemporal forecasting with physical guarantees remains underexplored, especially when the governing equations are unknown or difficult to obtain.

Diffusion model. Let $\boldsymbol{A}^{(m)}$ be a noised image at step $m \in \{0, \dots, M\}$, where the diffusion index m is distinct from the physical time index t. The forward corruption process is linear Gaussian: $\boldsymbol{A}^{(m)} = \alpha_m \boldsymbol{A}^{(0)} + \sigma_m \boldsymbol{\varepsilon}$ with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; the schedule $\{(\alpha_m, \sigma_m)\}_{m=0}^{M}$ yields $\boldsymbol{A}^{(M)} \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the x_0 parameterization, a neural denoiser predicts the clean sample from a noised input and an optional condition $\mathbf{y}$ via $\widehat{\boldsymbol{A}}^{(0)} = f_\theta([\boldsymbol{A}^{(m)}, \mathbf{y}], m)$. The reverse transition uses the closed-form Gaussian posterior with mean $\boldsymbol{\mu}_m(\boldsymbol{A}^{(m)}, \widehat{\boldsymbol{A}}^{(0)})$ and variance $\tilde{\sigma}_m^2 \mathbf{I}$, both fixed by the forward schedule. The network weights w_m are set by using Min-SNR-5 weighting (Hang et al., 2023). Training minimizes the following timestep-weighted reconstruction loss:

$$\mathcal{L}_{\mathrm{DDPM}}(\theta) = \mathbb{E}_{m, \boldsymbol{A}^{(0)}, \boldsymbol{\varepsilon}} \left[w_m \left\| f_\theta([\alpha_m \boldsymbol{A}^{(0)} + \sigma_m \boldsymbol{\varepsilon}, \mathbf{y}], m) - \boldsymbol{A}^{(0)} \right\|^2 \right].$$

Sampling procedure. Starting from $\boldsymbol{A}^{(M)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, generation iterates $\boldsymbol{A}^{(m-1)} = \boldsymbol{\mu}_m(\boldsymbol{A}^{(m)}, \widehat{\boldsymbol{A}}^{(0)}) + \tilde{\sigma}_m \boldsymbol{z}$ with $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; the stochastic reverse process can generate diverse trajectories conditioned on $\mathbf{y}$.

2.2 Gaussian Process Distributed Port-Hamiltonian Systems

Based on Hamiltonian dynamics, GP-dPHS is a physics-informed PDE learning method that both generalizes well from sparse data and provides uncertainty quantification (Tan et al., 2024). Composing Hamiltonian systems through input and output ports yields port-Hamiltonian systems, a class of dynamical systems in which ports formalize interactions among components. The Hamiltonian can also be interpreted as an energy representation of the system. This framework applies to the classical finite-dimensional setting (Beckers et al., 2022) and extends naturally to distributed-parameter and multivariable cases. In the infinite-dimensional formulation, the interconnection, damping, and input/output matrices are replaced by matrix differential operators that do not explicitly depend on the state or energy variables. Under this structure, the system model follows systematically once the Hamiltonian is specified. This general formulation is sufficiently versatile to represent various PDEs and has been shown to capture a wide range of physical phenomena, including heat conduction, piezoelectricity, and elasticity. We next recall the definition of distributed port-Hamiltonian systems presented by (Macchelli et al., 2004).

More formally, let $\mathcal{Z}$ be a compact subset of $\mathbb{R}^n$ representing the spatial domain, and consider a skew-adjoint constant differential operator J together with a constant differential operator G_d. Define the Hamiltonian functional $\mathcal{H}\colon \mathcal{X} \to \mathbb{R}$ as $\mathcal{H}(\boldsymbol{x}) = \int_{\mathcal{Z}} H(z, x)\, dV$, where $H\colon \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ is the energy density. Let $\mathcal{W}$ denote the space of vector-valued smooth functions on $\partial\mathcal{Z}$ representing the boundary terms, $\mathcal{W} := \{w \mid w =$

$B_{\mathcal{Z}}(\delta_{\boldsymbol{x}}\mathcal{H}, \boldsymbol{u})\}$, as defined by the boundary operator $B_{\mathcal{Z}}$. The general formulation of a multivariable dPHS Σ is then

$$\Sigma(J, R, \mathcal{H}, G) = \begin{cases} \frac{\partial \boldsymbol{x}}{\partial t} = (J - R)\delta_{\boldsymbol{x}}\mathcal{H} + G_d\boldsymbol{u} \\ \boldsymbol{y} = G_d^*\delta_{\boldsymbol{x}}\mathcal{H} \\ w = B_{\mathcal{Z}}(\delta_{\boldsymbol{x}}\mathcal{H}, \boldsymbol{u}), \end{cases} \tag{1}$$

where R is a constant differential operator that accounts for energy dissipation. Furthermore, $\boldsymbol{x}(t, \boldsymbol{z}) \in \mathbb{R}^n$ denotes the state (also called the energy variable) at time $t \in \mathbb{R}_{\geq 0}$ and location $\boldsymbol{z} \in \mathcal{Z}$, while $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}^m$ denote the input/output ports; see (Tan et al., 2024) for details. In general, J defines the interconnection among elements in the dPHS, whereas the Hamiltonian H characterizes their dynamical behavior. The structure of J primarily involves partial differential operators. The port variables $\boldsymbol{u}$ and $\boldsymbol{y}$ are conjugate in the sense that their duality product defines the energy flow exchanged between the system and its environment.

When the system dynamics are only partially known, the Hamiltonian can be modeled within a probabilistic framework using Gaussian processes. A Gaussian process is fully specified by a mean function and a covariance function, and as a nonparametric Bayesian prior, it is well-suited for smooth Hamiltonian functionals. Its invariance under linear transformations further supports consistent representation propagation through the operators that define the dynamics (Jidling et al., 2017).

Combining these concepts, we encode the unknown latent Hamiltonian of a distributed system within a dPHS model to ensure physical consistency. The unknown dynamics are captured by approximating the Hamiltonian functional with a GP while treating the matrices J, R, and G (more precisely, their estimates $\hat{J}_\Theta$, $\hat{R}_\Theta$, and $\hat{G}_\Theta$) as hyperparameters. This yields the following GP representation of the system dynamics

$$\frac{\partial \boldsymbol{x}}{\partial t} \sim \mathcal{GP}(\hat{G}_\Theta \boldsymbol{u}, k_{dphs}(\boldsymbol{x}, \boldsymbol{x}')),$$

with a physics-informed kernel function defined as

$$k_{dphs}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2(\hat{JR}_\Theta)\delta_{\boldsymbol{x}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\varphi_l^2}\right) \delta_{\boldsymbol{x}'}^\top (\hat{JR}_\Theta)^\top,$$

where $\hat{JR}_\Theta = \hat{J}_\Theta - \hat{R}_\Theta$, and the kernel is based on the squared-exponential function. Training this GP-dPHS model involves optimizing the hyperparameters Θ, φ_l, and σ_f by minimizing the negative log marginal likelihood. Thus, the GP learns the physics representation prior without assuming a particular functional form; the learned prior is fully described by its structured mean and variance functions.

Following the GP-dPHS formulation of (Tan et al., 2024), a GP prior is assigned to the Hamiltonian $\hat{\mathcal{H}}$. The closure of GPs under linear operators induces a GP model of the dynamics under the prescribed dPHS structure, enabling unknown dynamics to be learned from observations with predictive uncertainty. The separable isotropic Hamiltonian parameterization used for the two-dimensional membrane is detailed in (Appendix A.8).

3 Proposed PHDME

In this section, we discuss the assumptions and problem formulation, followed by a detailed introduction to the proposed **P**ort-**H**amiltonian **D**iffusion **M**odel without **E**xplicit underlying equations (PHDME). The method enhances predictive performance by leveraging learned energy representations and observation-dependent confidence from the GP-dPHS prior.

3.1 Assumptions and Settings

We consider a physical state $\boldsymbol{x}(t, z)$ evolving over a spatial domain $\mathcal{Z}$ and a finite time horizon $[0, T]$. Only a few trajectories of the system are available. Our goal is to predict a trajectory across space and time for known boundary conditions but potentially previously unseen initial conditions, specified by the initial

observations $\mathbf{c}_{\text{init}}$. We seek a conditional distribution over complete future trajectories, so that multiple plausible evolutions can be sampled rather than returning only a single forecast.

The desired forecasts should agree with held-out observations and respect the physical structure of the underlying system, including its boundary conditions and energy behavior. The complete analytic governing equations are unavailable for training and prediction. The task is therefore to support accurate and physically consistent forecasting from sparse data and structural knowledge alone.

Many physical systems admit a distributed port-Hamiltonian representation, including systems describing elasticity, heat conduction, and piezoelectricity (Macchelli et al., 2004). Motivated by this broad applicability, we make the following assumptions:

Assumption 1 *For each available trajectory, the state is observed at a finite set of known times and spatial locations. Together with the recorded inputs, these observations form the dataset*

$$\mathcal{D} = \left\{ \big(t_i, z_j, \boldsymbol{x}(t_i, z_j), \boldsymbol{u}(t_i)\big) \right\}_{i=0,\dots,N_t-1;\ j=0,\dots,N_z-1}.$$

The initial observations and boundary conditions for each forecasting task are available. Any external input $\boldsymbol{u}$ affecting the trajectory is known over the prediction horizon; in an unforced system, $\boldsymbol{u} = 0$.

Assumption 2 *The underlying system admits a distributed port-Hamiltonian representation of the form*

$$\partial_t \boldsymbol{x} = (J - R)\delta_{\boldsymbol{x}}\mathcal{H} + G_d \boldsymbol{u}.$$

The admissible operator templates are known: J describes power-preserving interconnection, R describes nonnegative dissipation, and G_d specifies where inputs act. The Hamiltonian $\mathcal{H}$, its variational derivatives, and unknown coefficients within these templates are not supplied. The solution $\boldsymbol{x}(t, z)$ is continuous in time and space on $[0, T] \times \mathcal{Z}$. In addition, the state and energy functional have the regularity required for the derivatives in this representation to be well defined.

These assumptions distinguish knowledge of the system's physical structure from knowledge of its exact equations. We assume the former, together with sparse state observations, while the constitutive relations and unspecified physical coefficients remain unknown.

3.2 PHDME Framework

Rather than integrating a learned vector field through a long autoregressive rollout, PHDME generates a complete spatiotemporal trajectory through one reverse-diffusion chain conditioned on the initial two frames. The training pipeline has two stages, as illustrated in Figure 2. In the first stage, scarce observations are encoded through the dPHS structure to learn a probabilistic energy representation using Gaussian processes, and posterior sample draws from this model generate an augmented training dataset for diffusion training. Hence, a powerful physics-informed prior is obtained in the first stage using very limited data, which will enhance the dataset for the diffusion training, as well as serving the role of physics guidance for diffusion training in the next stage. In the second stage, the diffusion model is trained with learned physics. To be more detailed, we will introduce the floor-aware projection physics loss in the following part. Under the limited observation assumption, the Gaussian prior can be uncertain and rough in derivatives in some regions, direct physics loss like (Bastek et al., 2024) may lead to non-smooth solutions that are hard to optimize. Essentially, this uncertainty-weighted projection loss calculates the smooth version of the energy residual to inform the diffusion model, while trusting the physics prior more where the confidence is high.

