FIXED-POINT REASONING: LEARNING OPTIMAL PHYSICAL REPRESENTATIONS OF DYNAMICAL SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning dynamical systems from data is fundamental to scientific modeling, prediction, and control. Recurring physical patterns across heterogeneous systems further motivate foundation models that learn general representations. However, existing approaches often rely on fully or partially known governing physics, with physical information typically introduced as input conditioning or output regularization rather than used to systematically shape internal representations. We propose the concept of *optimal physical representation* and formulate its learning as an optimization framework. The key is a feedback link that feeds estimated dynamics back through physics-aware encoding to endow tokens with explicit dynamical identities and refine their representations. This yields a cooperative Refiner-Estimator objective that iteratively minimizes learn-estimate inconsistency, realized as a contractive *Fixed-Point Transformer* with guaranteed convergence to a self-consistent equilibrium. Hence, our foundation model performs fixed-point reasoning, turning the estimation of complex physics into recurrent computation that progressively improves both representations and estimation. Additionally, we propose efficient structured local-operator learning with explicit stability conditions. Experiments across diverse dynamical systems demonstrate accurate prediction, stable long-horizon rollout, and progressive refinement toward the fixed point.

1 INTRODUCTION

Learning dynamical systems from data is a central problem in scientific machine learning, supporting simulation (Dou et al., 2026), forecasting (Li et al., 2026), digital twins (Rasheed et al., 2019), and control across engineered systems (Li et al., 2025; Mai et al., 2025), environmental processes (Lorenz, 2017), and health systems (Qian et al., 2021). Many such systems can be modeled by ordinary differential equations (ODEs), where learning extends beyond forecasting discrete data points to modeling the underlying continuous-time evolution. Recent progress in Transformer-based foundation models has shown that large-scale pretraining can learn general representations that adapt across varying operating conditions and generalize to previously unseen settings (Brown et al., 2020; Das et al., 2024; Liu et al., 2025). Since heterogeneous dynamical systems often share recurring dynamical structures, learning across diverse ODE systems offers a promising path toward general representation learning for dynamics.

Existing approaches address different aspects of dynamical learning. Symbolic regression seeks explicit governing equations from data but typically requires system-specific search and estimation (Brunton et al., 2016b; Li et al., 2022). Physics-informed learning imposes differential equations as training constraints, while Neural ODEs and their variants learn continuous-time vector fields from observations (Cuomo et al., 2022; Chen et al., 2018; Rubanova et al., 2019; Kidger et al., 2020; Li et al., 2026). These methods are effective for individual systems but do not naturally yield representations transferable across heterogeneous dynamics. In contrast, recent time-series foundation models leverage large-scale pretraining for reusable representations and zero-shot prediction (Garza et al., 2023; Rasul et al., 2023; Liu et al., 2024a;b; 2025), with similar ideas extended to heterogeneous physical systems (McCabe et al., 2024). However, they are primarily prediction-driven, with limited mechanisms for physical fidelity.

Recent scientific foundation models incorporate physics through operator structures, physical conditioning, or physics-informed objectives (Herde et al., 2024; Rahman et al., 2024; Sun et al., 2025; Soares et al., 2025; Zhu et al., 2025b), as illustrated in Section 5. Yet these approaches typically rely on known or externally provided physics through architectural priors, inputs, or regularization. In

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

many systems, governing physics may instead be unknown, nonlinear, or time-varying and must be inferred from data. Even then, it remains unclear how inferred physics should systematically shape and refine internal representations rather than serve as additional context or regularization.

Together, these limitations leave a coupled representation-learning problem unresolved: what representation should be learned to infer unknown dynamics, and how should the inferred physical knowledge refine the representation itself? Addressing this requires moving beyond one-way physical conditioning or output regularization toward a coupled process between representation learning and dynamics inference. We therefore introduce the notion of an *optimal physical representation*, where the representation and inferred dynamics are jointly refined to minimize their inconsistency. We formalize a theoretical framework that establishes a feedback link: inferred dynamics are fed back through physics-aware encoding to refine token representations, assigning each token an explicit dynamical identity beyond generic features. This leads to a cooperative Refiner-Estimator framework, where the Estimator infers dynamics and the Refiner uses the inference to update representations. We realize this recurrence as a *Fixed-Point Transformer* with contractive constraints, guaranteeing convergence to a self-consistent equilibrium. In Section 5, we compare our design with other recurrent Transformers. The whole framework is dubbed *LASS-ODE* (Large-scale System of Small ODEs). Our contributions are threefold:

- **Physics-grounded fixed-point reasoning.** LASS-ODE performs fixed-point reasoning via recurrent refinement: it gradually estimates the underlying physics, feeds the estimate back to refine the representation, and progressively improves learning and estimation toward an equilibrium.
- **Structured local dynamics with adaptive stability.** LASS-ODE learns structured local operators whose parameterization maps directly to fundamental dynamical characteristics, while automatically adapting local stability across different dynamical regimes.
- **Large-scale validation across heterogeneous dynamics.** Experiments across diverse ODE systems demonstrate accurate prediction, zero-shot generalization, stable long-horizon rollout, and further gains from lightweight LoRA fine-tuning (Hu et al., 2022).

2 Definitions of Problem and Optimal Physical Representation

We consider a family of dynamical systems governed by ODEs. For system i with state $\boldsymbol{x}_i \in \mathcal{X}_i$, where $\mathcal{X}_i$ denotes its state space, the evolution is described by $f_i : \mathcal{X}_i \times \Theta_i \to \mathcal{X}_i$ as $\dot{\boldsymbol{x}}_i = f_i(\boldsymbol{x}_i; \boldsymbol{\theta}_i)$, where $\boldsymbol{\theta}_i \in \Theta_i$ denotes the dynamic parameters of system i. We observe $\{\boldsymbol{x}_i(t_\ell^i)\}_\ell$ at timestamps $\mathcal{T}_{\text{obs}}^i := \{t_\ell^i\}_\ell$. Across systems, the state dimensions, sampling resolutions, and physical timescales may vary. We normalize the timestamps in each $\mathcal{T}_{\text{obs}}^i$ while preserving their relative temporal locations, and define the unified continuous temporal domain as $\mathcal{T} := [0, 1]$, with $\mathcal{T}_{\text{obs}}^i \subset \mathcal{T}$. The inferred trajectory of system i is denoted by $\widehat{\boldsymbol{x}}_i : \mathcal{T} \to \mathcal{X}_i$. Our goal is to learn transferable representations across a family of ODE systems, infer their underlying dynamics, and recover $\{\widehat{\boldsymbol{x}}_i\}_i$ over $\mathcal{T}$ from observations $\{\boldsymbol{x}_i(t_\ell^i) \mid t_\ell^i \in \mathcal{T}_{\text{obs}}^i\}_i$. We omit the system index i hereafter for notational simplicity.

Given the above problem setup, we consider three coupled variables: the learned representation E, the inferred physical trajectory $\widehat{\boldsymbol{x}}$, and the inferred dynamic parameters $\widehat{\boldsymbol{\theta}}$. A physically meaningful representation should make them mutually consistent. This motivates the following definition.

Definition 1. *A learned representation $E^\star$ is called a self-consistent optimal physical representation if the induced triplet $\left(E^\star, \widehat{\boldsymbol{x}}^\star, \widehat{\boldsymbol{\theta}}^\star\right)$ satisfies the following three consistency requirements:*

- ***Coordinate consistency*** $E^\star \leftrightarrow \widehat{\boldsymbol{x}}^\star$**.** *$E^\star$ and $\widehat{\boldsymbol{x}}^\star$ should be defined over consistent physical coordinates, where temporal coordinates are aligned on the unified domain $\mathcal{T}$ and spatial coordinates are determined by the state variables of each ODE system.*
- ***Learn-estimate consistency*** $E^\star \leftrightarrow \widehat{\boldsymbol{\theta}}^\star$**.** *Let* Est *denote the estimator that infers the dynamic parameters from the learned representation, and let* Ref *denote the refiner that updates the representation using the inferred physics. The optimal representation must satisfy:*

$$\widehat{\boldsymbol{\theta}}^\star = \text{Est}\left(E^\star\right), \qquad E^\star = \text{Ref}\left(E^\star, \widehat{\boldsymbol{\theta}}^\star\right). \tag{1}$$

- ***Governing consistency*** $\widehat{\boldsymbol{x}}^\star \leftrightarrow \widehat{\boldsymbol{\theta}}^\star$**.** *The inferred trajectory satisfies the governing dynamics associated with the inferred dynamic parameters: $\dot{\widehat{\boldsymbol{x}}}^\star(\tau) = f\left(\widehat{\boldsymbol{x}}^\star(\tau); \widehat{\boldsymbol{\theta}}^\star\right), \forall\, \tau \in \mathcal{T}$.*

3 METHODOLOGY

3.1 TIME-AWARE ENCODING FOR UNIFIED COORDINATES

Since E is derived from observations on $\mathcal{T}_{\text{obs}}$, it does not naturally share the temporal coordinates of $\widehat{\boldsymbol{x}}$ over $\mathcal{T}$. To guarantee *coordinate consistency*, we develop an encoding-modulation process.

Variable-wise processing. Different ODE systems may have different state spaces and numbers of variables. We therefore adopt variable-wise processing (Nie et al., 2022), where each physical variable $x^{(j)}$ is independently encoded and tokenized, preserving the spatial coordinates. This enables the same architecture to handle heterogeneous systems with different state dimensions. Dependencies among variables are then captured through the following attention layers.

Temporal coordinate-aware encoding. For each variable $x^{(j)}$, we first extract a trajectory-level temporal representation from its observations over $\mathcal{T}_{\text{obs}}$. A time-aware GRU takes the observed values and the corresponding sampling intervals as inputs: $\boldsymbol{h}^{(j)} = \text{GRU}\left(\left\{\left[x^{(j)}(t_\ell); \Delta t_\ell\right]\right\}_{t_\ell \in \mathcal{T}_{\text{obs}}}\right)$, where Δt_ℓ denotes the interval between consecutive observation timestamps.

Coordinate-conditioned modulation. The trajectory-level representations $\{\boldsymbol{h}^{(j)}\}_j$ can help construct E but are not yet defined over the unified temporal domain $\mathcal{T}$. We therefore construct K_{token} localized tokens with uniformly sampled temporal coordinates $\{\tau_k\}_{k=1}^{K_{\text{token}}}$ over $\mathcal{T}$. Then, we use radial basis functions (RBFs) to allow each token to emphasize its nearby temporal region while retaining graded influence from neighboring regions. Given K_{rbf} Gaussian kernels with centers $\{\mu_\ell\}_{\ell=1}^{K_{\text{rbf}}}$ evenly spaced over $\mathcal{T}$ and bandwidth σ, the response of the ℓth kernel at τ_k is $\phi_{\text{rbf}}^{(\ell)}(\tau_k) = \exp\left(-\frac{(\tau_k-\mu_\ell)^2}{2\sigma^2}\right), \ell \in \{1, \ldots, K_{\text{rbf}}\}$. We denote the stacked RBF embedding by $\boldsymbol{\phi}_{\text{rbf}}(\tau_k)$. The temporal coordinate τ_k then modulates the trajectory-level representation $\boldsymbol{h}^{(j)}$:

$$(\boldsymbol{\gamma}_k; \boldsymbol{\beta}_k) = \text{MLP}\left(\text{LN}\left(\boldsymbol{\phi}_{\text{rbf}}(\tau_k)\right)\right), \qquad \boldsymbol{e}_{jk} = \boldsymbol{\gamma}_k \odot \text{MLP}\left(\boldsymbol{h}^{(j)}\right) + \boldsymbol{\beta}_k. \tag{2}$$

To identify different state variables within a dynamical system, we introduce a learnable lookup table $C \in \mathbb{R}^{d_x^{\max} \times d_{\text{model}}}$, where $d_x^{\max}$ denotes the maximum state dimension across all systems. The jth row $C^{(j)} \in \mathbb{R}^{d_{\text{model}}}$ provides the embedding associated with variable j. Then, we have $\widetilde{\boldsymbol{e}}_{jk} = \boldsymbol{e}_{jk} + C^{(j)}$, where the tokens $\{\widetilde{\boldsymbol{e}}_{jk}\}_{j,k}$ preserve the state-variable and temporal coordinates of the target $\widehat{\boldsymbol{x}}$. As shown in the left part of Fig. 1, these tokens are processed by a standard Transformer to capture interactions across variables and times, yielding the initial representation $E^{(0)}$.

3.2 PHYSICS-GROUNDED FIXED-POINT REASONING TO GAIN EQUILIBRIUM

We next guarantee the *learn-estimate consistency* in Definition 1 through two cooperative roles: the physics *Estimator* and the representation *Refiner*. They are jointly optimized to minimize the inconsistency: $\min_{\text{Est, Ref}} \|E - \text{Ref}\left(E, \text{Est}(E)\right)\|_{\text{F}}^2$. At iteration r, the recurrent update is defined as $\widehat{\boldsymbol{\theta}}^{(r)} = \text{Est}\left(E^{(r)}\right), E^{(r+1)} = \text{Ref}\left(E^{(r)}, \widehat{\boldsymbol{\theta}}^{(r)}; E^{(0)}\right)$. Here, $E^{(0)}$ is retained as a fixed anchor that preserves the information extracted from the input observations.

Physical representation refinement. How should the Refiner improve the representation? We introduce *two complementary sources*: (1) **Intrinsic physics-aware encoding.** The physics-aware encoding $P\left(\widehat{\boldsymbol{\theta}}^{(r)}\right)$ feeds the inferred dynamics back into the representation, *endowing each token with an explicit dynamical identity and grounding it in physical meaning rather than generic features*. $\text{Est}\left(E^{(r)}\right)$ and $P\left(\widehat{\boldsymbol{\theta}}^{(r)}\right)$ are detailed in Eq. (6) in Section 3.3 and Eq. (8) in Section 3.4, respectively. (2) **Extrinsic cross-system transfer.** We propose inter-intra multi-head attention (MHA), denoted by $\text{MHA}_{\text{II}}(\cdot)$, to jointly extract intra- and inter-system features. In this sense, cross-system knowledge serves as an extrinsic source of refinement. $\text{MHA}_{\text{II}}(\cdot)$ is introduced in Eq. (10) in Section 3.4.

Refiner formulation. As shown in the middle part of Fig. 1, we integrate the two refinement mechanisms into a Transformer-based Refiner $\text{Ref}(\cdot)$. At iteration r, the update is:

$$\begin{aligned} \widetilde{E}^{(r)} &= E^{(r)} + P\left(\widehat{\boldsymbol{\theta}}^{(r)}\right), & \overline{E}^{(r)} &= \widetilde{E}^{(r)} + \text{MHA}_{\text{II}}\left(\text{LN}_1\left(\widetilde{E}^{(r)}\right)\right), \\ \widehat{E}^{(r+1)} &= \overline{E}^{(r)} + \text{MoE}\left(\text{LN}_2\left(\overline{E}^{(r)}\right)\right), & E^{(r+1)} &= (1-\rho)E^{(0)} + \rho\widehat{E}^{(r+1)}. \end{aligned} \tag{3}$$

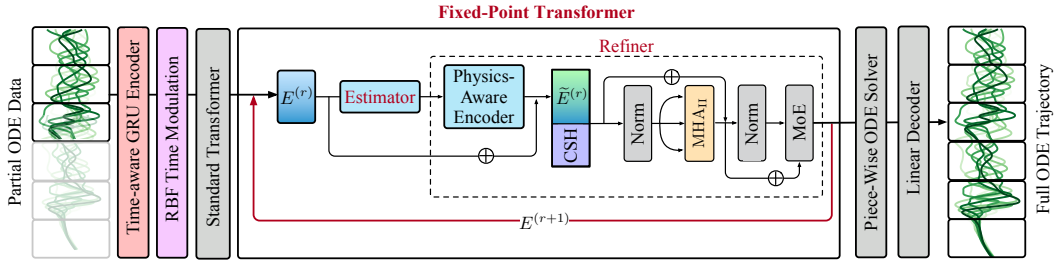

Figure 1: The proposed LASS-ODE framework.

Compared with a standard Transformer block, Eq. (3) has three key components, marked in blue. First, $P(\widehat{\boldsymbol{\theta}}^{(r)})$ feeds the estimated dynamics back into the representation, providing intrinsic physics-aware refinement. Second, $\mathrm{MHA}_{\mathrm{II}}$ augments conventional self-attention with recurring cross-system representations summarized by the prototype bank, providing extrinsic refinement across heterogeneous systems. Finally, the ρ-weighted anchoring to $E^{(0)}$, with $\rho \in (0, 1)$, regulates the recurrent update and enables the contraction condition for fixed-point convergence established below.

Fixed-Point Transformer for equilibrium convergence. Our goal is to reach the equilibrium in Eq. (1). Specifically, we restrict the computation as follows: a *bounded representation space*, *restricted scaling factors*, and *restricted transformations*. Let $\mathcal{H}$ denote a nonempty, closed, convex, and bounded admissible representation space. Then, we have:

$$E^{(r)} \leftarrow \mathrm{Proj}_{\mathcal{H}}\left(E^{(r)}\right),\ \boldsymbol{\gamma}_{\mathrm{LN}} \leftarrow \mathrm{clip}\left(\boldsymbol{\gamma}_{\mathrm{LN}}, -\gamma_{\mathrm{LN}}^{\max}, \gamma_{\mathrm{LN}}^{\max}\right),\ W \leftarrow \frac{s_W W}{\max\left\{s_W, \sigma_{\max}(W)\right\}}. \quad (4)$$

Here, $\boldsymbol{\gamma}_{\mathrm{LN}}$ denotes the trainable LayerNorm scaling factors, while W represents the learned weight matrices in the attention, MLP, physics-aware encoding, and Estimator modules. Moreover, $\sigma_{\max}(W)$ denotes the largest singular value of W, and $s_W > 0$ specifies its upper bound. The following theorem shows that these architectural conditions are sufficient to guarantee convergence.

Theorem 1. *Consider the recurrent Estimator-Refiner update in Eq. (3) for a fixed input representation $E^{(0)}$. Suppose that the Estimator and physics-aware encoding are composed of Lipschitz operations on $\mathcal{H}$, and that the architectural conditions in Eq. (4) are enforced. Then the undamped candidate update $E^{(r)} \to \widehat{E}^{(r+1)}$ admits a finite Lipschitz constant L. If the anchoring coefficient satisfies $0 < \rho < \frac{1}{\max\{1,L\}}$, then the complete recurrent update is contractive with contraction factor $\kappa \le \rho L < 1$. Consequently, there exists a unique equilibrium $(E^\star, \widehat{\boldsymbol{\theta}}^\star)$ conditioned on $E^{(0)}$, and the recurrent representations converge geometrically: $\left\|E^{(r)} - E^\star\right\|_{\mathrm{F}} \le \kappa^r \left\|E^{(0)} - E^\star\right\|_{\mathrm{F}}$. Moreover, $\widehat{\boldsymbol{\theta}}^{(r)}$ converges to $\widehat{\boldsymbol{\theta}}^\star = \mathrm{Est}(E^\star)$, thereby reaching the equilibrium in Eq. (1).*

The proof can be found in Appendix B. The recurrent refinement terminates when $\|E^{(r+1)} - E^{(r)}\|_{\mathrm{F}} < \epsilon$. Implementation details for convergence are in Appendix F. This process is *think-deep reasoning in the representation space* to repeatedly estimate physics and refine representations. This differs from a typical "think-long" paradigm in LLM reasoning, which often allocates more computation to generate longer token sequences (Wei et al., 2022). Further comparisons, including recurrent Transformer, are provided in Section 5. Our fixed-point reasoning opens new opportunities for scientific discovery by learning E^* that corresponds to the underlying physical reality.

3.3 Structured Local Operator Learning

In this subsection, we instantiate the Estimator $\mathrm{Est}(\cdot)$. To begin with, we introduce a universal modeling form for the nonlinear and time-varying dynamics: we first lift the system state into a shared latent space, and then approximate the dynamics using token-wise linear operators. Specifically, with the trajectory-level representations $\boldsymbol{h}^{(j)}$ in Section 3.1, the latent dynamics are given by

$$\boldsymbol{z}_0 = \phi\Big(\frac{1}{d_x}\sum\nolimits_{j=1}^{d_x} \boldsymbol{h}^{(j)}\Big) \in \mathbb{R}^{d_z}, \qquad \dot{\boldsymbol{z}}(\tau) = A_k \boldsymbol{z}(\tau) + \boldsymbol{b}_k, \quad \tau \in [\tau_k, \tau_{k+1}). \quad (5)$$

Here, $\phi(\cdot)$ is an MLP-based projection into the shared latent space. The mean aggregation provides a dimension-invariant initialization. Starting from $\boldsymbol{z}_0$, the latent trajectory can be efficiently computed using batched ODE solvers (Chen et al., 2018). Collectively, the token-wise operator parameters $\{A_k, \boldsymbol{b}_k\}_k$ instantiate the inferred dynamics $\widehat{\boldsymbol{\theta}}$. The capability of Eq. (5) to approximate nonlinear

dynamics is analyzed in Theorem 2 in Appendix C. The theorem bounds the worst-case mismatch between a ground-truth latent trajectory $\boldsymbol{z}$ and a trajectory $\widehat{\boldsymbol{z}}_k$ propagated by the learned affine operator within one token, assuming they start that token from the same state. The bound decomposes the approximation error into lifting, tokenization, and operator estimation errors, and shows that the tokenization error decreases quadratically as the temporal discretization is refined, providing a controllable approximation of nonlinear latent dynamics.

Following standard stability analysis of continuous-time linear systems (Söderlind, 2006), we decompose $A_k = \frac{A_k - A_k^\top}{2} + \frac{A_k + A_k^\top}{2} = \Omega_k + Q_k \operatorname{diag}(\alpha_{k,1}, \ldots, \alpha_{k,d_z}) Q_k^\top$. Here, $\Omega_k^\top = -\Omega_k$ characterizes the rotational component, while $\{\alpha_{k,m}\}_{m=1}^{d_z}$ quantify the instantaneous growth/decay rates along the orthogonal directions defined by Q_k. For these dynamic parameters, we apply an estimator:

$$\boldsymbol{e}_k = \frac{1}{d_x} \sum\nolimits_{j=1}^{d_x} \boldsymbol{e}_{jk}, \qquad (B_k,\, \{\boldsymbol{v}_{k,\ell}\}_{\ell=1}^{L_Q},\, \boldsymbol{\alpha}_k,\, \boldsymbol{b}_k) = \mathrm{Est}\left(\boldsymbol{e}_k\right). \tag{6}$$

Here, $\boldsymbol{\alpha}_k = [\alpha_{k,1}, \ldots, \alpha_{k,d_z}]^\top$, $B_k \in \mathbb{R}^{d_z \times d_z}$ is an unconstrained matrix, and $\{\boldsymbol{v}_{k,\ell}\}_{\ell=1}^{L_Q}$ are Householder vectors. We construct the skew-symmetric and orthogonal components by $\Omega_k = \frac{B_k - B_k^\top}{2}$, $H(\boldsymbol{v}_{k,\ell}) = I - 2\frac{\boldsymbol{v}_{k,\ell}\boldsymbol{v}_{k,\ell}^\top}{\boldsymbol{v}_{k,\ell}^\top \boldsymbol{v}_{k,\ell}}$, $Q_k = \prod_{\ell=1}^{L_Q} H(\boldsymbol{v}_{k,\ell})$, where the Householder parameterization follows (Mhammedi et al., 2017). Such computation to obtain Q_k requires $\mathcal{O}(L_Q d_z)$ operations, compared with $\mathcal{O}(d_z^3)$ for dense QR decomposition-based orthogonalization.

