Structured Temporal Inference in Hybrid State-Space Models

Hamidreza Hashempoor
Institute for AI, University of Stuttgart

Abstract

We propose a framework for structured temporal inference in nonlinear state-space models (SSMs) with hybrid latent dynamics that mix discrete and continuous variables. Our method follows a two-stage inference: continuous states are estimated via Kalman inspired updates, while discrete variables are sampled by a neural model conditioned on these states, avoiding explicit Markov assumptions. To handle instabilities arising from recurrent dynamics, we introduce stabilization approach, and train all components jointly using surrogate gradient estimators that support REINFORCE-style updates. This design achieves SOTA results across synthetic and real-world datasets, in state estimation, regime detection, and imputation under noise and partial observability.

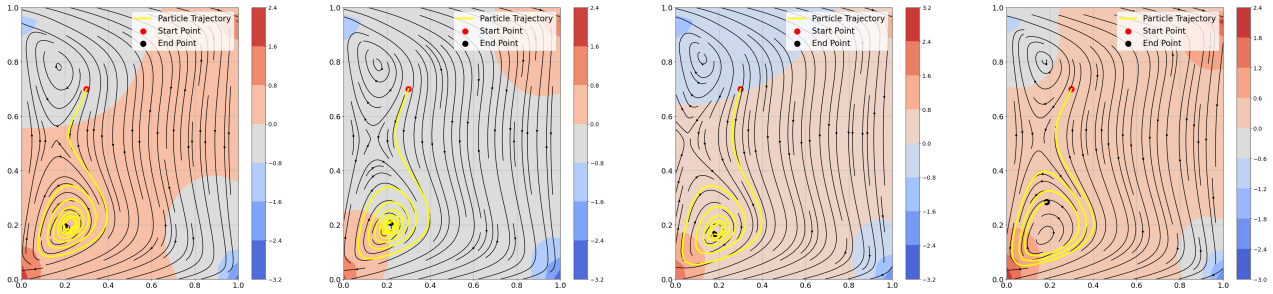
1 INTRODUCTION

SSMs provide a flexible probabilistic framework for modeling sequential data generated by latent dynamical processes. The central goal is to infer hidden trajectories or denoise partial, corrupted observations. A broad range of inference strategies have been proposed, including sampling-based methods (Neal et al., 2011), variational approximations (Kingma and Welling, 2013), Expectation-Maximization (EM) (Bishop and Nasrabadi, 2006), and message passing via Belief Propagation (BP) (Koller and Friedman, 2009). For linear-Gaussian systems with Markovian structure, classical solutions such as the Kalman filter-smoother (KF)(Kalman, 1960; Rauch et al., 1965) offer closed-form updates. Extensions like the Extended and Unscented KF (EKF, UKF) (Wan and Van Der Merwe, 2000; Ljung, 1979) address mild nonlinearities in more complex environments. However, these methods are fundamentally Model-Based (MB): their success depends on accurate domain knowledge and well-specified dynamics. Moreover, their reliance on costly matrix inversions limits scalability in high-dimensional settings.

In this work, we present a fundamentally different perspective on latent temporal modeling: we introduce π-SSM, a hybrid state-space framework that jointly models continuous latent states and discrete switching dynamics. Unlike traditional approaches such as Switching Linear Dynamical Systems (SLDSs) or neural SSMs with fixed transition structures, π-SSM infers both the continuous state $\mathbf{x}_t$ and a discrete latent variable z_t that governs the system's mode at each timestep, where we parameterize $z_t \sim p(z_t|\mathbf{x}_{t-1})$ with a neural network, allowing the discrete regime to be selected based directly on the latent trajectory. This formulation removes the need for hand-designed mode transition priors and enables more flexible, context-aware mode detection. This modeling approach aligns with real-world systems where mode-switching behavior emerges from continuous physical processes—such as contact-driven transitions in mechanical systems or context-dependent localization in autonomous agents (Linderman et al., 2016).

To achieve this, we construct a two-stage, message-passing-inspired inference algorithm. First, the continuous latent states $\mathbf{x}_t$ are inferred using recursive Kalman-style updates, parameterized by a compact RNN. Second, given the inferred $\mathbf{x}_t$, we sample the discrete latent variable z_{t+1} from a neural posterior conditioned on $\mathbf{x}_t$. This structure enables us to approximate full message passing integration via nested sampling—effectively computing messages of the form $m_{f\to v}$, where f and v denote factors and variables in the factor graph model. Factors are instantiated based on the generative structure, and messages are constructed recursively to recover beliefs over latent states. Training is performed by maximizing the predictive (lower bounded) log-likelihood, where gradients with respect to continuous parameters are computed

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.



(a) Ground Truth (b) π-SSM (c) SPINN (d) PINN

Figure 1: Inferred 1k length particle trajectories in incompressible fluid with Navier-Stokes dynamics. Colorbar represents pressure field, directed lines are velocity fields. Yellow line is the trajectory.

via backpropagation, while those for discrete variables are estimated using REINFORCE with control variates to reduce variance (Sutton et al., 1999).

At the core of π-SSM lies a compact recurrent architecture that complements the classical Kalman update framework to enable efficient and scalable inference. Rather than relying on explicit expensive matrix computations at each step, our model uses RNN-based update mechanisms that streamline the data flow. This design allows inference to be performed linearly faster compared to traditional approaches, which is especially beneficial for long sequences or high-dimensional systems. However, the nonlinear dynamics introduced by recurrent networks can lead to instability during training, such as exploding gradients or sensitivity to bifurcation behavior (Pascanu et al., 2013). To address this, we propose a stability-aware training strategy that enforces input-to-state stability (ISS) conditions during the forward pass, and applies gradient manipulation techniques during backpropagation through time (BPTT) to ensure stable and consistent gradient flow through time.