(stage 1, physics prior learning) Data collection and structured prior training. We observe the state $\mathbf{x}(t, z)$ at discrete times and spatial locations, $\mathcal{D} = \left\{ t_i,\ z_j,\ \mathbf{x}(t_i, z_j),\ \boldsymbol{u}(t_i) \right\}_{i=0,j=0}^{i=N_t-1,\, j=N_z-1}$. Because the measurements are sparse and Hamiltonian learning requires derivatives, we fit a smooth Gaussian process interpolant to $\mathbf{x}(t, z)$ using a squared-exponential kernel and exploit closed-form GP differentiation to obtain the required state derivatives. At each t_i, we evaluate the interpolated state and its time derivative on a refined spatial grid, yielding the stacked vectors $\tilde{\boldsymbol{x}}(t_i)$ and $\partial_t \tilde{\boldsymbol{x}}(t_i)$. Collecting these vectors over time gives

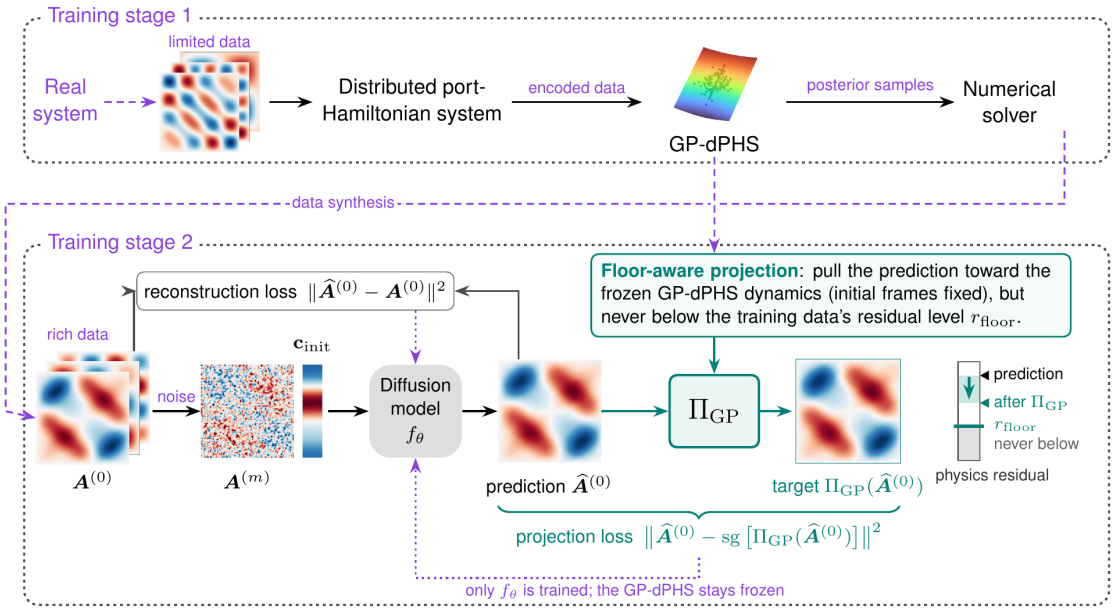

Figure 2: Two-stage training of PHDME. **Stage 1:** limited observations of the real system are encoded through the distributed port-Hamiltonian structure to fit the a Gaussian process over the system's Hamiltonian. Posterior samples of its energy gradient each define one plausible dynamics, which a numerical solver integrates to synthesize a rich training set of trajectories $\boldsymbol{A}^{(0)}$. **Stage 2:** each trajectory $\boldsymbol{A}^{(0)}$ is an image of the state over space (vertical) and time (horizontal). Gaussian noise is added to obtain $\boldsymbol{A}^{(m)}$ at diffusion step m, and the diffusion model f_θ, with trainable parameters θ, predicts the clean trajectory $\widehat{\boldsymbol{A}}^{(0)}$ from $\boldsymbol{A}^{(m)}$ together with the initial condition $\mathbf{c}_{\text{init}}$, the two given initial frames. The reconstruction loss compares $\widehat{\boldsymbol{A}}^{(0)}$ with $\boldsymbol{A}^{(0)}$. The floor-aware projection Π_{GP} (Algorithm 1) moves $\widehat{\boldsymbol{A}}^{(0)}$ toward trajectories that follow the frozen GP-dPHS dynamics. It reduces the GP uncertainty-weighted mismatch between the trajectory's time derivative given by the diffusion model and the derivative predicted by the GP-dPHS (Eq. 2). The projected trajectory $\Pi_{\text{GP}}(\widehat{\boldsymbol{A}}^{(0)})$ is used as a fixed target in the projection loss $\|\widehat{\boldsymbol{A}}^{(0)} - \text{sg}[\Pi_{\text{GP}}(\widehat{\boldsymbol{A}}^{(0)})]\|^2$, where $\|\cdot\|^2$ sums squared differences over all grid points and the stop-gradient operator $\text{sg}[\cdot]$ keeps the target's values but blocks gradients through its construction. Gradients of both losses update only θ; the GP-dPHS stays fixed after stage 1. In the full objective (Eq. 5), both losses also carry diffusion-step weights and a fixed physics weight, which the figure omits. For more details, see the paragraph.

$X = \big[\tilde{\boldsymbol{x}}(t_0), \ldots, \tilde{\boldsymbol{x}}(t_{N_t-1})\big], \dot{X} = \big[\partial_t\tilde{\boldsymbol{x}}(t_0), \ldots, \partial_t\tilde{\boldsymbol{x}}(t_{N_t-1})\big]$, and the training set $\mathcal{E} = [\,X, \dot{X}\,]$, aligned with the input sequence $\{\boldsymbol{u}(t_i)\}_{i=1}^{N_t}$. This construction provides derivative information from $x(t,z)$ while expanding the spatial coverage for subsequent GP-dPHS training. Based on this dataset, we learn a distributed port-Hamiltonian representation in which the dynamics satisfy

$$\partial_t \boldsymbol{x}(t,z) = (J - R)\,\delta_{\boldsymbol{x}}\hat{\mathcal{H}}(t,z) + G_d\,\boldsymbol{u}(t,z),$$

with interconnection matrix J, dissipative term R, and Hamiltonian $\hat{\mathcal{H}}$. The unknown Hamiltonian is modeled by a Gaussian process, and any unknown coefficients of J, R, and G_d are treated as hyperparameters Θ. Using the invariance of Gaussian processes under linear operators, we place a GP prior on the energy derivatives and obtain a GP over the time derivative of the state, $\partial_t\boldsymbol{x} \sim \mathcal{GP}\big(\hat{G}_\Theta\,\boldsymbol{u},\ k_{dphs}(\boldsymbol{x},\boldsymbol{x}')\big)$. We train the model on $\mathcal{E}$ by maximizing the marginal likelihood with respect to Θ and the kernel hyperparameters (φ_l, σ_f). The resulting posterior induces a stochastic Hamiltonian $\hat{\mathcal{H}}$ and yields the learned dPHS

$$\partial_t \boldsymbol{x}(t,z) = (\hat{J}_\Theta - \hat{R}_\Theta)\,\delta_{\boldsymbol{x}}\hat{\mathcal{H}}(t,z) + \hat{G}_\Theta\,\boldsymbol{u}(t,z),$$

which serves as a probabilistic physics prior for subsequent data generation and diffusion training. Because numerically solving the GP-dPHS is computationally demanding, we train a physics-informed diffusion model rather than using the GP-dPHS directly for prediction.

(stage 1, data augmentation) Dataset generation using GP samples. Since the trained GP-dPHS prior needs to address the problem of data scarcity, we leverage it to perform the data augmentation. The fitted GP-dPHS provides a posterior over energy representations. Rather than generating all trajectories from its posterior mean, we draw Hamiltonian gradients from this posterior and integrate the resulting dPHS under different initial conditions. Each function draw defines one plausible dynamical system and is held fixed throughout its rollout. The resulting training and validation trajectories reflect uncertainty in the learned physics as well as variation in initial conditions.

(stage 2) Training the diffusion model with learned physics. Only the enhanced dataset alone is not enough; we also want to inform the diffusion model of the physics knowledge behind it and the corresponding confidence level. We now use the trajectories generated in stage 1 as the training data for the diffusion model, while keeping the fitted GP-dPHS fixed. Each trajectory is represented as an image and paired with its initial observations. During training, we add noise to the trajectory and ask the denoiser to recover the clean field. The generated trajectory supplies the reconstruction target; the learned GP-dPHS provides additional physics supervision.

This supervision proceeds in three steps. First, we evaluate how well the denoised prediction satisfies the learned dynamics, accounting for the GP uncertainty. Second, we iteratively correct the prediction while keeping its initial and boundary values fixed, stopping when the residual reaches the level attained by observed trajectories or the iteration budget is exhausted. Third, we use the corrected/projected prediction as a fixed target for the denoiser, without passing gradients through the correction procedure. After an initial period of reconstruction training alone, this physics term is applied at selected noise levels alongside the reconstruction loss. The following paragraphs define the trajectory representation, learned residual, projection, and combined training objective.

(stage 2, first step) Measuring diffusion output with the learned dynamics. Since we trained the GP-dPHS on limited data, the physics prior can be uncertain of derivatives in some regions. Before correcting $\widehat{\boldsymbol{A}}^{(0)}$, we need to measure where it disagrees with the GP-dPHS prior and how much confidence to place in that discrepancy. Let $\boldsymbol{B}$ denote a candidate corrected trajectory on the grid of the diffusion model output. The linear finite-difference operator T_{op} computes its time derivatives. Independently, we evaluate the GP posterior mean of the energy gradients at $\boldsymbol{B}$ and apply the learned dPHS operators to obtain $\boldsymbol{g}_{\mathrm{GP}}(\boldsymbol{B})$, this $\boldsymbol{g}_{\mathrm{GP}}(\boldsymbol{B})$ is also the time derivatives predicted by the learned physics prior, including any known input. Their difference is the learned residual $\boldsymbol{r}_{\mathrm{GP}}(\boldsymbol{B})$:

$$\boldsymbol{r}_{\mathrm{GP}}(\boldsymbol{B}) = T_{\mathrm{op}}\boldsymbol{B} - \boldsymbol{g}_{\mathrm{GP}}(\boldsymbol{B}). \tag{2}$$

Each residual entry therefore measures a mismatch between the candidate's temporal evolution and the evolution prescribed by the learned dynamics.

The GP also provides uncertainty about these rates of change. We propagate its predictive variance through the spatial-difference operators and form a diagonal matrix $W(\boldsymbol{B})$ whose entries are the reciprocals of the resulting regularized variances. A positive variance floor prevents excessively large weights. The quantity $\|W(\boldsymbol{B})^{1/2}\boldsymbol{r}_{\mathrm{GP}}(\boldsymbol{B})\|_2^2$ thus gives more weight to discrepancies where the GP is confident and less where it is uncertain. This is how GP uncertainty of the learned GP-dPHS enters the correction. For every candidate sample $\boldsymbol{B}$, we always have the corresponding residual $\boldsymbol{r}_{\mathrm{GP}}(\boldsymbol{B})$ between the model and physics prior and its uncertainty weighting term $W(\boldsymbol{B})$.

(stage 2, second step) Floor-aware physics projection. As you may be aware, the core idea of our physics-informed learning is based on the state derivative. However, under very limited observation, the GP-dPHS may not provide smooth derivatives in all regions as we wanted in the first step. Hence, we need to project the target in order to compute a regularized correction in state space, then use the projected trajectory as a detached training target.

Algorithm 1 Floor-aware learned-physics projection

Require: Prediction $\widehat{\boldsymbol{A}}^{(0)}$, condition $\mathbf{c}_{\text{init}}$, learned GP operator, data residual floor r_{floor}

1: Set $\boldsymbol{B}^{(0)} \leftarrow \widehat{\boldsymbol{A}}^{(0)}$ and pin the constrained entries
2: **for** $\nu = 0, \ldots, N_{\text{out}} - 1$ **do**
3: Evaluate $\boldsymbol{g}_\nu = \boldsymbol{g}_{\text{GP}}(\boldsymbol{B}^{(\nu)})$ and $W_\nu = W(\boldsymbol{B}^{(\nu)})$; hold both fixed for this solve at projection iteration ν
4: Solve Equation 3 on free entries by damped conjugate gradient
5: Re-pin constraints and stop if $\|W_\nu^{1/2} \boldsymbol{r}_{\text{GP}}(\boldsymbol{B}^{(\nu+1)})\|_2 \le r_{\text{floor}}$
6: **end for**
7: **return** $\Pi_{\text{GP}}(\widehat{\boldsymbol{A}}^{(0)}) \leftarrow \boldsymbol{B}^{(\nu+1)}$

We now use this residual $\boldsymbol{r}_{\text{GP}}(\boldsymbol{B})$ weighted by $W(\boldsymbol{B})$ to find a nearby trajectory that is smoother and better align with the learned physics. Initialize $\boldsymbol{B}^{(0)}$ with $\widehat{\boldsymbol{A}}^{(0)}$ and impose the prescribed initial and boundary values. Here, $\boldsymbol{B}^{(\nu)}$ is the corrected trajectory after ν projection iterations. At iteration ν, we evaluate $\boldsymbol{g}_\nu = \boldsymbol{g}_{\text{GP}}(\boldsymbol{B}^{(\nu)})$ and $W_\nu = W(\boldsymbol{B}^{(\nu)})$, then hold these two quantities constant during the next solve. It gives a quadratic problem in $\boldsymbol{B}$, which can be solved as a linear system. After the solve, the GP quantities are evaluated again at the updated trajectory. The fitted GP parameters remain unchanged throughout.