As for $\boldsymbol{\alpha}_k$, we first use Theorem 3 in Appendix D to show that the ODE rollout error is bounded by $\alpha_k^{\max} := \max_{m \in \{1,\ldots,d_z\}} \alpha_{k,m}$. Specifically, a negative $\alpha_k^{\max}$ exponentially attenuates existing rollout errors. Such stabilized behavior is common in physical systems dominated by damping, dissipation, or restoration, while positive $\alpha_{k,m}$ may still be required for intrinsically expansive regimes such as bifurcations, phase transitions, turbulent flows, or chaotic dynamics.

Automatic stability-adaptation learning. We further elaborate the estimation of $\boldsymbol{\alpha}_k$ to favor negative $\alpha_{k,m}$ by default to encourage a stable rollout, while allowing the model to automatically activate positive $\alpha_{k,m}$ when locally expansive dynamics are needed:

$$\alpha_{k,m} = -\operatorname{softplus}\left(-\widetilde{\alpha}_{k,m}\right) + \lambda(s) g_k \operatorname{softplus}\left(\widetilde{\alpha}_{k,m}\right), \qquad m = 1, \ldots, d_z, \tag{7}$$

where $\widetilde{\alpha}_{k,m}$ denotes the raw estimator output, $g_k \in [0,1]$ is a token-wise gate predicted from $\boldsymbol{e}_k$, and $\lambda(s) \in [0,1]$ is a monotone schedule over the training progress s. At the beginning of training, $\lambda(s) \approx 0$, which enforces $\alpha_{k,m} < 0$. As training proceeds, $\lambda(s)$ increases, while g_k determines whether the local dynamics require relaxation toward non-negative growth rates.

Given the inferred local dynamic parameters Ω_k, Q_k, $\boldsymbol{\alpha}_k$, we obtain the continuous latent trajectory by integrating Eq. (5) with batched linear ODE solvers and reconstruct the physical trajectory as $\widehat{\boldsymbol{x}}(\tau) = W_{\mathrm{dec}} \boldsymbol{z}(\tau) + \boldsymbol{b}_{\mathrm{dec}}, \tau \in \mathcal{T}$, as shown in the right part of Fig. 1. Since $\boldsymbol{z}(\tau)$ is directly evolved under the inferred local operators, our $\widehat{\boldsymbol{x}}(\tau)$ and $\widehat{\boldsymbol{\theta}}$ satisfy the *governing consistency* in Definition 1. Unlike Latent ODEs (Rubanova et al., 2019), which repeatedly evaluate a nonlinear neural vector field during integration, LASS-ODE evaluates inexpensive affine dynamics $A_k \boldsymbol{z} + \boldsymbol{b}_k$, enabling substantially faster ODE rollout. A detailed analysis is provided in Appendix A.

3.4 Physics-Grounded Representation Refinement

In this subsection, we detail these two refinement mechanisms, i.e., $P(\widehat{\boldsymbol{\theta}})$ and $\mathrm{MHA}_{\mathrm{II}}$, as follows.

Intrinsic physics-aware encoding via probe-based dynamic signature. A naive choice is to directly encode the estimated decomposition parameters $\widehat{\boldsymbol{\theta}} = \{\Omega_k,\, Q_k,\, \boldsymbol{\alpha}_k,\, \boldsymbol{b}_k\}_{k=1}^{K_{\mathrm{token}}}$. However, Q_k is not uniquely represented due to permutation ambiguities and can become numerically sensitive when values of $\alpha_{k,m}$ are close. We therefore characterize each local operator *through its responses to a set of shared latent probes*. Specifically, we first compute $S_k = Q_k \operatorname{diag}(\boldsymbol{\alpha}_k) Q_k^\top$ and then evaluate both Ω_k and S_k on a shared probe matrix $R \in \mathbb{R}^{d_z \times d_R}$ with $R^\top R = I$, where R is generated once by orthogonalizing a random Gaussian matrix and is shared across all tokens and systems. The dimension $d_R \le d_z$ controls the probe dimension. The resulting probe responses are encoded as

$$P(\widehat{\boldsymbol{\theta}})_{jk} = h_{\mathrm{phy}}\left(\left[\mathrm{vec}(\Omega_k R); \mathrm{vec}(S_k R); \boldsymbol{b}_k; \alpha_k^{\max}; \alpha_k^{\min}\right]\right) \in \mathbb{R}^{d_{\mathrm{model}}}, \tag{8}$$

where $P(\widehat{\boldsymbol{\theta}})_{jk}$ denotes the physics-aware encoding at temporal token k, shared across variables since the local operator is defined at the token level. $h_{\mathrm{phy}}(\cdot)$ is our physics-aware encoder. $\mathrm{vec}(\Omega_k R)$ and

$\mathrm{vec}(S_k R)$ capture the rotational and directional responses of the local operator, respectively, while $\boldsymbol{b}_k$, $\alpha_k^{\max}$, and $\alpha_k^{\min}$ preserve drift and stability information.

Extrinsic inter-intra system attention via common structure hub (CSH). We construct the CSH $\Pi = [\boldsymbol{\pi}_1; \ldots; \boldsymbol{\pi}_{K_{\mathrm{CSH}}}] \in \mathbb{R}^{K_{\mathrm{CSH}} \times d_{\mathrm{model}}}$ from the equilibrium representations $E^\star$ in Eq. (1), where each vector summarizes a recurring token-level pattern across ODE systems. CSH is updated online:

$$\mathcal{B}_l = \{\mathrm{sg}(E^\star_{jk}) : l = \arg\min_q \left\|\mathrm{sg}(E^\star_{jk}) - \boldsymbol{\pi}_q\right\|_2^2\}, \quad \boldsymbol{\pi}_l \leftarrow \eta \boldsymbol{\pi}_l + (1-\eta)\frac{1}{|\mathcal{B}_l|}\sum\nolimits_{\boldsymbol{e}\in\mathcal{B}_l} \boldsymbol{e}, \quad (9)$$

where $\mathcal{B}_l$ contains the equilibrium tokens assigned to the common structure l, $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator and prevents the clustering update into $E^\star$, and $\eta \in (0, 1)$ is the exponential-moving-average coefficient controlling the update rate. CSH is updated only after $E^\star$ is obtained. Then, we define the intra-inter-system $\mathrm{MHA}_{\mathrm{II}}$ (see the middle part in Fig. 1) for Eq. (3):

$$Z = [\mathrm{LN}_1\left(\widetilde{E}\right); \mathrm{LN}_1(\Pi)], \qquad \mathrm{MHA}_{\mathrm{II}}\left(\mathrm{LN}_1\left(\widetilde{E}\right)\right) = [\mathrm{MHA}\left(Z, Z, Z\right)]_{\widetilde{E}}. \quad (10)$$

Here, $[\cdot]_{\widetilde{E}}$ retains only the outputs corresponding to the current-system tokens. The resulting joint self-attention allows each token to interact with both ODE system tokens and CSH tokens.

4 Experiments

Datasets. We employ a curated dataset across diverse real-world ODE systems. Dataset descriptions, system portions, and representative training trajectories are provided in Appendix E. **Model and environment** are described in Appendix F. **Benchmarks**. Our baseline suite comprises neural ODE and hybrid continuous-time models, task-specific Transformer forecasting models, cross-system time-series foundation models, state-space sequence models, liquid neural networks, and scientific operator models and ODE adaptations, as detailed in Appendix G. **Additional numerical analyses and results** are provided in the Appendix, including additional in-domain prediction and zero-shot generalization results (Appendix I and Appendix J), fine-tuning evaluation (Appendix K), CSH effectiveness through attention visualization (Appendix L), iterative dynamical reasoning visualization (Appendix M), training convergence analysis (Appendix N), real-world engineering system evaluation (Appendix O), and robustness evaluation under noisy observations (Appendix P).

4.1 Generalization across Dynamical Systems

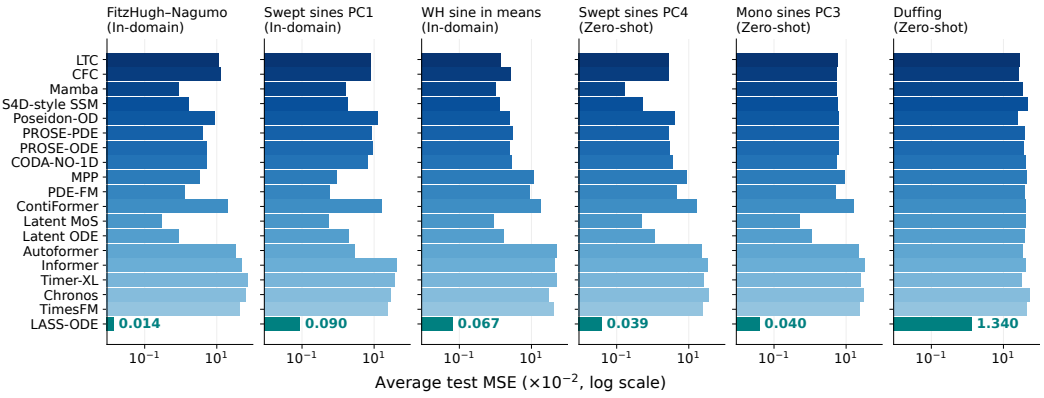

Figure 2: Test MSE ($\times 10^{-2}$) for in-domain and zero-shot ODE systems.

As shown in Fig. 2, we evaluate generalization under two settings: in-domain prediction on known ODE systems with unseen initial conditions and physical parameters, and zero-shot transfer to unseen system types. Complete prediction results are provided in Appendix I and Appendix J. Across diverse in-domain systems, LASS-ODE achieves the lowest prediction error in nearly all evaluated settings, reducing MSE by over 70% compared with existing methods. While generic time-series foundation models capture broad sequence patterns, they often fail to represent system-specific dynamical structures. Recent scientific foundation models, including MPP, PDE-FM, and related physics-informed architectures, incorporate physical priors through operator structures, additional conditioning, or physics-based objectives, but are primarily developed for specific physical domains and often require explicit physical information. ODE-based models provide stronger continuous-time dynamical modeling, but typically remain system-specific without transferable representations across heterogeneous systems. For many zero-shot test systems such as Swept sines PC4 and Monosines

PC3, LASS-ODE achieves accurate predictions. In more challenging nonlinear systems, such as Duffing dynamics, LASS-ODE still provides a competitive zero-shot prediction and rapidly improves through lightweight LoRA fine-tuning. After approximately 10 epochs of adaptation, LASS-ODE accurately recovers the target dynamics, demonstrating efficient adaptation to previously unseen physical systems (Fig. 3). Additional fine-tuning results are provided in Appendix K.

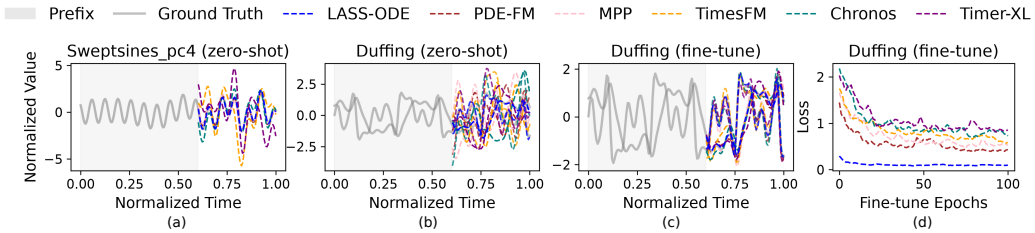

Figure 3: (a) Representative zero-shot result with strong performance. (b) Representative zero-shot result with comparatively weaker performance. (c) Fine-tuned result. (d) Loss curve in fine-tuning.

4.2 Effectiveness of Fixed-point Reasoning for Dynamic Structure Discovery

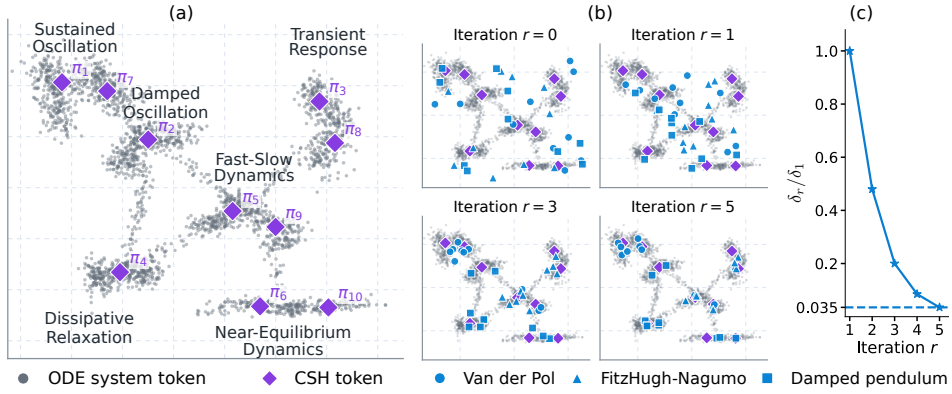

Figure 4: **(a)** 2D visualization of ODE system tokens and CSH tokens ($k_{\text{CSH}} = 10$). **(b)** Fixed-point reasoning of test system tokens at iterations $r = 0, 1, 3, 5$. **(c)** Normalized convergence of the token updates. For iteration r, $\delta_r = \left\|E^{(r+1)} - E^{(r)}\right\|_{\text{F}}$, and the plotted quantity is δ_r/δ_1.

As shown in panel (a) of Fig. 4, the CSH tokens, clustered from E^* in Eq. (9), form distinct clusters with clear dynamical meanings. Panel (b) shows the fixed-point reasoning process in the representation space. We test new systems (Van der Pol, FitzHugh–Nagumo, and damped pendulum) and visualize the iterations of $E^{(r)}$ in Eq. (3). Their representative token representations gradually converge toward the corresponding dynamical clusters, showing that fixed-point reasoning can effectively capture the underlying physical behaviors. Panel (c) shows the decreasing normalized update magnitude δ_r/δ_1, indicating convergence. More dynamical interpretations are provided in Appendix H. Additional visualizations of ODE trajectory reasoning are shown in Fig. 19 in Appendix M.

Table 1: Real-world dynamical-system results. MSE values are reported as $(\times 10^{-2})$.

System	LASS-ODE	TimesFM	Latent ODE	Informer	ContiFormer	PDE-FM	MPP	CoDA-NO 1D	PROSE ODE	PROSE PDE	Poseidon 0D	S4D-style SSM	Mamba	CfC	LTC
Frequency	0.35	16.98	7.94	42.77	35.53	8.85	10.17	9.563	9.592	10.185	9.121	5.291	2.390	8.403	8.730
Voltage	0.18	11.45	3.88	6.98	2.47	5.31	8.62	5.871	5.791	6.030	7.397	3.565	1.864	5.520	7.218

4.3 Tests on Real-World Dynamical Systems

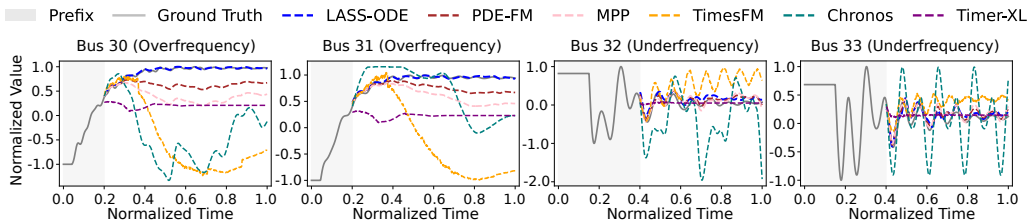

Figure 5: Realistic power system frequency prediction results.

We further evaluate LASS-ODE on power system frequency dynamics and AI data center voltage dynamics. The results in Table 1 and Fig. 5 show that LASS-ODE consistently outperforms representative baselines and accurately reconstructs transient dynamics. Detailed dataset descriptions, evaluation protocols, and case settings are provided in Appendix O.

4.4 Effectiveness of Automatic Stability-Adaptation Rollout

We also evaluate the adaptive learning of the local growth/decay rates in Eq. (7), where negative rates are favored for stable rollout while positive rates are allowed when locally expansive dynamics are required. The top panel of Fig. 6 shows the learned $\alpha_k^{\max}$ across representative systems and temporal tokens, where positive values appear only in limited locally expansive regimes. We further inject equal-magnitude random perturbations with different directions and track their propagation during rollout, as shown in the bottom panel. Although some perturbations exhibit temporary amplification, the errors eventually decrease for most systems, indicating stabilized rollout errors.

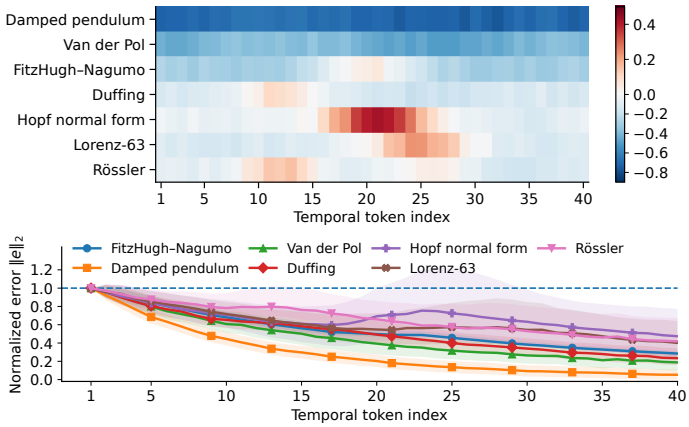

Figure 6: **Top:** Learned $\alpha_k^{\max}$ across systems and temporal tokens. **Bottom:** Normalized perturbation error under random perturbation directions.

4.5 Ablation Study, Inference Time, and Sensitivity Analysis

Following the pipeline in Fig. 1 (left to right), we conduct ablations that: (1) use variable-dependent processing; (2) replace GRU and time modulation with an MLP encoder; (3) use Fourier feature (FF) time encoding; (4) use rotary positional embeddings (RoPE) for time encoding; (5) remove state-variable encoding; (6) remove Fixed-Point Transformer; (7) remove the CSH module; (8) replace MoE with a single MLP; and (9) replace the piecewise-linear (PWL) evolution in Eq. (5) with a nonlinear MLP. Fig. 7 (a) summarizes the ablation results. The largest degradation comes from removing the Fixed-Point Transformer or enforcing variable-dependent processing, both increasing the error by 2.6-2.7×. The temporal encoding design is also important: replacing the GRU and time modulation with an MLP encoder, or replacing RBF time encoding with FF or RoPE, leads to roughly 2–3× higher error. Removing state-variable encoding, CSH, or MoE further degrades performance by about 1.5–2.2×. Finally, replacing the piecewise-linear evolution in Eq. (5) with a nonlinear MLP slightly worsens accuracy while increasing training time by more than 15×, supporting the efficiency of our local linearization.

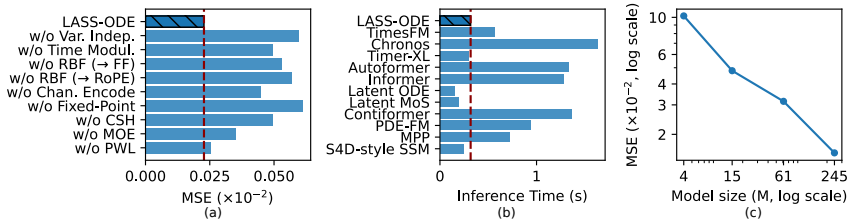

Figure 7: **(a)** Ablation results. **(b)** Inference time. **(c)** Scaling result.

In Fig. 7, panel (b) reports the inference time for predicting 16 trajectories across different ODE systems. Thanks to the sparse MoE and the token-wise linear ODE decoder, LASS-ODE achieves moderate latency, capable of real-time ODE prediction. We further evaluate zero-shot scaling by increasing the model size from 4M to 245M parameters. As shown in panel (c), the trend demonstrates that LASS-ODE benefits from model-size scaling in the zero-shot setting. For sensitivity analysis, we vary K_{CSH}, the MoE expert count, the latent dimension d_z of $\boldsymbol{z}$ in Eq. (5), and the number of tokens per trajectory K_{token}, which together control model

capacity. Fig. 8 summarizes the results. The test MSE generally decreases as these hyperparameters increase. We also found that beyond a moderate point ($K_{\text{CSH}} \geq 10$, #experts ≥ 8, $d_z \geq 15$, and $K_{\text{token}} \geq 40$), further increases in the corresponding hyperparameters do not substantially improve predictive accuracy.

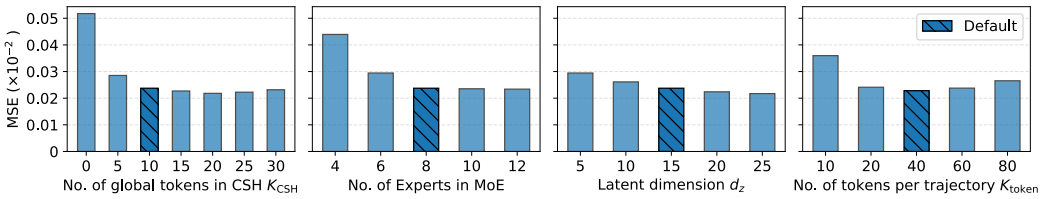

Figure 8: Sensitivity analysis results.

5 Related Work

Dynamical Learning. Symbolic regression, physics-informed learning, Neural ODE families, and time-series foundation models are introduced in Section 1.

Physics-Informed Foundation Models. Recent scientific foundation models incorporate physics through several complementary mechanisms. Operator-structured models encode physical evolution into the architecture or training process, as in Poseidon, which exploits the semigroup structure of time-dependent PDEs (Herde et al., 2024), and CoDA-NO, which extends attention and representation learning to function spaces across multiphysics systems (Rahman et al., 2024). Physics-conditioned models instead provide governing equations or physical information as model inputs, including PROSE-PDE (Sun et al., 2025) and PDE-FM (Soares et al., 2025). Physics can also be introduced through training objectives: PI-MFM constructs PDE-residual losses from symbolic equations (Zhu et al., 2025b), while PITA derives physics-informed temporal alignment directly from observed trajectories (Zhu et al., 2025a). Collectively, these approaches establish physical fidelity through architectural structure, supplied physical information, or learning objectives. When the governing dynamics are unknown and must be inferred jointly with the representation, however, a principled representation-level target for physical fidelity remains unclear.

Recurrent and Equilibrium Transformers. Prior recurrent Transformer designs have evolved along three related directions. First, Universal and looped Transformers introduce recurrence to provide the iterative inductive bias that standard Transformers inherently lack (Dehghani et al., 2019; Giannou et al., 2023; Yang et al., 2024). Second, deep equilibrium models (DEQs) formulate recurrent computation as a fixed point to obtain an implicit infinite-depth representation (Bai et al., 2019); however, the resulting equilibrium is computational and carries no explicit physical meaning. Third, a recent trend in recurrent-depth Transformers interprets repeated latent computation as reasoning, increasing reasoning depth and test-time compute without longer token sequences (Saunshi et al., 2025; Geiping et al., 2025). This trend further supports the potential of iterative computation in latent representation space as a reasoning mechanism. Together, these advances bring recurrent Transformers close to our notion of fixed-point reasoning, but they still lack an explicit physics-grounded reasoning loop that iteratively feeds physical estimates back to refine both the representation and subsequent estimation toward a self-consistent equilibrium.

6 Conclusion, Limitation, and Future Work

We introduced LASS-ODE for learning optimal physical representations across heterogeneous dynamical systems. LASS-ODE performs physics-grounded fixed-point reasoning by recurrently estimating the underlying physics and feeding it back to refine the representation toward a self-consistent equilibrium. We further developed structured local-operator learning with adaptive stability and cross-system knowledge transfer through recurring dynamical structures. Experiments across diverse ODE systems demonstrate accurate prediction, zero-shot generalization, stable long-horizon rollout, and progressive fixed-point refinement. A current limitation is that variable-wise processing does not explicitly characterize the coupling structure among state variables in a physical system. A natural extension is to infer a state-interaction graph G and construct a structural encoding $P(G)$, analogous to the physics-aware encoding $P(\widehat{\boldsymbol{\theta}})$ in Eq. (8), to feed structural positions and distances back into the representations. This preserves the Fixed-Point Transformer principle while extending the task from dynamical estimation to structural coupling discovery and joint physical reasoning.

