SOLIS: Physics-Informed Learning of Interpretable Neural Surrogates for Nonlinear Systems

Murat Furkan Mansur and Tufan Kumbasar

***Abstract*—Nonlinear system identification must balance physical interpretability with model flexibility. Classical methods yield structured, control-relevant models but rely on rigid parametric forms that often miss complex nonlinearities, whereas Neural ODEs are expressive yet largely black-box. Physics-Informed Neural Networks (PINNs) sit between these extremes, but inverse PINNs typically assume a known governing equation with fixed coefficients, leading to identifiability failures when the true dynamics are unknown or state-dependent. We propose SOLIS, which models unknown dynamics via a *state-conditioned second-order surrogate model* and recasts identification as learning a Quasi-Linear Parameter-Varying (Quasi-LPV) representation, recovering interpretable natural frequency, damping, and gain without presupposing a global equation. SOLIS decouples trajectory reconstruction from parameter estimation and stabilizes training with a cyclic curriculum and Local Physics Hints windowed ridge-regression anchors that mitigate optimization collapse. Experiments on benchmarks show accurate parameter-manifold recovery and coherent physical rollouts from sparse data, including regimes where standard inverse methods fail.**

***Index Terms*—Physics-Informed Neural Network, System Identification, Quasi-LPV, Physics Hints, Second Order Surrogate.**

I. Introduction

System Identification (SysID) connects observed data to models that can be analyzed and used for control synthesis [1]. A persistent tension is between *interpretability* and *flexibility*. Classical SysID pipelines yield structured and control-oriented models, but they can become restrictive when the dynamics are strongly nonlinear. At the other extreme, modern deep learning approaches offer highly expressive dynamical models [2]. In particular, Neural Ordinary Differential Equations (NODEs) learn continuous-time vector fields from data [3], [4], yet the learned representation is typically unstructured, making it difficult to extract control-oriented quantities such as damping ratios, natural frequencies, or stability margins.

Physics-Informed Neural Networks (PINNs) partially address this by embedding differential constraints into the learning objective [5]. For SysID, Inverse PINN (IPINN) training can reconstruct trajectories and estimate parameters from sparse or irregular measurements without explicit numerical integration [5]. In practice, however, training is often ill-conditioned and sensitive to loss balancing and gradient pathologies, which can lead to convergence to poor solutions or outright failure [6]–[8]. Beyond optimization, IPINN settings face identifiability issues when coefficients are underdetermined or effectively state-dependent, motivating explicit analyses of identifiability and predictability [9]. Related modeling families reflect the same trade-off: discrete-time sequence models (RNN/GRU/LSTM) can predict well but hide dynamics in latent states [1], [10]–[12], and neural state-space models improve expressiveness and uncertainty handling but typically do not expose classical parameters in a directly usable form [13]–[15]. Continuous-time neural modeling, including latent NODE variants, accommodates irregular sampling and has motivated practical identification pipelines for continuous-time state-space models [3], [16]–[18], yet mapping a generic learned vector field into control-relevant structure remains nontrivial even when rollouts are accurate. Hybrid "hidden-physics" approaches attempt to learn unknown components while enforcing known structure [19], but they inherit the same optimization fragilities [6]–[8] and identifiability limits highlighted in applied studies [20], [21].

We propose **SOLIS** (Second-Order Local Identification of Surrogates) to bridge data-driven flexibility with physically interpretable surrogate models of nonlinear systems. Rather than learning an unconstrained black-box vector field, SOLIS identifies a *state-conditioned second-order surrogate* that models local trajectory geometry as a mass–spring–damper system with state-varying coefficients. This yields an interpretable Quasi-Linear Parameter-Varying (Quasi-LPV) representation suitable for gain scheduling and stability analysis, while retaining the physics-regularized benefits of IPINN-style learning.

SOLIS is a structured framework that decouples trajectory approximation from physical-law identification and is designed to mitigate the coupled non-convex failure modes reported in inverse settings [6]–[8]. Our contributions are threefold:

1) *Quasi-LPV Surrogate Formulation:* We define nonlinear identification as learning a state-dependent affine second-order model, mapping complex nonlinearities to interpretable scalar fields.
2) *Solution-Parameter Decomposition:* We introduce a two-network architecture: a *Solution Network* reconstructs trajectories from sparse measurements, while a *Parameter Network* identifies the state-conditioned physics.
3) *Curriculum Learning:* We stabilize the coupled optimization via a cyclic curriculum and sliding-window ridge regression, which provides analytical parameter anchors during early training.

To show the superiority of SOLIS, we conduct comparative experiments on parameter-manifold recovery and physically consistent rollout performance across benchmark systems.