With $\mathcal{C}(\mathbf{c}_{\text{init}})$ denoting trajectories that satisfy the prescribed initial and boundary values, the update is

$$\boldsymbol{B}^{(\nu+1)} = \underset{\boldsymbol{B} \in \mathcal{C}(\mathbf{c}_{\text{init}})}{\arg\min} \; \frac{1}{2}\|\boldsymbol{B} - \widehat{\boldsymbol{A}}^{(0)}\|_2^2 + \frac{\lambda_\nu}{2}\big\|W_\nu^{1/2}\big(T_{\text{op}}\boldsymbol{B} - \boldsymbol{g}_\nu\big)\big\|_2^2, \tag{3}$$

The first term keeps the corrected trajectory close to the original denoiser prediction; the second reduces disagreement with the learned dynamics, weighted by GP confidence W_ν. The positive parameter λ_ν controls this tradeoff within the projection, separately from the physics weight in diffusion training. Only entries not fixed by the initial or boundary conditions are optimized. We solve the resulting normal equations by conjugate gradient with Levenberg–Marquardt damping for numerical stability.

Because the GP is an approximation, even observed trajectories have a nonzero learned residual. We use their residual level, measured with the same weighting convention, as the stopping threshold r_{floor}. Iterations stop when the corrected trajectory reaches this threshold or the iteration budget is exhausted. This avoids demanding exact agreement with an imperfect operator. The final trajectory is $\Pi_{\text{GP}}(\widehat{\boldsymbol{A}}^{(0)})$, summarized in Algorithm 1.

(stage 2, third step) Projection-target training. The corrected trajectory supplies a detached training target. We penalize its squared distance from the original prediction:

$$\ell_{\text{proj}}(\widehat{\boldsymbol{A}}^{(0)}) = \left\| \widehat{\boldsymbol{A}}^{(0)} - \text{sg}\left[\Pi_{\text{GP}}(\widehat{\boldsymbol{A}}^{(0)})\right] \right\|_2^2. \tag{4}$$

Here $\|\cdot\|^2$ sums squared differences over all grid points and the stop-gradient operator $\text{sg}[\cdot]$ keeps the target's values but blocks gradients through its construction.

The complete objective is

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{DDPM}}(\theta) + c_{\text{res}}\,\mathcal{L}_{\text{phys}}(\theta), \qquad c_{\text{res}} > 0, \tag{5}$$

where c_{res} is a fixed training weight, and $\mathcal{L}_{\text{phys}} = w_m^{\text{phys}} \ell_{\text{proj}}$, which is uncertainty weighted loss. Training begins with reconstruction alone, followed by the gated projection term. Experimental settings are specified in Section 4.

Optional projection after sampling. PHDME uses the standard reverse-diffusion transitions and may apply the same projection procedure once to the completed trajectory. This optional correction uses only the learned GP-dPHS and preserves the prescribed initial and boundary values. We denote samples corrected in this way by +proj.

4 Experiments

Benchmarks and data. We evaluate a synthetic string, a real-world spring, and a nonlinear membrane in two spatial dimensions. In each benchmark, the GP-dPHS is fitted to 20 ground-truth trajectories: the string trajectories are downsampled by a factor of 50 (1,639 observation rows), the spring GP uses the same budget of 1,639 rows per equation, and the membrane GP uses 200 observation rows per energy-gradient component. Trajectories integrated from posterior draws of this GP-dPHS form the training and validation data of every learned model, so no model sees ground-truth trajectories beyond these 20. Test references are held-out ground-truth trajectories (string $N = 1000$, spring $N = 250$, membrane $N = 500$): every model is trained on learned physics and tested on the true system, and analytic equations are used only for evaluation.

Metrics. Each metric is computed per test trajectory and averaged over the test set. *Accuracy:* relL2 $= \|\widehat{\boldsymbol{x}} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$. MSE averages the squared error over the same entries. *Physics:* oracle-phys is the mean squared residual of the analytic governing equations (string: $\partial_t p = \partial_z q$, $\partial_t q = \partial_z p$; membrane: Appendix A.8) on the full predicted field, with derivatives taken by finite differences on the evaluation grid. Because of this discretization, even the reference trajectories have a nonzero residual (string 0.056, membrane 0.045); this ground-truth level is the practical floor, and a value below it indicates agreement with the discretized equations rather than better physics. Oracle-phys is not reported for the spring, whose governing equations are not available in closed form. E-cons is the temporal standard deviation of the total energy divided by its mean (string and spring: $\frac{1}{2}\sum_z (p^2 + q^2)$); it should be small for the conservative string and membrane and should match the reference value 0.527 for the damped spring. The suffix +proj marks PHDME samples corrected by the projection after sampling.

Baselines. All learned baselines are trained on the same GP-dPHS augmented trajectories as PHDME. DDPM has PHDME's network, data, sampler, and checkpoint selection but no projection target, so it isolates the learned-physics loss. ConvNODE integrates a learned convolutional vector field with RK4. FieldPHNN learns a neural Hamiltonian and converts its energy gradients into state rates with the same discrete interconnection operator and fixed ends as the GP-dPHS; it shares the GP-dPHS structure but not its learned energy. GP rollout integrates the posterior mean of the fitted GP-dPHS from the first reference frame with fixed ends; it is the teacher that generated PHDME's training data.

Training and projection settings. All benchmarks use the same stage-2 recipe: $c_{\text{res}} = 0.3$, $w_{\max}^{\text{phys}} = 50$, training uses Adam (learning rate 10^{-3}) with gradient-norm clipping at 1. The training projection uses $N_{\text{out}} = 2$, $N_{\text{CG}} = 15$, $\eta = 10^{-6}$, and $\lambda = 10^{-3}$ (string, membrane) or 3×10^{-6} (spring). For +proj, we allow five outer iterations with at most 60 conjugate-gradient iterations each; λ is chosen from $\{0.3, 1, 3, 10\} \times 10^{-3}$ (string) or $\{0.3, 1, 3, 10, 30\} \times 10^{-6}$ (spring) by the discrepancy rule on the first 200 or 100 samples, and is fixed to 10^{-3} on the membrane. Diffusion checkpoints are selected on GP-world validation trajectories.

4.1 Quantitative Results

Comparison with the GP-dPHS prior and other baselines. GP rollout integrates the same posterior-mean dynamics that define PHDME's projection, so it shows what amortizing the GP-dPHS costs or gains. On the string, GP rollout (0.085) and FieldPHNN (0.097) are accurate. FieldPHNN shares the GP-dPHS's port-Hamiltonian structure and is trained on the trajectories that the GP-dPHS generated, so it re-learns the teacher's law rather than new physics: its per-case errors track those of the GP-dPHS rollout (correlation 0.89), and it stays slightly less accurate than its teacher. On the linear, conservative string, integrating this accurate local law from the first frame is hard to beat, but the advantage does not carry over: FieldPHNN's structure has no dissipation, so on the **real world damping spring**, it keeps too much energy (E-cons 0.386 versus 0.527; relL2 0.5205), and on the membrane its rollouts violate the equations strongly (oracle-phys 2.784; relL2 0.498). On the spring, PHDME and GP rollout differ by only 0.0006: models trained on the GP world inherit the error of a GP-dPHS fitted to 20 trajectories. On the **membrane**, GP rollout has the lowest relL2 (0.266) but four times PHDME's analytic residual (0.423 versus 0.106), because its operator is less accurate on reference states than on GP-world ones (residual 0.148). PHDME's gains over its teacher are thus a 20× speedup (Table 5), samples instead of one rollout, and better physical consistency on the

Table 1: Test results on held-out ground-truth trajectories: string ($N = 1000$), real-world spring ($N = 250$), and membrane ($N = 500$). All learned models are trained on trajectories augmented by a GP-dPHS learned from limited trajectories; Lower is better except for spring E-cons, which should match the reference value 0.527 and is not ranked. Bold marks the best value per system and column; † entries are not ranked. Ground-truth oracle-phys: string 0.056, membrane 0.045.

System	Model	relL2	oracle-phys	MSE	E-cons
String	PHDME +proj (Ours)	0.397	0.121	1.47×10^{-3}	0.084
	PHDME (Ours)	0.413	0.183	1.58×10^{-3}	0.093
	DDPM	0.637	0.574	3.06×10^{-3}	0.150
	ConvNODE	0.537	0.121	2.32×10^{-3}	**0.025**
	FieldPHNN	0.097	0.014†	$\mathbf{4.35\times10^{-4}}$	0.087
	GP rollout	**0.085**	**0.091**	7.40×10^{-4}	0.103
Membrane	PHDME +proj (Ours)	0.298	**0.106**	7.42×10^{-3}	0.018
	PHDME (Ours)	0.297	0.116	7.36×10^{-3}	0.015
	DDPM	0.380	0.150	1.20×10^{-2}	0.016
	ConvNODE	0.337	0.114	9.29×10^{-3}	0.018
	FieldPHNN	0.498	2.784	2.24×10^{-2}	0.028
	GP rollout	**0.266**	0.423	$\mathbf{6.70\times10^{-3}}$	**0.014**
Real World Spring Data	PHDME +proj (Ours)	**0.1137**	—	9.21×10^{-4}	0.529
	PHDME (Ours)	**0.1137**	—	$\mathbf{9.20\times10^{-4}}$	0.529
	DDPM	0.1144	—	9.33×10^{-4}	0.529
	ConvNODE	0.1208	—	9.95×10^{-4}	0.521
	FieldPHNN	0.5205	—	1.73×10^{-2}	0.386
	GP rollout	0.1143	—	9.33×10^{-4}	0.529

†Below the ground-truth level (0.056): FieldPHNN integrates the same finite-difference operator that oracle-phys evaluates, so its rollouts satisfy the discretized equations more closely than the reference trajectories do.

Table 2: Evaluation of the role of data enhancement under extreme data scarcity. As table 1 shows that the learned physics prior makes the generative model more accurate and physically consistent using enhanced dataset, we also want to make fair comparisons against the baselines under the original limited data observation settings without our framework. In general, the data budget should be the same as the one to train the GP-dPHS as 1,639 sparse points. However, A diffusion model needs fixed number of input and output, so the DDPM is trained on one whole field trajectory (4,096 values per field, which is a lot more than other baselines but the smallest set it can use. ConvNODE needs whole frames and is trained on 12 pairs of consecutive frames from the same trajectories (1,536 values per field as comparable budget. E-cons computed on the normalized fields, it is 0.104 for the ground truth. Ground-truth oracle-phys: 0.056.

Training data	Model	relL2	oracle-phys	MSE	E-cons
Pipeline	PHDME +proj (Ours)	0.397	0.121	1.47×10^{-3}	0.084
Augmented Data	PHDME (Ours)	0.413	0.183	1.58×10^{-3}	0.093
1,639 points	GP rollout	**0.085**	**0.091**	$\mathbf{7.40\times10^{-4}}$	0.103
1,639 points	FieldPHNN	0.535	4.61×10^{2}	1.24×10^{-1}	0.195
12 frame pairs	ConvNODE	0.422	1.649	3.62×10^{-3}	0.152
1 trajectory	DDPM	2.297	2.002	4.36×10^{-2}	0.398

membrane, not higher accuracy. ConvNODE has the best string E-cons (0.025) despite relL2 0.537: a nearly constant energy does not imply a correct wave. Finally, +proj mainly improves physics, lowering oracle-phys on the string ($0.183 \to 0.121$) and membrane ($0.116 \to 0.106$) while changing relL2 by at most 0.016; it has no measurable effect on the spring.