To summarize the contributions, we introduce π-SSM, a novel state-space modeling framework that jointly infers discrete and continuous latent variables through structured, partially non-Markovian inference. We develop a nested sampling-based inference strategy inspired by message passing, combining Kalman-style continuous updates with neural discrete selection mechanisms. To optimize the model, we propose a theoretically grounded training objective based on predictive likelihoods, where continuous parameters are updated via BPTT, and discrete latent variables are trained jointly using REINFORCE with control variates. Finally, we propose a general stability-aware recurrent architecture that subsumes popular RNNs such as LSTM and GRU, enabling efficient inference while preserving stable training dynamics. We then validate our method through extensive experiments on both synthetic and real-world systems. These include: (1) a bouncing ball environment with switching dynamics, (2) chaotic Lorenz attractor sequences, (3) complex fluid-like systems generated by noisy Navier-Stokes PDEs (inferred system visualized in Figure 1), (4) real-world robot localization in the NCLT dataset, and (5) convergence tests across random seeds to assess training stability.

2 RELATED WORKS

Graphical models provide a powerful framework for representing complex dependencies in structured data. Besides classical filtering techniques, sampling-based inference methods such as MCMC algorithms—including Gibbs sampling (Gelfand, 2000)—and importance sampling variants (Friedman and Van den Broeck, 2018; Lou et al., 2019; Marinescu et al., 2019) have been widely used, offering the advantage that inference accuracy improves over time without additional memory cost. However, their improvement often slows significantly with computation time in practice (Bathla and Vasudevan, 2023). Another widely adopted approach is BP, which structures inference through local message passing. Yet in loopy or strongly coupled graphs, BP offers no general convergence guarantees and can produce inaccurate, overconfident marginals (Guo et al., 2024).

Kanai et al. (2017) and GIN (Hashempoorikderi and Choi, 2024)—a Markovian SSM without discrete modes— both analyze stability only for the GRU case under zero input, a strong and non-general assumption. Like π-SSM, hybrid GNN (Garcia Satorras et al., 2019), SSI (Ruhe and Forré, 2021), and KalmanNet Revach et al. (2021) employ recurrent mechanisms for state evolution, but generally rely on known (complete or partial) dynamics without any particular stability handling for RNNs.

The SLDS family of models provides a natural base-

line for hybrid discrete-continuous inference. Recent variants such as rSLDS (Linderman et al., 2016), SNLDS (Dong et al., 2020), REDSDS (Ansari et al., 2021), and the Bayesian nonparametric infinite rSLDS (Linderman et al., 2017) extend classical SLDS through recurrent parameterizations and improved inference schemes. Additionally, hybrid SSM approaches like HiP-RSSM (Shaj et al., 2022) and models using learned switching behavior such as LSD-VBS (Becker-Ehmck et al., 2019) offer alternative perspectives on combining recurrent updates with structured discrete mode transitions. While these methods address hybrid dynamics in various ways, they typically employ explicit Markovian transitions (e.g., $p(z_t \mid z_{t-1})$) or rely on classical Kalman gains, contrasting with our design choice of state-dependent discrete sampling $p(z_t \mid x_{t-1})$ and learned Kalman-style updates.

This section highlights the key related works. A more in-depth discussion—including system identification (SI) using the EM algorithm, auto-regressive (AR) models, SLDS, neural ODEs (e.g., NODE (Chen et al., 2018) and MoNODE (Auzina et al., 2024)), and PINNs (Raissi et al., 2019)—is available in Appendix D.1 and Table 4. In addition, we provide a comprehensive discussion of variational inference methods, including both classical and recent deep extensions, in Appendix D.1. Finally, we report an empirical runtime complexity analysis in Appendix D.3, comparing execution times per iteration across baselines using wall-clock measurements.

3 BACKGOUND

Message passing. Message passing offers a structured approach to inference in graphical models by propagating information between variables and factors through local update rules. In general, the message from a variable node $\mathbf{v}$ to a factor node f is given by $m_{\mathbf{v}\to f} = \prod_{f'\in \mathbf{ne}(\mathbf{v}\setminus f)} m_{f'\to \mathbf{v}}$ where $\mathrm{ne}(\mathbf{v}\backslash\mathbf{f})$ denotes the set of neighboring factors of $\mathbf{v}$ except f. Conversely, the message from a factor node f to a variable node $\mathbf{v}$ is computed by marginalizing over the other variables connected to f. When both discrete and continuous variables are involved, the message takes the form: $m_{f\to \mathbf{v}} = \sum_{\mathcal{D}_f} \int_{\mathcal{C}_f} \left(f(\mathbf{v}, \mathcal{D}_f, \mathcal{C}_f) \prod_{\mathbf{v}'\in \mathbf{V}'_{f\setminus \mathbf{v}}} m_{\mathbf{v}'\to f} \right) d\mathcal{C}_f$. Here $\mathbf{V}'_{f\setminus \mathbf{v}}$ is the set of all variables connected to f except $\mathbf{v}$, and $\mathcal{D}_f$, $\mathcal{C}_f$ denote the discrete and continuous subsets of $\mathbf{V}'_{f\setminus \mathbf{v}}$, respectively. Then the *Belief* of the variable $\mathbf{v}$ is given by $\mathrm{Belief}(\mathbf{v}) \propto \prod_{f\in \mathbf{ne}(\mathbf{v})} m_{f\to \mathbf{v}}$.This framework serves as the foundation for our approximate inference algorithms.