AI USE STATEMENT

We used generative AI tools to polish sentence clarity and grammar, correct typographical errors, check notation consistency, and assist in checking the integrity of theorem proofs. All technical content, derivations, and conclusions were developed and verified by the authors.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, pp. 690–701, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11(2):e0150171, 2016a. doi: 10.1371/journal.pone.0150171.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016b.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36:47143–47175, 2023.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

Jan Decuyper, Tim De Troyer, and M. C. Runacres. Canonical systems for unsteady fluid mechanics: A new family of nonlinear benchmarks. Technical report, FLOW, Vrije Universiteit Brussel, Belgium, 2024. URL `https://www.nonlinearbenchmark.org/benchmarks/unsteady-fluid-mechanics`.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Jingyu Dou, Wentao Lyu, Feng Chen, Qing Nie, and Chunhe Li. Inferring stochastic dynamics by biophysical neural ode using single-cell transcriptomics. *Nature Communications*, 2026.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2206.11893.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7657–7666, 2021. doi: 10.1609/aaai.v35i9.16936. URL https://arxiv.org/abs/2006.04439.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4:992–1003, 2022. doi: 10.1038/s42256-022-00556-7. URL https://arxiv.org/abs/2106.13898.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. In *Advances in Neural Information Processing Systems*, volume 37, pp. 72525–72624, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

Haoran Li, Yang Weng, and Hanghang Tong. Console: Convex neural symbolic learning. *Advances in Neural Information Processing Systems*, 35:5766–5774, 2022.

Haoran Li, Chenhan Xiao, Muhao Guo, and Yang Weng. Latent mixture of symmetries for sample-efficient dynamic learning. *arXiv preprint arXiv:2510.03578*, 2025.

Haoran Li, Muhao Guo, Yang Weng, and Hanghang Tong. Neural predictive control to coordinate discrete-and continuous-time models for time-series analysis with control-theoretical improvements. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 669–680, 2026.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024a.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. *arXiv preprint arXiv:2410.04803*, 2024b.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *arXiv preprint arXiv:2502.00816*, 2025.

Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. PROSE: Predicting multiple operators and symbolic expressions using multimodal transformers. *Neural Networks*, 180:106707, 2024c. doi: 10.1016/j.neunet.2024.106707.

Edward N Lorenz. Deterministic nonperiodic flow 1. In *Universality in Chaos, 2nd edition*, pp. 367–378. Routledge, 2017.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9:4950, 2018. doi: 10.1038/s41467-018-07210-0.

Lihao Mai, Haoran Li, Yang Weng, Erik Blasch, and Xiaodong Zheng. Guaranteed conversion from static measurements into dynamic ones based on manifold feature interpolation. *IEEE Transactions on Power Systems*, 40(5):3763–3777, 2025. doi: 10.1109/TPWRS.2025.3540724.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2401–2409. PMLR, 2017.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Jean-Philippe Noël, Maarten Schoukens, and Koen Tiels. Nonlinear benchmarks. `https://www.nonlinearbenchmark.org/benchmarks`, 2025.

Zhaozhi Qian, William Zame, Lucas Fleuren, Paul Elbers, and Mihaela van der Schaar. Integrating expert odes into neural odes: pharmacology and disease progression. *Advances in Neural Information Processing Systems*, 34:11364–11383, 2021.

Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics pdes. In *Advances in Neural Information Processing Systems*, volume 37, pp. 104035–104064, 2024.

Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers. *arXiv preprint arXiv:1910.01719*, 2019.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

Maarten Schoukens and Jean Philippe Noël. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC-PapersOnLine*, 50(1):446–451, 2017.

Maarten Schoukens, Anna Marconato, Rik Pintelon, Gerd Vandersteen, and Yves Rolain. Parametric identification of parallel wiener–hammerstein systems. *Automatica*, 51:111–122, 2015.

Eduardo Soares, Emilio Vital Brazil, Victor Shirasuna, Breno W. S. R. de Carvalho, and Cristiano Malossi. Towards a foundation model for partial differential equations across physics domains. *arXiv preprint arXiv:2511.21861*, 2025.

Gustaf Söderlind. The logarithmic norm: History and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006. doi: 10.1007/s10543-006-0069-9.

Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Physical Review E*, 111(3): 035304, 2025. doi: 10.1103/PhysRevE.111.035304.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Congcong Zhu, Xiaoyan Xu, Jiayue Han, and Jingrun Chen. Physics-informed temporal alignment for auto-regressive PDE foundation models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80223–80258. PMLR, 2025a.

Min Zhu, Jingmin Sun, Zecheng Zhang, Hayden Schaeffer, and Lu Lu. PI-MFM: Physics-informed multimodal foundation model for solving partial differential equations. *arXiv preprint arXiv:2512.23056*, 2025b.

A Computational Complexity of LASS-ODE Rollout

We provide a detailed complexity comparison between our proposed LASS-ODE and Latent ODE (Chen et al., 2018). We focus on the ODE rollout stage after the token representation $\boldsymbol{e}_k$ is obtained; the representation-learning cost shared by the overall architecture is therefore excluded from the comparison.

LASS-ODE. For each temporal token k, LASS-ODE estimates the local dynamics $(A_k, \boldsymbol{b}_k)$ only once from $\boldsymbol{e}_k$. Consider an N_{Est}-layer Estimator with hidden width K_{width}. The output contains B_k, L_Q Householder vectors, the quantities for constructing $\boldsymbol{\alpha}_k$, and $\boldsymbol{b}_k$, with output dimension

$$d_{\text{out}} = d_z^2 + (L_Q + 2)d_z + 1. \tag{11}$$

The corresponding one-time estimation cost is

$$C_{\text{Est}} = d_{\text{model}} K_{\text{width}} + (N_{\text{Est}} - 2)K_{\text{width}}^2 + K_{\text{width}} d_{\text{out}}. \tag{12}$$

After the local operator has been estimated, each ODE evaluation only computes the affine dynamics $A_k \boldsymbol{z} + \boldsymbol{b}_k$. Using the structured form of A_k, the orthogonal transformations can be applied implicitly through the L_Q Householder vectors without explicitly constructing Q_k. The resulting cost is

$$C_{\text{lin}} = \mathcal{O}\left(d_z^2 + L_Q d_z\right) = \mathcal{O}\left(d_z^2\right), \tag{13}$$

where the last equality holds for a fixed small L_Q. Therefore, for n_{step} ODE evaluations, the total rollout cost is

$$C_{\text{LASS}} = \mathcal{O}\left(C_{\text{Est}} + n_{\text{step}} C_{\text{lin}}\right). \tag{14}$$

Latent ODE. In contrast, a Latent ODE evaluates a nonlinear neural vector field at every ODE step. For a conditioned N_{MLP}-layer MLP with hidden width K_{width} and input $[\boldsymbol{z}; \boldsymbol{e}_k]$, one vector-field evaluation requires

$$C_{\text{MLP}} = (d_z + d_{\text{model}})K_{\text{width}} + (N_{\text{MLP}} - 2)K_{\text{width}}^2 + K_{\text{width}} d_z. \tag{15}$$

Here, N_{Est} and N_{MLP} denote the numbers of layers in the Estimator and the Latent ODE vector-field MLP, respectively.

Since this nonlinear network must be repeatedly evaluated throughout integration, the total cost is

$$C_{\text{LatentODE}} = \mathcal{O}\left(n_{\text{step}} C_{\text{MLP}}\right). \tag{16}$$

Table 2 summarizes the difference. The key advantage is that LASS-ODE pays the neural estimation cost only once per token interval, whereas Latent ODE repeatedly pays the nonlinear MLP cost at every ODE evaluation.

Table 2: Computational complexity of ODE rollout per token interval.

Method	One-time cost	Per ODE evaluation	Total cost
LASS-ODE	C_{Est}	$\mathcal{O}(d_z^2 + L_Q d_z)$	$\mathcal{O}(C_{\text{Est}} + n_{\text{step}} C_{\text{lin}})$
Latent ODE	–	C_{MLP}	$\mathcal{O}(n_{\text{step}} C_{\text{MLP}})$

The computational advantage becomes increasingly significant as the number of ODE evaluations grows. In particular, the relative complexity ratio is

$$\frac{C_{\text{LatentODE}}}{C_{\text{LASS}}} = \frac{n_{\text{step}} C_{\text{MLP}}}{C_{\text{Est}} + n_{\text{step}} C_{\text{lin}}}, \tag{17}$$

which approaches

$$\frac{C_{\text{MLP}}}{C_{\text{lin}}} = \mathcal{O}\left(\frac{(d_z + d_{\text{model}})K_{\text{width}} + (N_{\text{MLP}} - 2)K_{\text{width}}^2 + K_{\text{width}} d_z}{d_z^2 + L_Q d_z}\right) \tag{18}$$

as n_{step} increases. When $K_{\text{width}} \gg d_z$, which is typical when a high-capacity neural vector field is used, the nonlinear MLP evaluation is substantially more expensive than the local affine evaluation. The advantage is further amplified for adaptive or high-accuracy ODE solvers that require many vector-field evaluations. In addition, the local affine dynamics across tokens and samples can be evaluated in parallel using batched linear ODE solvers, further improving practical rollout efficiency.

B PROOF OF CONVERGENCE

We introduce the following theorem to prove that our Fixed-Point Transformer is contractive.

Theorem 1 (Existence, uniqueness, and convergence of the Fixed-Point Transformer)**.** *Consider the recurrent Estimator–Refiner update in Eq. (3) for a fixed input representation $E^{(0)}$ and a fixed token configuration. Let $\mathcal{H}$ be a nonempty, closed, convex, and bounded representation space. Suppose that the Estimator, physics-aware encoding, and attention operations are composed of Lipschitz operations on $\mathcal{H}$, and that the architectural conditions in Eq. (4) are enforced. Then the undamped candidate update $E^{(r)} \to \widehat{E}^{(r+1)}$ admits a finite Lipschitz constant L.*

If the anchoring coefficient satisfies

$$0 < \rho < \frac{1}{\max\{1, L\}}, \tag{19}$$

then the complete recurrent update is contractive with contraction factor

$$\kappa \leq \rho L < 1. \tag{20}$$

Consequently, there exists a unique equilibrium $(E^\star, \widehat{\boldsymbol{\theta}}^\star)$ conditioned on $E^{(0)}$, and the recurrent representations converge geometrically:

$$\left\| E^{(r)} - E^\star \right\|_{\mathrm{F}} \leq \kappa^r \left\| E^{(0)} - E^\star \right\|_{\mathrm{F}}. \tag{21}$$

Moreover, $\widehat{\boldsymbol{\theta}}^{(r)} \to \widehat{\boldsymbol{\theta}}^\star = \mathrm{Est}(E^\star)$, thereby reaching the learn–estimate equilibrium in Eq. (1).

Proof. We first establish that the undamped candidate update $E^{(r)} \mapsto \widehat{E}^{(r+1)}$ has a finite Lipschitz constant over the admissible representation space $\mathcal{H}$.

The first condition in Eq. (4) restricts every recurrent representation to $\mathcal{H}$. Since $\mathcal{H}$ is closed and convex, the Euclidean projection onto $\mathcal{H}$ is nonexpansive. Therefore, for any E_1, E_2,

$$\left\| \mathrm{Proj}_{\mathcal{H}}(E_1) - \mathrm{Proj}_{\mathcal{H}}(E_2) \right\|_{\mathrm{F}} \leq \left\| E_1 - E_2 \right\|_{\mathrm{F}}. \tag{22}$$

We next consider the learned transformations within one recurrent Estimator–Refiner iteration. The second condition in Eq. (4) bounds the trainable LayerNorm scaling factors by $\gamma_{\mathrm{LN}}^{\max}$. Together with the positive numerical stabilization constant used in LayerNorm, this ensures that each LayerNorm operation has a finite Lipschitz constant on $\mathcal{H}$.

The third condition in Eq. (4) rescales every learned weight matrix W such that

$$\sigma_{\max}(W) \leq s_W. \tag{23}$$

Hence, the learned linear transformations in the attention, MLP, physics-aware encoding, and Estimator modules all have bounded operator norms.

Since $E^{(r)} \in \mathcal{H}$ and $\mathcal{H}$ is bounded, the recurrent token representations are bounded. Together with Eq. (23), this implies that the query, key, and value representations in multi-head attention remain bounded. The attention logits therefore have finite sensitivity to their inputs, and the softmax operation is Lipschitz in finite dimensions. Consequently, the complete attention operation admits a finite Lipschitz constant on $\mathcal{H}$.

Similarly, the MLP block admits a finite Lipschitz constant because its linear transformations have bounded spectral norms and GELU is Lipschitz. By assumption, the Estimator and physics-aware encoding are also composed of Lipschitz operations on $\mathcal{H}$. Therefore, the feedback path

$$E^{(r)} \to \widehat{\boldsymbol{\theta}}^{(r)} = \mathrm{Est}\left(E^{(r)}\right) \to P\left(\widehat{\boldsymbol{\theta}}^{(r)}\right) \tag{24}$$

has a finite Lipschitz constant.

The undamped candidate $\widehat{E}^{(r+1)}$ in Eq. (3) consists of a finite composition and sum of these Lipschitz operations. Therefore, there exists a finite constant L such that, for any $E_1, E_2 \in \mathcal{H}$,

$$\left\| \widehat{E}_1^+ - \widehat{E}_2^+ \right\|_{\mathrm{F}} \leq L \left\| E_1 - E_2 \right\|_{\mathrm{F}}, \tag{25}$$

where $\widehat{E}_1^+$ and $\widehat{E}_2^+$ denote the corresponding undamped candidate updates.

We now consider the complete recurrent update. For the same fixed input anchor $E^{(0)}$, enforcing the bounded-space condition gives

$$E_i^+ = \mathrm{Proj}_{\mathcal{H}}\left[(1-\rho)E^{(0)} + \rho\widehat{E}_i^+\right], \qquad i \in \{1,2\}. \tag{26}$$

Using the nonexpansiveness in Eq. (22),

$$\begin{aligned}\left\|E_1^+ - E_2^+\right\|_{\mathrm{F}} &\le \left\|\rho\widehat{E}_1^+ - \rho\widehat{E}_2^+\right\|_{\mathrm{F}} \\ &= \rho\left\|\widehat{E}_1^+ - \widehat{E}_2^+\right\|_{\mathrm{F}} \\ &\le \rho L\left\|E_1 - E_2\right\|_{\mathrm{F}}.\end{aligned} \tag{27}$$

The fixed anchor $E^{(0)}$ cancels when comparing two recurrent updates. Thus, it preserves the information extracted from the input observations without increasing the recurrent amplification.

Under Eq. (19), we have $\rho L < 1$. Therefore, the complete recurrent update is a contraction with $\kappa \le \rho L < 1$, establishing Eq. (20).

Since $\mathcal{H}$ is closed in a finite-dimensional Euclidean space, it is complete under the Frobenius norm. The Banach fixed-point theorem therefore guarantees the existence and uniqueness of a fixed point $E^\star$ conditioned on $E^{(0)}$. Moreover, repeated application of the recurrent update converges geometrically:

$$\left\|E^{(r)} - E^\star\right\|_{\mathrm{F}} \le \kappa^r\left\|E^{(0)} - E^\star\right\|_{\mathrm{F}}, \tag{28}$$

which gives Eq. (21).

Finally, the Estimator is Lipschitz and therefore continuous. Hence, $E^{(r)} \to E^\star$ implies

$$\widehat{\boldsymbol{\theta}}^{(r)} = \mathrm{Est}\left(E^{(r)}\right) \to \mathrm{Est}\left(E^\star\right) = \widehat{\boldsymbol{\theta}}^\star. \tag{29}$$

Therefore, $(E^\star, \widehat{\boldsymbol{\theta}}^\star)$ is the unique input-conditioned equilibrium satisfying the learn–estimate consistency in Eq. (1), completing the proof. □

C Proof of Approximation Capability

We introduce the following theorem to characterize the approximation capacity of lifting with token-wise linearization for nonlinear dynamics. Theorem 2 decomposes the trajectory approximation error $\|\boldsymbol{z}(\tau) - \widehat{\boldsymbol{z}}_k(\tau)\|_2$ into three sources: the first term captures the error from approximating the lifted nonlinear dynamics by time-varying affine dynamics, the second captures the tokenization error from representing the time-varying dynamics by token-wise constant operators, and the third captures the estimation error between the learned operators $(A_k, \boldsymbol{b}_k)$ and their ideal counterparts. The bound is conditional and applies within one token: its tokenization contribution decreases with finer temporal discretization when the variation bounds remain uniform, while the other contributions depend on the affine-approximation residual and operator-estimation accuracy.

Theorem 2 (Approximation capacity of token-wise latent linearization)**.** *Let $\boldsymbol{z}(\tau) \in \mathbb{R}^{d_z}$ be a reference latent trajectory in a d_z-dimensional latent space used in Eq. (5) that satisfies $\dot{\boldsymbol{z}} = g(\boldsymbol{z}, \tau)$. Here g is the reference latent vector field. Assume that, along the trajectory, there exist a time-varying matrix $A(\tau) \in \mathbb{R}^{d_z \times d_z}$ and vector $\boldsymbol{b}(\tau) \in \mathbb{R}^{d_z}$ such that*

$$g(\boldsymbol{z}(\tau), \tau) = A(\tau)\boldsymbol{z}(\tau) + \boldsymbol{b}(\tau) + \boldsymbol{\delta}_{\mathrm{lift}}(\tau), \qquad \|\boldsymbol{\delta}_{\mathrm{lift}}(\tau)\|_2 \le \varepsilon_{\mathrm{lift}}. \tag{30}$$

where $\boldsymbol{\delta}_{\mathrm{lift}}$ is the bounded residual of the time-varying affine approximation. Here $\varepsilon_{\mathrm{lift}} \ge 0$ is the assumed uniform bound on its norm. Consider the k-th token interval $I_k = [\tau_k, \tau_{k+1}]$, whose width is $\Delta_k = \tau_{k+1} - \tau_k > 0$. Assume that, throughout this interval,

$$\|\dot{A}(\tau)\|_2 \le L_k^{(A)}, \qquad \|\dot{\boldsymbol{b}}(\tau)\|_2 \le L_k^{(b)}, \qquad \|\boldsymbol{z}(\tau)\|_2 \le L_k^{(z)}. \tag{31}$$

The nonnegative constants $L_k^{(A)}$, $L_k^{(b)}$, and $L_k^{(z)}$ bound the rates of change of A and $\boldsymbol{b}$ and the norm of the reference trajectory, respectively. Throughout, $\|\cdot\|_2$ is the Euclidean norm for vectors and the induced spectral norm for matrices. Define the interval-averaged coefficients $\bar{A}_k = \frac{1}{\Delta_k}\int_{I_k} A(\tau)\,d\tau$, $\bar{\boldsymbol{b}}_k = \frac{1}{\Delta_k}\int_{I_k} \boldsymbol{b}(\tau)\,d\tau$. Let A_k and $\boldsymbol{b}_k$ be the learned, constant affine coefficients assigned to this token. Let $\widehat{\boldsymbol{z}}_k(\tau)$ be the resulting approximate trajectory, defined by $\dot{\widehat{\boldsymbol{z}}}_k = A_k\widehat{\boldsymbol{z}}_k + \boldsymbol{b}_k$, $\widehat{\boldsymbol{z}}_k(\tau_k) = \boldsymbol{z}(\tau_k)$. Then the mismatch at every time in the token interval satisfies

$$\sup_{\tau\in I_k} \|\boldsymbol{z}(\tau) - \widehat{\boldsymbol{z}}_k(\tau)\|_2 \le e^{\|A_k\|_2\Delta_k}\Bigg[\underbrace{\varepsilon_{\mathrm{lift}}\Delta_k}_{\text{affine-approximation residual}} + \underbrace{\frac{1}{2}\big(L_k^{(z)}L_k^{(A)} + L_k^{(b)}\big)\Delta_k^2}_{\text{tokenization error}} + \underbrace{\big(L_k^{(z)}\|A_k - \bar{A}_k\|_2 + \|\boldsymbol{b}_k - \bar{\boldsymbol{b}}_k\|_2\big)\Delta_k}_{\text{operator estimation error}}\Bigg]. \tag{32}$$

Proof. Since $\|\dot{A}(\tau)\|_2 \le L_k^{(A)}$ for all $\tau \in I_k$, $A(\tau)$ is Lipschitz continuous on I_k. Hence, for any $\tau \in I_k$,

$$\|A(\tau) - \bar{A}_k\|_2 = \left\|\frac{1}{\Delta_k}\int_{I_k}\big(A(\tau) - A(s)\big)\,ds\right\|_2 \le \frac{L_k^{(A)}}{\Delta_k}\int_{I_k}|\tau - s|\,ds \le \frac{1}{2}L_k^{(A)}\Delta_k. \tag{33}$$

Similarly, $\|\dot{\boldsymbol{b}}(\tau)\|_2 \le L_k^{(b)}$ implies

$$\|\boldsymbol{b}(\tau) - \bar{\boldsymbol{b}}_k\|_2 \le \frac{1}{2}L_k^{(b)}\Delta_k. \tag{34}$$

For the error dynamics relative to the learned token operator, let $\xi_k(\tau) := \boldsymbol{z}(\tau) - \widehat{\boldsymbol{z}}_k(\tau)$. By construction, $\xi_k(\tau_k) = 0$. Using $\dot{\boldsymbol{z}}(\tau) = A(\tau)\boldsymbol{z}(\tau) + \boldsymbol{b}(\tau) + \boldsymbol{\delta}_{\mathrm{lift}}(\tau)$ and $\dot{\widehat{\boldsymbol{z}}}_k(\tau) = A_k\widehat{\boldsymbol{z}}_k(\tau) + \boldsymbol{b}_k$, we obtain

$$\dot{\xi}_k(\tau) = A_k\xi_k(\tau) + \big(A(\tau) - A_k\big)\boldsymbol{z}(\tau) + \big(\boldsymbol{b}(\tau) - \boldsymbol{b}_k\big) + \boldsymbol{\delta}_{\mathrm{lift}}(\tau). \tag{35}$$

Decompose

$$A(\tau) - A_k = \big(A(\tau) - \bar{A}_k\big) + \big(\bar{A}_k - A_k\big), \tag{36}$$

$$\boldsymbol{b}(\tau) - \boldsymbol{b}_k = \big(\boldsymbol{b}(\tau) - \bar{\boldsymbol{b}}_k\big) + \big(\bar{\boldsymbol{b}}_k - \boldsymbol{b}_k\big). \tag{37}$$

Using $\|\boldsymbol{z}(\tau)\|_2 \le L_k^{(z)}$, $\|\boldsymbol{\delta}_{\mathrm{lift}}(\tau)\|_2 \le \varepsilon_{\mathrm{lift}}$, and Eqs. (33)–(34), we have

$$\begin{aligned}&\big\|\big(A(\tau)-A_k\big)\boldsymbol{z}(\tau)+\big(\boldsymbol{b}(\tau)-\boldsymbol{b}_k\big)+\boldsymbol{\delta}_{\mathrm{lift}}(\tau)\big\|_2\\&\quad\le \varepsilon_{\mathrm{lift}}+\frac{1}{2}\left(L_k^{(z)}L_k^{(A)}+L_k^{(b)}\right)\Delta_k+L_k^{(z)}\|A_k-\bar{A}_k\|_2+\|\boldsymbol{b}_k-\bar{\boldsymbol{b}}_k\|_2.\end{aligned}\tag{38}$$

The variation-of-constants formula applied to Eq. (35) gives

$$\xi_k(\tau)=\int_{\tau_k}^{\tau} e^{A_k(\tau-s)}\boldsymbol{\zeta}_k(s)\,ds,\tag{39}$$

where $\boldsymbol{\zeta}_k(s) := \big(A(s)-A_k\big)\boldsymbol{z}(s)+\big(\boldsymbol{b}(s)-\boldsymbol{b}_k\big)+\boldsymbol{\delta}_{\mathrm{lift}}(s)$. Since A_k is constant over the kth token, its state-transition matrix satisfies

$$\|e^{A_k(\tau-s)}\|_2 \le e^{\|A_k\|_2(\tau-s)} \le e^{\|A_k\|_2\Delta_k}, \qquad \tau_k \le s \le \tau \le \tau_{k+1}.\tag{40}$$

Combining Eqs. (38)–(40) and using $\tau-\tau_k \le \Delta_k$ yields

$$\begin{aligned}\|\xi_k(\tau)\|_2 \le e^{\|A_k\|_2\Delta_k}\Big[&\varepsilon_{\mathrm{lift}}\Delta_k+\frac{1}{2}\left(L_k^{(z)}L_k^{(A)}+L_k^{(b)}\right)\Delta_k^2\\&+\left(L_k^{(z)}\|A_k-\bar{A}_k\|_2+\|\boldsymbol{b}_k-\bar{\boldsymbol{b}}_k\|_2\right)\Delta_k\Big].\end{aligned}\tag{41}$$

Since the right-hand side of Eq. (41) is independent of τ, taking the supremum over $\tau \in I_k$ yields the bound in Theorem 2. □

Notation and scope. In Eq. (5), $\boldsymbol{z}$ denotes the model-generated latent trajectory, whereas in Theorems 2 and 3 it denotes a reference latent trajectory. $\widehat{\boldsymbol{z}}_k$ denotes a trajectory propagated by the learned operator. Theorem 2 initializes both trajectories at the same token-start state to isolate the error generated within that token. This bound does not include error carried over from earlier tokens.