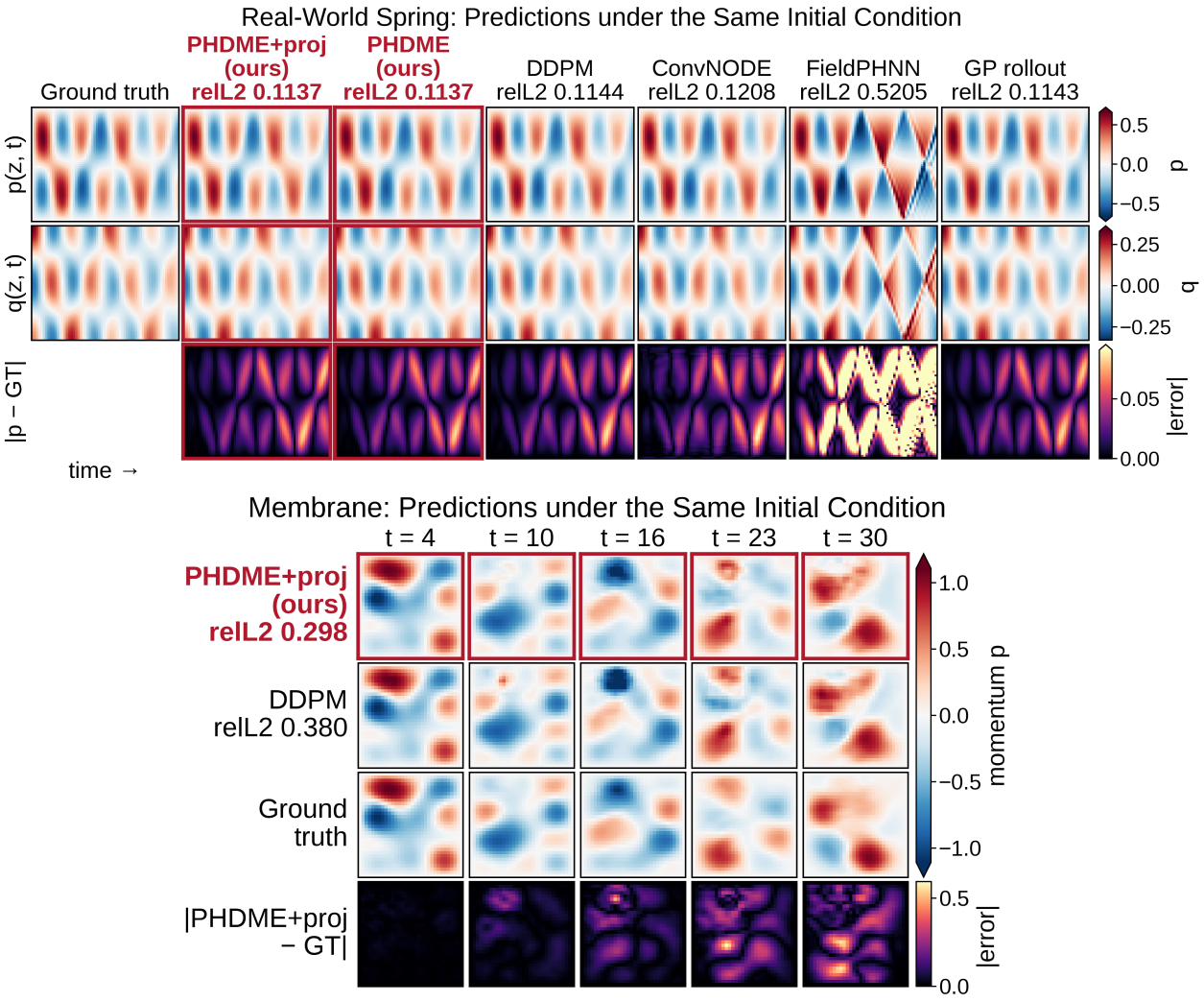

Figure 3: Representative test cases. The number with each method is its test-set relL2 from Table 1, not the error of the case shown. Top: a real-world spring test case p and q over position z (vertical) and time (horizontal), and the absolute error in p. Bottom: a membrane test case the momentum at five of the 32 frames for PHDME +proj, DDPM, and the ground truth, and the absolute PHDME +proj error.

4.2 Qualitative Results

Figure 3 shows one real-world spring and one membrane test case; the spring observation pipeline and the reverse-diffusion visualization are in Appendix A.1.

In the spring example, PHDME with and without +proj, DDPM, and GP rollout are nearly indistinguishable: their predictions differ from one another by less than 1% of the signal, while each is about 11% from the ground truth, and their errors grow with time. FieldPHNN develops sharp spurious fronts in the second half of the run, with about five times the error of the other methods. In the membrane example, PHDME +proj places the main lobes correctly; its error grows with time and lies where the field changes fastest, and DDPM's error is larger in every frame shown. These examples illustrate, but do not replace, the averages in Table 1.

4.3 Ablation Study

We use the string benchmark because its analytic physics gives the cleanest evaluation signal. Each variant changes one component of the final $c_{\text{res}} = 0.3$ recipe; A2–A8 and B1 are trained for 100k iterations, and A0 and A1 reuse the runs of Table 1. Every variant is selected on 200 GP-world validation trajectories within 70k–100k iterations and evaluated on the $N = 1000$ string test trajectories with seed 0, so A0 and A1 differ slightly from Table 1, which selects over the full runs (relL2 0.428 versus 0.413, 0.670 versus 0.637). The reference includes floor-aware stopping, two outer projection iterations with at most 15 conjugate-gradient iterations each, GP-variance whitening, the SNR gate, and a 10,000-step warm-up.

Table 3: Component leave-one-out on the string test set ($N = 1000$). Each row changes one component of full PHDME; all metrics are lower-is-better. Oracle-phys is measured after +proj (A1: raw samples, since DDPM receives no projection, hence the dash); E-cons is measured on raw samples.

Variant	One-knob change	relL2 raw	relL2 +proj	oracle-phys	E-cons
A0	Full PHDME (reference)	**0.428**	**0.410**	**0.115**	**0.076**
A1	Remove learned physics (DDPM)	0.670	—	0.702	0.200
A2	Replace projection with raw residual loss	1.115	1.115	9.510	0.918
A3	Remove floor-aware stopping	0.934	0.900	0.375	0.189
A5	Remove GP-variance whitening	0.662	0.623	0.270	0.249
A6	Remove SNR gate	3.072	3.072	57.94	0.359
A7	Remove data-only warmup	0.618	0.606	0.359	0.152
A8	Replace fixed weight with GradNorm	0.569	0.541	0.312	0.254

Every removal degrades the full model in every column. The SNR gate is the most sensitive component: applying physics supervision at all noise levels raises relL2 from 0.410 to 3.072. Replacing the projection target with the raw learned residual is also destructive (relL2 1.115, worse than DDPM's 0.670). These two ablations identify the projection and its noise gate as the load-bearing elements; removing floor-aware stopping, variance whitening, warm-up, or the fixed weight gives smaller but consistent losses. Qualitative failures are shown in Appendix A.2.

5 Related Work

Equation-informed diffusion methods can impose known PDE residuals during training or use guidance and projection during inference (Shu et al., 2023; Bastek et al., 2024; Jacobsen et al., 2025). PHDME addresses a different setting in which the constitutive physics is learned from sparse trajectories. Its projection is also used differently: the projected sample is a stop-gradient training target rather than solely an inference-time correction. The optional post-sampling projection is reported separately. Continuous-time data-driven models such as NeuralODE (Chen et al., 2018) and ConvNODE learn vector fields without a learned Hamiltonian constraint and may accumulate errors on held-out test trajectories. PHDME instead amortizes a structured, uncertain operator into a full-trajectory generative model.

6 Conclusion

PHDME uses the learned GP-dPHS as a local physical constraint rather than a long horizon integrator. A floor-aware Gauss-Newton projection transforms its ill-conditioned residual into a stable stop-gradient target for diffusion training. Across the benchmarks, PHDME improves the balance between predictive accuracy and physical consistency. Ablations confirm that both the port-Hamiltonian structure and the projection mechanism are essential. ConvNODE remains competitive, direct GP rollout achieves reasonable membrane accuracy but poor analytic consistency, and DDPM attains the lowest prediction error when the augmented training distribution is sufficiently saturated.

References

Jan-Hendrik Bastek, WaiChing Sun, and Dennis M Kochmann. Physics-informed diffusion models. *arXiv preprint arXiv:2403.14404*, 2024.

Thomas Beckers and Leonardo Colombo. Data-driven boundary control of distributed port-hamiltonian systems. *arXiv preprint arXiv:2604.04266*, 2026.

Thomas Beckers, Jacob Seidman, Paris Perdikaris, and George J Pappas. Gaussian process port-hamiltonian systems: Bayesian learning with physics prior. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1447–1453. IEEE, 2022.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Hyungjin Chung, Dohoon Ryu, Michael T McCann, Marc L Klasky, and Jong Chul Ye. Solving 3d inverse problems using pre-trained 2d diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22542–22551, 2023.

Sławomir Cieślik. Mathematical modeling of the dynamics of linear electrical systems with parallel calculations. *Energies*, 14(10):2930, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Zhifang Guo, Jianguo Mao, Rui Tao, Long Yan, Kazushige Ouchi, Hong Liu, and Xiangdong Wang. Audio generation with multiple conditional diffusion model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18153–18161, 2024.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7441–7451, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. Cocogen: Physically consistent and conditioned score-based generative models for forward and inverse problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025.

Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B Schön. Linearly constrained Gaussian processes. *Advances in Neural Information Processing Systems*, 30, 2017.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Peilun Li, Kaiyuan Tan, and Thomas Beckers. Napi-mpc: Neural accelerated physics-informed mpc for nonlinear pde systems. In *7th Annual Learning for Dynamics\& Control Conference*, pp. 1230–1242. PMLR, 2025.

Jingyun Liang, Yuchen Fan, Kai Zhang, Radu Timofte, Luc Van Gool, and Rakesh Ranjan. Movideo: Motion-aware video generation with diffusion model. In *European Conference on Computer Vision*, pp. 56–74. Springer, 2024.

Alessandro Macchelli, Arjan J Van Der Schaft, and Claudio Melchiorri. Port Hamiltonian formulation of infinite dimensional systems i. modeling. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 4, pp. 3762–3767. IEEE, 2004.

Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *arXiv preprint arXiv:2306.10619*, 2023.

Marcell Missura and Maren Bennewitz. Predictive collision avoidance for the dynamic window approach. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8620–8626. IEEE, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Ralf Römer, Alexander von Rohr, and Angela Schoellig. Diffusion predictive control with constraints. In *7th Annual Learning for Dynamics\& Control Conference*, pp. 791–803. PMLR, 2025.

Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in neural information processing systems*, 36:45259–45287, 2023.

Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Kaiyuan Tan, Jun Wang, and Yiannis Kantaros. Targeted adversarial attacks against neural network trajectory predictors. In *Learning for Dynamics and Control Conference*, pp. 431–444. PMLR, 2023.

Kaiyuan Tan, Peilun Li, and Thomas Beckers. Physics-constrained learning of pde systems with uncertainty quantified port-hamiltonian models. In *6th Annual Learning for Dynamics & Control Conference*, pp. 1753–1764. PMLR, 2024.

Arjan van der Schaft and Dimitri Jeltsema. Port-hamiltonian systems theory: An introductory overview. *Foundations and Trends® in Systems and Control*, 1(2–3):173–378, 2014.

Arjan J. van der Schaft and Bernhard M. Maschke. Hamiltonian formulation of distributed-parameter systems with boundary energy flow. *Journal of Geometry and Physics*, 42(1–2):166–194, 2002. doi: 10.1016/S0393-0440(01)00083-3.

Bin Xia, Yulun Zhang, Shiyin Wang, Yitong Wang, Xinglong Wu, Yapeng Tian, Wenming Yang, and Luc Van Gool. Diffir: Efficient diffusion model for image restoration. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 13095–13105, 2023.

Xingqian Xu, Zhangyang Wang, Gong Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7754–7765, 2023.

Tongjia Zheng and Hai Lin. PDE-based dynamic control and estimation of soft robotic arms. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2702–2707. IEEE, 2022.

Navid Zobeiry and Keith D Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021.

Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, Nand Vinchhi, Kaushik Balakrishnan, Devesh Upadhyay, and Chris Rackauckas. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations, 2021. URL `https://arxiv.org/abs/2107.09443`.

A Appendix

A.1 Additional Experimental Visualizations

Figure 4 shows the reverse-diffusion trajectory for the string benchmark. Figure 5 documents the observation pipeline used to construct the repaired real-world spring trajectories.

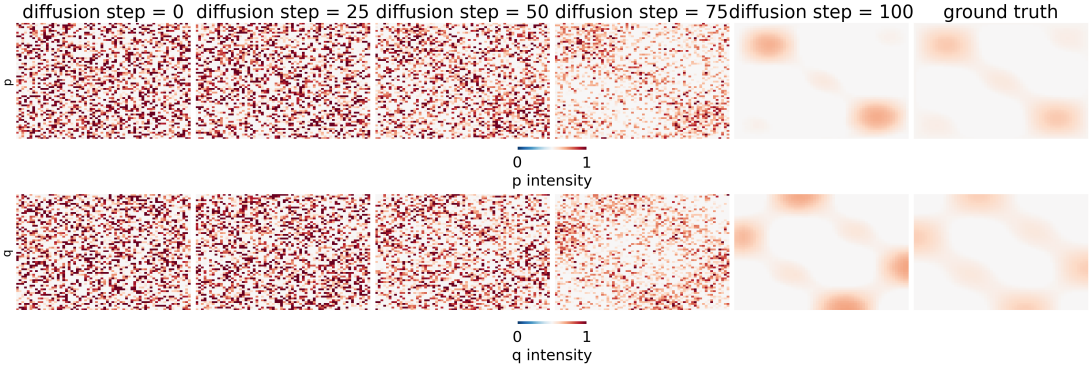

Figure 4: Intermediate reverse-diffusion states for a representative string trajectory. Coherent oscillations emerge progressively from noise while retaining the conditioning frames.