This work was supported by the project (125E231) of the Scientific and Technological Research Council of Türkiye (TUBITAK)

Murat Furkan Mansur and Tufan Kumbasar are with AI and Intelligent Systems Laboratory, Istanbul Technical University, 34469, Istanbul, Türkiye mansur17@itu.edu.tr, kumbasart@itu.edu.tr

II. Preliminaries on PINN

PINNs provide a framework for solving forward differential equation problems by embedding physical constraints into the learning process. In the standard PINN setting, we assume the system dynamics are governed by a *known* model structure $\hat{\mathbf{f}}$:

$$\dot{\mathbf{x}}(t) = \hat{\mathbf{f}}(\mathbf{x}(t), u(t)). \tag{1}$$

To approximate the solution of (1), PINNs employ a neural network $\hat{\mathbf{x}}(t) = \mathcal{N}(t; \mathbf{W})$, parametrized by weights $\mathbf{W}$, which maps time t to the state estimates.

The network is trained to satisfy both the measurement data and the differential equation. This is achieved by minimizing a composite loss function:

$$\min_{\mathbf{W}} \left(\mathcal{L}_{data}(\mathbf{W}) + \lambda_p \mathcal{L}_{phys}(\mathbf{W}) \right). \tag{2}$$

Here, $\mathcal{L}_{data}$ enforces data fidelity, while $\mathcal{L}_{phys}$ acts as a physics-consistency proxy by taking the mean squared ℓ_2 discrepancy, evaluated over collocation points $t \in \mathcal{T}_c$, between the predicted time derivative $\dot{\hat{\mathbf{x}}}(t)$ and the model right-hand side $\hat{\mathbf{f}}(\hat{\mathbf{x}}(t), u(t))$. Crucially, the time derivatives are computed via automatic differentiation on the computation graph of $\mathcal{N}$, allowing for mesh-free gradient-based optimization.

IPINNs extend the PINN framework to SysID tasks by parametrizing the model structure $\hat{f}$ with $\boldsymbol{\theta}$:

$$\dot{\mathbf{x}}(t) = \hat{\mathbf{f}}(\mathbf{x}(t), u(t); \boldsymbol{\theta}). \tag{3}$$

The goal of IPINNs is to jointly estimate the state trajectory $\hat{\mathbf{x}}(t)$ and the system parameters $\boldsymbol{\theta}$. This is formulated as a simultaneous optimization problem:

$$\min_{\mathbf{W}, \boldsymbol{\theta}} \left(\mathcal{L}_{data}(\mathbf{W}) + \lambda_p \mathcal{L}_{phys}(\mathbf{W}, \boldsymbol{\theta}) \right). \tag{4}$$

Thus, the IPINN discovers the parameters $\boldsymbol{\theta}$ that best explain the observed data while maintaining consistency with (3).

III. Identification Objective and Hypothesis

We consider the problem of identifying a nonlinear dynamical system from sparse and noisy measurements. The underlying system is assumed to evolve based on

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)), \tag{5}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the system state, $u(t)$ is a known exogenous input, and the functional form of $\mathbf{f}(\cdot)$ is unknown. Neither the system order nor an explicit parametric structure is assumed a priori.

A. Identification Objective and Setting

The objective of this work is not to recover the true governing equations of the system, but to infer a data-consistent and physically meaningful surrogate model that explains the observed system trajectories and yields interpretable parameters suitable for control-oriented analysis.

To achieve this goal, we assume to have measurements from $J (j = 1, \ldots, J)$ trajectories. The data is composed of:

- A known exogenous input signal $u^{(j)}(t)$, available as a continuous-time function over the horizon $t \in [0, T^{(j)}]$.
- Output measurements $\mathcal{D}_m^{(j)} = \{(t_i^{(j)}, \mathbf{y}_i^{(j)})\}_{i=1}^{N_d^{(j)}}$, where $\mathbf{y}_i^{(j)}$ denotes the observed system states at time $t_i^{(j)}$.

The complete measurement dataset is $\mathcal{D}_m = \bigcup_{j=1}^{J} \mathcal{D}_m^{(j)}$. Besides, for each trajectory, we assume to have a dense set of collocation points $\mathcal{T}_c^{(j)} = \{t_k^{(j)}\}_{k=1}^{N_c^{(j)}}$, with $N_c^{(j)} \gg N_d^{(j)}$.

B. The Second-Order Surrogate Hypothesis

Applying physics-informed learning to the problem (5) presents a fundamental challenge. While standard IPINNs are effective when the exact mathematical structure of $\mathbf{f}$ is known, they face significant difficulties when the model structure is unknown or when the effective parameters vary dynamically across the state space. To bridge this gap, we introduce the **Second-Order Surrogate Hypothesis**:

The local temporal geometry of smooth system trajectories can be approximated by a second-order Quasi-LPV surrogate dynamics.