The error bound in Theorem 2 separates three sources of within-token approximation error. The first is the residual of approximating the reference latent dynamics by a time-varying affine system. This residual can be small under mild conditions. For example, if the reference latent coordinates span a finite-dimensional Koopman-invariant subspace, their evolution is exactly linear and the affine residual vanishes (Brunton et al., 2016a). More locally, if the latent vector field has bounded second derivatives and the trajectory remains close to a reference state, a first-order affine approximation has a residual that decreases quadratically with the size of that neighborhood. Learned Koopman embeddings have also demonstrated approximately linear evolution for particular nonlinear systems (Lusch et al., 2018). The second term measures the error from replacing smoothly varying affine dynamics with one operator per token and decreases quadratically with token width under the stated assumptions. The third measures the discrepancy between the learned operator and its ideal token-level counterpart. Thus, when the affine residual and operator-estimation error are small, within-token variation is bounded, and the amplification factor remains controlled, the theorem guarantees a small trajectory mismatch within a token for trajectories initialized at the same state.

D Proof of Local Perturbation Dynamics and Stability

We provide the proof by analyzing the perturbation dynamics induced by the structured local operator in $A_k = \frac{A_k - A_k^\top}{2} + \frac{A_k + A_k^\top}{2} = \Omega_k + Q_k \operatorname{diag}(\alpha_{k,1}, \ldots, \alpha_{k,d_z}) Q_k^\top$. Unlike the matching-initial-state comparison in Theorem 2, the learned trajectory in the following theorem is the restriction of the uninterrupted model rollout to token k. Its token-start state may differ from the reference state.

Theorem 3 (Directional perturbation growth and local stability)**.** *Consider the learned token-wise dynamics $\dot{\widehat{\boldsymbol{z}}}_k(\tau) = A_k \widehat{\boldsymbol{z}}_k(\tau) + \boldsymbol{b}_k$ and the reference latent dynamics represented as $\dot{\boldsymbol{z}}(\tau) = A_k \boldsymbol{z}(\tau) + \boldsymbol{b}_k + \boldsymbol{r}_k(\tau),\ \tau \in [\tau_k, \tau_{k+1})$, where $\boldsymbol{r}_k(\tau)$ denotes the local approximation/model-mismatch residual. Under the assumptions of Theorem 2, this residual equals $\boldsymbol{r}_k(\tau) = (A(\tau) - A_k)\boldsymbol{z}(\tau) + (\boldsymbol{b}(\tau) - \boldsymbol{b}_k) + \boldsymbol{\delta}_{\mathrm{lift}}(\tau)$. Define the rollout error $\xi_k(\tau) := \boldsymbol{z}(\tau) - \widehat{\boldsymbol{z}}_k(\tau)$. Under the decomposition in $A_k = \frac{A_k - A_k^\top}{2} + \frac{A_k + A_k^\top}{2} = \Omega_k + Q_k \operatorname{diag}(\alpha_{k,1}, \ldots, \alpha_{k,d_z}) Q_k^\top$, where $\Omega_k^\top = -\Omega_k$, Q_k is orthogonal, and $\alpha_{k,m}$ are the eigenvalues of $(A_k + A_k^\top)/2$, its energy satisfies*

$$\frac{d}{d\tau} \frac{1}{2} \|\xi_k(\tau)\|_2^2 = \sum_{m=1}^{d_z} \alpha_{k,m} \big(\boldsymbol{q}_{k,m}^\top \xi_k(\tau)\big)^2 + \xi_k(\tau)^\top \boldsymbol{r}_k(\tau), \tag{42}$$

where $\boldsymbol{q}_{k,m}$ is the m-th column of the orthogonal matrix Q_k. Let $\alpha_k^{\max} := \max_m \alpha_{k,m}$. Then, for any $\tau_k \le \tau_0 \le \tau < \tau_{k+1}$,

$$\|\xi_k(\tau)\|_2 \le \underbrace{e^{\alpha_k^{\max}(\tau - \tau_0)} \|\xi_k(\tau_0)\|_2}_{\textit{existing-error propagation}} + \underbrace{\int_{\tau_0}^{\tau} e^{\alpha_k^{\max}(\tau - s)} \|\boldsymbol{r}_k(s)\|_2 \, ds}_{\textit{model-mismatch accumulation}}. \tag{43}$$

Proof. From $\xi_k(\tau) = \boldsymbol{z}(\tau) - \widehat{\boldsymbol{z}}_k(\tau)$, the error dynamics satisfy

$$\dot{\xi}_k(\tau) = A_k \xi_k(\tau) + \boldsymbol{r}_k(\tau).$$

Using $A_k = \Omega_k + Q_k \operatorname{diag}(\boldsymbol{\alpha}_k) Q_k^\top$, together with $\Omega_k^\top = -\Omega_k$, we have $\xi_k^\top \Omega_k \xi_k = 0$. Therefore,

$$\begin{aligned} \frac{d}{d\tau} \frac{1}{2} \|\xi_k(\tau)\|_2^2 &= \xi_k^\top A_k \xi_k + \xi_k^\top \boldsymbol{r}_k \\ &= \sum_{m=1}^{d_z} \alpha_{k,m} \big(\boldsymbol{q}_{k,m}^\top \xi_k\big)^2 + \xi_k^\top \boldsymbol{r}_k, \end{aligned} \tag{44}$$

which establishes Eq. (42).

Let $\alpha_k^{\max} = \max_m \alpha_{k,m}$. Since Q_k is orthogonal,

$$\xi_k^\top A_k \xi_k \le \alpha_k^{\max} \|\xi_k\|_2^2.$$

Together with Cauchy–Schwarz, $\xi_k^\top \boldsymbol{r}_k \le \|\xi_k\|_2 \|\boldsymbol{r}_k\|_2$, this gives

$$D^+ \|\xi_k(\tau)\|_2 \le \alpha_k^{\max} \|\xi_k(\tau)\|_2 + \|\boldsymbol{r}_k(\tau)\|_2.$$

Here D^+ is the upper right Dini derivative. When $\xi_k(\tau) = 0$, the inequality follows directly from $\dot{\xi}_k(\tau) = \boldsymbol{r}_k(\tau)$; otherwise, it follows by dividing the preceding energy inequality by $\|\xi_k(\tau)\|_2$. Applying the integrating-factor argument over $[\tau_0, \tau] \subseteq [\tau_k, \tau_{k+1})$ yields

$$\|\xi_k(\tau)\|_2 \le e^{\alpha_k^{\max}(\tau - \tau_0)} \|\xi_k(\tau_0)\|_2 + \int_{\tau_0}^{\tau} e^{\alpha_k^{\max}(\tau - s)} \|\boldsymbol{r}_k(s)\|_2 \, ds.$$

□

E Dataset Description

To evaluate LASS-ODE, we combine established nonlinear dynamical systems benchmarks with a large and diverse library of simulated dynamical systems, focusing on systems governed by ordinary differential equations (ODEs).

We first use datasets from the Nonlinear Benchmark repository Noël et al. (2025), which provides standardized real-world and synthetic time-series from nonlinear dynamical systems. For example, from this repository, we include several Wiener-Hammerstein (WH) process benchmarks Schoukens et al. (2015); Schoukens & Noël (2017): the standard WH system, a Parallel WH variant with multiple branches, a version with dominant process noise entering before the nonlinearity, and so on. These datasets exhibit nonlinear temporal dynamics with memory, internal state evolution, and realistic noise.

We also include datasets from other physical domains to broaden dynamical diversity. The Unsteady Fluid Mechanics benchmark Decuyper et al. (2024) is derived from computational fluid dynamics (CFD) simulations of flow past a transversely oscillating cylinder. The dataset contains multiple types of imposed motions (sinesweep, sine, multisine) on the cylinder, and the velocity, pressure, and vorticity of the flow in the wake of the cylinder are measured on a 31 x 31 grid. This dataset introduces high-dimensional, spatially distributed, and strongly nonlinear temporal dynamics.

Beyond these established benchmarks, we construct a large curated library of simulated time-series generated from classical ODE models. The goal of this library is to expose the model to a wide range of dynamical behaviors, regimes, and time scales. This library spans:

- Low-dimensional test equations include Bernoulli, Cauchy-Euler, and Emden-Fowler equations.
- Nonlinear oscillatory systems include the Van der Pol oscillator (across multiple stiffness parameters), damped and undamped pendulum systems, the Duffing oscillator, and the Ermakov-Pinney equation.
- Biological and reaction dynamics include logistic growth, the Lotka-Volterra predator–prey model, the FitzHugh-Nagumo neuron model, and the Hopf normal form.
- Chaotic systems include the Lorenz-63 and Rössler attractors, as well as the higher-dimensional Lorenz-96 model.
- Power system dynamics include swing-equation and droop-controlled grid frequency models.

For each ODE system, we generate multiple trajectories under diverse initial conditions and parameter settings to capture both transient and long-term behaviors. Simulations are sampled at multiple temporal resolutions, ranging from millisecond-scale to second-scale sampling, to encourage robustness across time scales. All trajectories are stored in .CSV format for data loading.

Across all established benchmark datasets and our simulated ODE library, the total data volume used in our experiments is approximately 40 GB, comprising roughly 500 million time points.

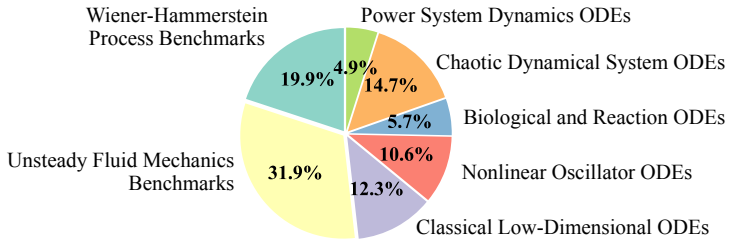

Figure 9: Ratios of data sources used in this work. Example trajectories from the constructed dataset are shown in Fig. 10.

Fig. 10 presents example trajectories from the constructed dataset used in this work. Each panel corresponds to a single dynamical system, and multiple curves within a panel represent different

state dimensions of that system. For high-dimensional systems, we divide the state variables across multiple panels, with each panel showing two or three dimensions for visual clarity.

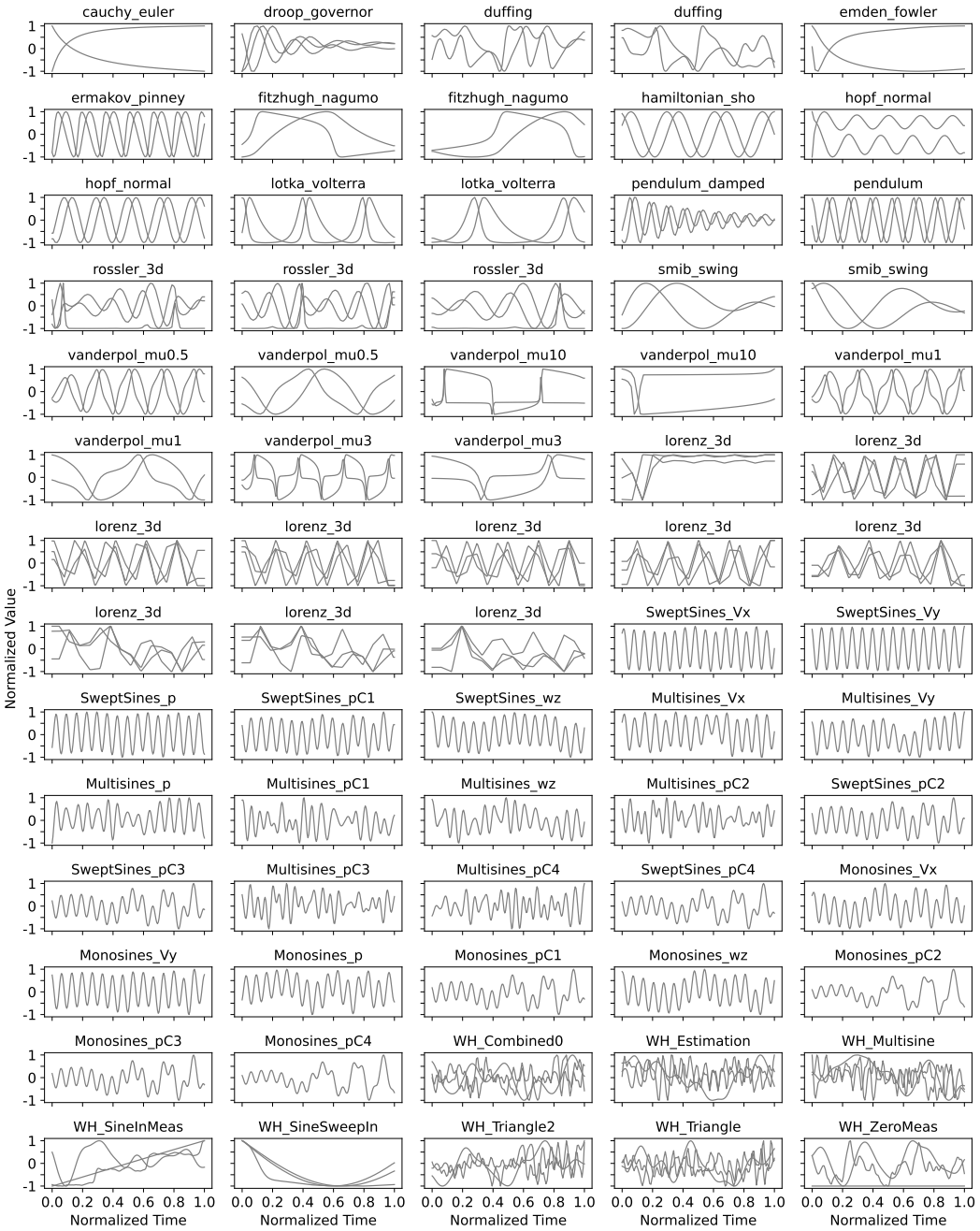

Figure 10: Example trajectories from the constructed dataset, with each panel corresponding to a different dynamical system.

F MODELS AND ENVIRONMENTS

Environment. All experiments were conducted on a high-performance computing cluster equipped with 4 NVIDIA A100 GPUs, each with 80 GB of GPU memory. Our implementation is based on PyTorch. ODE integration is performed using `torchdiffeq` (Chen et al., 2018). Unless otherwise specified, all reported results use the default configuration summarized below.

Hyperparameters (default run).

MODEL DETAILS

GRU trajectory encoder. For each state variable, we independently encode its irregularly sampled observations using a 2-layer GRU with hidden dimension 128. At an observation time t_ℓ, the GRU input is $[x^{(j)}(t_\ell); \Delta t_\ell] \in \mathbb{R}^2$, where Δt_ℓ explicitly provides the local sampling interval. Processing the variables independently makes the encoder applicable to systems with different state dimensions and produces $\boldsymbol{h}^{(j)} \in \mathbb{R}^{128}$ for each state variable j.

The GRU is part of the initial observation encoder rather than the recurrent fixed-point map. Therefore, its weight matrices are not subject to the spectral constraints used to certify the recurrent Refiner. This allows the initial representation $E^{(0)}$ to retain sufficient expressive capacity while the subsequent recurrent corrections remain controlled.

RBF-based temporal modulation. We use $K_{\mathrm{rbf}} = 64$ Gaussian kernels with bandwidth $\sigma = 0.25$ to encode the normalized temporal coordinate τ_k. We use $K_{\mathrm{token}} = 40$ localized temporal tokens over $\mathcal{T} = [0, 1]$. The RBF representation is processed by a two-layer Tanh MLP with hidden dimension 256, which produces

$$(\boldsymbol{\gamma}_k; \boldsymbol{\beta}_k) \in \mathbb{R}^{2d_{\mathrm{model}}} = \mathbb{R}^{256}.$$

Thus, $\boldsymbol{\gamma}_k, \boldsymbol{\beta}_k \in \mathbb{R}^{128}$. The trajectory representation $\boldsymbol{h}^{(j)}$ is separately mapped to $\mathbb{R}^{128}$ by a two-layer MLP with hidden dimension 256 before the element-wise temporal modulation in Eq. (2).

The temporal modulation is applied only when constructing the initial tokens and is therefore not included in the Lipschitz constant of the recurrent Fixed-Point Transformer.

Variable encoding. The maximum supported number of state variables is $d_x^{\max} = 10$. We therefore use the learnable lookup table

$$C \in \mathbb{R}^{d_x^{\max} \times d_{\mathrm{model}}} = \mathbb{R}^{10 \times 128}. \tag{45}$$

The corresponding row $C^{(j)}$ is added to every temporal token associated with state variable j. This encoding differentiates physical variables without requiring a system-specific input dimension.

Initial Transformer encoder. The resulting coordinate-aware tokens $\{\widetilde{\boldsymbol{e}}_{jk}\}_{j,k}$ are processed by a standard 4-layer Transformer to obtain $E^{(0)}$. We use $d_{\mathrm{model}} = 128$, $H = 4$ attention heads, and hence $d_h = d_{\mathrm{model}}/H = 32$ dimensions per head. The feed-forward dimension is 512.

Importantly, this Transformer is evaluated only once before fixed-point reasoning. Its parameters are not shared across recurrent iterations and are not included in the contraction certificate. We therefore do not impose the restrictive recurrent spectral bounds on this initial Transformer. The recurrent stability constraints are applied only after $E^{(0)}$ has been constructed.

Fixed-point Refiner. The recurrent Refiner follows Eq. (3). At iteration r, the current representation first receives the estimated physics through

$$\widetilde{E}^{(r)} = E^{(r)} + P\left(\widehat{\boldsymbol{\theta}}^{(r)}\right).$$

It then undergoes inter-intra-system attention followed by the MoE block, producing the undamped candidate $\widehat{E}^{(r+1)}$. The candidate is anchored to the initial observation representation through

$$E^{(r+1)} = (1-\rho)E^{(0)} + \rho\widehat{E}^{(r+1)}, \qquad \rho = 0.20.$$

The same Estimator, physics-aware encoder, inter-intra attention, and MoE parameters are reused at every recurrent iteration.

Table 3: Default model, fixed-point reasoning, and training hyperparameters.

Category	Name	Value
Data	ODE time points	50–200
	State dimension d_x	1–5
	Maximum state dimension $d_x^{\max}$	10
	Number of prefixes	56
	Prefix ratio	0.3–0.9
Representation	Embedding dimension d_{model}	128
	GRU hidden dimension	128
	GRU layers	2
	RBF kernels K_{rbf}	64
	RBF bandwidth σ	0.25
	Temporal tokens K_{token}	40
	Time-modulation hidden dimension	256
	Initial Transformer depth	4
	Attention heads H	4
	Attention head dimension d_h	32
	Transformer FFN dimension	512
Fixed-point Refiner	CSH size K_{CSH}	10
	MoE experts	8
	Activated experts	2
	Expert hidden dimension	512
	Representation radius $B_{\mathcal{H}}$	$2\sqrt{128} \approx 22.63$
	Refiner LayerNorm gain bound $\gamma_{\text{LN}}^{\max}$	0.6
	Refiner LayerNorm epsilon ϵ_{LN}	0.05
	Attention spectral bound s_{attn}	0.27
	MoE/FFN spectral bound s_{FFN}	0.35
	Estimator spectral bound s_{Est}	0.80
	Physics-encoder spectral bound s_{phy}	0.70
	Anchoring coefficient ρ	0.20
	Maximum recurrent iterations $R_{\max}$	10
	Convergence tolerance ϵ	10^{-4}
	Recurrent dropout	0
Local dynamics	Latent dimension d_z	15
	Householder reflections L_Q	15
	Estimator hidden dimension	256
	Estimator layers	2
	Probe dimension d_R	8
	Physics-encoder hidden dimension	256
	Physics-encoder layers	2
	Stability schedule $\lambda(s)$	$\min\{1, s/0.3\}$
	ODE solver	`rk4`
Cross-system transfer	CSH EMA coefficient η	0.99
Training	Optimizer	AdamW
	Learning rate	1×10^{-3}
	Betas	$(0.9,\ 0.95)$
	Weight decay	0.05
	Batch size	16
	Epochs	1500
Fine-tuning	Method	LoRA
	Learning rate	5×10^{-3}
	Epochs	30–100

We terminate recurrent inference when

$$\left\| E^{(r+1)} - E^{(r)} \right\|_{\text{F}} < \epsilon, \qquad \epsilon = 10^{-4}, \tag{46}$$

or when $R_{\max} = 10$ iterations have been performed. We set dropout to zero inside the recurrent Refiner so that the recurrent map is deterministic during both training and inference.

Bounded representation space. We implement the admissible representation space $\mathcal{H}$ as a product of token-wise Euclidean balls:

$$\mathcal{H} = \{E : \|E_{jk}\|_2 \le B_{\mathcal{H}},\ \forall j,k\}, \qquad B_{\mathcal{H}} = 2\sqrt{d_{\text{model}}} = 2\sqrt{128} \approx 22.63. \tag{47}$$

The projection $\text{Proj}_{\mathcal{H}}$ is applied independently to each token. The radius $2\sqrt{d_{\text{model}}}$ leaves substantial room beyond the typical norm scale of a normalized 128-dimensional token while ensuring that the recurrent representation remains in a closed, convex, and bounded set. Since Euclidean projection onto $\mathcal{H}$ is nonexpansive, the projection itself does not increase the Lipschitz constant of the recurrent map.

Inter-intra-system attention. The common structure hub contains $K_{\text{CSH}} = 10$ prototype tokens,

$$\Pi \in \mathbb{R}^{K_{\text{CSH}} \times d_{\text{model}}} = \mathbb{R}^{10\times 128}. \tag{48}$$

During each fixed-point solve, Π is treated as fixed external memory. Following Eq. (10), the current-system tokens and CSH tokens are jointly processed by multi-head self-attention, while only the outputs associated with the current-system tokens are retained.