A.2 Extended Ablation Results

Figure 6 first places all leave-one-out predictions under a shared initial condition. It visually confirms the table-level result: removing the SNR gate or replacing projection with a raw residual destroys the wave structure, while the other removals introduce smaller phase and amplitude errors.

The prior-structure experiment uses the same 1,639 observations and RBF ExactGP family for every learned prior. GP-dPHS learns spatially integrated Hamiltonian gradients, whereas the plain GP regresses raw time derivatives pointwise. The latter target is ill-posed from local state alone: its held-out unexplained variance exceeds one, and its true-data operator floor is 3.7, compared with 0.033 for GP-dPHS.

Table 4: Prior-structure ladder on the reference string test set. Downstream relL2 is raw/+proj where a valid learned-prior projection is available.

Rung	Physics prior	relL2 raw/+proj	$1 - R^2$, p/q	Data floor
0	No learned prior (DDPM)	0.670/—	—	—
1	Plain GP on raw derivatives	0.617/—	1.47/1.30	3.7
2	GP-dPHS with uncertainty	**0.428/0.410**	0.0002/0.0000	0.033

The correct structure with the wrong raw-residual loss fails, and the projection recipe with an unstructured prior provides little benefit; only their combination attains the full result. Figures 7 and 8 provide enlarged examples of the individual failures.

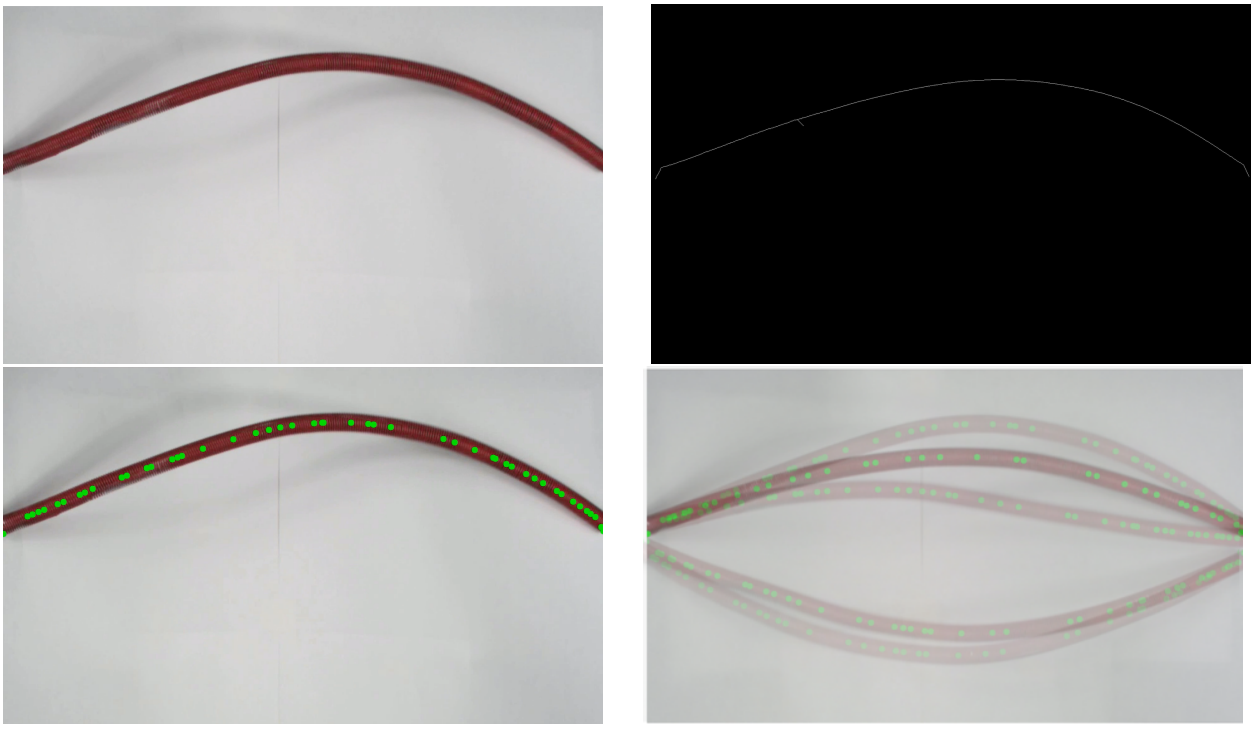

Figure 5: Construction of the real-world spring benchmark. Top-left: raw camera frame. Top-right: skeletonized centerline. Bottom-left: spatially ordered tracked points. Bottom-right: tracked centerlines at several times. The trajectories are filtered and interpolated before conversion to the physically consistent representation described in Appendix A.5.

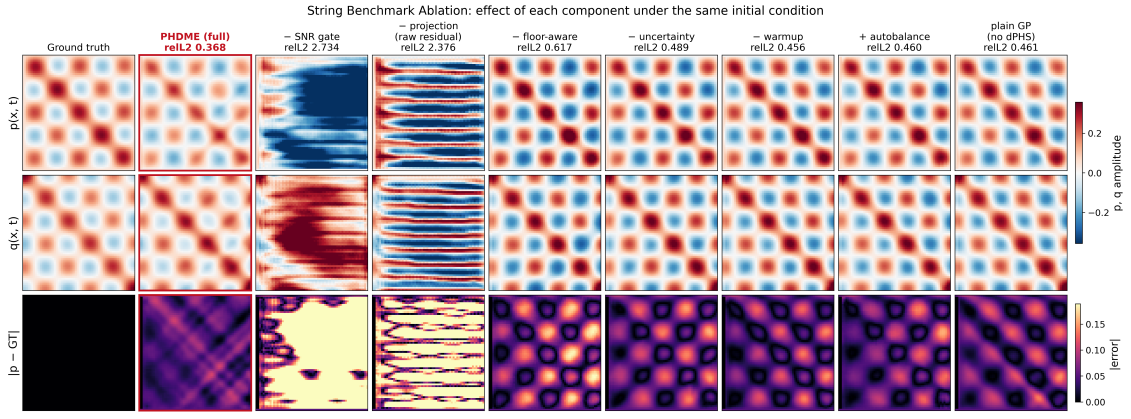

Figure 6: String leave-one-out comparison under a shared initial condition. Removing the SNR gate or replacing projection with a raw residual destroys the wave structure; the remaining variants preserve the broad pattern but introduce larger phase and amplitude errors.

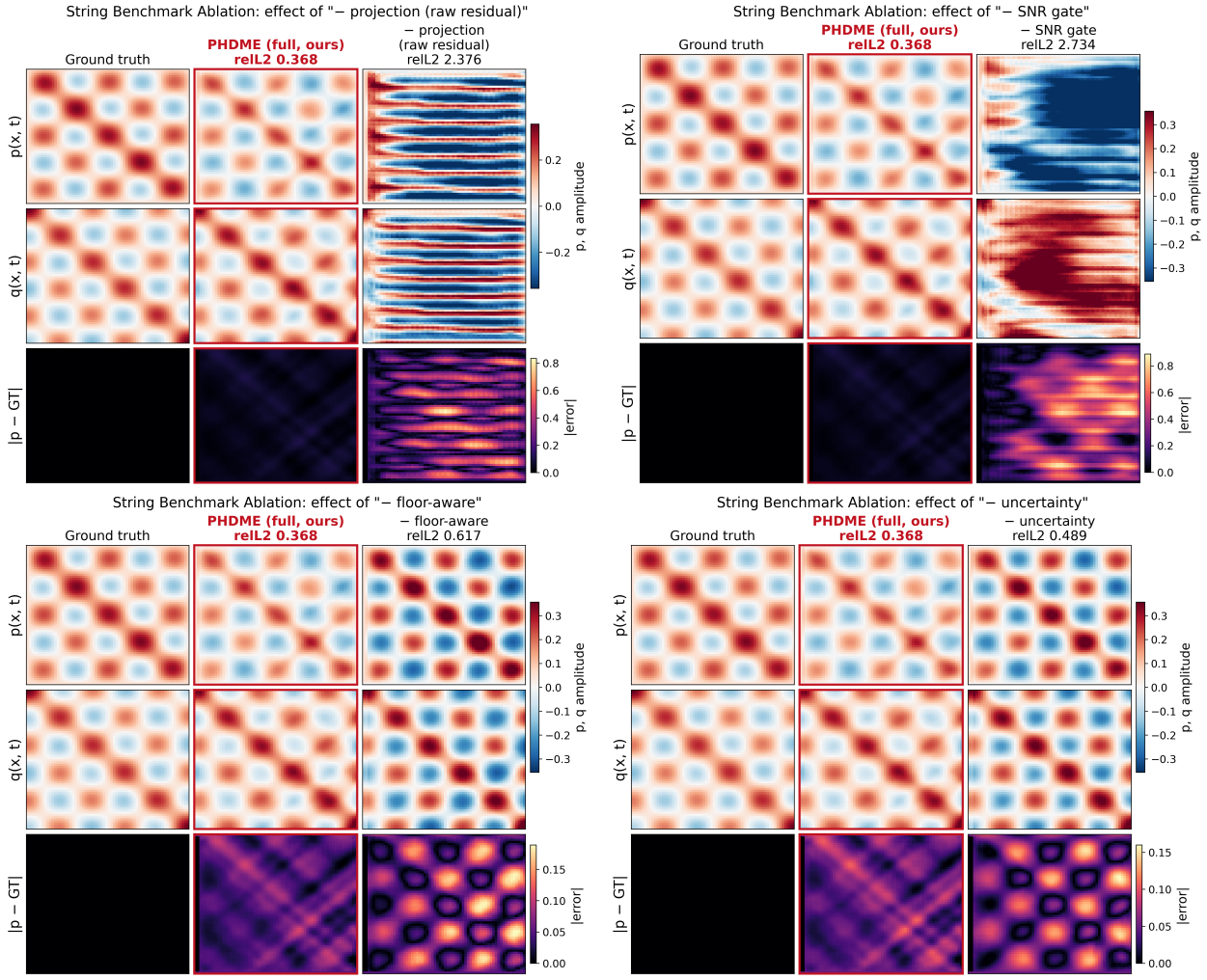

Figure 7: Detailed mechanism ablations. Top: raw residual instead of projection and removal of the SNR gate, both of which cause global structural failure. Bottom: removal of floor-aware stopping and uncertainty whitening, which produces more localized but substantial errors.

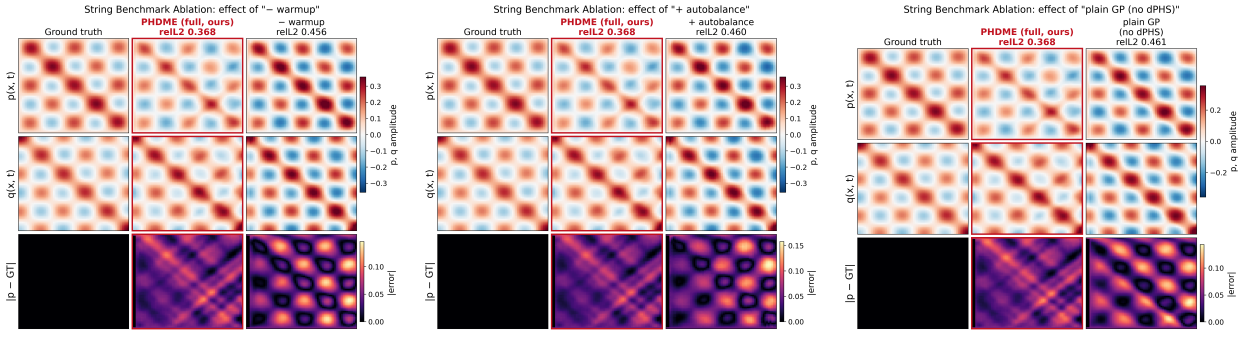

Figure 8: Additional ablations: no warmup, adaptive rather than fixed loss balancing, and a plain GP in place of GP-dPHS. All retain recognizable oscillations but reduce forecasting accuracy.