Accordingly, we define the surrogate state as $\mathbf{x}(t) = [y(t), v(t)]^\top$, where $y(t)$ is the output and $v(t) = \dot{y}(t)$ is the (potentially latent) velocity. We postulate that the approximating physics model $\hat{\mathbf{f}}$ takes the following affine structure:

$$\hat{\mathbf{f}}\big(\mathbf{x}, u; \boldsymbol{\theta}(\mathbf{x})\big) = \begin{bmatrix} v \\ -k(\mathbf{x})y - d(\mathbf{x})v + g(\mathbf{x})u \end{bmatrix}. \tag{6}$$

In contrast to standard IPINNs where $\boldsymbol{\theta}$ is a constant vector, here the parameter set is defined as a *vector field* conditioned on the state: $\boldsymbol{\theta}(\mathbf{x}) = [k(\mathbf{x}), d(\mathbf{x}), g(\mathbf{x})]^\top$. This reframes identification as learning a state-conditioned coefficient field $\boldsymbol{\theta}(\mathbf{x})$ that renders the observed trajectories dynamically consistent under the surrogate in (6). The Quasi-LPV formulation preserves control interpretability at each state $\mathbf{x}$ as the learned coefficients map to canonical second-order quantities, namely natural frequency, damping ratio, and DC gain,:

$$\omega_n(\mathbf{x}) = \sqrt{k(\mathbf{x})}, \quad \zeta(\mathbf{x}) = \frac{d(\mathbf{x})}{2\sqrt{k(\mathbf{x})}}, \quad K(\mathbf{x}) = \frac{g(\mathbf{x})}{k(\mathbf{x})}, \tag{7}$$

enabling local, state-dependent analysis.

IV. Quasi-LPV Surrogate Modeling via Physics-Informed Learning: SOLIS

The proposed formulation creates an intrinsically coupled learning problem. The coefficient fields $k(\mathbf{x}), d(\mathbf{x}), g(\mathbf{x})$ define the differential constraints, but they must be evaluated at the system state $\mathbf{x}(t)$. However, the true state trajectory is unknown (due to sparsity and noise) and must be reconstructed simultaneously. Accordingly, the objective is to jointly learn:

1) A continuous-time reconstruction of the trajectories $\hat{\mathbf{x}}^{(j)}(t)$ for all $j = 1 \ldots J$.
2) A shared parameter network mapping $\mathbf{x} \mapsto [k, d, g]^\top$ that renders the reconstructed trajectories physically consistent according to (6).

To resolve the intrinsic coupling, we adopt a two-network architecture trained under a two-phase curriculum. As shown in Fig. 1, the framework comprises two interacting neural components: (i) a *solution network* $\mathcal{N}_{sol}$ that reconstructs

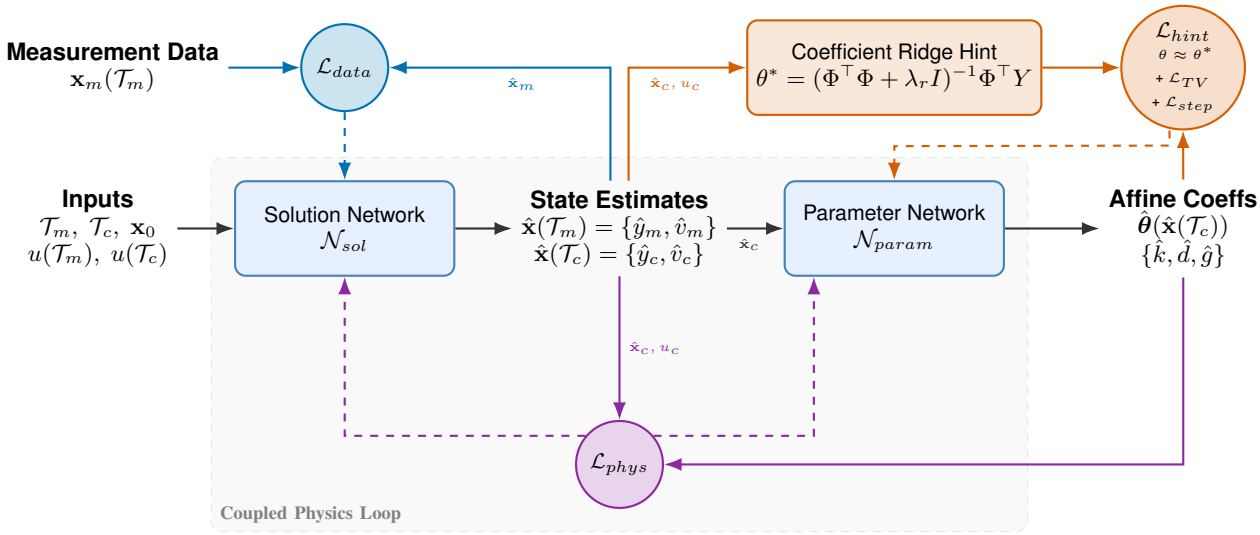

Fig. 1. Overview of the SOLIS architecture. The solution network reconstructs continuous-time state trajectories at measurement times $\mathcal{T}_m$ and collocation times $\mathcal{T}_c$. The data loss uses only $\hat{\mathbf{x}}(\mathcal{T}_m)$, whereas the physics, ridge, and Phase 2 regularization terms are enforced using $\hat{\mathbf{x}}(\mathcal{T}_c)$ and the induced coefficients $\hat{\boldsymbol{\theta}}(\hat{\mathbf{x}}(\mathcal{T}_c))$. Loss terms are color-coded by phase: **data loss** (Phase 1), **hint/regularization losses** (Phase 2), and the **physics loss** coupling both networks.