We use $H = 4$ heads with head dimension $d_h = 32$. The learned query, key, value, and output projection matrices are constrained by

$$\sigma_{\max}(W) \le s_{\text{attn}} = 0.27, \qquad W \in \{W_Q, W_K, W_V, W_O\}. \tag{49}$$

The value 0.27 is deliberately tighter than the bounds used for the Estimator and physics-aware encoder because the sensitivity of dot-product attention depends on both the query/key transformations and their interaction. Restricting the recurrent attention does not directly restrict the expressiveness of the initial Transformer encoder; instead, it limits the magnitude of each recurrent attention-based correction.

Mixture-of-experts block. The recurrent MoE contains 8 experts, with the top 2 experts activated for each token. Each expert is implemented as

$$\mathbb{R}^{128} \to \mathbb{R}^{512} \to \mathbb{R}^{128},$$

with GELU activation. For the learned matrices in the expert transformations, we use

$$\sigma_{\max}(W) \le s_{\text{FFN}} = 0.35. \tag{50}$$

The expert matrices are therefore less strongly constrained than the attention projections. This reflects the fact that an MLP has a simpler Lipschitz dependence on its weight matrices than dot-product attention. The router maps the 128-dimensional token representation to 8 expert scores and uses top-2 routing.

Refiner LayerNorm. The two pre-LayerNorm operations inside the recurrent Refiner use

$$\epsilon_{\text{LN}} = 0.05, \qquad \|\boldsymbol{\gamma}_{\text{LN}}\|_\infty \le \gamma_{\text{LN}}^{\max} = 0.6. \tag{51}$$

After every optimizer update, the learned LayerNorm scaling factors are clipped element-wise to $[-\gamma_{\text{LN}}^{\max}, \gamma_{\text{LN}}^{\max}]$. These values are used only for LayerNorm operations inside the recurrent Refiner. They need not be imposed on normalization layers in the non-recurrent initial encoder.

The relatively larger numerical constant $\epsilon_{\text{LN}} = 0.05$ controls the worst-case sensitivity of normalization when the input variance becomes small. For representations with unit-order variance, its effect on the usual normalization scale is mild, while it substantially improves the worst-case LayerNorm Lipschitz bound.

Structured Parameter Estimator. For each temporal token k, we first aggregate the variable-wise representations according to

$$\boldsymbol{e}_k = \frac{1}{d_x}\sum_{j=1}^{d_x} \boldsymbol{e}_{jk} \in \mathbb{R}^{128}. \tag{52}$$

The mean aggregation is dimension invariant and nonexpansive.

The Estimator then follows Eq. (6) and returns the structured quantities

$$\Big(B_k,\ \{\boldsymbol{v}_{k,\ell}\}_{\ell=1}^{L_Q},\ \boldsymbol{\alpha}_k,\ \boldsymbol{b}_k\Big) = \text{Est}(\boldsymbol{e}_k), \tag{53}$$

rather than directly estimating a dense A_k. We use $d_z = 15$ and $L_Q = 15$. Therefore,

$$B_k \in \mathbb{R}^{15\times 15}, \qquad \boldsymbol{v}_{k,\ell} \in \mathbb{R}^{15}, \qquad \boldsymbol{\alpha}_k, \boldsymbol{b}_k \in \mathbb{R}^{15}.$$

The corresponding number of externally defined structured quantities is

$$\begin{aligned} d_{\text{struct}} &= \underbrace{d_z^2}_{B_k} + \underbrace{L_Q d_z}_{\{\boldsymbol{v}_{k,\ell}\}} + \underbrace{d_z}_{\boldsymbol{\alpha}_k} + \underbrace{d_z}_{\boldsymbol{b}_k} \\ &= 15^2 + 15^2 + 15 + 15 = 480. \end{aligned} \tag{54}$$

Internally, the stability-adaptation mechanism in Eq. (7) requires the raw vector $\widetilde{\boldsymbol{\alpha}}_k$ and a scalar gate g_k. We therefore implement the final raw Estimator head with

$$d_{\text{raw}} = d_z^2 + L_Q d_z + d_z + d_z + 1 = 481 \tag{55}$$

outputs. The scalar raw gate is passed through a sigmoid to obtain $g_k \in [0, 1]$, and $\widetilde{\boldsymbol{\alpha}}_k$ is transformed according to Eq. (7) to obtain the final $\boldsymbol{\alpha}_k$. Thus, Eq. (6) continues to expose the 480 structured quantities, while the additional scalar is used only internally to implement automatic stability adaptation.

Concretely, the Estimator is

$$128 \to 256 \to 481,$$

with Tanh activation between the two linear transformations. We constrain both learned matrices by

$$\sigma_{\max}(W) \le s_{\text{Est}} = 0.80. \tag{56}$$

We intentionally use a substantially looser bound for the Estimator than for recurrent attention, since the Estimator must recover a relatively rich structured dynamical description from each token representation.

The predicted B_k and Householder vectors are deterministically converted to

$$\Omega_k = \frac{B_k - B_k^\top}{2}, \qquad Q_k = \prod_{\ell=1}^{L_Q} H(\boldsymbol{v}_{k,\ell}), \tag{57}$$

and the final local operator is

$$A_k = \Omega_k + Q_k \operatorname{diag}(\boldsymbol{\alpha}_k) Q_k^\top. \tag{58}$$

Therefore, A_k is constructed from the learned dynamical characteristics and is never directly predicted by a separate dense operator head.

Automatic stability adaptation. We use the monotone schedule

$$\lambda(s) = \min\left\{1, \frac{s}{0.3}\right\}, \qquad s \in [0, 1], \tag{59}$$

where s denotes normalized training progress. Consequently, the first 30% of training gradually relaxes the initial preference for negative growth rates. This allows the model to first learn predominantly stable local dynamics while subsequently activating locally expansive modes when required by the data.

Physics-aware encoding. We use the probe-based dynamic signature in Eq. (8). The latent dimension is $d_z = 15$ and the probe dimension is set to

$$d_R = 8.$$

The shared probe matrix

$$R \in \mathbb{R}^{15\times 8}, \qquad R^\top R = I,$$

is generated once by orthogonalizing a random Gaussian matrix and is then kept fixed across all systems, temporal tokens, training iterations, and recurrent refinement steps.

The input to h_{phy} is

$$\left[\operatorname{vec}(\Omega_k R); \operatorname{vec}(S_k R); \boldsymbol{b}_k; \alpha_k^{\max}; \alpha_k^{\min}\right].$$

Its dimension is therefore

$$\begin{aligned} d_{\text{phy}} &= 2d_z d_R + d_z + 2 \\ &= 2(15)(8) + 15 + 2 \\ &= 257. \end{aligned} \tag{60}$$

The physics-aware encoder is implemented as

$$257 \to 256 \to 128,$$

with Tanh activation. Its output is therefore directly compatible with the 128-dimensional recurrent token representation. Both learned matrices are constrained by

$$\sigma_{\max}(W) \le s_{\text{phy}} = 0.70. \tag{61}$$

This constraint is looser than that of recurrent attention because the physics-aware encoder acts on an already structured dynamical signature, rather than constructing pairwise token interactions.

Common Structure Hub. We use

$$K_{\text{CSH}} = 10$$

common-structure prototypes, giving

$$\Pi \in \mathbb{R}^{10\times 128}.$$

The prototypes are updated only from converged representations $E^\star$, as described in Eq. (9). We use the exponential moving-average coefficient

$$\eta = 0.99.$$

The stop-gradient operator prevents the prototype update from introducing an additional gradient path into $E^\star$. During a single recurrent fixed-point solve, the CSH is held fixed; the prototype update is performed only after the equilibrium representation has been obtained.

Module-specific spectral projection. The recurrent modules have substantially different roles and Lipschitz sensitivities. In particular, dot-product attention is considerably more sensitive to its projection matrices than the structured Estimator. We therefore use module-specific spectral limits rather than forcing every learned transformation to share a single unnecessarily restrictive bound.

For every constrained matrix W, after each optimizer update we apply

$$W \leftarrow \frac{s(W)W}{\max\{s(W), \sigma_{\max}(W)\}}, \tag{62}$$

where

$$s(W) = \begin{cases} 0.27, & W \in \text{recurrent MHA}, \\ 0.35, & W \in \text{MoE/FFN}, \\ 0.80, & W \in \text{Est}, \\ 0.70, & W \in h_{\text{phy}}. \end{cases} \tag{63}$$

For recurrent MHA, the constraint is applied to W_Q, W_K, W_V, and W_O. For every MoE expert, it is applied to both linear transformations. For the Estimator and physics-aware encoder, it is applied to every learned linear transformation. Bias vectors are retained and are not spectrally constrained because they do not affect the input-output Lipschitz constant.

We compute $\sigma_{\max}(W)$ after the optimizer update and rescale the matrix before the next forward pass. Thus, the spectral constraints are hard architectural constraints rather than soft regularization penalties.

Numerical contraction certificate. We finally verify that the above default configuration satisfies the contraction condition in Theorem 1. For the recurrent LayerNorm,

$$L_{\text{LN}} \le \frac{\gamma_{\text{LN}}^{\max}}{\sqrt{\epsilon_{\text{LN}}}} = \frac{0.6}{\sqrt{0.05}} \approx 2.6833, \tag{64}$$

and the norm of a normalized token is bounded by

$$r_{\text{LN}} = \gamma_{\text{LN}}^{\max}\sqrt{d_{\text{model}}} = 0.6\sqrt{128} \approx 6.7882. \tag{65}$$

For $H = 4$, $d_h = 32$, and $s_{\mathrm{attn}} = 0.27$, the recurrent multi-head attention satisfies the conservative bound

$$\begin{aligned} L_A &\le L_{\mathrm{LN}}\sqrt{H}\, s_{\mathrm{attn}}^2 \left(1 + \frac{4 r_{\mathrm{LN}}^2 s_{\mathrm{attn}}^2}{\sqrt{d_h}}\right) \\ &\le 1.3206. \end{aligned} \tag{66}$$

Using the maximum derivative of GELU, approximately 1.129, the corresponding FFN/Expert bound is

$$\begin{aligned} L_F &\le L_{\mathrm{LN}} \left(1.129\, s_{\mathrm{FFN}}^2\right) \\ &\le 0.3712. \end{aligned} \tag{67}$$

Consequently, the attention and feed-forward residual transformations give

$$L_\Phi \le (1 + L_A)(1 + L_F) \le 3.1817. \tag{68}$$

The two-layer Tanh Estimator satisfies

$$L_{\mathrm{Est}} \le s_{\mathrm{Est}}^2 = 0.80^2 = 0.64, \tag{69}$$

while the two-layer Tanh physics-aware encoder satisfies

$$L_{\mathrm{phy}} \le s_{\mathrm{phy}}^2 = 0.70^2 = 0.49. \tag{70}$$

Under the regular structured-operator parameterization assumed in our fixed-point analysis, the physics-feedback path therefore has

$$L_{\mathrm{fb}} \le L_{\mathrm{Est}} L_{\mathrm{phy}} = 0.3136. \tag{71}$$

The undamped candidate map is consequently bounded by

$$\begin{aligned} L &\le (1 + L_{\mathrm{fb}}) L_\Phi \\ &\le 1.3136 \times 3.1817 \\ &\le 4.1795. \end{aligned} \tag{72}$$

With $\rho = 0.20$, the complete recurrent map therefore satisfies

$$\kappa \le \rho L \le 0.20 \times 4.1795 \approx 0.836 < 1. \tag{73}$$

Thus, the default recurrent configuration satisfies the sufficient contraction condition used in Theorem 1. The bound also implies

$$\kappa^{10} \lesssim 0.167,$$

which motivates the choice $R_{\max} = 10$: even under the conservative certificate, ten recurrent steps substantially reduce the distance to the fixed-point equilibrium, while the residual-based stopping criterion usually allows earlier termination.

Latent initialization and ODE solver. The initial latent state is obtained from the variable-wise GRU representations according to Eq. (5):

$$\boldsymbol{z}_0 = \phi\left(\frac{1}{d_x}\sum_{j=1}^{d_x} \boldsymbol{h}^{(j)}\right) \in \mathbb{R}^{15}. \tag{74}$$

The projection ϕ is implemented as an MLP from the 128-dimensional aggregated trajectory representation to the 15-dimensional latent space. Since ϕ is used to initialize the ODE trajectory and does not belong to the recurrent representation update, it is not included in the Fixed-Point Transformer spectral constraints.

After the local operators have been inferred, we integrate

$$\dot{\boldsymbol{z}}(\tau) = A_k \boldsymbol{z}(\tau) + \boldsymbol{b}_k, \qquad \tau \in [\tau_k, \tau_{k+1}), \tag{75}$$

using the fourth-order Runge–Kutta solver `rk4`. Other solver settings follow the defaults of `torchdiffeq.odeint` (Chen et al., 2018). Since the vector field within each temporal token is affine, ODE evaluations require only the structured linear transformation and drift rather than repeated evaluation of a nonlinear neural vector field.

Linear decoder. The physical trajectory is reconstructed from the latent trajectory as

$$\widehat{\boldsymbol{x}}(\tau) = W_{\mathrm{dec}}\boldsymbol{z}(\tau) + \boldsymbol{b}_{\mathrm{dec}}, \tag{76}$$

where

$$W_{\mathrm{dec}} \in \mathbb{R}^{d_x \times d_z} = \mathbb{R}^{d_x \times 15}, \qquad \boldsymbol{b}_{\mathrm{dec}} \in \mathbb{R}^{d_x}. \tag{77}$$

The decoder is applied after latent ODE evolution and does not participate in the recurrent fixed-point map; consequently, its parameters are not subject to the recurrent spectral constraints.

Training and fine-tuning. We train LASS-ODE using AdamW with learning rate 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.05, and batch size 16 for 1500 epochs. Spectral projection and LayerNorm-gain clipping are applied after every optimizer update so that the recurrent model satisfies the architectural constraints throughout training rather than only at inference time.

For system-specific adaptation, we use LoRA (Hu et al., 2022) with learning rate 5×10^{-3} for 30–100 epochs. Unless otherwise stated, all other model components and the fixed-point inference procedure remain unchanged during fine-tuning.

G BASELINE MODELS

We organize our baseline suite into six groups that provide complementary comparisons for learning and forecasting trajectories across heterogeneous dynamical systems.

(1) Neural ODE and hybrid continuous-time models. Latent ODE (Rubanova et al., 2019) and Latent MoS (Li et al., 2025) represent the neural ODE family, which generates trajectories through latent dynamical evolution. ContiFormer (Chen et al., 2023) combines Transformer representations with continuous-time dynamics. These models provide comparisons with latent ODE modeling and hybrid attention–ODE architectures.

(2) Task-specific Transformer forecasting models. Informer (Zhou et al., 2021) and Autoformer (Wu et al., 2021) are supervised Transformer forecasting baselines. They provide task-specific comparisons based on discrete sequence prediction, without an explicit latent ODE solver.

(3) Cross-system time-series foundation models. We include TimesFM (Das et al., 2024), TimeGPT (Garza et al., 2023), Chronos (Ansari et al., 2024), and Timer-XL (Liu et al., 2024b). These models learn transferable temporal representations from large time-series collections and are evaluated primarily through zero-shot forecasting, with fine-tuning where supported. They provide a comparison with general-purpose temporal prediction without an explicit learned ODE representation.

(4) State-space sequence models. We add S4D-style SSM, inspired by diagonal state-space layers (Gu et al., 2022), and Mamba (Gu & Dao, 2023). Our S4D-style implementation uses stable real negative diagonal dynamics discretized with the observed time intervals, four residual blocks of width 128, and 16 state coordinates per feature; it is a custom diagonal SSM rather than a full S4 or canonical S4D reproduction. Mamba uses the Hugging Face Mamba-1 backbone with four layers, width 128, state size 16, and linear trajectory-input/output adapters. Sampling intervals enter Mamba as input features, while its internal selective discretization remains learned.

(5) Liquid neural networks. Closed-form continuous-time networks (CfC) (Hasani et al., 2022) use explicit time-dependent recurrent updates, whereas liquid time-constant networks (LTC) (Hasani et al., 2021) evolve hidden states with input-dependent effective time constants. We use the official `ncps` implementations, stacking four 128-unit recurrent layers with a linear readout and supplying the observation intervals to their continuous-time updates. LTC uses six internal solver unfolds per interval. These baselines test whether flexible continuous-time recurrent dynamics improve ODE-trajectory forecasting.

(6) Scientific operator models and ODE adaptations. MPP (McCabe et al., 2024) and PDE-FM (Soares et al., 2025) provide PDE-oriented scientific surrogate comparisons. We additionally examine the following numerical adaptations of scientific operator architectures on ODE trajectories.

CoDA-NO-1D. CoDA-NO (Rahman et al., 2024) performs attention across physical-variable functions. Our adaptation uses time as the one-dimensional operator domain and treats each observed state component as a variable token. It retains the official spectral and attention layers, with one-dimensional normalization and masked future states. The small configuration has two operator layers and 16 Fourier modes.

PROSE-ODE and PROSE-PDE. PROSE-ODE (Liu et al., 2024c) and PROSE-PDE (Sun et al., 2025) combine numerical operator learning with symbolic modeling in their original formulations. Here, both use their official numerical-only encoder–operator-decoder branches, without equation inputs or symbolic targets. The ODE branch encodes observed time–state pairs and predicts states at query times. The PDE branch places the padded state variables on an artificial one-dimensional grid and maps the predicted field back to a state vector. Both use width 64 and two layers in each encoder and decoder.

Poseidon-0D. Poseidon (Herde et al., 2024) uses a time-conditioned multiscale operator Transformer. Our small adaptation lifts the last observed ODE state, its channel mask, and observation time to spatially constant 8×8 fields, applies the official ScOT backbone with elapsed-time conditioning, and spatially averages the output to recover a state vector. It uses only the last observation, whereas the other three operator adaptations use the observed prefix.

Additional-baseline training and evaluation. S4D-style SSM (Gu et al., 2022), Mamba (Gu & Dao, 2023), CfC (Hasani et al., 2022), and LTC (Hasani et al., 2021) use the LASS-ODE data loader and a common masked-prefix forecasting interface. They are trained for 1,500 epochs on 38 systems with seeds 42, 43, and 44; their comparisons use the mean over seeds. The learning rate is 3×10^{-4} for the first three and 10^{-4} for LTC, selected by a held-out-system validation search. The four operator adaptations—CoDA-NO-1D (Rahman et al., 2024), PROSE-ODE (Liu et al., 2024c), PROSE-PDE (Sun et al., 2025), and Poseidon-0D (Herde et al., 2024)—are smaller architecture-level experiments trained from scratch for 100 epochs on six ODE systems with seed 42, without pretrained PDE weights. Each training system supplies 30 trajectory segments, and evaluation scores the complete unobserved suffix at prefix ratios of 30%, 60%, and 90%. These adaptations therefore assess the specified numerical architectures, rather than reproducing the original pretrained or multimodal foundation models.

We exclude traditional system-identification methods such as SINDy (Brunton et al., 2016b) from the shared multi-system baseline suite because their usual fitting procedure estimates a separate governing model for each system.

H TOKEN-SPACE STRUCTURE AND FIXED-POINT REFINEMENT

To visualize how LASS-ODE organizes heterogeneous ODE trajectories around reusable dynamical structures, we project the trajectory-token embeddings into a shared two-dimensional space using t-SNE, shown in panel (a) in Fig. 11. Gray dots denote ODE trajectory tokens, while purple diamonds denote the ten learned CSH prototypes $\{\boldsymbol{\pi}_\ell\}_{\ell=1}^{10}$. The tokens organize into recurring regions associated with sustained oscillation, damped oscillation, transient response, dissipative relaxation, fast–slow dynamics, and near-equilibrium dynamics. We assign them post hoc by comparing the trajectories represented in each cluster with the known qualitative behavior of their corresponding benchmark systems. Panel (b) illustrates the corresponding fixed-point refinement process at iterations $r = 0, 1, 3$, and 5. Starting from dispersed system-specific token embeddings at $r = 0$, the blue tokens progressively move toward regions associated with shared dynamical structures.

In particular, the Van der Pol tokens approach the sustained-oscillation and fast–slow regions, the FitzHugh–Nagumo tokens approach the fast–slow and transient-response regions, and the damped-pendulum tokens approach the damped-oscillation, dissipative-relaxation, and near-equilibrium regions. These assignments are consistent with the qualitative dynamics of the corresponding systems. Panel (c) summarizes the refinement rate using the normalized update magnitude. The decreasing δ_r indicates that successive refinement updates become smaller, providing qualitative evidence that the Fixed-Point Transformer progressively organizes heterogeneous trajectories around shared dynamical structures.

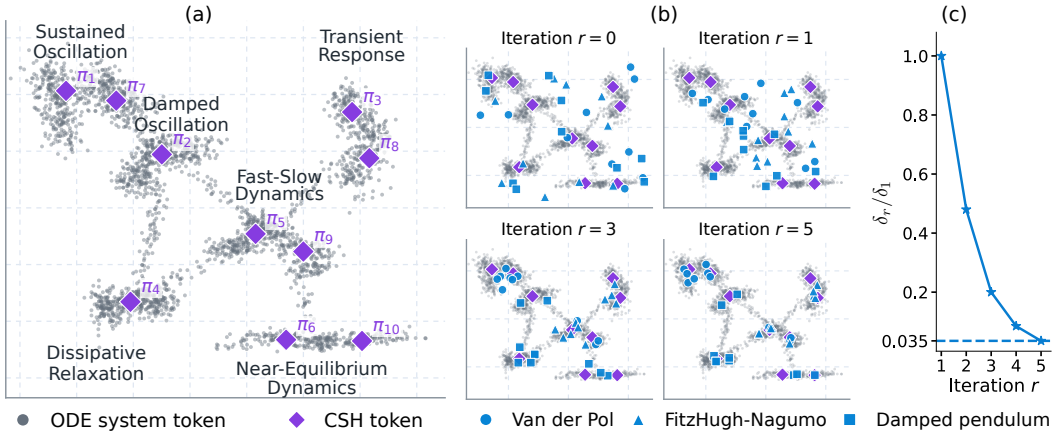

Figure 11: **(a)** 2D visualization of ODE system tokens and CSH tokens ($k_{\text{CSH}} = 10$). **(b)** Fixed-point reasoning of test system tokens at iterations $r = 0, 1, 3, 5$. **(c)** Normalized convergence of the token updates. For iteration r, $\delta_r = \left\|E^{(r+1)} - E^{(r)}\right\|_{\text{F}}$, and the plotted quantity is δ_r/δ_1.

I RESULTS OF IN-DOMAIN TEST TRAJECTORIES AND MSE

We show the table of MSE of in-domain test in Table 4 and 5. We show the trajectory comparison of us and time-series foundation models in Figs. 12-16. The prefix ratios are 30%, 60%, and 90%.

Table 4: Test MSE ($\times 10^{-2}$) for in-domain systems.