SLDS. SLDS models complex time series data by decomposing trajectories into sequences of simpler, locally Linear Gaussian SSM (LG). By fitting an SLDS to data, one can capture piecewise linear dynamics while simultaneously segmenting the sequence into coherent discrete regimes. The generative process is as follows. At each time step $t = 1, \ldots, T$, a discrete latent state $z_t \in 1, \ldots, K$ evolves according to a Markovian process, where the next state z_{t+1} is sampled from a categorical distribution conditioned on the current state as $z_{t+1} \sim \mathrm{Categorical}(p(z_{t+1}|z_t))$. Conditioned on z_{t+1}, the continuous latent state $\mathbf{x}_t \in \mathbb{R}^M$ evolves linearly as $\mathbf{x}_{t+1} = \mathbf{A}_{z_{t+1}}\mathbf{x}_t + \mathbf{b}_{z_{t+1}} + \mathbf{q}_{t+1}$ where $\mathbf{q}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{z_{t+1}})$, with matrices $\mathbf{A}_{z_{t+1}}, \mathbf{Q}_{z_{t+1}} \in \mathbb{R}^{M\times M}$ and bias vector $\mathbf{b}_{z_{t+1}} \in \mathbb{R}^M$. Finally, an observation $\mathbf{y}_t \in \mathbb{R}^N$ is generated linearly from $\mathbf{x}_t$ according to $\mathbf{y}_t = \mathbf{C}_{z_t}\mathbf{x}_t + \mathbf{d}_{z_t} + \mathbf{r}_t$ where $\mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{z_t})$, with matrices $\mathbf{C}_{z_t} \in \mathbb{R}^{N\times M}, \mathbf{R}_{z_t} \in \mathbb{R}^{N\times N}$ and bias vector $\mathbf{d}_{z_t} \in \mathbb{R}^N$. Full posterior approximate inference can be performed via conditional sampling, where a Gibbs sampling procedure alternates between discrete state updates using the forward-backward algorithm and continuous state updates via filtering and smoothing (see Appendix A for details).

4 π-SSM: GENERATIVE ASSUMPTIONS AND GRAPHICAL STRUCTURE

Most of the symbols follow the notation introduced in the Background section, including $\mathbf{x}_t$, z_t, $\mathbf{y}_t$, and the discrete-state dependent transition and emission matrices $\mathbf{A}_{z_t}$ and $\mathbf{C}_{z_t}$. We denote by $\boldsymbol{\mu}_{t|t}$ and $\boldsymbol{\Sigma}_{t|t}$ the mean and covariance of the filtered posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, and by $\boldsymbol{\mu}_{t|t-1}$ and $\boldsymbol{\Sigma}_{t|t-1}$ those of the predictive distribution $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. We use θ to refer to parameters governing the discrete latent variable, and ϕ for those associated with the continuous states. (A summary of all notations, categorized by discrete and continuous variables, is in Appendix Table 3 and Figure 7.) For simplicity, we omit explicit input terms in the transition and observation (e.g., $\mathbf{b}_{z_{t+1}}$, $\mathbf{d}_{z_t}$) and assume known mode-independent noise covariances $\mathbf{Q}_t$, $\mathbf{R}_t$, although these can be incorporated without loss of generality. The generative model jointly defines a distribution over continuous latent states $\mathbf{x}_{1:T}$, discrete mode indices $\mathbf{z}_{1:T}$, and observations $\mathbf{y}_{1:T}$. The factorization is given by:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{x}_0) \prod_{t=1}^{T} p(z_t|\mathbf{x}_{t-1}).p(\mathbf{x}_t|\mathbf{x}_{t-1}, z_t).p(\mathbf{y}_t|\mathbf{x}_t, z_t) \quad (1)$$

where $p(\mathbf{x}_0)$ is the prior over the initial continuous state, $p(z_t \mid \mathbf{x}_{t-1})$ denotes the state-dependent categorical distribution over discrete modes, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)$ is

the Gaussian transition model parameterized by the selected mode, and $p(\mathbf{y}_t \mid \mathbf{x}_t, z_t)$ represents the observation model. Importantly, unlike standard SLDS where discrete transitions are explicitly modeled via terms like $p(z_t \mid z_{t-1})$ (see eq.(16) in Appendix), in our factorization z_t depends only on the current continuous state $\mathbf{x}_t$ and is independent of previous discrete assignments. This structural design ensures that, when optimizing the model, each discrete variable z_t only influences the local observation $\mathbf{y}_t$ through the immediate continuous state $\mathbf{x}_t$, without requiring backward dependency propagation through $\mathbf{z}_{1:t-1}$. As a result, gradient updates for discrete variable z_t can be performed independently at each time step without necessitating fully back-propagate in time, enabling scalable training via local surrogate objectives such as REINFORCE (See Theorem 6). The graphical model is in Figure 2. Beyond training advantages, this factorization also plays a central role during inference. By preserving a clear local dependency structure between $\mathbf{x}_t$, z_t, and $\mathbf{x}_{t+1}$, the model enables a modular message-passing interpretation of inference. Each latent state $\mathbf{x}_t$ aggregates local information from associated variables $(\mathbf{y}_t, z_t, \mathbf{x}_{t-1})$ through messages determined by the corresponding conditional distributions. Furthermore, since some factors such as $p(z_{t+1} \mid \mathbf{x}_t)$ or $p(\mathbf{y}_t \mid \mathbf{x}_t, z_t)$ may not admit tractable parametric forms, the model naturally accommodates their approximation via flexible function approximators (e.g., neural networks). This dual capability—to support local message-passing-based inference while simultaneously enabling principled approximation of individual factors—forms the foundation of our hybrid inference strategy.

A natural concern is whether omitting the discrete Markov edge $p(z_t \mid z_{t-1})$ (used in SLDS-family models) sacrifices expressivity. Our key observation is that regime changes in the systems we consider are tightly coupled to changes in continuous dynamics, making $p(z_t \mid \mathbf{x}_{t-1})$ sufficient for accurate mode detection. Empirically, this is validated across our experiments: for instance, in the Pong environment, we observe sharp shifts in eigenvalues of the transition matrices during regime changes (aligned with ball-wall collisions), indicating that $\mathbf{x}_t$ carries strong signals about boundaries without explicit discrete persistence. Moreover, the state-dependent factorization enables fully online inference where modes are inferred sequentially, contrasting with SLDS-family methods that typically require forward-backward passes over entire trajectories. Our design thus trades explicit discrete mode persistence for scalability and real-time capability, while empirically maintaining effective segmentation. An analysis of our model's expressivity, limitations and extension compared with SLDSs appears in Appendix D.2.