continuous-time state trajectories, and (ii) a *parameter network* $\mathcal{N}_{param}$ that identifies a state-conditioned affine surrogate of the dynamics.

These components are coupled through a physics-based consistency loss enforcing the surrogate dynamics along the reconstructed trajectory. The learning process alternates between trajectory reconstruction and state-conditioned parameter identification, stabilizing optimization while enforcing consistency with the surrogate dynamics.

A. Solution Network ($\mathcal{N}_{sol}$)

This network approximates the continuous-time state trajectory $\hat{\mathbf{x}}(t) = [\hat{y}(t), \hat{v}(t)]^\top$. Unlike standard PINNs that fit a single time-series, $\mathcal{N}_{sol}$ is trained to represent a family of trajectories corresponding to different initial conditions and input profiles. To achieve this, the network is conditioned on a compact representation of the input through a recurrent context embedding and feature-wise modulation.

1) Context Embedding: For each trajectory, the control input $\mathbf{u}_c = \{u(t)\}_{t\in\mathcal{T}_c^{(m)}}$ is encoded into a fixed-dimensional context vector $\mathbf{c}$ using a Gated Recurrent Unit (GRU) [11]:

$$\mathbf{c} = \mathrm{GRU}(\mathbf{u}_c; \mathbf{W}_{context}). \tag{8}$$

Rather than performing sequential state integration, the GRU is used strictly to embed the discrete exogenous input profile into a static context, preserving the continuous-time nature of the solution network.

2) FiLM Conditioning: The context embedding $\mathbf{c}$ modulates the internal feature maps of the solution network via Feature-wise Linear Modulation (FiLM) [22]. For the l-th hidden layer with pre-activation $\mathbf{z}_l$, the modulation is

$$\mathbf{z}_l' = \boldsymbol{\gamma}_l(\mathbf{c}) \odot \mathbf{z}_l + \boldsymbol{\beta}_l(\mathbf{c}), \tag{9}$$

where $\boldsymbol{\gamma}_l$ and $\boldsymbol{\beta}_l$ are learned affine functions of $\mathbf{c}$. This conditioning enables a single shared network to represent multiple trajectory geometries while preserving a consistent temporal coordinate system.

3) Temporal Fourier Encoding (Optional): To capture high-frequency oscillatory dynamics for complex systems, we optionally use Random Fourier Features (RFF) [23]. The scalar time is mapped via $\mathbf{e}(t) = [\cos(2\pi\mathbf{B}t),\ \sin(2\pi\mathbf{B}t)]^\top$, where $\mathbf{B}$ is a fixed vector of frequencies sampled from $N(0, \sigma^2)$.

B. Parameter Network ($\mathcal{N}_{param}$)

The parameter network maps the instantaneous state and input to the affine surrogate coefficients:

$$[\hat{k}, \hat{d}, \hat{g}]^\top = \mathcal{N}_{param}(\hat{y}, \hat{v}, u; \mathbf{W}_{param}). \tag{10}$$

Since $\mathcal{N}_{\mathrm{param}}$ is conditioned on $\hat{\mathbf{x}}$, physical consistency is enforced across trajectories, ensuring that the learned parameter field represents a global, state-conditioned model.

1) Mixture-of-Experts (Optional): For systems exhibiting highly nonlinear or regime-dependent dynamics, the parameter network may optionally employ a Mixture-of-Experts (MoE) head to improve local expressivity. In this case, the surrogate coefficients are computed as

$$[\hat{k}, \hat{d}, \hat{g}]^\top = \sum_{j=1}^{M} \alpha_j(\mathbf{x}, u)\, \mathcal{E}_j(\mathbf{x}, u), \tag{11}$$

where the mixing weights $\boldsymbol{\alpha}(\mathbf{x}, u)$ are produced by a Softmax gating network. For simpler systems, a standard feed-forward parameter network is used.

2) State Feature Augmentation (Optional): To accelerate the discovery of complex parameter manifolds, we optionally augment the input space of $\mathcal{N}_{param}$ with fixed nonlinear transformations. The raw state vector is expanded to include polynomial terms (e.g., y^2, y^3), absolute values, and interaction terms (e.g., $y\cdot v$). This simple feature engineering provides a strong inductive bias for complex physical systems.