A.3 Wave PDE Data Generation

We synthesize supervision using a physically faithful simulator of a fixed-end string that solves the one-dimensional wave equation on a fine grid and then projects the solution onto the learning grid. Let $s(z,t)$ denote displacement, with spatial domain $z \in [0, L]$ and time $t \in [0, T]$. The continuous dynamics satisfy

$$\partial_{tt} s(z,t) = c^2\, \partial_{zz} s(z,t), \qquad s(0,t) = 0,\ s(L,t) = 0, \tag{6}$$

with initial conditions $s(z,0) = s_0(z)$ and $\partial_t s(z,0) = w_0(z)$. The learned state is the derivative pair

$$p(z,t) = \partial_z s(z,t), \qquad q(z,t) = \partial_t s(z,t), \tag{7}$$

and we form the state vector $\boldsymbol{x}(z,t) = [p(z,t),\, q(z,t)]^\top$.

Fine–to–coarse simulation. We integrate an equivalent first-order system on a fine grid and then downsample to the learning resolution. Let $\boldsymbol{\chi}_{\text{wave}}(t) = [s(\cdot,t);\ w(\cdot,t)]$ with $w = \partial_t s$. We discretize space on N_z^{fine} nodes with step Δz^{fine} and approximate the Laplacian using a second-order central stencil. The semi-discrete dynamics are

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} s \\ w \end{bmatrix} = \begin{bmatrix} w \\ c^2\, \mathbf{D}_{\text{fine}}^{zz}\, s \end{bmatrix}, \qquad s_1(t) = s_{N_z^{\text{fine}}}(t) = 0, \tag{8}$$

where $\mathbf{D}_{\text{fine}}^{zz}$ is the tridiagonal second–difference operator with Dirichlet boundary rows. We integrate (8) over N_t^{fine} fine time points using an adaptive ODE solver. From the fine solution we compute

$$p^{\text{fine}} = \partial_z s \approx \mathbf{D}_{\text{fine}}^{z}\, s, \qquad q^{\text{fine}} = \partial_t s \approx \mathbf{D}_{\text{fine}}^{t}\, s, \tag{9}$$

where $\mathbf{D}_{\text{fine}}^{z}$ is the centered first-difference operator in z and $\mathbf{D}_{\text{fine}}^{t}$ is a centered time stencil. Optional Gaussian smoothing with standard deviation σ_{smooth} may be applied to s before differencing. We then downsample $(p^{\text{fine}}, q^{\text{fine}})$ to the learning grid of size $N_z \times N_t$ to obtain

$$p \in \mathbb{R}^{N_z \times N_t}, \qquad q \in \mathbb{R}^{N_z \times N_t}. \tag{10}$$

Randomized, valid initial conditions. To span smooth, physically consistent excitations, we sample s_0 and w_0 as finite Fourier–sine series that respect fixed ends:

$$s_0(z) = \sum_{n=1}^{N_{\text{m}}} a_n \sin\Big(\frac{n\pi z}{L}\Big) \cos\phi_n, \qquad w_0(z) = \sum_{n=1}^{N_{\text{m}}} a_n \sin\Big(\frac{n\pi z}{L}\Big) \sin\phi_n\, \frac{n\pi c}{L}, \tag{11}$$

with amplitudes a_n in a symmetric range and phases $\phi_n \sim \mathcal{U}([0, 2\pi))$.

Four-channel tensor with boundary conditioning. Each sample is packaged into

$$\big[\texttt{p_field},\, \texttt{full_p},\, \texttt{q_field},\, \texttt{full_q}\big] \in \mathbb{R}^{4 \times N_z \times N_t},$$

where $\texttt{full_p} = p$ and $\texttt{full_q} = q$. The conditioning channels encode the first two time frames with zeros elsewhere. The first frame of $\texttt{p_field}$ is set to zero to anchor the spatial–slope channel:

$$\texttt{p_field}[:,0] = \mathbf{0}, \quad \texttt{p_field}[:,1] = p[:,1], \quad \texttt{p_field}[:,t] = \mathbf{0} \text{ for } t \geq 2, \tag{12}$$

$$\texttt{q_field}[:,0] = q[:,0], \quad \texttt{q_field}[:,1] = q[:,1], \quad \texttt{q_field}[:,t] = \mathbf{0} \text{ for } t \geq 2. \tag{13}$$

Normalization. To harmonize dynamic range, we apply channelwise min–max normalization to $[v_{\min}, v_{\max}]$ with $v_{\min} = -1$ and $v_{\max} = 1$,

$$\widetilde{V} = v_{\min} + \frac{v_{\max} - v_{\min}}{V_{\max} - V_{\min} + \epsilon}\,\big(V - V_{\min}\big), \qquad V \in \{\texttt{full_p}, \texttt{full_q}\}, \tag{14}$$

and use the same affine map for the corresponding conditioning frames.

Algorithm 2 Wave Data Generation (create_string_dataset v3.0, z–space, s–displacement)

Set $N_z^{\text{fine}} \leftarrow 4N_z$ and $N_t^{\text{fine}} \leftarrow 4N_t$
for each sample **do**
  Sample $\{a_n, \phi_n\}_{n=1}^{N_{\text{m}}}$ and construct s_0, w_0 via (11)
  Integrate (8) on the fine grid to obtain $s^{\text{fine}}(z_i, t_j)$
  Compute $p^{\text{fine}} = \partial_z s^{\text{fine}}$ and $q^{\text{fine}} = \partial_t s^{\text{fine}}$ using centered differences
  Downsample $p^{\text{fine}}, q^{\text{fine}}$ to $p, q \in \mathbb{R}^{N_z \times N_t}$
  Set targets `full_p` $\leftarrow p$ and `full_q` $\leftarrow q$
  Form conditioning `p_field`, `q_field` with the first two frames and zeros elsewhere, enforcing `p_field`$[:, 0] = \mathbf{0}$
  Apply channelwise normalization and write tensors to disk
end for

A.4 Nonlinear Two-Dimensional Membrane Data

The membrane benchmark is a two-dimensional Stokes–Dirac drum with the $(\text{div}, -\nabla)$ interconnection structure (van der Schaft & Maschke, 2002). To distinguish its vector strain-like field from the scalar string variable q, we write the membrane state as $\boldsymbol{x}_{\text{mem}} = (p_{\text{mem}}, \boldsymbol{q}_{\text{mem}})$, where $\boldsymbol{q}_{\text{mem}} = (q_x, q_y)$. Its co-energies are

$$e_p = p_{\text{mem}}, \qquad \boldsymbol{e}_q = \frac{\tanh(\|\boldsymbol{q}_{\text{mem}}\|)}{\|\boldsymbol{q}_{\text{mem}}\|} \boldsymbol{q}_{\text{mem}}, \tag{15}$$

where the ratio is defined by its continuous limit at $\|\boldsymbol{q}_{\text{mem}}\| = 0$. The evolution is

$$\partial_t p_{\text{mem}} = \text{div}\, \boldsymbol{e}_q, \qquad \partial_t q_x = \partial_x e_p, \qquad \partial_t q_y = \partial_y e_p. \tag{16}$$

The edge momentum is pinned to $p_{\text{mem}} = 0$. We integrate Equation 16 by RK4 on a $32 \times 32 \times 32$ space–time grid with `SUB` $= 8$ substeps. Gentle band-limited initial conditions have amplitude 0.15, and the generated fields remain approximately bounded by 2.25 in magnitude. The simulator is conservative up to discretization error, with ground-truth E-cons approximately 0.013.

Each volume contains six channels: three initial-conditioning fields and three complete target fields $(p_{\text{mem}}, q_x, q_y)$. The simulator is implemented by `create_membrane_dataset.py`. The low-budget augmented set, `membrane_v5_lowbudget`, is generated by posterior-sample rollouts of the separable GP described in Appendix A.8, analogous to the one-dimensional posterior-sample generator in Appendix A.7.

A.5 Real-World Spring Data Collection

Experimental setup. As an abstraction of deformable obstacles that robots may encounter in their operational environments, we consider a Home Depot extension spring (model #26455, length 41.91 cm, diameter 1.42 cm) mounted with fixed endpoints and excited into transverse oscillation. This setup mimics the dynamic behavior of flexible obstacles such as swinging cables or a soft robot arm.

High-speed acquisition and coordinate system. We record the motion of the spring using a high-speed RGB camera (Blackfly S USB3 FLIR BFS-U3-16S2C-CS) at 226 FPS. Let $i \in \{0, \ldots, N_t^{\text{raw}} - 1\}$ index video frames, and let $t_i = i\,\Delta t$, with $\Delta t = \frac{1}{226}$ s, denote the corresponding physical time. The spring endpoints are rigidly mounted, and their pixel locations are identified once at the beginning of each recording. Using the known physical length $L = 41.91$ cm, we define an arclength coordinate $z \in [0, L]$ along the spring and interpolate the extracted centerline onto a fixed grid $\{z_j\}_{j=1}^{N_z}$, resulting in discrete observations of the transverse deflection $s(t_i, z_j)$.

RGB segmentation and skeletonization. To facilitate robust segmentation, the spring is painted red and the background is chosen to provide strong color contrast. For each frame, we apply a simple RGB thresholding mask

$$[150, 0, 0] \;\le\; \text{RGB}(x, y) \;\le\; [255, 80, 80]$$

componentwise to isolate spring pixels from the background. The resulting binary mask encodes the body of the deformable obstacle. We then apply skeletonization to reduce the segmented region to a one–pixel–wide medial axis (Fig. 9), which provides a compact representation of the spring shape. Mapping this skeleton to the arclength grid $\{z_j\}$ produces a discrete, time–indexed centerline

$$\mathcal{D}_0 = \big\{(t_i, z_j, s(t_i, z_j))\big\},$$

where $s(t_i, z_j)$ denotes the transverse deflection at time t_i and arclength position z_j, measured in pixel units and later scaled to physical units using the known length L.

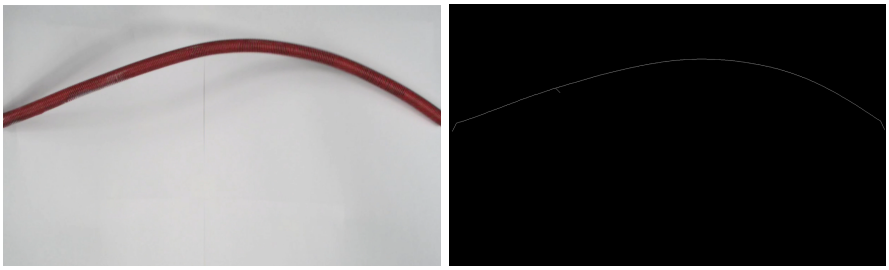

Figure 9: Visualization of the skeletonization process.

Denoising and Gaussian process representation. Real-world videos contain sensor noise, quantization artifacts, and occasional segmentation errors. To mitigate these effects while preserving the underlying dynamics, we first apply a Kalman filter along the temporal axis at each spatial location z_j to obtain a denoised trajectory $\hat{s}(t_i, z_j)$. The denoised dataset

$$\hat{\mathcal{D}}_1 = \big\{(t_i, z_j, \hat{s}(t_i, z_j))\big\}$$

is then used to train a Gaussian process (GP) model that acts as a continuous, differentiable representation of the spring dynamics. We treat $(t, z) \in [0, T] \times [0, L]$ as the input and model the state

$$s(t, z) \sim \mathcal{GP}\big(\mu_s(t, z),\, k_s\big((t, z), (t', z')\big)\big),$$

with a smooth covariance kernel k_s, such as a squared-exponential kernel in time and space, and a mean function μ_s fitted from data. Training proceeds by maximizing the GP marginal likelihood on $\hat{\mathcal{D}}_1$, resulting in the posterior distribution $p\big(s(\cdot, \cdot) \mid \hat{\mathcal{D}}_1\big)$.

Because derivatives of a GP are again GPs, the posterior directly provides a probabilistic estimate of temporal and spatial partial derivatives,

$$\frac{\partial s}{\partial t}(t, z), \qquad \frac{\partial s}{\partial z}(t, z), \tag{17}$$

which we use as physics–aware latent features. We define the representation fields

$$p(t, z) = \frac{\partial s}{\partial z}(t, z), \qquad q(t, z) = \frac{\partial s}{\partial t}(t, z), \tag{18}$$

consistent with the synthetic string system, and evaluate (p, q) on a refined spatial grid of size $N_z^{\text{ref}} \gg N_z$ to obtain a dense, noise-robust surrogate for the real-world string dynamics.