ODE Systems	Prefix ratio	**LASS-ODE** (Ours)	**TimesFM** (2024)	**Chronos** (2024)	**Timer-XL** (2024b)	**Informer** (2021)	**Autoformer** (2021)	**Latent ODE** (2019)	**Latent MoS** (2025)	**ContiFormer** (2023)	**PDE-FM** (2025)	**MPP** (2024)	**CoDA-NO-1D** (2024)	**PROSE-ODE** (2024c)	**PROSE-PDE** (2025)	**Poseidon-0D** (2024)	**S4D-style SSM** (2022)	**Mamba** (2023)	**CfC** (2022)	**LTC** (2021)
In-domain Test, part 1 of 2																				
Cauchy_euler	30%	**0.01**	74.10	69.00	84.40	33.40	19.00	0.63	0.30	20.20	1.50	3.90	2.29	4.40	4.78	4.59	0.04	0.02	4.01	4.66
	60%	**0.01**	13.80	27.70	23.90	57.20	32.60	0.54	0.14	16.90	0.45	1.20	3.19	4.33	4.84	7.10	0.04	0.03	7.29	7.66
	90%	**0.01**	0.32	0.06	0.64	67.10	6.20	0.68	0.06	1.70	0.01	0.01	1.64	4.52	5.86	8.02	0.03	0.02	9.39	9.49
Droop_governor	30%	**0.01**	4.20	33.80	45.30	3.00	1.80	0.16	0.09	4.40	0.22	0.68	2.12	3.24	2.94	3.43	0.02	0.06	10.93	9.41
	60%	**0.01**	1.60	17.40	8.50	5.10	3.20	0.16	0.06	4.20	0.09	0.38	2.82	2.37	3.33	4.15	0.08	0.05	12.11	10.44
	90%	**0.01**	3.00	0.69	1.60	5.00	0.96	0.16	0.05	1.10	0.02	0.06	1.10	1.25	2.31	2.96	0.07	0.05	9.87	8.25
Ermakov_pinney	30%	**0.06**	10.70	23.70	67.60	51.50	9.20	1.70	0.46	19.20	0.48	0.80	7.73	7.27	7.32	7.77	0.66	0.35	7.22	7.21
	60%	**0.05**	3.60	2.50	51.80	51.20	6.30	1.60	0.32	7.70	0.28	0.20	3.53	4.73	4.92	5.38	0.45	0.06	4.80	4.78
	90%	**0.05**	1.90	0.47	37.60	51.00	4.30	1.70	0.27	3.90	0.25	0.11	1.34	4.28	4.38	5.47	0.30	0.06	4.34	4.41
Hopf_normal	30%	**0.03**	1.70	6.70	57.20	49.90	21.10	1.20	0.24	10.50	0.29	0.34	0.78	8.44	7.16	9.67	1.13	0.53	13.93	12.05
	60%	0.03	**0.02**	0.66	48.50	50.10	8.20	1.20	0.08	4.20	0.17	0.04	0.45	6.51	7.03	9.86	0.81	0.44	11.97	9.81
	90%	**0.03**	0.04	0.08	38.60	49.70	1.30	1.20	0.07	1.80	0.16	0.03	0.27	4.80	4.93	12.19	0.68	0.31	16.20	12.91
Lotka_volterra	30%	**0.07**	51.20	297.80	54.90	57.60	43.30	1.90	0.91	56.40	5.00	8.20	23.52	21.29	25.14	18.06	4.76	1.75	21.02	15.40
	60%	**0.03**	35.30	43.00	54.50	58.70	49.20	1.40	0.57	24.50	2.20	2.30	9.44	14.89	12.34	7.27	1.30	0.34	10.28	7.09
	90%	**0.04**	24.40	1.00	21.50	59.40	20.10	1.50	0.48	5.60	0.82	0.59	9.61	10.77	7.84	4.35	0.68	0.25	6.39	5.32
Fitzhugh_nagumo	30%	**0.02**	97.83	136.04	136.72	48.78	79.06	0.91	0.52	33.15	2.39	7.28	9.23	10.05	7.59	14.94	2.67	2.03	19.37	16.74
	60%	**0.01**	28.08	52.27	64.29	48.86	19.32	0.81	0.27	22.13	1.04	1.99	5.09	4.07	3.89	8.06	1.43	0.56	12.03	11.04
	90%	**0.01**	3.09	1.38	11.98	53.29	4.86	0.83	0.13	5.38	0.53	0.71	1.40	1.16	0.99	3.65	0.89	0.09	6.95	7.13
Pendulum	30%	**0.10**	14.20	32.10	64.90	48.60	41.60	2.20	0.92	22.30	16.00	12.60	11.82	11.00	10.78	11.52	7.62	4.00	10.70	10.72
	60%	**0.10**	9.50	1.10	51.80	48.40	32.20	2.30	0.81	5.80	22.20	16.60	8.52	9.10	9.00	9.62	5.28	2.20	8.90	8.90
	90%	**0.11**	4.60	0.14	42.60	48.40	22.70	2.30	0.66	2.50	31.10	23.60	2.88	7.14	7.12	8.12	1.16	0.34	6.98	6.94
Rossler_3d	30%	**0.07**	20.20	84.30	34.50	49.80	11.60	1.90	0.61	31.90	2.40	3.70	4.28	5.38	5.36	6.28	2.42	1.72	13.58	12.90
	60%	**0.05**	17.00	35.40	34.00	49.70	8.50	1.60	0.45	21.80	2.60	3.00	1.48	2.94	2.92	3.56	1.40	0.68	7.36	6.80
	90%	**0.05**	6.50	16.50	19.00	50.00	5.50	1.60	0.34	16.10	3.30	2.70	1.20	3.10	3.16	3.52	1.10	0.16	8.84	8.10
Smib_swing	30%	**0.02**	41.00	68.60	52.50	23.20	50.80	0.61	0.39	17.30	0.64	1.60	14.52	12.07	22.81	3.00	0.42	0.15	4.73	11.98
	60%	**0.01**	7.10	35.70	10.20	8.90	25.70	0.36	0.22	8.20	0.08	0.38	5.44	12.29	14.07	3.80	0.28	0.10	4.01	3.91
	90%	0.02	0.24	0.69	0.96	7.20	0.42	0.34	0.05	1.50	**0.01**	0.02	1.50	2.08	2.19	1.13	0.06	0.02	1.05	1.19
Vanderpol_mu0.5	30%	**0.01**	85.40	105.70	61.20	26.30	0.85	0.61	0.18	21.60	1.40	4.30	17.52	40.89	39.64	41.76	4.76	0.91	38.54	39.15
	60%	**0.01**	9.50	67.00	83.50	45.10	1.50	0.79	0.13	23.50	0.84	1.60	4.73	27.54	27.88	30.13	3.04	0.18	27.02	27.30
	90%	0.01	1.50	**0.01**	16.70	45.10	1.60	0.78	0.09	0.76	0.16	0.07	2.43	12.44	11.76	14.77	0.55	0.05	11.93	12.94
Vanderpol_mu10	30%	**1.10**	56.50	121.00	66.40	38.20	18.30	6.50	4.20	42.40	4.80	7.70	13.22	27.36	27.25	28.69	5.59	3.70	29.86	29.33
	60%	**0.29**	40.60	42.50	53.10	65.50	31.40	4.40	2.00	32.10	3.80	5.30	1.14	6.42	6.33	8.82	1.18	0.74	8.24	8.24
	90%	**0.10**	12.90	19.10	19.60	70.80	31.50	2.60	0.82	21.70	1.20	1.70	0.14	0.82	0.67	1.59	0.42	0.10	2.21	2.17
Vanderpol_mu1	30%	**0.02**	58.50	144.70	48.00	23.90	1.00	0.65	0.20	23.90	1.10	3.60	5.01	5.44	5.34	5.57	0.48	0.09	5.18	5.22
	60%	**0.02**	14.10	69.00	59.90	40.90	1.80	0.80	0.16	23.00	0.68	1.30	2.90	4.98	5.17	5.74	0.44	0.03	4.90	4.99
	90%	**0.02**	7.50	8.10	19.40	40.50	1.90	0.79	0.14	9.70	0.17	0.16	4.23	4.79	4.90	5.69	0.22	0.03	4.83	5.33
Vanderpol_mu3	30%	**0.03**	58.20	65.60	50.80	20.60	3.90	0.77	0.35	17.00	1.20	3.40	5.10	2.82	2.88	2.94	0.35	0.16	2.74	2.77
	60%	**0.03**	44.30	63.80	35.60	35.50	6.60	0.95	0.34	21.70	0.91	2.50	3.69	2.53	2.74	2.73	0.33	0.10	2.55	2.60
	90%	**0.03**	8.40	7.80	11.60	35.00	2.00	0.96	0.20	9.40	0.18	0.43	3.05	2.25	2.75	2.21	0.25	0.12	2.45	2.63
Lorenz_3d	30%	**0.08**	39.30	80.20	35.90	27.20	26.10	1.50	0.87	23.40	10.10	10.70	6.80	3.54	3.86	6.39	2.60	2.16	63.23	35.21
	60%	**0.06**	33.50	62.50	28.00	29.10	26.80	1.30	0.72	21.30	13.60	13.80	4.22	3.90	4.65	8.81	2.64	2.51	65.07	27.03
	90%	**0.07**	19.60	12.50	19.20	27.60	17.70	1.40	0.64	11.00	16.40	14.20	3.32	3.54	3.30	8.06	2.93	1.53	59.91	15.17
Sweptsines_vx	30%	**0.12**	37.30	46.10	46.30	46.30	2.80	2.40	0.73	25.60	0.76	2.00	9.42	9.79	9.36	10.35	1.47	0.51	9.03	9.07
	60%	**0.12**	31.80	1.40	42.30	48.20	1.20	2.50	0.60	6.40	0.44	0.45	7.82	8.20	7.67	9.51	1.38	0.21	7.68	7.68
	90%	**0.13**	18.20	0.83	46.00	40.50	0.50	2.30	0.45	4.80	0.38	0.20	9.77	6.34	6.41	8.68	0.56	0.17	6.51	6.54
Sweptsines_vy	30%	**0.12**	35.40	44.70	51.60	45.80	1.30	2.40	0.61	25.10	0.77	1.90	7.98	7.58	8.54	8.26	1.09	0.36	7.63	7.62
	60%	**0.12**	31.30	0.95	42.20	48.10	1.70	2.40	0.62	5.50	0.45	0.51	7.04	6.83	7.29	9.57	0.99	0.29	6.91	6.88
	90%	**0.13**	16.10	0.47	41.10	52.30	0.73	2.60	0.47	4.30	0.43	0.19	8.96	7.15	7.22	10.52	1.08	0.26	7.40	7.39
Sweptsines_p	30%	**0.12**	37.30	64.80	41.80	43.90	1.40	2.30	0.62	28.50	0.80	2.00	8.01	7.98	7.57	8.60	0.91	0.46	7.38	7.36
	60%	**0.12**	32.60	5.80	39.70	43.70	1.60	2.30	0.62	10.80	0.58	0.57	6.17	6.57	6.19	7.74	0.73	0.22	6.12	6.10
	90%	**0.12**	15.60	2.20	35.30	53.70	0.77	2.50	0.45	8.10	0.45	0.23	4.29	3.53	3.40	4.37	0.34	0.16	3.53	3.51
Sweptsines_pc1	30%	**0.09**	30.39	76.59	36.58	38.02	2.21	1.89	0.56	27.60	0.80	2.02	8.70	8.76	9.17	9.53	1.59	1.44	8.23	8.28
	60%	**0.09**	30.58	5.01	38.20	37.56	2.75	1.85	0.58	9.19	0.58	0.57	5.08	5.42	5.49	8.77	1.03	1.60	5.37	5.41
	90%	**0.09**	13.76	5.51	38.12	55.77	4.06	2.19	0.51	11.56	0.45	0.23	6.29	13.16	12.00	19.71	2.94	1.91	10.90	10.85

Table 5: Test MSE ($\times 10^{-2}$) for in-domain systems, continued.

ODE Systems	Prefix ratio	**LASS-ODE** (Ours)	**TimesFM** (2024)	**Chronos** (2024)	**Timer-XL** (2024b)	**Informer** (2021)	**Autoformer** (2021)	**Latent ODE** (2019)	**Latent MoS** (2025)	**ContiFormer** (2023)	**PDE-FM** (2025)	**MPP** (2024)	**CoDA-NO-1D** (2024)	**PROSE-ODE** (2024c)	**PROSE-PDE** (2025)	**Poseidon-0D** (2024)	**S4D-style SSM** (2022)	**Mamba** (2023)	**CfC** (2022)	**LTC** (2021)
In-domain Test, part 2 of 2																				
Sweptsines_wz	30%	**0.09**	24.70	39.00	31.60	32.70	7.60	1.70	0.68	19.50	17.10	18.80	2.79	4.44	3.23	2.42	2.79	1.61	3.43	3.04
	60%	**0.09**	29.10	41.70	26.50	34.40	6.90	1.70	0.66	20.40	16.20	15.60	2.19	4.41	2.61	3.36	2.72	1.27	3.10	2.78
	90%	**0.10**	11.20	2.20	31.30	25.00	6.00	1.60	0.57	5.50	27.00	26.20	1.85	2.57	1.83	1.75	2.22	0.68	1.92	1.77
Multisines_vx	30%	**0.13**	43.00	37.20	33.10	37.60	8.20	2.20	0.95	21.20	1.20	3.00	8.75	9.12	8.82	9.83	1.97	1.27	8.34	8.38
	60%	**0.13**	45.00	26.70	34.90	46.00	5.80	2.40	0.89	20.60	0.98	1.60	5.64	5.96	5.62	6.65	1.53	0.29	5.54	5.53
	90%	**0.13**	18.10	5.90	47.90	36.70	4.90	2.20	0.72	10.00	1.00	1.30	5.39	4.03	3.82	5.26	0.57	0.26	3.83	3.84
Multisines_vy	30%	**0.12**	55.10	52.00	34.10	32.10	6.60	2.00	0.92	22.30	2.00	4.10	24.85	23.34	27.34	24.67	3.30	2.40	23.56	23.44
	60%	**0.13**	38.20	25.50	32.80	34.90	9.10	2.10	0.95	17.60	1.80	2.20	20.98	19.83	20.47	25.60	3.84	0.98	19.85	19.72
	90%	**0.13**	12.10	14.20	38.80	53.20	6.00	2.60	0.68	17.20	3.00	2.70	15.52	13.19	13.16	19.29	1.20	0.24	13.14	13.14
Multisines_p	30%	**0.09**	36.00	37.30	26.70	29.50	9.80	1.70	0.78	18.20	4.20	5.30	6.38	5.99	6.10	6.96	1.05	1.04	5.67	5.69
	60%	**0.09**	29.30	35.60	23.90	35.30	23.60	1.80	0.87	19.50	4.30	4.80	6.11	5.84	5.53	6.41	1.04	1.04	5.64	5.65
	90%	**0.09**	5.90	8.50	17.00	25.80	2.60	1.50	0.41	9.40	3.90	3.70	3.86	3.70	3.74	6.23	0.86	0.37	4.04	4.06
Multisines_pc1	30%	**0.09**	39.90	43.40	38.90	23.20	20.20	1.40	0.88	17.00	18.40	16.50	17.04	4.76	4.37	9.51	12.66	10.74	8.90	8.96
	60%	**0.08**	38.90	30.60	17.80	24.90	40.30	1.40	0.97	15.30	10.20	9.10	4.68	3.30	3.17	7.19	4.59	10.40	3.50	3.44
	90%	**0.09**	14.80	23.30	18.10	16.90	14.60	1.20	0.70	11.40	10.70	10.00	2.73	2.63	3.56	2.49	3.41	4.35	2.75	2.81
Multisines_wz	30%	**0.14**	28.60	42.20	24.10	25.60	19.70	1.90	1.10	18.60	24.30	22.70	5.16	5.87	4.76	3.36	3.62	1.88	3.60	3.26
	60%	**0.14**	39.60	29.80	18.70	27.90	16.50	2.00	1.10	16.80	25.30	23.70	3.26	7.76	6.29	3.09	3.90	2.90	4.88	4.02
	90%	**0.13**	19.00	13.10	24.20	18.70	3.00	1.60	0.68	9.90	18.20	14.80	2.60	7.40	4.49	2.61	3.98	2.66	4.92	4.19
Multisines_pc2	30%	**0.26**	22.50	37.40	16.60	19.60	16.20	2.30	1.50	16.50	23.30	22.00	7.42	5.06	5.75	7.30	21.99	6.34	4.91	5.62
	60%	**0.28**	17.50	23.70	12.80	16.30	16.40	2.10	1.40	12.60	27.90	24.50	1.22	1.75	1.34	1.99	5.20	1.75	1.31	1.73
	90%	**0.25**	16.00	17.70	24.00	27.70	14.40	2.70	1.30	14.50	26.10	23.30	1.80	1.96	2.02	4.20	4.62	2.93	1.63	2.04
Sweptsines_pc2	30%	**0.07**	24.90	111.30	29.40	29.90	5.60	1.40	0.54	27.60	12.40	12.00	3.37	8.80	7.44	8.00	5.64	2.87	6.35	6.29
	60%	**0.07**	27.70	10.20	31.70	30.80	7.80	1.40	0.58	10.70	17.00	16.10	3.24	3.58	3.60	6.19	2.58	2.36	3.68	3.76
	90%	**0.06**	15.80	9.80	30.50	49.90	10.70	1.80	0.54	13.40	18.90	18.40	4.14	4.03	4.22	4.39	2.14	2.06	3.86	3.93
Sweptsines_pc3	30%	**0.06**	22.20	71.60	25.90	25.70	9.10	1.30	0.55	21.30	16.80	16.10	6.70	6.31	6.43	6.76	4.90	5.47	6.15	6.31
	60%	**0.06**	27.50	18.40	29.00	29.50	12.80	1.30	0.59	13.10	24.00	23.20	2.23	2.53	2.57	4.21	2.66	2.12	2.63	2.70
	90%	**0.06**	17.50	14.50	27.20	48.90	17.70	1.70	0.56	15.30	32.20	30.50	1.94	4.00	3.75	2.92	3.24	1.71	2.78	2.75
Multisines_pc3	30%	**0.26**	22.50	37.30	16.60	19.30	16.20	2.30	1.50	16.40	23.10	21.50	10.89	8.12	7.95	11.09	29.60	11.37	7.72	7.76
	60%	**0.28**	17.50	26.00	12.80	16.20	16.40	2.10	1.40	13.00	28.00	24.60	2.27	2.58	2.43	3.16	9.43	3.16	2.36	2.35
	90%	**0.25**	16.00	16.60	24.00	28.50	14.40	2.70	1.30	14.30	25.90	22.90	3.20	3.73	3.26	5.63	9.01	4.28	3.13	3.14
Monosines_vx	30%	**0.09**	31.90	45.20	32.50	34.40	10.00	1.70	0.74	21.10	0.91	2.00	5.90	6.24	5.83	6.55	0.90	0.46	5.63	5.66
	60%	**0.09**	26.60	15.70	30.40	35.40	3.50	1.70	0.56	14.00	0.59	0.72	4.44	4.80	4.43	5.45	0.80	0.32	4.43	4.42
	90%	**0.09**	17.10	5.30	32.00	25.10	2.30	1.50	0.49	7.70	0.47	0.46	5.31	3.60	3.44	4.71	0.35	0.33	3.53	3.53
Monosines_vy	30%	**0.12**	28.40	28.90	41.60	39.10	2.50	2.20	0.66	19.40	0.85	1.80	12.25	9.03	9.05	13.53	1.10	0.29	9.24	9.23
	60%	**0.12**	25.20	4.10	34.50	40.10	2.80	2.20	0.66	9.00	0.49	0.49	6.69	6.41	6.73	9.19	0.76	0.29	6.43	6.41
	90%	**0.12**	15.50	2.60	33.50	52.70	2.90	2.50	0.59	8.60	0.64	0.39	7.68	7.40	8.26	8.20	0.91	0.35	7.40	7.41
Monosines_p	30%	**0.07**	25.60	243.70	25.30	36.40	12.00	1.60	0.66	41.90	1.10	2.10	4.65	4.67	4.38	4.90	0.69	0.35	4.17	4.18
	60%	**0.07**	33.50	18.40	26.20	34.10	8.10	1.60	0.63	14.40	1.40	1.70	4.12	4.36	4.05	5.34	0.49	0.24	4.00	4.00
	90%	**0.07**	14.90	13.40	19.00	44.10	3.90	1.80	0.46	14.40	0.83	1.00	2.62	2.79	2.71	3.70	0.27	0.19	2.70	2.70
Monosines_pc1	30%	**0.06**	20.90	56.40	24.10	23.50	12.20	1.20	0.54	18.30	22.70	22.20	5.75	5.23	5.22	5.40	4.82	5.86	5.18	5.34
	60%	**0.05**	28.30	25.80	28.00	28.40	17.00	1.20	0.58	14.50	31.20	30.20	1.76	2.03	2.11	3.22	2.44	2.06	2.12	2.18
	90%	**0.05**	19.10	21.60	25.60	48.80	22.70	1.50	0.55	17.70	40.70	37.70	1.50	2.54	2.67	1.75	2.25	1.26	1.77	1.75
Wh_combined0	30%	**0.22**	42.10	57.60	36.60	24.90	35.90	2.30	1.70	21.70	36.60	33.60	2.17	2.18	2.70	2.70	2.17	2.28	1.74	1.82
	60%	**0.21**	30.40	38.50	24.20	25.10	26.90	2.30	1.50	18.40	35.70	31.10	2.14	2.13	2.63	2.50	1.90	2.15	1.60	1.66
	90%	**0.22**	16.70	19.30	17.50	25.70	24.70	2.40	1.30	14.20	32.50	26.70	2.65	2.07	2.61	1.87	2.04	1.99	1.70	1.68
Wh_estimation	30%	**0.23**	34.30	58.30	30.00	24.10	26.30	2.30	1.60	21.50	27.00	25.40	2.85	3.42	3.43	3.25	2.13	2.18	1.68	1.81
	60%	**0.22**	34.70	42.20	26.80	24.30	29.10	2.30	1.60	18.90	27.40	25.80	2.48	2.52	2.97	2.55	2.08	2.00	1.66	1.82
	90%	**0.23**	20.10	20.40	19.40	25.60	25.40	2.40	1.40	14.60	24.30	21.60	2.41	2.30	2.67	1.72	2.12	1.27	1.65	1.71
Wh_sineinmeas	30%	**0.08**	69.17	56.02	95.99	31.49	81.79	1.55	1.21	21.72	10.62	14.79	2.32	3.14	3.62	3.82	1.36	1.24	2.24	1.40
	60%	**0.06**	43.38	30.07	44.52	39.66	43.33	1.55	0.84	18.56	8.89	11.22	2.82	2.16	2.88	2.08	0.94	1.06	2.26	1.16
	90%	**0.07**	9.75	6.18	13.47	58.99	20.81	1.95	0.56	12.10	6.48	8.35	3.44	2.28	2.54	1.42	1.54	0.74	3.62	1.52
Wh_sinesweepin	30%	**0.05**	62.50	134.00	136.80	62.50	50.60	1.80	0.84	41.60	3.00	5.80	2.72	4.16	4.25	4.04	2.49	2.28	2.43	2.34
	60%	**0.03**	30.30	38.60	43.70	64.90	36.40	1.50	0.52	24.70	2.40	3.00	1.81	2.81	2.60	1.68	1.61	0.86	1.34	1.41
	90%	**0.04**	3.10	3.40	12.00	65.00	33.90	1.60	0.36	9.50	1.20	1.00	4.11	5.29	2.95	1.27	4.70	0.49	4.05	2.40
Wh_triangle2	30%	**0.29**	32.00	43.50	32.40	23.40	26.30	2.60	1.80	19.30	27.60	26.70	3.60	3.68	4.48	4.22	2.89	2.81	2.12	2.27
	60%	**0.28**	27.70	32.50	22.40	23.00	27.90	2.50	1.80	17.00	30.50	28.90	3.44	3.49	4.26	3.74	2.68	2.69	2.14	2.37
	90%	**0.28**	17.10	20.40	16.50	20.80	23.90	2.40	1.60	13.40	28.80	25.50	3.00	3.14	4.10	3.00	2.64	1.89	2.16	2.35

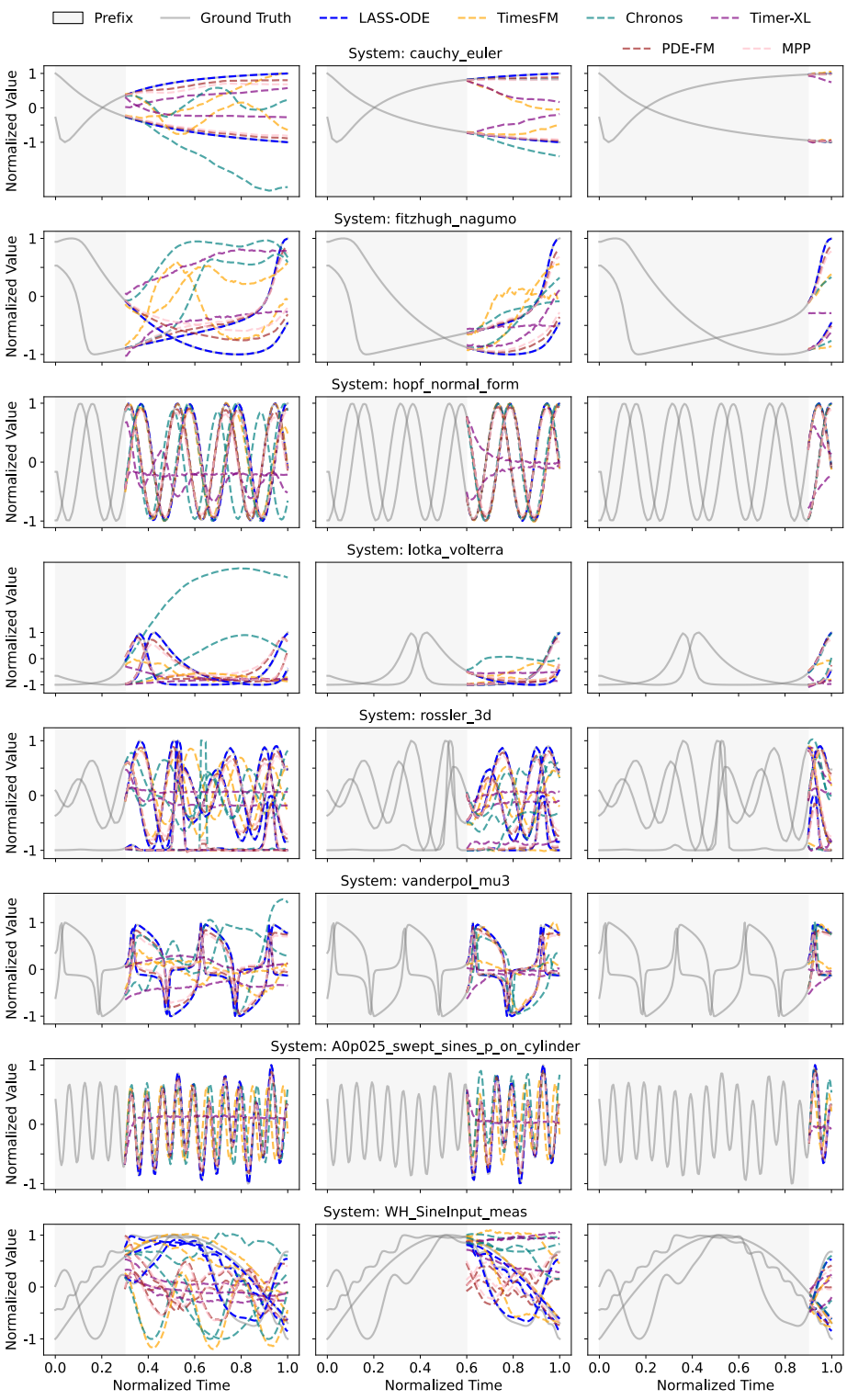

Figure 12: Examples of trajectories generated by time-series foundational models on the in-domain test set.