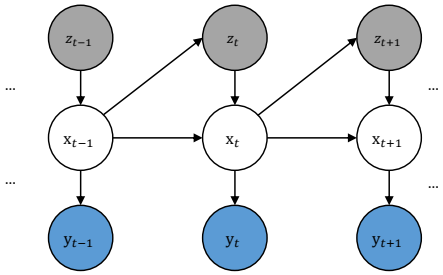


Figure 2: Grey and white nodes are latent representing discrete and continuous variables. Blue nodes are observables in continuous space.

5 APPROXIMATE INFERENCE FRAMEWORK

The goal of inference in our model is to approximate the posterior distribution $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{y}_{1:T})$ over the latent continuous states and discrete switching variables, given the observed sequence $\mathbf{y}_{1:T}$. The structured factorization in eq. (1) facilitates this by localizing dependencies: each discrete latent z_t depends only on the corresponding continuous state $\mathbf{x}_{t-1}$, and each continuous transition $\mathbf{x}_{t-1} \to \mathbf{x}_t$ is conditioned only on $(\mathbf{x}_{t-1}, z_t)$. This factorization enables us to design an inference procedure that naturally aligns with local message passing across time. To formalize this, we characterize the resulting belief updates and filtered posteriors using a nested message passing structure. Under the generative factorization in eq. (1), standard message passing yields beliefs over the latent variables $\mathbf{x}_t$ and z_t of the form: $\text{Belief}(\mathbf{x}_t) \propto m_{f_{\text{dyn}} \to \mathbf{x}_t} \times m_{f_{\text{obs}} \to \mathbf{x}_t}$, and $\text{Belief}(z_t) \propto m_{f_{\text{dyn}} \to z_t} \times m_{f_{\text{mode}} \to z_t}$.

where the incoming messages are given by (see Figure 3 for visualization of factor graph):

$$m_{f_{\text{dyn}} \to \mathbf{x}_t} = \sum_{z_t} \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, p(z_t \mid \mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1},$$

$$m_{f_{\text{obs}} \to \mathbf{x}_t} = p(\mathbf{y}_t \mid \mathbf{x}_t), \tag{2}$$

$$m_{f_{\text{dyn}} \to z_t} = \iint p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, \text{Belief}(\mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_t)\, d\mathbf{x}_{t-1}\, d\mathbf{x}_t, \tag{3}$$

$$m_{f_{\text{mode}} \to z_t} = \int p(z_t \mid \mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}. \tag{4}$$

And the posteriors are given by:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = p(\mathbf{y}_t \mid \mathbf{x}_t) \cdot \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})} \left[\mathbb{E}_{z_t \sim p(z_t|\mathbf{x}_{t-1})} \left[p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\right]\right] \propto \text{Belief}(\mathbf{x}_t) \tag{5}$$

$$p(z_t|\mathbf{y}_{1:t}) = \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})} \left[p(z_t \mid \mathbf{x}_{t-1}) \times \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}, z_t)} \left[p(\mathbf{y}_t \mid \mathbf{x}_t)\right]\right] \propto \text{Belief}(z_t) \tag{6}$$

Thus, local message passing over the graphical structure induces a nested inference procedure. Under the forward inference setting, the full posterior over all latent variables up to time t' approximately factorizes as: $p(\mathbf{x}_{1:t'}, z_{1:t'} \mid \mathbf{y}_{1:t'}) \approx \prod_{t=1}^{t'} p(\mathbf{x}_t, z_t \mid \mathbf{y}_{1:t}) \approx \prod_{t=1}^{t'} p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \times p(z_t \mid \mathbf{y}_{1:t})$, which follows directly from the recursive nature of the message updates and the assumed temporal independence in the forward direction. Detailed derivations/explanations provided in Appendix C.1.

The message passing formulations in eqs. (2-6) induce a recursive inference strategy that we refer to as nested message passing. While exact computation of the posterior terms may be intractable, the modular factorization enables tractable approximations via neural parameterizations. Specifically, we model $p(z_{t+1} \mid \mathbf{x}_t)$ with a neural network that produces a softmax distribution, denoted by $q(z_{t+1} \mid \mathbf{x}_t)$, mimicking a categorical distribution. This distribution is used to select the transition and emission parameters $\mathbf{A}_{z_t}$ and $\mathbf{C}_{z_t}$ corresponding to the inferred mode.

Having $z_t \sim q(z_t|\mathbf{x}_{t-1})$, we use $\mathbf{A}_{z_t}$ and $\mathbf{C}_{z_t}$ to approximate the filtered posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ with a Gaussian distribution $q(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{t|t}, \hat{\boldsymbol{\Sigma}}_{t|t})$, where both the mean and diagonal covariance are derived from a pseudo Kalman-style update. Specifically, we compute a gain matrix $\hat{\mathbf{K}}_t = \hat{\boldsymbol{\Sigma}}_{t|t-1}\mathbf{C}_{z_t}^T\mathbf{L}_t\mathbf{L}_t^T$, where the Cholesky-like factor $\mathbf{L}_t$ is produced by a RNN-based function: $\mathbf{L}_t = \mathrm{RNN}\big([\hat{\boldsymbol{\Sigma}}_{t|t-1}, \mathbf{r}_t]\big)$. The posterior parameters are then computed as:

$$\hat{\boldsymbol{\mu}}_{t|t} = \hat{\boldsymbol{\mu}}_{t|t-1} + \hat{\mathbf{K}}_t \left(\mathbf{y}_t - \mathbf{C}_{z_t}\hat{\boldsymbol{\mu}}_{t|t-1}\right),$$
$$\hat{\boldsymbol{\Sigma}}_{t|t} = \hat{\boldsymbol{\Sigma}}_{t|t-1} + \hat{\mathbf{K}}_t \left(\mathbf{C}_{z_t}\hat{\boldsymbol{\Sigma}}_{t|t-1}\mathbf{C}_{z_t}^T + \mathbf{R}_t\right)\hat{\mathbf{K}}_t^T. \quad (7)$$