From raw `.mat` files to a physically consistent dataset. An earlier v3.5 conversion scaled p independently for each sample while applying one global factor to q. This transformation broke the coupling between the two state channels and effectively assigned a different physical system to every sample, causing failures in both GP learning and downstream PHDME training. Real-world v4 was therefore rebuilt directly from the advisor-provided `p_i.mat` and `q_i.mat` files in consistent simulator units. The validated unit map is

$$p_{\text{sim}} = p_{\text{mat}}/6.87, \qquad q_{\text{sim}} = q_{\text{mat}}/2.28.$$

With this shared conversion, the held-out explained variance exceeds 99%. This data repair, rather than an unspecified model change, explains the transition from the obsolete real-world result to the v4 benchmark reported in Table 1. The repair and scale validation correspond to `build_real_world_v4.py`, `fit_advisor_gp_scales.py`, and the advisor-GP preprocessing module in the full experiment repository.

A.6 Synthesizing a Dataset Using Mean Predictions from GP-dPHS

This section describes how the version 4 data generator constructs spatiotemporal training pairs by simulating the mean-field dynamics implied by a trained Gaussian process distributed port-Hamiltonian system. The generator replaces the analytical wave operator with the posterior means of two Gaussian processes that approximate the Hamiltonian gradients and then integrates the induced first-order evolution to produce complete fields of p and q.

Learned energy gradients and induced evolution. Let $s(z,t)$ denote displacement on $z \in [0, L]$ and $t \in [0, T]$. Consistent with Appendix A.3, the representation uses

$$p(z,t) = \partial_z s(z,t), \qquad q(z,t) = \partial_t s(z,t), \tag{19}$$

stacked channelwise into $\boldsymbol{x}(z,t) = [p(z,t),\, q(z,t)]^\top$. The GP-dPHS module comprises two Gaussian processes trained on pairs (p, q) to regress the Hamiltonian gradients $g_p = \partial H/\partial p$ and $g_q = \partial H/\partial q$. Denote their posterior means by

$$\mu_p(p,q) = \mathbb{E}[g_p(p,q) \mid \mathcal{D}_e]\,, \qquad \mu_q(p,q) = \mathbb{E}[g_q(p,q) \mid \mathcal{D}_e]\,, \tag{20}$$

where $\mathcal{D}_e$ is the training set of derivative–integral pairs. The distributed port-Hamiltonian evolution induced by these learned gradients is

$$\partial_t p(z,t) = \partial_z \mu_q\big(p(z,t), q(z,t)\big), \qquad \partial_t q(z,t) = \partial_z \mu_p\big(p(z,t), q(z,t)\big), \tag{21}$$

with fixed–end constraints applied at the spatial boundaries for the p channel. Equation (21) specializes the canonical dPHS structure to the GP mean and consequently yields a learned but physically structured flow on the representation.

Space–time discretization and solver. We discretize the spatial domain on N_z nodes with spacing Δz and the time horizon into N_t frames with step Δt. Let $\mathbf{D}_z$ be the standard centered first-difference matrix on the interior nodes with Dirichlet boundary handling. We vectorize the state at time t as $\boldsymbol{x}(t) \in \mathbb{R}^{2N_z}$ with $\boldsymbol{x}(t) = [\mathbf{p}(t); \mathbf{q}(t)]$. The right-hand side used by the integrator is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{p}(t)\\ \mathbf{q}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{D}_z\, \mu_q\big(\mathbf{p}(t), \mathbf{q}(t)\big)\\ \mathbf{D}_z\, \mu_p\big(\mathbf{p}(t), \mathbf{q}(t)\big)\end{bmatrix}, \tag{22}$$

where μ_p and μ_q are evaluated pointwise at each spatial node using the trained GP posterior means. A standard adaptive explicit ODE solver advances (22) over $[0, (N_t - 1)\Delta t]$. After each step, the boundary rows of $\mathbf{p}$ are set to zero to enforce fixed ends, preserving the intended physical interpretation of p at the string endpoints.

Initialization and conditioning convention. The generator samples smooth, band-limited initial profiles that satisfy the boundary conditions. The convention follows the version 3 setup for compatibility with the downstream diffusion model. The first frame of the p channel is set to zero and the first two frames of the q channel are provided by the sampler. The solver then integrates (22) forward in time to obtain a complete trajectory $\{\mathbf{p}(t_j), \mathbf{q}(t_j)\}_{j=0}^{N_t-1}$ on the learning grid. This seeding strategy anchors the learned representation on early frames and stabilizes the subsequent generative steps.

Mean-only synthesis and uncertainty handling. The evolution in (22) uses only the posterior means μ_p, μ_q to synthesize an augmented trajectory. This choice yields one coherent learned-physics rollout for each initialization without posterior-sampling variation. The GP predictive variances remain available as quality indicators, but they do not perturb these mean-generated fields.

Packaging and normalization. For each realization, the generator writes a four-channel tensor of shape $[4, N_z, N_t]$,

$$\big[\texttt{p_field}, \texttt{full_p}, \texttt{q_field}, \texttt{full_q}\big]. \tag{23}$$

The targets are $\texttt{full_p} = \mathbf{p}$ and $\texttt{full_q} = \mathbf{q}$. The conditioning channels encode the two initial time frames with zeros elsewhere and respect the initialization convention for p. A channelwise affine normalization maps the targets to a symmetric range with the same transform applied to the corresponding conditioning frames to maintain consistency.

A.7 Synthesizing a Dataset Using Sampled Predictions from GP-dPHS

From limited observations to a generative physics prior. Let $\mathcal{D}_e = \{(\mathbf{x}_i, \mathbf{e}_i)\}_{i=1}^{N_{\text{obs}}}$ be the limited state/Hamiltonian-gradient data used to train the GP-dPHS. The Gaussian process does not return a single function; it yields a posterior distribution over Hamiltonian energy functionals. We exploit this posterior to draw realizations of the Hamiltonian gradients and simulate many physically consistent trajectories $\boldsymbol{x}(t,z) = [\,p(t,z),\, q(t,z)\,]^\top$ for diffusion training.

Random Fourier feature prior draw. Consider a real, continuous, shift-invariant kernel $k_e(\cdot,\cdot)$ for the energy-gradient field. By Bochner's theorem, there exists a spectral density $\rho_e(\boldsymbol{\omega})$ such that

$$k_e(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} e^{\,\mathrm{i}\,\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{x}')}\, \rho_e(\boldsymbol{\omega})\, d\boldsymbol{\omega}.$$

We approximate k_e using N_{RFF} random features,

$$\boldsymbol{\psi}(\mathbf{x}) = \sqrt{\tfrac{2}{N_{\text{RFF}}}}\,\big[\cos(\boldsymbol{\omega}_1^\top\mathbf{x} + \beta_1), \ldots, \cos(\boldsymbol{\omega}_{N_{\text{RFF}}}^\top\mathbf{x} + \beta_{N_{\text{RFF}}})\big]^\top,$$

with $\boldsymbol{\omega}_j \sim \rho_e$ and $\beta_j \sim \text{Uniform}[0, 2\pi]$. This gives $k_e(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}')$. A pathwise prior sample is then

$$\mathbf{e}_0(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

which provides one random realization for the stacked Hamiltonian gradients $\mathbf{e}(\mathbf{x}) = \big[\partial\widehat{H}/\partial p(\mathbf{x}),\, \partial\widehat{H}/\partial q(\mathbf{x})\big]^\top$.

Posterior correction on a finite set. Let $\mathbf{X}_e = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_{\text{obs}}}]$ collect the training inputs and let $\mathbf{e}_{\text{obs}}$ collect the corresponding Hamiltonian-gradient targets. Denote the learned mean by $\boldsymbol{\mu}_e(\cdot)$. Define the covariance blocks

$$\mathbf{C}_{ee} = \big[k_e(\mathbf{x}_i, \mathbf{x}_j)\big]_{ij}, \quad \mathbf{C}_{\star e} = \big[k_e(\mathbf{x}_i^\star, \mathbf{x}_j)\big]_{ij}, \quad \mathbf{C}_{\star\star} = \big[k_e(\mathbf{x}_i^\star, \mathbf{x}_j^\star)\big]_{ij},$$

for any query set $\mathbf{X}_\star = \{\mathbf{x}_j^\star\}_{j=1}^{N_\star}$. The random Fourier feature draw induces $\mathbf{e}_0(\mathbf{X}_e)$ and $\mathbf{e}_0(\mathbf{X}_\star)$. A gradient sample from the posterior at $\mathbf{x}^\star \in \mathbf{X}_\star$ is obtained by the exact conditioning correction

$$\mathbf{e}_{\text{post}}(\mathbf{x}^\star) = \boldsymbol{\mu}_e(\mathbf{x}^\star) + \mathbf{e}_0(\mathbf{x}^\star) + \mathbf{C}_{\star e}\big(\mathbf{C}_{ee} + \sigma_n^2\mathbf{I}\big)^{-1}\Big(\mathbf{e}_{\text{obs}} - \boldsymbol{\mu}_e(\mathbf{X}_e) - \mathbf{e}_0(\mathbf{X}_e)\Big), \tag{24}$$

applied entrywise to both gradient components. Equation (24) warps the prior draw so that it agrees with the observations in a kernel-consistent manner and yields an exact posterior sample in the finite-dimensional sense induced by $\mathbf{X}_e$ and $\mathbf{X}_\star$.

Insertion into the distributed port-Hamiltonian dynamics. The sampled gradients define the variational derivative $\delta_{\mathbf{x}}\hat{\mathcal{H}}(\cdot) = \big[\partial\widehat{H}/\partial p(\cdot),\, \partial\widehat{H}/\partial q(\cdot)\big]^\top$. On the spatial grid, we assemble the semi-discrete evolution

$$\frac{d}{dt}\begin{bmatrix}\mathbf{p}(t)\\ \mathbf{q}(t)\end{bmatrix} \;=\; \mathbf{L}_{\text{dPHS}}\,\delta_{\mathbf{x}}\hat{\mathcal{H}}\big(\mathbf{p}(t), \mathbf{q}(t)\big) \;+\; \mathbf{G}_h\,\mathbf{u}(t),$$

where $\mathbf{L}_{\text{dPHS}}$ is the discrete representation of $J - R$ and the boundary conditions, and $\mathbf{G}_h$ maps inputs. The right-hand side is evaluated by applying centered differences in the interior and consistent one-sided stencils at the boundaries to the sampled gradient fields. With the initial state fixed by the first two frames, we integrate in time using an adaptive Runge–Kutta scheme to obtain the trajectories

$$\text{full_p} = \{\mathbf{p}(t_i)\}_{i=1}^{N_t}, \qquad \text{full_q} = \{\mathbf{q}(t_i)\}_{i=1}^{N_t}.$$

Assembly of conditioning and targets. The conditioning channels keep only the first two frames,

$$\text{p_field}(:,:,1{:}2) = \text{full_p}(:,:,1{:}2), \qquad \text{q_field}(:,:,1{:}2) = \text{full_q}(:,:,1{:}2),$$

and are zero elsewhere. Stacking $[\text{p_field}, \text{full_p}, \text{q_field}, \text{full_q}]$ yields a tensor of shape $[4, N_z, N_t]$ that matches the diffusion model interface.

Why this sample-based generator helps representation learning. Drawing $\delta_{\mathbf{x}}\hat{\mathcal{H}}$ from the posterior produces a family of Hamiltonian-consistent vector fields that reflect the epistemic uncertainty learned from $\mathcal{D}_e$. The resulting collection of simulated trajectories covers a diverse yet physically structured region of the state space. This enlarged dataset supervises the diffusion reconstruction objective, while the floor-aware projection target aligns reliable denoised estimates with the learned port-Hamiltonian manifold.

Implementation notes in v5.0. The generator fixes the trained hyperparameters, constructs the random Fourier feature map, draws $\boldsymbol{\xi}$ to obtain $\mathbf{e}_0$, and applies the posterior correction in (24) on the grid required by the discrete operator.

A.8 Separable-Hamiltonian GP-dPHS for Two Dimensions

The two-dimensional membrane introduces an identification problem absent in one dimension. The equation $\partial_t p_{\text{mem}} = \operatorname{div} \boldsymbol{e}_q$ identifies $\boldsymbol{e}_q$ only up to a divergence-free component, so the one-dimensional line-integration construction does not carry over. Independently learned co-energy heads may fit observed derivatives without being the gradient of one Hamiltonian, and such fits were unstable when rolled out away from the observed manifold.