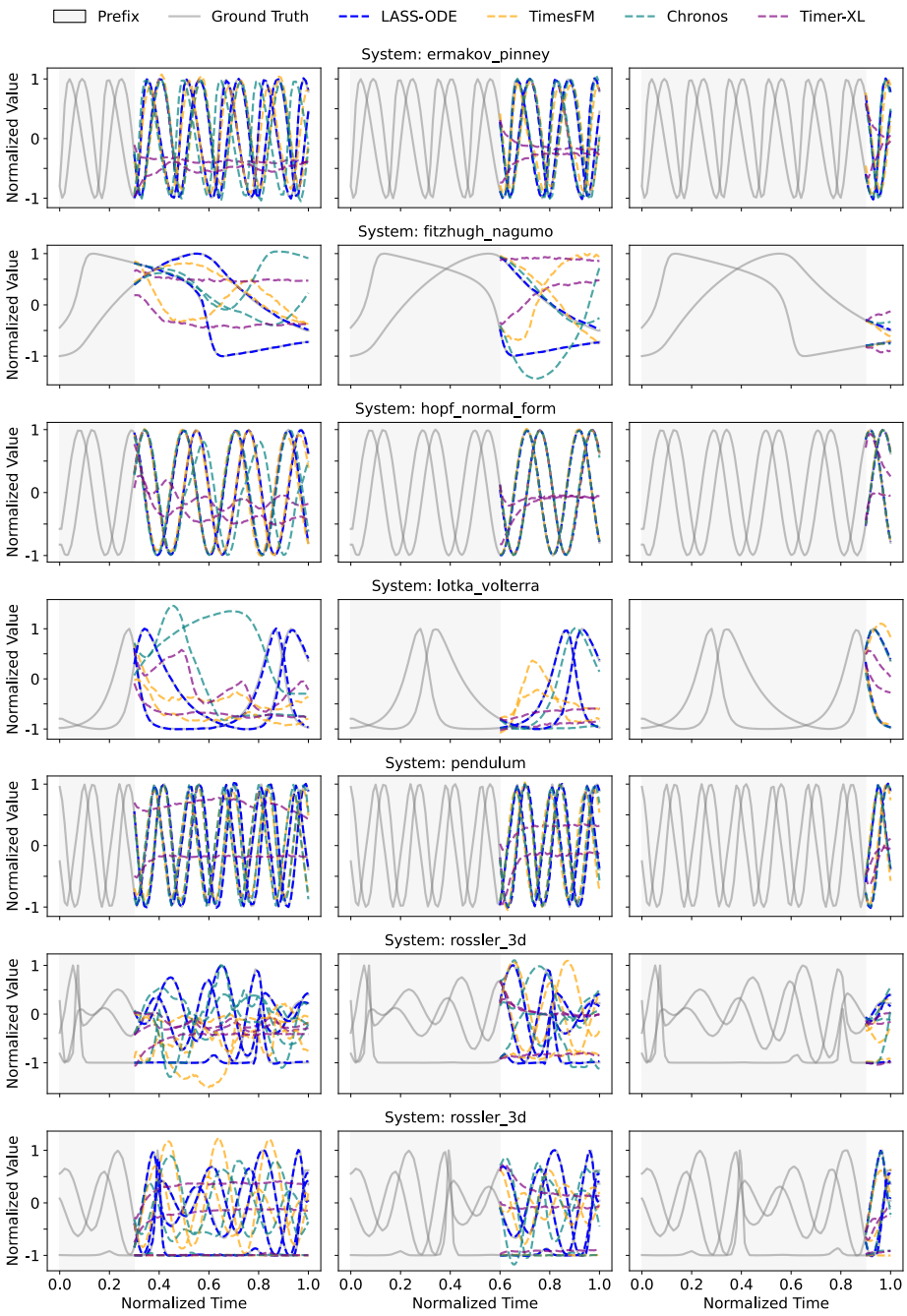

Figure 13: Examples of trajectories generated by time-series foundational models on the in-domain test set.

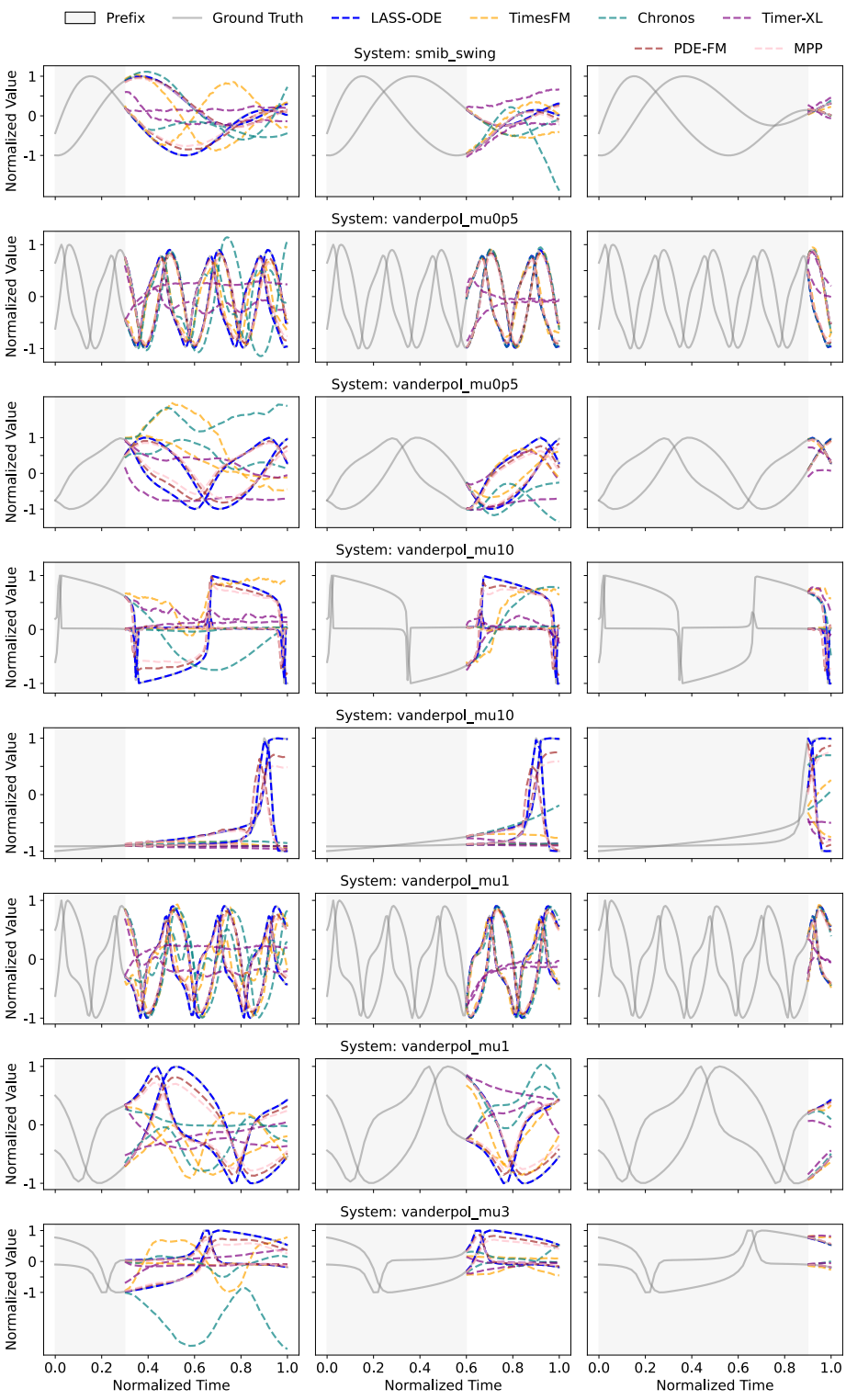

Figure 14: Examples of trajectories generated by time-series foundational models on the in-domain test set.

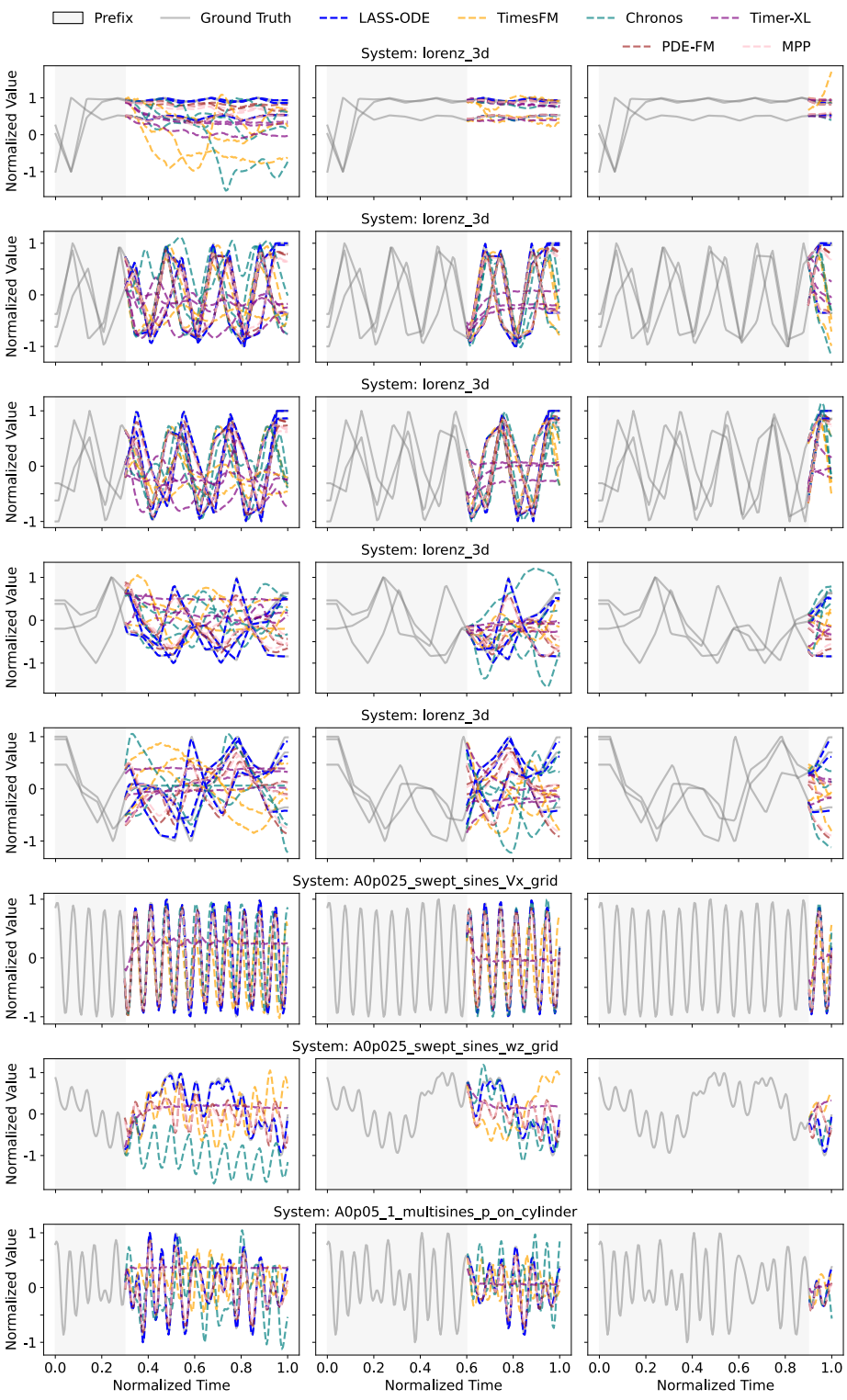

Figure 15: Examples of trajectories generated by time-series foundational models on the in-domain test set.

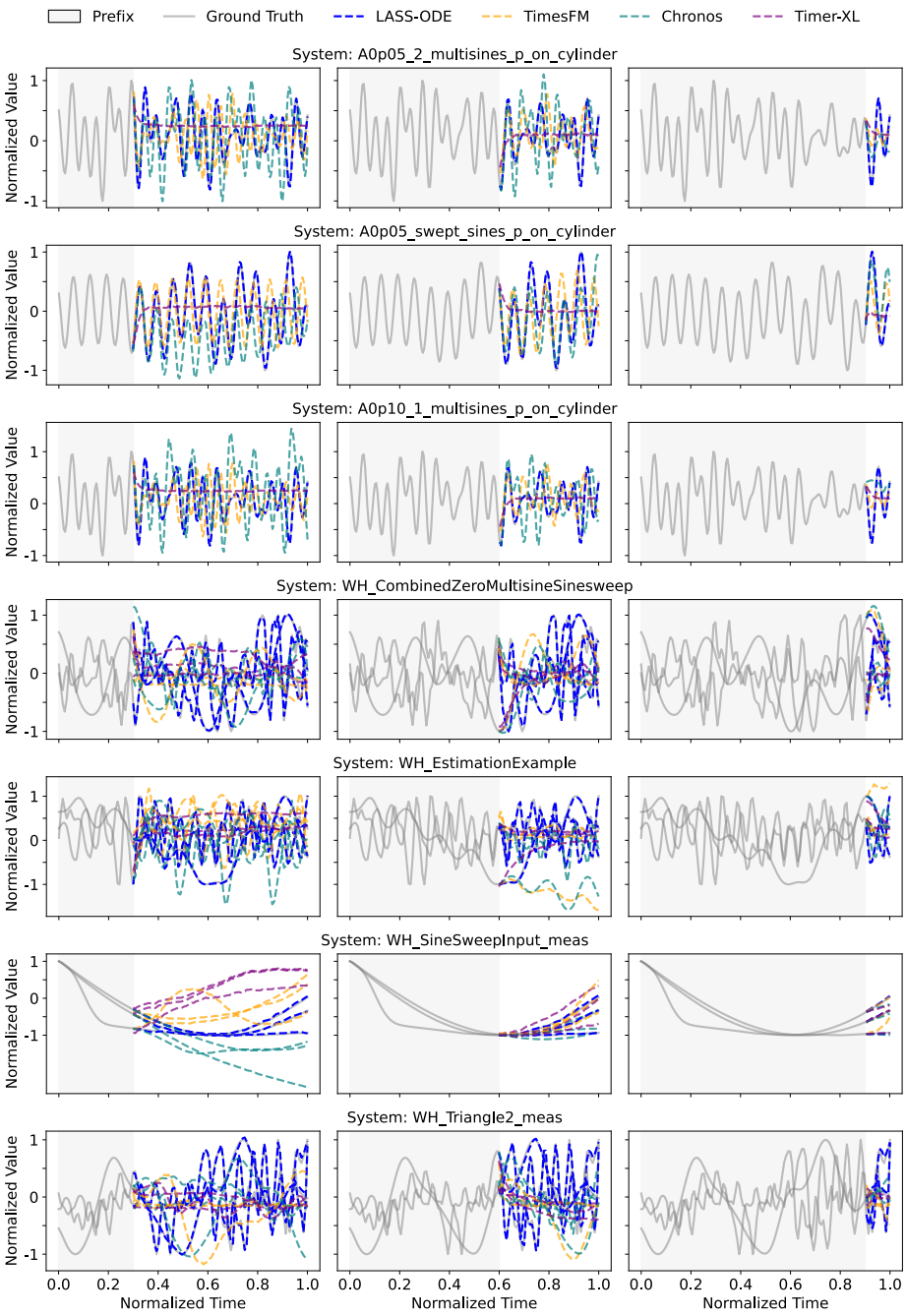

Figure 16: Examples of trajectories generated by time-series foundational models on the in-domain test set.

J RESULTS OF ZERO-SHOT ODE RESULTS

In this section, we show the complete zero-shot tests on systems that never appear in the training dataset. Table 6 reports the results. For most of the systems, LASS-ODE can directly output accurate predictions. Meanwhile, there are also systems that are hard to fit. All methods nearly fail in these systems. However, we find that with around 30 epochs of fine-tuning, LASS-ODE can quickly be able to capture the underlying ODE structure and achieve accurate extrapolation, as shown in Fig. 17.

Table 6: Test MSE ($\times 10^{-2}$) for zero-shot systems.

ODE Systems	Prefix ratio	**LASS-ODE** (Ours)	**TimesFM** (2024)	**Chronos** (2024)	**Timer-XL** (2024b)	**Informer** (2021)	**Autoformer** (2021)	**Latent ODE** (2019)	**Latent MoS** (2025)	**ContiFormer** (2023)	**PDE-FM** (2025)	**MPP** (2024)	**CoDA-NO-1D** (2024)	**PROSE-ODE** (2024c)	**PROSE-PDE** (2025)	**Poseidon-0D** (2024)	**S4D-style SSM** (2022)	**Mamba** (2023)	**CfC** (2022)	**LTC** (2021)
Zero-shot Test																				
Duffing	30.0%	**1.39**	67.71	80.24	32.67	20.46	40.22	38.45	47.91	48.35	43.05	47.86	43.38	32.85	35.40	30.42	52.17	45.06	28.11	30.03
	60.0%	**1.26**	46.77	56.98	31.43	39.72	43.61	37.15	44.41	41.23	41.20	45.93	44.94	39.66	40.86	25.77	47.88	35.37	25.23	27.00
	90.0%	**1.36**	13.98	26.46	32.54	60.41	19.17	38.15	29.92	30.93	33.15	35.57	32.79	37.20	43.26	20.25	38.31	21.03	25.71	26.28
Emden_fowler	30%	**7.30**	34.30	69.50	60.10	33.50	9.90	15.60	10.60	38.50	19.40	23.70	17.41	35.00	39.13	32.00	9.98	8.41	27.66	37.17
	60%	**3.80**	4.30	16.50	12.80	57.50	16.90	19.70	7.40	28.20	17.50	20.20	4.54	8.36	8.48	12.20	5.09	4.76	12.72	14.61
	90%	2.40	3.60	9.40	**0.89**	58.00	3.60	19.30	3.20	3.50	17.10	19.80	2.21	4.06	3.92	6.67	3.07	2.47	7.50	8.14
Hamiltonian_sho	30%	**0.97**	29.30	58.80	74.20	50.60	6.10	70.20	17.50	57.40	79.90	64.10	35.88	21.92	20.52	20.88	28.32	4.52	19.56	20.40
	60%	**0.91**	30.20	39.10	54.20	50.50	4.20	70.00	10.20	10.80	65.10	47.90	39.16	22.88	21.84	22.72	33.52	1.00	21.28	22.08
	90%	**0.86**	10.60	20.20	23.00	50.50	2.80	70.00	8.20	2.40	133.00	101.00	12.32	10.68	9.72	9.76	12.00	0.91	9.56	10.76
Pendulum_damped	30%	**3.50**	19.50	48.70	19.30	8.20	5.90	13.30	16.30	18.40	22.20	19.60	24.56	18.28	18.40	22.28	59.82	72.08	17.50	17.66
	60%	**0.80**	7.80	11.70	8.00	4.40	2.20	9.60	10.90	7.60	24.80	20.20	1.72	2.58	1.72	2.02	5.96	9.72	1.60	1.58
	90%	**0.20**	2.10	0.96	1.60	3.10	2.90	7.90	8.80	2.30	24.80	19.60	0.38	0.70	0.64	0.24	1.60	1.24	0.22	0.22
Multisines_pc4	30.0%	**7.70**	16.60	50.00	23.50	24.80	35.70	24.90	24.70	32.90	28.00	37.50	35.48	24.03	25.51	49.22	54.47	30.83	26.77	28.42
	60.0%	**8.70**	16.50	26.80	20.20	25.00	21.80	25.40	22.80	25.80	26.40	31.40	30.78	26.01	24.76	24.47	36.54	34.07	23.83	23.91
	90.0%	**8.30**	15.00	28.60	35.70	54.40	38.70	37.30	25.00	39.00	27.10	30.70	26.96	31.25	21.60	26.92	35.66	24.40	20.43	20.96
Sweptsines_pc4	30%	**0.04**	19.81	51.98	22.31	20.82	16.64	0.94	0.48	16.17	5.19	10.48	3.34	3.46	3.30	4.30	0.58	0.19	3.19	3.20
	60%	**0.04**	30.60	23.06	27.51	29.83	23.54	1.05	0.52	13.79	4.73	8.31	2.38	2.49	2.29	3.26	0.48	0.09	2.32	2.32
	90%	**0.04**	23.49	34.57	25.88	49.09	30.16	1.36	0.52	20.82	3.85	7.53	4.72	2.66	2.75	4.65	0.53	0.23	2.78	2.80
Monosines_wz	30%	**2.30**	21.60	52.60	27.10	28.20	14.10	8.00	5.10	28.10	25.80	24.60	6.65	11.47	9.55	5.41	8.73	7.48	11.54	8.78
	60%	**2.40**	27.40	30.70	23.90	25.70	13.10	7.80	5.50	21.80	30.00	27.80	7.92	23.20	17.12	6.52	8.45	8.13	11.73	9.07
	90%	**2.30**	16.00	12.60	14.80	31.50	3.10	8.50	3.60	16.80	50.10	46.80	4.45	12.91	11.70	5.15	6.96	5.66	6.91	5.80
Monosines_pc2	30%	**0.42**	22.50	34.00	24.00	21.80	24.40	3.00	2.10	17.60	28.90	27.50	5.16	4.64	4.51	4.37	5.32	6.59	4.76	4.93
	60%	**0.44**	38.50	42.20	31.40	31.60	36.80	3.70	2.60	23.10	38.50	36.90	3.69	4.43	4.87	6.10	6.01	7.21	4.68	4.81
	90%	**0.45**	30.80	32.20	27.70	60.10	33.60	5.20	2.50	28.70	43.50	40.00	5.40	7.04	7.95	4.73	6.63	4.58	4.78	4.74
Monosines_pc3	30%	**0.05**	19.76	29.56	22.30	20.79	16.64	0.97	0.49	12.95	5.85	10.76	10.80	9.82	9.80	10.14	9.04	11.00	9.70	10.02
	60%	**0.04**	30.55	26.36	27.47	30.61	23.34	1.08	0.52	14.77	5.11	8.93	3.34	3.86	4.02	6.14	4.64	3.90	4.04	4.16
	90%	**0.04**	23.31	38.02	25.75	51.48	29.98	1.40	0.52	22.17	4.34	7.81	2.86	4.84	5.08	3.32	4.28	2.40	3.36	3.32
Monosines_pc4	30%	**0.42**	22.50	28.40	24.00	21.80	24.40	3.00	2.10	16.30	0.76	1.70	9.59	10.27	9.25	12.16	2.88	1.27	9.07	9.10
	60%	**0.44**	38.50	36.20	31.40	34.40	36.90	3.90	2.60	22.70	0.85	1.80	6.27	6.61	5.93	8.40	2.20	0.79	6.02	6.01
	90%	**0.45**	30.80	38.00	27.70	60.10	33.60	5.20	2.50	30.70	0.62	1.60	4.13	2.27	2.27	3.93	1.17	0.56	2.32	2.34
Wh_multisine	30%	**10.70**	46.50	42.50	49.00	23.50	44.00	25.10	33.00	30.20	30.80	29.70	33.45	37.52	44.35	25.32	26.01	16.83	20.09	22.05
	60%	**11.20**	39.20	40.70	28.60	24.60	23.50	25.40	27.80	30.30	28.10	26.70	28.93	29.03	32.87	30.59	23.74	20.01	16.74	19.43
	90%	**12.30**	26.20	23.60	28.80	24.00	36.20	24.80	26.50	24.00	30.20	26.90	26.58	29.95	33.05	30.48	21.89	21.77	16.49	17.34
Wh_triangle	30%	**16.00**	31.60	33.40	32.70	22.80	28.30	24.30	27.50	26.90	27.90	26.70	27.89	27.58	31.31	33.21	21.38	22.56	16.84	17.69
	60%	**13.50**	27.90	30.10	24.00	22.60	33.50	24.30	27.80	25.70	28.90	27.10	22.24	23.36	25.73	24.86	17.11	16.84	14.07	15.98
	90%	**10.80**	19.50	31.90	21.90	26.20	27.40	26.00	24.70	28.30	25.50	22.70	14.16	16.23	19.08	12.49	11.67	11.41	11.05	11.40
Wh_zeromeas	30%	**9.10**	27.80	36.90	39.50	51.80	22.10	46.80	34.20	44.30	42.40	39.70	39.40	29.78	33.12	38.72	49.92	47.06	37.07	49.75
	60%	**8.40**	35.10	35.60	34.10	51.70	31.10	47.90	39.50	43.90	38.90	51.40	18.51	20.45	21.65	25.90	35.70	30.45	26.35	31.98
	90%	**5.90**	8.70	16.70	31.00	46.40	14.20	46.20	26.00	30.80	30.10	66.50	7.73	9.85	9.19	10.16	19.74	6.42	12.52	11.94