This construction ensures the updated covariance remains positive definite and allows $\mathbf{y}_t$ to modulate the posterior via a learnable correction mechanism, moreover, this parameterization of the data flow is linearly faster than traditional LGs (see Appendix C.2 for details). The resulting $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ is propagated forward by sampling $z_{t+1} \sim q(z_{t+1} \mid \mathbf{x}_t)$ and evaluating the transition model $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, z_{t+1})$. Marginalizing over z_{t+1} yields a Gaussian predictive distribution $q(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{t+1|t}, \hat{\boldsymbol{\Sigma}}_{t+1|t})$, which serves as the basis for the next filtering step.

In practice, at each time step t, inference proceeds in two steps: (i) approximating $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ using forward message updates (eq.(5)) and replacing it with $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ via eq.(7), and (ii) replacing $p(z_t \mid \mathbf{x}_{t-1})$ with $q(z_t \mid \mathbf{x}_{t-1})$, followed by sampling $z_t \sim q(z_t \mid \mathbf{y}_{1:t})$ using the structure in eq. (6). While the inference at time t focuses on estimating $q(\mathbf{x}_t, z_t \mid \mathbf{y}_{1:t})$, the model can also sample $z_{t+1} \sim q(z_{t+1} \mid \mathbf{x}_t)$ and propagate to the next state via $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, z_{t+1})$ to construct a predictive prior for the next step. This defines a forward procedure that decomposes inference into a sequence of locally parameterized computations, enabling scalable and differentiable implementation while preserving the graphical model structure.

6 TRAINING

To train the model, we aim to maximize the marginal log-likelihood of the observed sequence $\mathbf{y}_{1:T}$. However, exact inference is intractable due to the presence of both discrete latent variables $\mathbf{z}_{1:T}$ and continuous latent states $\mathbf{x}_{1:T}$. To circumvent this, we construct a tractable approximation to the predictive likelihood at each time step . Specifically, we define a local evidence surrogate: $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) = \mathbb{E}_{\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{x}_t}$ $[p(\mathbf{y}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{z}_t)\, q(\mathbf{z}_t \mid \mathbf{x}_{t-1})\, q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})]$ which approximates the intractable predictive density $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ via nested sampling over latent variables. The full training objective is then given by the sum of predictive log-likelihoods $\mathcal{L}(\mathbf{y}_{1:T}) = \sum_{t=1}^{T} \log\, q(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ which can be efficiently estimated via Monte Carlo sampling of the latent variables. This formulation supports online training and enables local credit assignment, without requiring gradient propagation through the entire sequence. In the following Theorem, we state how much our approximated objective is lower bounded. The proof is in Appendix C.3.

Theorem 1 (Lower Bound on Predictive Log-Likelihood)**.** *The surrogate predictive log-likelihood defined in $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ provides a lower bound on the true predictive likelihood at each time step. Specifically, $\log p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) \geq \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) + \mathcal{E}_t$, where the gap term $\mathcal{E}_t$ is given by the sum of KL divergences of approximate and true latent distributions: $\mathcal{E}_t = \mathrm{KL}\left(q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \,\|\, p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\right) + \mathbb{E}_{\mathbf{x}_{t-1}}\left[\mathrm{KL}\left(q(\mathbf{z}_t \mid \mathbf{x}_{t-1}) \,\|\, p(\mathbf{z}_t \mid \mathbf{x}_{t-1})\right)\right]$*

To optimize the surrogate objective $\mathcal{L}(\mathbf{y}_{1:T})$, we estimate gradients with respect to both continuous and discrete variational parameters. Denoting by θ the parameters of the discrete inference model $q(z_t \mid \mathbf{x}_{t-1})$, and by ϕ the parameters of the continuous filtering model $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, we compute:

$$\nabla_\phi \mathcal{L}(\mathbf{y}_{1:T}) \approx \sum_{t=1}^{T} \nabla_\phi \log\, q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) \quad (8)$$

$$\nabla_\theta \mathcal{L}(\mathbf{y}_{1:T}) \approx \sum_{t=1}^{T} \mathbb{E}_{z_t \sim q(z_t|\mathbf{x}_{t-1})} \left[(\log\, q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) - b_t)\, \nabla_\theta \log q(z_t \mid \mathbf{x}_{t-1})\right]. \quad (9)$$

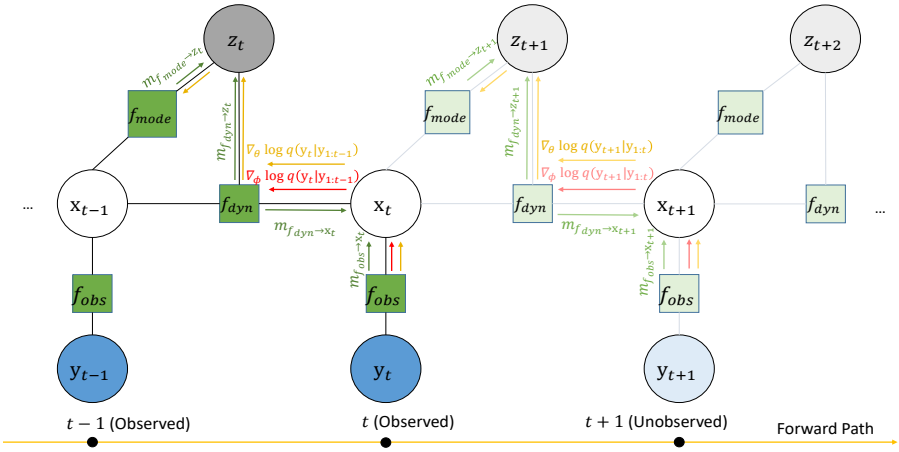


Figure 3: Factor graph. During inference, at each time step t, messages are passed from factor nodes (green) to latent variables (z_t, $\mathbf{x}_t$) as shown by green arrows. In training, gradient flows with respect to continuous parameters ϕ and discrete parameters θ are shown in red and yellow, respectively.