C. Curriculum Training Strategy

Naïve joint optimization of the solution and parameter networks creates a highly ill-conditioned landscape, often collapsing to trivial solutions (e.g., $\hat{y} \to 0, \hat{k} \to 0$) due to gradient conflict. To stabilize training, we employ a *cyclic curriculum* that alternates between two distinct phases (see Algorithm 1*). This strategy decouples the learning of trajectory geometry from the discovery of the parameter manifold, preventing early-stage noise from corrupting the physics identification.

- **Phase 1**-Trajectory Reconstruction: We freeze $\mathcal{N}_{param}$ and optimize only $\mathcal{N}_{sol}$. The objective is to recover a smooth continuous-time trajectory that fits the sparse measurements.

$$\mathcal{L}_{\mathrm{P1}} = \lambda_d \mathcal{L}_{data} + \lambda_{ic} \mathcal{L}_{ic} + \lambda_p \mathcal{L}_{phys}. \tag{12}$$

  The physics residual acts as a *geometric regularizer*, guiding the interpolation to respect the differential constraints induced by the current parameter estimates.
- **Phase 2**-Parameter Identification: We freeze $\mathcal{N}_{sol}$ and optimize $\mathcal{N}_{param}$.

$$\begin{aligned}\mathcal{L}_{\mathrm{P2}} = \lambda_p \mathcal{L}_{phys} + \lambda_{roll} \mathcal{L}_{rollout} + \lambda_h \mathcal{L}_{hint} \\ + \lambda_{reg}(\mathcal{L}_{TV} + \mathcal{L}_{div}).\end{aligned} \tag{13}$$

  Here, to prevent optimization collapse in early iterations, we introduce the **Local Physics Hint** loss $\mathcal{L}_{hint}$. It anchors the network to analytical parameter estimates $\boldsymbol{\theta}^*_{ridge}$ derived from sliding-window ridge regression, providing a convex "warm-start" for the non-convex physics loss.

Phases 1 and 2 are executed in an alternating loop to promote mutual convergence.

Each phase employs a phase-specific composite loss aligned with its learning objective. The remainder of this section details the individual loss components.

1) Physics Consistency (Phase 1 & 2): The core coupling term enforces the surrogate dynamics at all collocation points:

$$\mathcal{L}_{phys} = \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} \left\| \dot{\hat{v}}(t) + \hat{k}\, \hat{y}(t) + \hat{d}\, \hat{v}(t) - \hat{g}\, u(t) \right\|_2^2, \tag{14}$$

2) Trajectory Fidelity (Phase 1): To anchor $\mathcal{N}_{sol}$, we impose consistency with measurements and initial conditions:

$$\begin{aligned}\mathcal{L}_{data} &= \frac{1}{|\mathcal{D}_m|} \sum_{(t, \mathbf{x}_m) \in \mathcal{D}_m} \left\| \hat{\mathbf{x}}_m(t) - \mathbf{x}_m(t) \right\|_2^2, \\ \mathcal{L}_{ic} &= \left\| \hat{\mathbf{x}}_0 - \mathbf{x}_0 \right\|_2^2.\end{aligned} \tag{15}$$

3) Local Ridge Hints (Phase 2): To prevent parameter collapse early in training, we compute a local analytical estimate $\boldsymbol{\theta}^*_{ridge}(t)$ using a *random rolling window* scheme. At each optimization step, we sample an odd window length $w \sim \mathcal{U}\{5, |\mathcal{T}_c|\}$ and form, for every $t \in \mathcal{T}_c$, a local window $W_t^{(w)}$ of size w extracted by a stride-1 rolling. Within each window, the dynamics are approximated as $\mathbf{Y} \approx \boldsymbol{\Phi}\boldsymbol{\theta}$:

$$\mathbf{Y} = [\dot{\hat{v}}(\tau)]_{\tau \in W_t^{(w)}}, \ \boldsymbol{\Phi} = [-\hat{y}(\tau), -\hat{v}(\tau), u(\tau)]_{\tau \in W_t^{(w)}}. \tag{16}$$