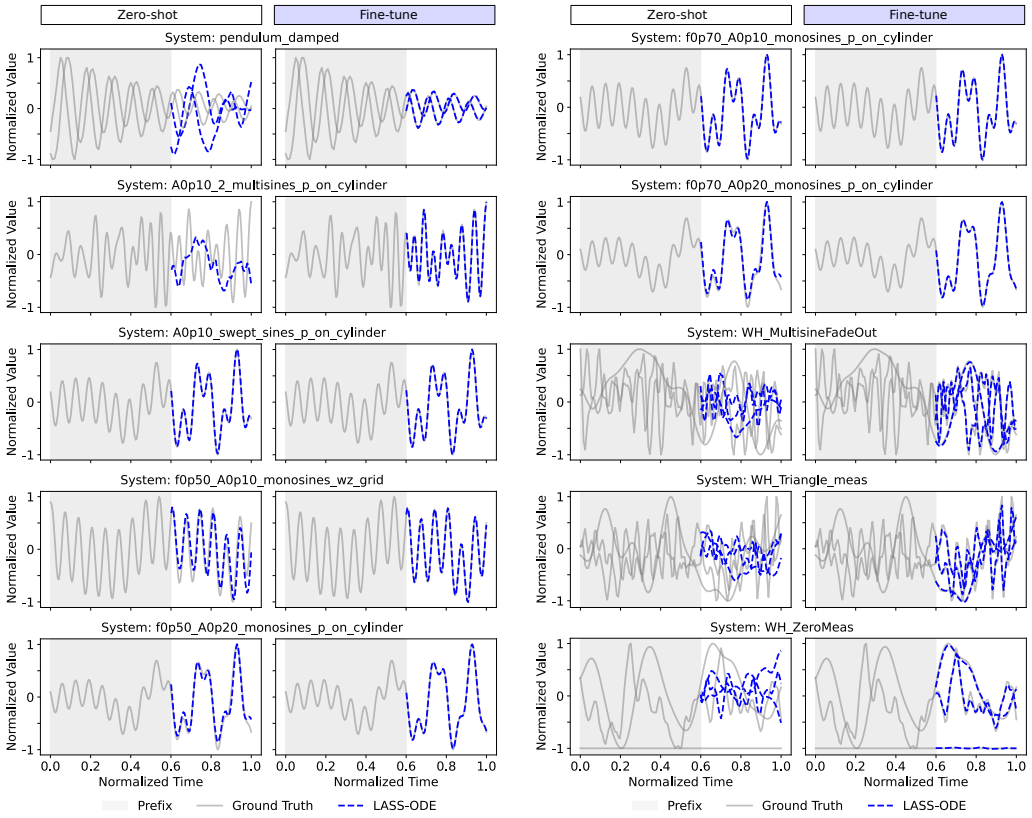

Figure 17: Performance comparison of the same base system under zero-shot inference (left) and after LoRA fine-tuning (right) for 30 epochs. The prefix ratio is set to 60%.

K LoRA-based Fine-tuning

To evaluate the adaptability of LASS-ODE to target systems, we further perform lightweight LoRA fine-tuning. Specifically, we fine-tune the MHA weight matrices using system-specific trajectories and test on new trajectories from the same ODE type with unseen initial conditions and ODE parameters. Figs. 3(b)–(c) compare zero-shot and fine-tuned predictions on Duffing system, and Fig. 3(d) shows the fine-tuning loss curve. LASS-ODE adapts rapidly and achieves accurate predictions with only lightweight updates. Table 7 reports the test MSE on the fine-tuned systems, with additional visualizations provided in Fig. 17.

Table 7: Test MSE ($\times 10^{-2}$) for fine-tuned systems.

ODE Systems	Prefix ratio	**LASS-ODE** (Ours)	**TimesFM** (2024)	**Chronos** (2024)	**Timer-XL** (2024b)	**PDE-FM** (2025)	**MPP** (2024)	**CoDA-NO** 1D (2024)	**PROSE** ODE (2024c)	**PROSE** PDE (2025)	**Poseidon** 0D (2024)	**S4D-style** SSM (2022)	**Mamba** (2023)	**CfC** (2022)	**LTC** (2021)
Multisines_pc4	30.0%	**0.197**	1.882	8.473	6.006	1.213	1.731	2.974	1.981	2.062	5.451	5.012	3.284	2.401	3.208
	60.0%	**0.194**	1.779	7.212	4.816	1.185	1.659	2.521	2.238	2.463	2.593	3.396	3.129	2.067	2.776
	90.0%	**0.191**	1.693	1.853	3.248	1.171	1.615	3.215	3.170	2.524	2.228	4.199	2.348	2.401	2.274
	Avg.	**0.194**	1.785	5.846	4.690	1.190	1.668	2.903	2.463	2.350	3.424	4.202	2.920	2.290	2.753
Duffing	30.0%	**0.218**	6.415	11.20	11.75	3.313	5.217	6.078	3.718	4.886	4.242	6.759	6.564	3.112	3.384
	60.0%	**0.182**	6.783	7.234	10.01	3.201	4.736	4.862	5.329	4.952	3.159	5.576	4.090	3.079	2.724
	90.0%	**0.194**	3.134	3.477	5.910	2.580	2.859	4.189	4.591	5.892	2.357	5.681	2.137	2.722	2.936
	Avg.	**0.198**	5.444	7.304	9.223	3.031	4.271	5.043	4.546	5.243	3.253	6.005	4.264	2.971	3.015

L CSH EFFECTIVENESS IN ATTENTION

To understand how the CSH enables cross-system knowledge transfer, we visualize the first-layer attention map between ODE system tokens and CSH tokens. Fig. 18 shows four attention patterns. First, ODE system tokens exhibit strong intra-system attention, indicating that the model preserves local dynamical interactions within each individual system. Second, system-token queries also attend to CSH tokens with non-trivial attention weights, suggesting that system tokens retrieve reusable dynamical structures from the shared prototype bank. Third, CSH token queries attend to system tokens, allowing CSH tokens to aggregate system-dependent evidence. Finally, the attention among CSH prototypes remains limited, preventing redundant prototype-to-prototype communication and avoiding collapse into a single global representation.

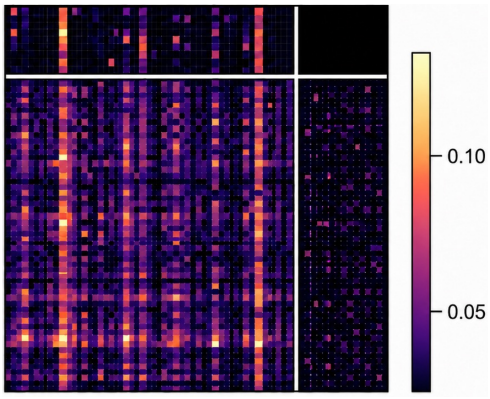

Figure 18: Attention patterns between system-specific tokens and CSH prototypes.

Together, these attention patterns support the intended role of CSH as a compact memory of recurring dynamical structures: system tokens preserve individual dynamics while selectively interacting with shared CSH tokens for cross-system knowledge transfer.

M VISUALIZATION OF ITERATIVE DYNAMICAL REASONING IN THE TEST INFERENCE

We further visualize the recurrent reasoning process by decoding the model prediction after each of six inference iterations, $r = 0, \ldots, 5$. At iteration r, the current representation produces an intermediate dynamical characterization $\widehat{\boldsymbol{\theta}}^{(r)} = \mathrm{Est}(E^{(r)})$, which is fed back to refine the representation $E^{(r+1)} = \mathrm{Ref}\left(E^{(r)}, \widehat{\boldsymbol{\theta}}^{(r)}; E^{(0)}\right)$.

We therefore obtain an iteration-wise sequence of trajectory predictions $\{\widehat{\boldsymbol{x}}^{(r)}(\tau)\}_{r=0}^{5}$ from the same trained model and the same test instance. The purpose of this visualization is to reveal whether recurrent inference changes the model's *dynamical hypothesis*, rather than merely applying additional computation to reduce pointwise prediction error.

Across representative ODE systems, the intermediate predictions exhibit a coarse-to-structured refinement of the inferred dynamics. Early iterations may fail to reproduce even the correct dynamical family or global trajectory topology. Subsequent iterations recover system-specific motifs—e.g., recurrence and double-scroll organization in Lorenz dynamics, relaxation structure in Van der Pol, coupled phase relationships in Lotka–Volterra, fast–slow transitions in FitzHugh–Nagumo, and timescale separation in stiff Robertson kinetics—before refining the remaining local geometry. Conceptually, the observed progression is

$$\text{coarse / incorrect dynamics} \;\rightarrow\; \text{correct dynamical family} \;\rightarrow\; \text{system-specific motif} \;\rightarrow\; \text{refined local geometry}. \tag{78}$$

This behavior provides a direct qualitative view of the intended role of dynamical-characterization feedback: intermediate physical inference is reused to revise the latent representation, allowing the model to progressively correct *how it interprets the underlying dynamics*, rather than only correcting the final state trajectory.

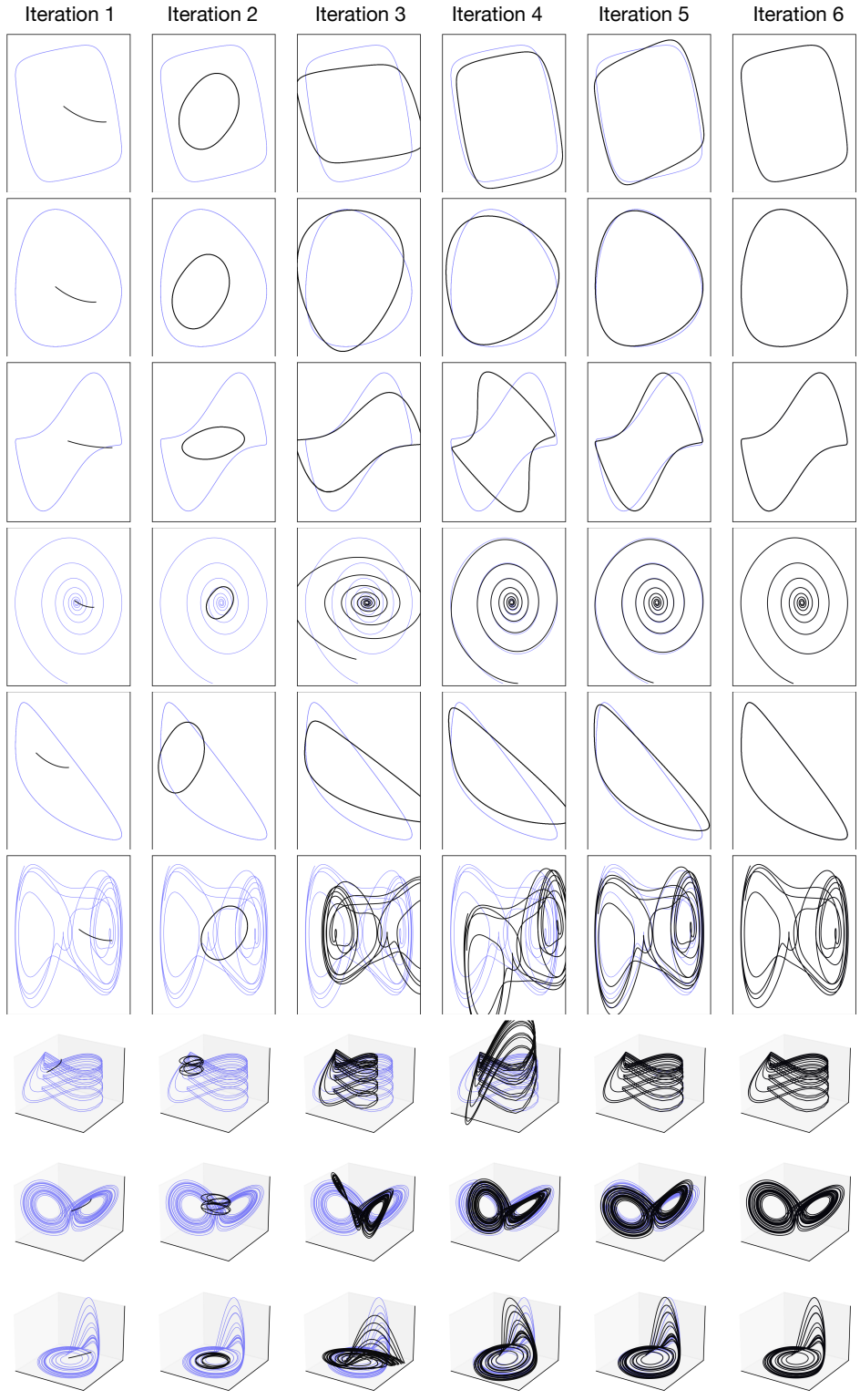

Figure 19: Intermediate trajectory predictions across six recurrent inference iterations for representative ODE systems. Each sequence is produced by the same trained model on the same test trajectory. The examples illustrate progressive refinement from an initially incorrect dynamical hypothesis toward the characteristic system-specific structure of the target ODE.

N CONVERGENCE OF LASS-ODE TO GROUND-TRUTH ODE TRAJECTORIES OVER TRAINING

This section provides a qualitative visualization of how LASS-ODE progressively learns the underlying dynamics of ODE-governed time-series during training. Each row in Fig. 20 corresponds to a distinct dynamical system from the training set, while columns show model predictions at increasing training epochs (0, 150, 300, 450, 600, 900, and 1500). Within each panel, the ground-truth trajectory is plotted alongside the trajectory reconstructed by LASS-ODE. At early epochs (0, 150, and 300), the predicted trajectories exhibit noticeable geometric and phase mismatches, reflecting the model's limited understanding of the governing dynamics at this stage. As training proceeds, the predicted curves gradually capture the global shape, local curvature, and long-term evolution patterns of the true trajectories. By later epochs (900 and 1500), LASS-ODE produces trajectories that closely align with the ground truth across a variety of dynamical regimes, indicating that the model has internalized system-level structural information rather than merely fitting short-term signal patterns.

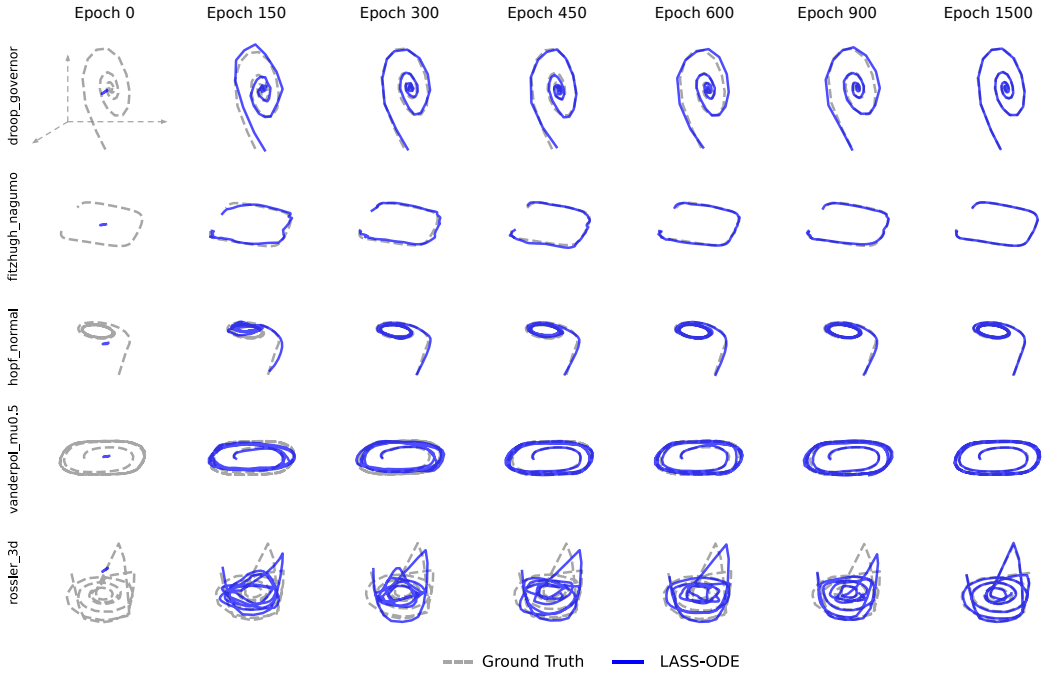

Figure 20: Progressive learning of ODE trajectories by LASS-ODE across training epochs. Each row corresponds to a different ODE system from the training dataset, and each column shows model predictions at increasing training epochs (0, 150, 300, 450, 600, 900, and 1500). In each panel, the ground-truth trajectory is plotted together with the trajectory reconstructed by LASS-ODE.

O Real-World and Realistic Power-System Evaluation Details

We further evaluate LASS-ODE on two real-world industrial dynamical systems. The first is generator dynamics in power plants, where generator dynamic states, including rotor-angle/speed estimates and bus frequency, are obtained from synchronized PMU-based monitoring at 30 samples/s. Each transient trajectory spans approximately 20 seconds and contains 600 time points with three measured dimensions. We use approximately 3×10^4 sequences, corresponding to around 200MB of data, for fine-tuning.

The second is voltage dynamics in data centers, where voltage, current, and power are measured at a shared power bus at 1kHz. Each transient window spans approximately 50ms and contains 50 time points with three measured dimensions. We use approximately 1.8×10^6 sequences, corresponding to around 1GB of data, for fine-tuning. For both systems, we vary the observed prefix ratio from 10% to 90% and reconstruct the full trajectory. Table 1 reports the MSE. LASS-ODE substantially outperforms representative baselines from different comparison groups, including TimesFM, Latent ODE, Informer, and ContiFormer.

In addition, we evaluate realistic power-system frequency dynamics under transient overfrequency and underfrequency events. Specifically, we use four IEEE-39 bus cases: Bus 30 and Bus 31 correspond to overfrequency events, while Bus 32 and Bus 33 correspond to underfrequency events. For the overfrequency cases, we condition on the first 20% of the trajectory; for the underfrequency cases, we condition on the first 40%. Given the observed prefix, each model reconstructs the remaining normalized frequency trajectory. Fig. 5 compares LASS-ODE with TimesFM, Chronos, Timer-XL, PDE-FM, and MPP. Across both overfrequency and underfrequency events, LASS-ODE closely follows the ground-truth dynamics after the observed prefix, while the baselines show larger deviations under transient oscillations. These results further demonstrate that LASS-ODE can capture realistic frequency dynamics and generalize across different buses and disturbance types.

P NOISY-INPUT TESTING

We evaluate noisy-input robustness by injecting Gaussian noise into the observed input prefix while keeping the target output trajectory clean. This setting tests whether each model can recover the underlying clean dynamics from corrupted observations. In the test, we select 10 ODE systems with a prefix of 60%, and inject different levels of noise. We consider signal-to-noise ratios (SNRs) from 60dB to 10dB and report representative results under the noiseless, 60dB, and 10dB settings. Table 8 shows that LASS-ODE remains robust under both zero-shot and fine-tuning settings. In particular, the zero-shot MSE of LASS-ODE increases only from 4.86 to 6.11 $(\times 10^{-2})$ when the input changes from noiseless to 10dB noise, while representative baselines degrade much more significantly.

Table 8: Noisy-input robustness results. MSE values are reported as $(\times 10^{-2})$. Gaussian noise is injected into the input prefix, while the target output trajectory remains clean.

	Zero-shot			Fine-tuning		
Method	Noiseless	60dB	10dB	Noiseless	60dB	10dB
LASS-ODE	4.86	5.13	6.11	0.10	0.11	0.16
TimesFM	30.49	33.42	57.48	22.72	29.35	50.81
Latent ODE	20.02	21.25	31.08	5.76	5.91	9.81
Informer	37.38	40.12	66.06	28.12	33.83	47.78
ContiFormer	22.39	23.98	37.78	16.06	17.65	27.86