The gradient in eq. (8) is computed via standard backpropagation through the continuous reparameterized path. The second term in eq. (9) uses the REINFORCE estimator, applying the score function method to the discrete sampling path. Here, b_t is a baseline (control variate) used to reduce gradient variance, typically set as an exponential moving average of past log-likelihood values (Kool et al., 2019). The log term $\log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ is treated as a constant with respect to θ—i.e., gradients are not propagated through it—which is implemented by `detach()` in practice. (See appendix C.4)

Importantly, due to the structural factorization of our model (see Section 4), each discrete variable z_t only influences the local observation $\mathbf{y}_t$ through the immediate continuous state $\mathbf{x}_t$, and is independent of the previous discrete assignments $\mathbf{z}_{1:t-1}$. This contrasts with standard SLDS approaches that model transitions via $p(z_t \mid z_{t-1})$, which introduce backward temporal dependencies. As a result, our design enables localized updates to the parameters θ of $q(z_t \mid \mathbf{x}_{t-1})$ using only the current observation $\mathbf{y}_t$, without requiring full BPTT, making training more efficient and naturally suited for online or streaming settings as explained below (proof is in Appendix C.5).

Gradient Efficiency via Local Factorization. In our model, the local factorization $p(z_t \mid x_t)$ enables per-step gradient updates using REINFORCE with complexity $\mathcal{O}(T)$, in contrast to standard SLDS models with Markovian transitions $p(z_t \mid z_{t-1})$, where cumulative dependencies induce a gradient complexity of $\mathcal{O}(T^2)$.

To compute gradients using eqs. (8 -9), we must mitigate instability due to gradient explosion in RNN updates (used to compute $\hat{\boldsymbol{\mu}}_{t|t}$ and $\hat{\boldsymbol{\Sigma}}_{t|t}$ in eq. (7)). We therefore propose a general stability scheme for a broad class of gated, saturating RNN modules (GSRNN), and then specialize it to GRU and LSTM. Consider a general GSRNN with input $u_t \in \mathbb{R}^{n_u}$, hidden states $h_t \in \mathbb{R}^{n_h}$, with $g(h_t, u_t) \in (0,1)^{n_h}$ as a nonlinear gate, and $\Psi(h_t, u_t)$ as mixing operator:

$$h_{t+1} = g(h_t, u_t) \odot h_t + \big(1 - g(h_t, u_t)\big) \odot \phi\big(W u_t + U\, \Psi(h_t, u_t) + b\big), \quad (10)$$

where g, Ψ act componentwise, $\phi : \mathbb{R} \to [-1, 1]$ is a saturating nonlinearity (e.g., tanh), and W, U, b are parameters. Stability scheme is detailed in Theorem 2 with proof in Appendix C.6. For this we first give two definitions (details with examples in Appendix A.5):

Definition 1 (Classes $\mathcal{K}_\infty$ and $\mathcal{KL}$)**.** A function $\gamma : [0, \infty) \to [0, \infty)$ belongs to $\mathcal{K}_\infty$ if it is continuous, strictly increasing, satisfies $\gamma(0) = 0$, and $\lim_{r\to\infty} \gamma(r) = \infty$. A function $\beta : [0, \infty) \times [0, \infty) \to [0, \infty)$ belongs to $\mathcal{KL}$ if, for each fixed $t \geq 0$, $\beta(\cdot, t) \in \mathcal{K}_\infty$, and for each fixed $r \geq 0$, $\beta(r, \cdot)$ is decreasing with $\lim_{t\to\infty} \beta(r, t) = 0$.

Definition 2 (ISS)**.** System eq. (10) is ISS if there exist $\beta \in \mathcal{KL}$ and $\gamma_u, \gamma_b \in \mathcal{K}_\infty$ that, for all $t \geq 0$: $\|h_t\|_\infty \leq \beta(\|h_0\|_\infty, t) + \gamma_u(\|u\|_{\infty,1:t}) + \gamma_b(\|b\|_\infty)$, with $\|u\|_{\infty,1:t} := \max_{1\leq\tau\leq t} \|u_\tau\|_\infty$.

Theorem 2 (ISS for GSRNN)**.** *Assume bounded inputs $\|u_t\|_\infty \leq U_{\max}$, a Lipschitz saturating nonlinearity ϕ with $L_\phi \leq 1$, and gates/mixing maps g, Ψ satisfying $0 \leq \underline{g} \leq g \leq \bar{g} < 1$ and $\|\Psi(h, u)\|_\infty \leq \hat{\psi}\|h\|_\infty$ for some $\hat{\psi} < 1$. Define the effective recurrent gain $c := L_\phi \|U\|_\infty \hat{\psi}$. If $c < 1$, then the system eq. (10) is ISS in $\|\cdot\|_\infty$: for a matrix $\mathbf{A}$, $\|\mathbf{A}\|_\infty = \max_i \sum_j |a_{ij}|$.*

Following Theorem 2, let $c = L_\phi \|U\|_\infty \hat{\psi}$ and fix a safety margin $\varepsilon > 0$. After each gradient step we