*Python implementation: https://github.com/Assaciry/solis

Algorithm 1 SOLIS Training Procedure

Input: Measurements $\mathcal{D}_m$, Collocation points $\mathcal{D}_c$, Phase lengths $K_{1,2}$, Hint decay γ, Ridge λ_r
Initialize: Networks $\mathcal{N}_{sol}, \mathcal{N}_{param}$
for $epoch = 1$ to E **do**
  Sample minibatch $\mathcal{B}$ from $\mathcal{D}_m$ and $\mathcal{D}_c$
  $\hat{\mathbf{x}}, \dot{\hat{\mathbf{x}}} \leftarrow \mathcal{N}_{sol}(\mathcal{B})$ ▷ Forward & Autograd
  $\hat{\boldsymbol{\theta}} \leftarrow \mathcal{N}_{param}(\hat{\mathbf{x}}, \mathbf{u})$ ▷ Parameter prediction
  Compute $\mathcal{L}_{phys}$
  $k \leftarrow epoch \bmod (K_1 + K_2)$
  if $k < K_1$ **then** ▷ Phase 1: Trajectory Learning
    Freeze $\mathcal{N}_{param}$, Unfreeze $\mathcal{N}_{sol}$
    Compute $\mathcal{L}_{data}$ and $\mathcal{L}_{ic}$
    $\mathcal{L}_{total} \leftarrow \lambda_d \mathcal{L}_{data} + \lambda_{ic} \mathcal{L}_{ic} + \lambda_p \mathcal{L}_{phys}$
    Step $\mathcal{N}_{sol}$ optimizer
  else ▷ Phase 2: Parameter Identification
    Freeze $\mathcal{N}_{sol}$, Unfreeze $\mathcal{N}_{param}$
    Decay hint weight: $\lambda_h \leftarrow \lambda_h \cdot \gamma$
    Physics Hint:
      Construct matrices $\boldsymbol{\Phi}, \mathbf{Y}$ locally on batch
      Compute $\boldsymbol{\theta}^*_{ridge}$ via Eq. (17)
      Compute $\mathcal{L}_{hint}$ and weights w_t
    Compute $\mathcal{L}_{roll}$ and $\mathcal{L}_{TV}$
    $\mathcal{L}_{total} \leftarrow \lambda_p \mathcal{L}_{phys} + \lambda_h \mathcal{L}_{hint} + \lambda_{reg}(\mathcal{L}_{roll} + \mathcal{L}_{TV})$
    Step $\mathcal{N}_{param}$ optimizer
  end if
end for
Return Trained networks $\mathcal{N}_{sol}$ and $\mathcal{N}_{param}$

The corresponding ridge estimator is computed in closed form,

$$\boldsymbol{\theta}^*_{ridge}(t) = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda_r \mathbf{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{Y}, \tag{17}$$

where λ_r may be constant or scaled with the local feature energy (i.e., adaptive ridge) to avoid over-regularization in low-excitation windows.

The hint loss penalizes deviation from this analytical proxy, weighted by a local reliability score:

$$\mathcal{L}_{hint} = \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} w_t \left\| \hat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}^*_{ridge}(t) \right\|_2^2. \tag{18}$$

The weight $w_t \in [0, 1]$ is derived from the eigen-structure of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$, which suppresses hints in ill-conditioned or weakly excited regimes. Randomizing the window length across iterations prevents the hint from locking onto a single temporal scale and encourages $\hat{\boldsymbol{\theta}}(t)$ to remain consistent under multiple local linearizations, improving stability in Phase 2.

4) Parameter Regularization (Phase 2): To ensure temporal consistency, we employ Total Variation (TV) regularization on the parameter field and a short-horizon (H-step) rollout loss:

$$\begin{aligned}\mathcal{L}_{TV} &= \frac{1}{|\mathcal{T}_c|} \sum_t \left\| \nabla_x \hat{\boldsymbol{\theta}}(\hat{\mathbf{x}}_t) \right\|_1, \\ \mathcal{L}_{roll} &= \frac{1}{H} \sum_{h=1}^{H} \left\| \mathrm{RK4}(\hat{\mathbf{x}}_t, \hat{\boldsymbol{\theta}}_t, \dots) - \hat{\mathbf{x}}_{t+h} \right\|_2^2.\end{aligned} \tag{19}$$

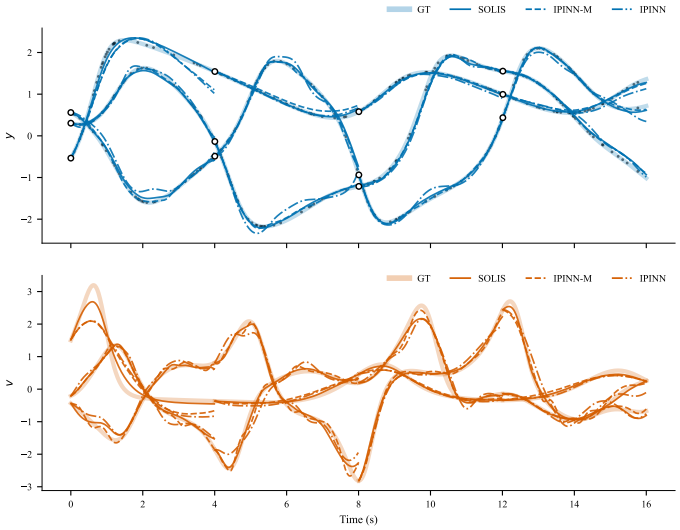

Fig. 2. Van der Pol in-sample reconstruction. Top: $y(t)$ (markers) with Ground Truth (GT) and reconstructed trajectories. Bottom: corresponding $v(t)$.