We instead restrict the learned energy density to the separable isotropic form

$$\widehat{H}(p_{\text{mem}}, \boldsymbol{q}_{\text{mem}}) = K(p_{\text{mem}}) + \Omega(\|\boldsymbol{q}_{\text{mem}}\|^2), \qquad e_p = \kappa(p_{\text{mem}}), \qquad \boldsymbol{e}_q = \tau(\|\boldsymbol{q}_{\text{mem}}\|^2)\boldsymbol{q}_{\text{mem}}. \tag{25}$$

The scalar functions κ and τ are fit by one-dimensional random-Fourier-feature ridge regressions through the finite-difference operators for $\partial_t q_x = \partial_x e_p$, $\partial_t q_y = \partial_y e_p$, and $\partial_t p_{\text{mem}} = \operatorname{div} \boldsymbol{e}_q$. Only interior pixels enter these regressions; rows on pinned edges are excluded because they carry constraint forces. This parameterization is conservative by construction and permits coercivity control on the densely covered one-dimensional input domains. The recovered tension law has correlation greater than 0.99 with $\tanh(\|\boldsymbol{q}_{\text{mem}}\|)/\|\boldsymbol{q}_{\text{mem}}\|$.

The rollout structure ablation emphasizes that off-manifold stability is not determined by training fit alone. Independent co-energy heads reach rollout relL2 of approximately 14; a generic three-dimensional RFF Hamiltonian reaches approximately 53 and has a minimum Hessian eigenvalue near -196, with observed off-manifold blow-up. The separable Hamiltonian reduces rollout relL2 to 0.205. These results support the narrower conclusion that structural coercivity matters more than on-manifold derivative fit for this rollout.

A.9 Shift-Gated Augmented-World Contract

Before generating posterior-sample training data, we require the GP-generated trajectories to differ meaningfully but stably from the reference simulator. The gate, implemented by `eval_membrane_gp_shift.py`, accepts a GP configuration only when its posterior-draw displacement from ground truth lies in $[0.15, 0.9]$, posterior draws have nonzero diversity, and the regularization or scaling parameter γ keeps rollout divergence at or below 10%. An excessively accurate GP makes the augmented training set nearly identical to the reference data and removes the identifiable value of physics guidance; insufficient regularization produces unstable or noisy rollouts.

The headline membrane setup passes this contract with 200 observation rows per head, displacement 0.264, diversity 0.096, and $\gamma = 1.0$. The v5-lowbudget dataset is generated only after these quantities are checked. The gate is a benchmark-design contract, not a mechanism for arbitrarily weakening DDPM: PHDME and DDPM receive the same accepted augmented training set, and the projection target is the sole training difference.

Saturated budget-300 study. With roughly 10^8 stacked observations and only 0.2–0.5% unexplained variation, the budget-300 GP makes the augmented training set nearly indistinguishable from ground truth. On $N = 500$ reference membrane trajectories, PHDME and PHDME +proj have relL2 0.201 and oracle-phys 0.068, while DDPM has relL2 0.175 and oracle-phys 0.070. DDPM therefore slightly wins predictive accuracy, while the physics metric is effectively tied at the residual floor. This saturated result motivates the gate: when the training generator already reproduces the reference system, the additional effect of learned-physics guidance is not identifiable.

A.10 Projection Implementation and Learned Representation

The one-dimensional learned residual is evaluated on the full mini-batch space–time grid using centered interior stencils and consistent one-sided boundary stencils. The GP posterior mean supplies the co-energy fields, and its predictive variance is propagated through the same spatial-difference operator to construct the diagonal precision matrix W. Numerical variance floors prevent a small posterior variance from dominating the projection metric.

At each outer iteration, the projection freezes the GP right-hand side and precision matrix, pins the conditioning and boundary entries, and solves the linear system induced by Equation 3 only on free entries. Levenberg–Marquardt damping regularizes the normal equations. The conjugate-gradient solve is exact with respect to this frozen linearized system up to its numerical tolerance; no analytic PDE coefficient enters the solve. The outer loop recomputes the learned right-hand side and stops at the residual floor measured on genuine training trajectories.

The projected trajectory is detached before Equation 4 is evaluated. Consequently, gradients flow through the denoiser displacement $\widehat{\boldsymbol{A}}^{(0)} - \mathrm{sg}[\Pi_{\mathrm{GP}}(\widehat{\boldsymbol{A}}^{(0)})]$, not through the iterative projection algorithm. The string and real-world experiments use the two-dimensional image implementation, while the membrane uses the corresponding volume operator with the separable co-energies from Appendix A.8.

Loss-mechanism ablation. Section 4.3 reports the matched one-knob comparison used for causal interpretation: replacing the projection target with a raw residual penalty increases string test relL2 from 0.428 to 1.115. This matched sweep retrains every variant to 100k iterations and selects within a common validation window.

A central component of PHDME is the use of Gaussian processes to learn an energy representation of the distributed port-Hamiltonian string system from limited data. Unlike purely data-driven models that fit trajectories directly, our GP-dPHS surrogates approximate the underlying Hamiltonian gradients, $\partial H/\partial p$ and $\partial H/\partial q$, providing a structured representation aligned with physical laws.

Learning energy gradients. The training data consist of spatiotemporal fields p and q generated by the wave system. From these fields, we compute integrated derivatives that serve as training targets for the GP models. One GP learns the mapping $(p, q) \mapsto \partial H/\partial p$, and the other learns $(p, q) \mapsto \partial H/\partial q$, thereby embedding the system in an implicit energy functional.

Visualization of learned surfaces. Figure 10 shows the learned GP surfaces for $\partial H/\partial p$ and $\partial H/\partial q$, overlaid with the sparse training observations. The string configuration uses approximately 1,639 GP observations obtained from 20 downsampled trajectories. The smooth posterior means provide the co-energy representation used by both posterior-sample generation and learned-physics projection.

Energy behavior of a representative prediction. For the conservative string, we compute $H(p, q) = \int (p^2 + q^2)/2 \, \mathrm{d}x$ for a representative ground-truth trajectory and PHDME prediction. Figure 11 illustrates close agreement for this sample. The aggregate E-cons values in Table 1, rather than this individual example, provide the quantitative evaluation.

Role in PHDME. These GP-learned energy gradients form the learned residual operator without introducing analytic PDE coefficients. During diffusion training, the residual is expressed through the floor-aware

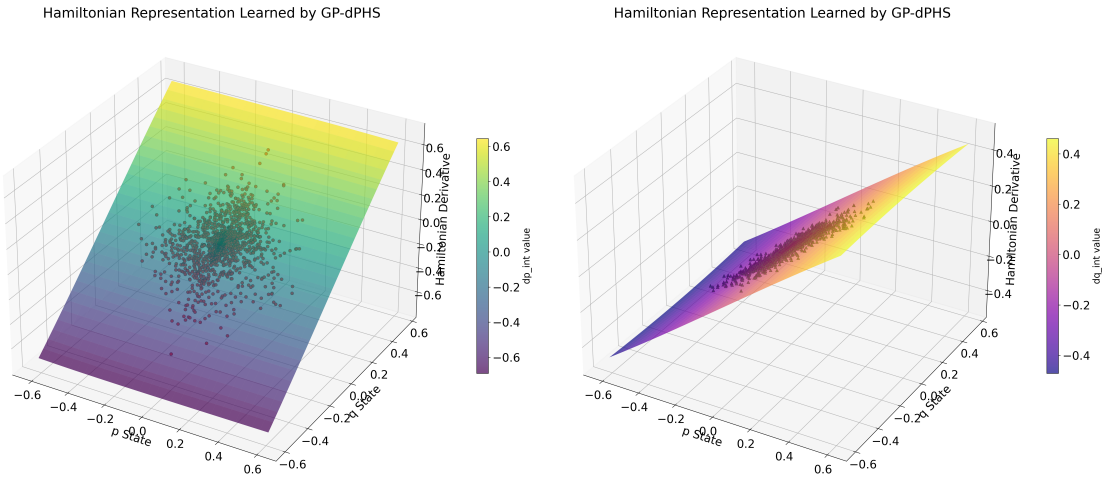

Figure 10: GP posterior means for $\partial H/\partial p$ and $\partial H/\partial q$, with the sparse training observations overlaid. These learned co-energy surfaces define the string residual operator.

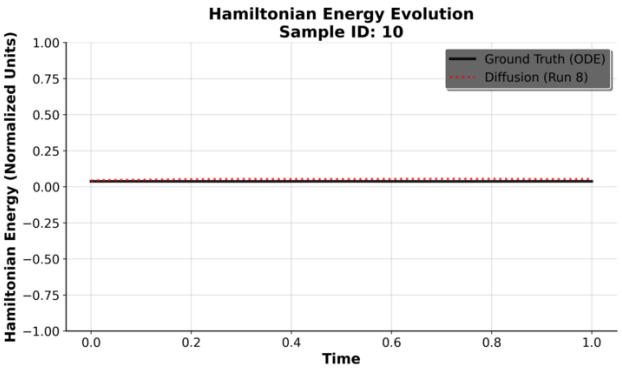

Figure 11: Hamiltonian energy of a representative string ground-truth trajectory and PHDME prediction.

projection target. The posterior is therefore used both to construct the augmented training set and to define a local correction toward the learned port-Hamiltonian manifold.

A.11 Baselines and Integrator Failure Modes

Matched DDPM. DDPM has the same diffusion architecture, optimizer, training trajectories, and sampling schedule as PHDME. It omits the projection target and never receives learned-GP projection at inference. This is the primary ablation isolating the value of the learned-physics constraint.

ConvNODE. ConvNODE is a method-of-lines neural ODE whose convolutional right-hand side is unrolled with RK4. It is competitive on several reference-test accuracy and physics metrics and remains a strong deterministic comparator, but it does not produce a predictive distribution.

FieldPHNN. FieldPHNN represents a pointwise Hamiltonian by an MLP, obtains co-energies by automatic differentiation, and applies the same Stokes–Dirac stencils in learned-operator mode. Its rollout relL2 is approximately 0.94, 2.10, and 0.50 on the string, real-world, and membrane benchmarks, respectively. Its string test residual of 0.008 is deceptive: the model produces nearly constant fields, so satisfying a derivative-based residual does not imply useful forecasting.

Direct learned-physics rollout. The GP-dPHS and advisor-GP baselines directly integrate the learned operator. The real-world relL2 of 2.032, and the membrane test combination of relL2 0.266 with oracle-phys 0.423, show that an operator may fit its generated training trajectories yet remain physically inaccurate for the reference system. These failures motivate amortizing the learned operator into the diffusion model.

Table 5: Generation-speed comparison. PHDME generates a complete trajectory through reverse-diffusion steps in approximately 0.2 seconds per sample, whereas direct GP-dPHS rollout on the fine grid takes approximately 4.1 seconds per sample, a roughly 20× speedup. The relevant forecasting comparisons also favor PHDME over direct GP integration.

Method	100 samples	1,000 samples	10,000 samples	Avg. speed (s/sample)
GP-dPHS	4:40	1:06:40	11:23:20	4.1
PHDME	0:20	3:21	33:23	0.2

A.12 Further Discussion and Limitations

Limitations under extreme scales. While the GP-based representation is robust to moderate data scarcity, it exhibits limitations when the dynamics evolve near extremely small state magnitudes. In these regimes, the training data provide only sparse coverage of the (p, q) space, and the GP posterior surfaces tend to flatten, resulting in poor approximation of the true energy gradients. Consequently, when the diffusion model is conditioned on such representations, generated samples may fail to capture fine-scale oscillatory behavior. This effect is visible in the tails of the learned surfaces, where variance grows and predictions become less structured.

Relative performance. PHDME does not win every metric on the reference test sets. ConvNODE is competitive on several accuracy and physics metrics; direct GP rollout has the lowest membrane relL2 but a substantially worse analytic residual; and DDPM slightly wins relL2 in the saturated budget-300 membrane study. The recurring result is narrower: with a lightweight but stable learned prior, projection-target PHDME improves the accuracy–physics tradeoff relative to matched DDPM.

Ablation: GP-dPHS versus a quadratic-Hamiltonian estimator. Using the same data, a controlled linear-regression estimator fits $\partial H/\partial p$ and $\partial H/\partial q$, corresponding to a global quadratic Hamiltonian. The earlier prior-fitting pipeline reports MSE 0.1818 for GP-dPHS and 0.2967 for this estimator; these values are included as rung 2 of Table 4. They are prior-fit errors rather than downstream forecasting metrics and

therefore support only the narrow claim that the nonparametric representation fits the evaluated energy-gradient targets more accurately.