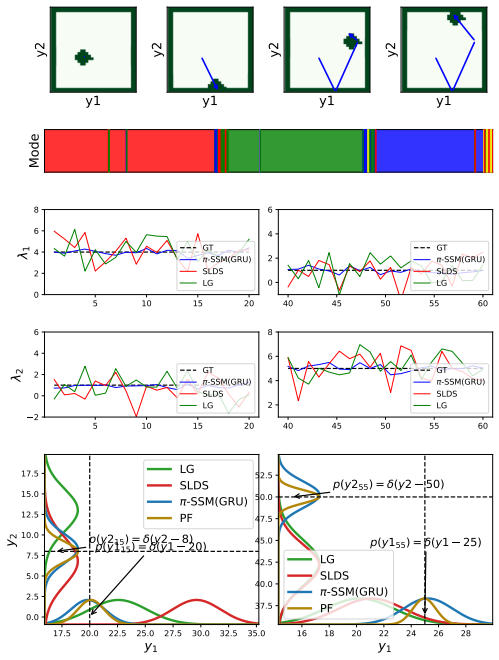


Figure 4: Predicted observation (position) at 15-th and 55-th time steps (last row). The first row shows the ground truth ball position in 10, 20, 50 and 60-th time steps, respectively.

enforce $c < 1-\varepsilon$ by solving the row–wise projection: $\min_U \|U-\hat{U}\|_F^2$ s.t. $\|U_{i,\cdot}\|_1 \leq \frac{1-\varepsilon}{L_\phi \hat{\psi}}$, $\forall i$. It is projecting each row of U onto an ℓ_1-ball of radius $(1-\varepsilon)/(L_\phi\hat{\psi})$. Rows already within the bound remain unchanged, ensuring the ISS condition for eq. (10) (Detailed in Appendix C.9).

Remark 1. This general scheme also covers standard gated architectures such as GRU and LSTM to provide their generic stability. In the Appendix C.7 and C.8 we derive two proof schemes for each: (i) Using specific ISS inequalities and (ii) using Lyapunov function. Then we provide the corresponding projection schemes for each in Appendix C.9.

7 EXPERIMENTS

The first experiment involves a bouncing ball in a dynamic environment, where the underlying dynamics change consistently. Then we focus on a non-linear Lorenz attractor and a Navier-Stokes PDE system, respectively, showcasing π-SSM's ability to infer nonlinear states. While not specifically designed for solving PDEs, π-SSM demonstrates its adaptability and potential as an alternative to PINNs. The next experiment covers NCLT dataset for real world data. The last experiment shows the effectiveness of Theorem 2 for stability handling. The appendix provides intuitive Python code and a detailed training algorithm (see Appendix E). Additional explanations on hyperparameter optimization, network architecture, and practical strategies to avoid poor local minima are included in Appendix F. We use general LSTM and GRU cells modeling RNN is eq. (7), shown by π-SSM$_{\text{LSTM}}$ and SSM$_{\text{GRU}}$. Runtime comparisons in Appendix, Table 5.

To validate that the proposed learned Kalman-style update is a meaningful proxy for the true posterior, we introduce a variant π-SSM(LG) where the RNN-based gain in Eq. (7) is replaced with the classical exact Kalman gain (computed via matrix inversion), while preserving the discrete switching mechanism. This variant maintains the graphical-model design and joint optimization but uses traditional filtering without neural adaptation. Results in Table 1 demonstrate that π-SSM(LG) consistently achieves lower test log-likelihood than the full π-SSM across all datasets, empirically validating that the learned gain better approximates the true posterior in both linear-Gaussian and nonlinear/switching regimes. Moreover, the learned parameterization avoids the cubic complexity of exact matrix inversions, enabling efficient scaling to higher dimensions (see Appendix D.3 for runtime analysis).

Pong. To demonstrate the adaptability of π-SSM, we evaluated it in a switching dynamics environment. We generated 5k sequences, each with 80 time steps, simulating a ball moving within a four-sided enclosure. The ball's initial position and velocity were randomized, and no external forces were applied. Collisions with the walls were modeled as elastic reflections. For the ablation study, we first compare π-SSM against a simple observation model without latent parameterization, and then against a classic LG variant, where RNN cells are replaced by standard filtering equations and discrete latents are omitted (See Appendix G.1 for details). We further assess the importance of latent parameterization by replacing the π-SSM core with GRU and LSTM cells that directly parameterize the output observations. In addition, we compare various variational inference (VI) models—including VI-GRU, VI-LSTM (Chung et al., 2015)—and EM-based methods such as KVAE, EKVAE, and MVAE.
We also include comparisons with latent SLDS and

(a) Observation (b) π-SSM$_{\text{GRU}}$

Figure 5: Inferred 5k-length Lorenz attractor.

Table 1: Predictive (lower bounded) likelihood for Pong, Lorenz, Navier-Stokes and NCLT experiments. MSE results are included in Appendix H.3.