V. Performance Analysis

In this section, we evaluate the performance of SOLIS on

- Duffing Oscillator (Simulated): A nonlinear benchmark exhibiting cubic stiffness effects, with dynamics governed by a state-dependent stiffness $k(x) = \alpha + \beta x^2$.
- Van der Pol Oscillator (Simulated): Assesses the identification of self-sustained oscillations induced by nonlinear, state-dependent damping $d(x) = \mu(x^2 - 1)$.
- Two-Tank System (Real dataset): A laboratory-scale fluid process characterized by coupled tank dynamics and nonlinear level–flow relationships under measurement noise.

against the following methods:

- **IPINN:** A baseline inverse PINN assuming constant global parameters (k, d, g).
- **IPINN-M:** IPINN with a multi-trajecory setting.
- **TF:** A second-order transfer function estimated with MATLAB's built-in `tfest` function.

Trajectory and rollout performance are quantified via *Accuracy*, defined as $(1 - \text{NRMSE}) \times 100\%$, with NRMSE normalized by the true signal's peak-to-peak range.

A. Solution Reconstruction on Observed Data

We first evaluate *in-sample* solution reconstruction on the observed trajectories to verify that $\hat{\mathbf{x}}(t)$ accurately fits the measurements while remaining physics consistent.

Fig. 2 shows a representative Van der Pol training subset. SOLIS closely tracks the ground-truth trajectories, whereas IPINN and IPINN-M exhibit drift and local mismatches, particularly near peaks and high-curvature regions. Table I confirms this behavior across all training trajectories, with SOLIS achieving the highest reconstruction accuracy.

B. Identification and Generalization

Here, we evaluate the generative capabilities of the learned surrogate $\mathcal{N}_{param}$, i.e., whether the identified Quasi-LPV dynamics recover the underlying vector field and produce stable open-loop predictions beyond the training windows.

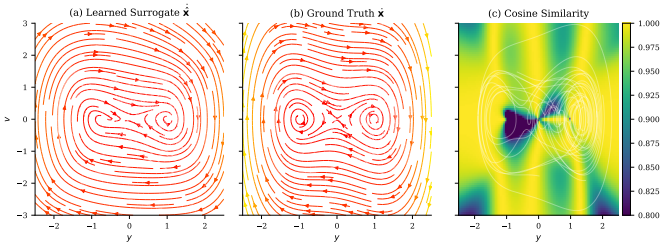

Fig. 3. Duffing Oscillator Phase portrait comparison (a) Surrogate phase portrait calculated via (20), (b) ground-truth phase portrait, (c) cosine similarity between surrogate and ground-truth vector fields over the state space, with training trajectories shown in white.

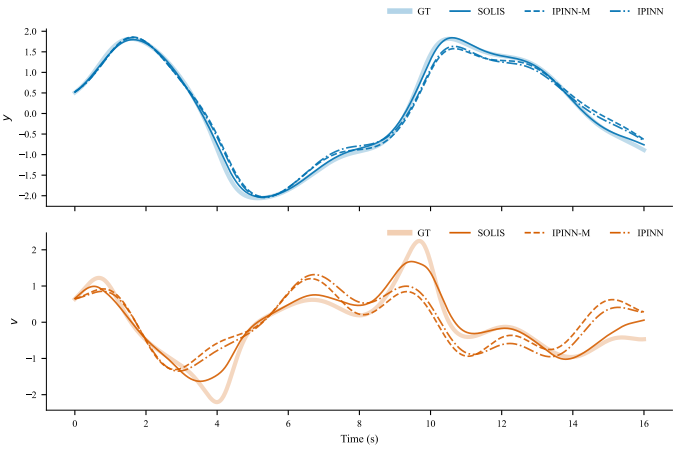

Fig. 4. Van der Pol oscillator open-loop test rollout. Trajectories obtained by forward integrating the learned surrogate dynamics (20) from the same initial condition are compared against the ground truth for both state channels.

a) Phase Portrait Comparison: For Duffing and Van der Pol, we compare the surrogate-induced phase portrait with the ground-truth field. As illustrated in Fig. 3, SOLIS reproduces the qualitative flow structure and matches the ground-truth directions over most of the explored state space; the cosine-similarity map is highest along the regions covered by the training trajectories. Table II confirms this quantitatively: SOLIS attains the highest average similarity on both systems, indicating more faithful recovery of the underlying vector field.

b) Predictive Rollout: To assess generalization, we perform open-loop forward integration on unseen test trajectories. Starting from an initial condition $\mathbf{x}_0$, we integrate the learned Quasi-LPV dynamics using the identified parameter network:

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \hat{v} \\ -\mathcal{N}_{param}(\hat{\mathbf{x}}, u) \cdot [\hat{y}, \hat{v}, -u]^\top \end{bmatrix}. \tag{20}$$

Fig. 4 shows that SOLIS maintains coherent test-time rollouts that track the reference evolution, whereas the baselines accumulate drift over time. The aggregate results in Table III confirm this trend across all benchmarks, with SOLIS achieving the highest accuracy, indicating that its surrogate captures dynamics that generalize beyond in-sample reconstruction.

TABLE I
SOLUTION LEARNING ACCURACY (↑) ON TRAINING TRAJECTORIES

System	IPINN	IPINN-M	SOLIS
Van der Pol	95.94%	98.07%	98.89%
Duffing	95.43%	97.56%	99.01%
Two-Tank	88.64%	91.25%	98.62%

TABLE II
PHASE PORTRAIT SIMILARITY (↑) MEASURED VIA AVERAGE COSINE SIMILARITY TO THE GROUND-TRUTH FIELD.

System	IPINN	IPINN-M	SOLIS
Van der Pol	0.70	0.67	0.86
Duffing	0.81	0.82	0.96

TABLE III
IDENTIFICATION ROLLOUT ACCURACY (↑) ON TEST TRAJECTORIES.

System	IPINN	IPINN-M	TF	SOLIS
Van der Pol	83.72%	81.84%	70.22%	90.54%
Duffing	76.60%	75.27%	74.60%	83.07%
Two-Tank	82.74%	82.9%	77.74%	84.29%

VI. CONCLUSION AND FUTURE WORK

This work proposed SOLIS, a PINN framework for identifying interpretable second-order surrogate models of nonlinear systems. By decomposing the learning problem into trajectory geometry and parameter manifold identification, we addressed the identifiability and stability challenges common in IPINNs.

Our key technical contribution lies in the structured curriculum strategy, specifically the introduction of *Local Physics Hints*. We demonstrated that using a sliding-window ridge regression proxy to warm-start the parameter network acts as an effective convex regularization, preventing the optimization from collapsing into trivial solutions. Furthermore, by framing the interaction between the solution and parameter networks as a self-regulated feedback loop, we established a robust mechanism for discovering physical laws that are consistent with observed data. Experimental results confirm that SOLIS can recover stable, state-dependent parameter profiles even in regimes where global parameter identification fails.

While our current evaluation validates the framework on fundamental low-dimensional benchmarks, future work will scale this methodology to higher-dimensional and more complex nonlinear systems. Additionally, we plan to extend this framework to multi-input multi-output systems and explore the application of the learned surrogates in predictive control settings.

ACKNOWLEDGMENT

The authors acknowledge using ChatGPT and Gemini to refine the grammar and enhance the expression of English.

REFERENCES

[1] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep networks for system identification: A survey," *Automatica*, vol. 171, p. 111907, 2025.

[2] T. Dai, K. Aljanaideh, R. Chen, R. Singh, A. Stothert, and L. Ljung, "Deep learning of dynamic systems using system identification toolbox™," *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 580–585, 2024.

[3] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[4] M. A. Ferah and T. Kumbasar, "Introducing interval neural networks for uncertainty-aware system identification," in *Int. Congress on Human-Computer Interaction, Optimization and Robotic Applications*, 2025.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[6] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[7] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[8] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[9] E. Kharazmi, M. Cai, X. Zheng, Z. Zhang, G. Lin, and G. E. Karniadakis, "Identifiability and predictability of integer- and fractional-order epidemiological models using physics-informed neural networks," *Nature Computational Science*, vol. 1, pp. 744–753, 2021.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] K. Cho, B. van Merriënboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[12] A. Dong, A. Starr, and Y. Zhao, "Neural network-based parametric system identification: a review," *International Journal of Systems Science*, vol. 54, no. 13, pp. 2676–2688, 2023.

[13] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021.

[14] J. Lin and G. Michailidis, "Deep learning-based approaches for state space models: A selective review," arXiv preprint, 2024.

[15] A. E. Sertbaş and T. Kumbasar, "Stable-by-design neural network-based lpv state-space models for system identification," in *Conference of Image Processing, Wavelet and Applications on Real World Problems*, 2025.

[16] Y. Rubanova, T. Q. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 5321–5331.

[17] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *European Journal of Control*, vol. 59, pp. 69–81, 2021.

[18] G. I. Beintema, M. Schoukens, and R. Tóth, "Continuous-time identification of dynamic state-space models by deep subspace encoding," in *International Conference on Learning Representations*, 2023.

[19] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," arXiv preprint, 2018.

[20] J. Stiasny, G. S. Misyris, and S. Chatzivasileiadis, "Physics-informed neural networks for non-linear system identification for power system dynamics," arXiv preprint, 2020.

[21] Z. Lai, C. Mylonas, S. Nagarajaiah, and E. Chatzi, "Structural identification with physics-informed neural ordinary differential equations," *Journal of Sound and Vibration*, vol. 508, p. 116196, 2021.

[22] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 3942–3950.

[23] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 7537–7547.