Model	Pong	Lorenz	Navier-Stokes	NCLT
Observation	4.112 ± 0.621	4.374 ± 0.462	3.591 ± 0.397	-30.87 ± 1.07
VI-LSTM	4.592 ± 0.388	5.271 ± 0.241	4.401 ± 0.306	-25.64 ± 1.32
VI-GRU	4.691 ± 0.227	5.267 ± 0.364	4.375 ± 0.194	-25.51 ± 1.63
KVAE	4.492 ± 0.339	5.162 ± 0.255	4.299 ± 0.241	-25.98 ± 1.80
EKVAE	4.688 ± 0.282	5.231 ± 0.198	4.411 ± 0.293	-25.16 ± 1.88
MVAE	4.701 ± 0.176	5.371 ± 0.273	4.511 ± 0.258	-24.76 ± 1.93
DeepAR	4.342 ± 0.422	4.985 ± 0.392	4.194 ± 0.409	-29.14 ± 3.57
SLDS	4.768 ± 0.344	5.241 ± 0.441	4.481 ± 0.198	-25.10 ± 1.63
rSLDS	4.952 ± 0.291	5.322 ± 0.347	4.501 ± 0.298	-24.81 ± 1.73
REDSDS	4.939 ± 0.182	5.510 ± 0.296	4.580 ± 0.311	-24.59 ± 1.98
SNLDS	4.815 ± 0.163	5.377 ± 0.214	4.521 ± 0.238	-24.94 ± 1.52
irSLDS	4.925 ± 0.291	5.449 ± 0.397	4.592 ± 0.264	-24.57 ± 2.09
NODE	4.632 ± 0.312	5.184 ± 0.337	4.382 ± 0.299	-25.81 ± 2.11
MoNODE	4.791 ± 0.391	5.395 ± 0.294	4.511 ± 0.411	-24.90 ± 1.73
RKN	4.661 ± 0.229	5.179 ± 0.214	4.326 ± 0.159	-26.14 ± 1.50
CRU	4.692 ± 0.197	5.315 ± 0.359	4.481 ± 0.174	-25.64 ± 1.61
KalmanNet	4.897 ± 0.341	5.268 ± 0.214	4.431 ± 0.246	-26.15 ± 1.37
GIN	5.032 ± 0.280	5.418 ± 0.228	4.655 ± 0.247	-24.95 ± 1.54
Hybrid GNN	5.034 ± 0.240	5.511 ± 0.297	4.691 ± 0.181	-24.55 ± 1.62
LSTM	4.580 ± 0.322	5.182 ± 0.392	4.571 ± 0.192	-27.91 ± 1.39
GRU	4.611 ± 0.391	5.215 ± 0.430	4.558 ± 0.267	-27.94 ± 1.47
HiP-RSSM	4.722 ± 0.199	5.307 ± 0.255	4.502 ± 0.284	-25.70 ± 1.13
LSD-VBS	4.981 ± 0.241	5.402 ± 0.319	4.530 ± 0.204	-25.61 ± 1.64
LG	4.993 ± 0.310	5.434 ± 0.520	4.711 ± 0.311	-24.42 ± 1.29
π-SSM(LG)	5.385 ± 0.173	5.691 ± 0.227	5.044 ± 0.201	-23.51 ± 0.79
π-SSM$_{\text{GRU}}$	5.401 ± 0.197	**5.856 ± 0.387**	5.097 ± 0.247	**-23.18 ± 1.07**
π-SSM$_{\text{LSTM}}$	**5.475 ± 0.217**	5.844 ± 0.292	**5.137 ± 0.180**	-23.25 ± 0.94

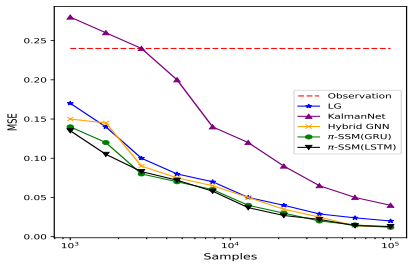


Figure 6: MSE of Lorenz attractor.

Table 2: Comparison of our ISS and GC for stability handling for Pong.

Stability Handling		Objective	Success
π-SSM$_{\text{GRU}}$(ISS)	$\epsilon = 0.1$	5.401 ± 0.197	100%
	$\epsilon = 0.2$	5.084 ± 0.281	100%
π-SSM$_{\text{LSTM}}$(ISS)	$\epsilon = 0.1$	5.475 ± 0.217	100%
	$\epsilon = 0.2$	5.172 ± 0.202	100%
π-SSM$_{\text{GRU}}$(GC)	$\delta = 10$	5.249 ± 1.12	60%
	$\delta = 20$	N/A	0%
π-SSM$_{\text{LSTM}}$(GC)	$\delta = 10$	5.251 ± 1.31	45%
	$\delta = 20$	N/A	0%

irSLDS. For a broader evaluation, we incorporate DeepAR (an autoregressive model), as well as CRU, RKN, neural ODEs, and other VI-based approaches. Numerical results are reported in Table 1. Figure 4 shows samples from the predictive distributions $q(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ of π-SSM$_{\text{GRU}}$, LG, and SLDS compared with ground-truth Pong trajectories. Top row: a trajectory of 4 frames with two collisions, each triggering a mode switch. Second row: π-SSM$_{\text{GRU}}$ inferred modes, where each color marks a regime; intervals of constant ground-truth mode align with consistent colors, and color changes coincide with mode switches. Third and fourth rows: comparison of the first two eigenvalues of learned transition matrices of π-SSM$_{\text{GRU}}$ with ground truth, demonstrating accurate recovery of dynamics across regimes. Bottom row: predictive distributions at $t = 15$ and $t = 55$ ($\mathbf{y}_t$ = ball position), highlighting alignment of models with ground-truth transitions. To provide a conventional high-cost posterior reference for Figure 4, we also include Particle Filter (PF) estimates in the last row: PF is a computationally expensive sampling-based posterior approximator, and its inclusion shows that the predictive posterior of π-SSM remains close to this stronger reference while being much more efficient. Further explanations with detailed example of 8 frames and the strategy to extend $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ to perform sequence generation/imputation are in Appendix H.1. [1]

[1]Animated files demonstrating sequence generation/imputation are available at: Google site: https://sites.google.com/view/pissm-aistats.
Blog page: Link.

Lorenz Attractor. The Lorenz system is a set of nonlinear ordinary differential equations originally developed to model atmospheric convection. Due to its chaotic and highly nonlinear dynamics (see Appendix G.2), it serves as a strong benchmark for evaluating the π-SSM cell. We assess π-SSM on a trajectory of length 5k, where each observation is perturbed by zero-mean Gaussian noise with covariance $\mathbf{R}_t = 0.5\mathbf{I}$. During training, the predictive likelihood $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ is modeled as a Gaussian and optimized accordingly. In addition to the log-likelihood scores in Table 1, we report the MSE across different sample sizes in Figure 6. Due to the system's inherent nonlinearity, LG must linearize the transition model before applying standard filtering, which limits its performance. Sample trajectories are visualized in Figure 5.

Real World Dynamics: NCLT dataset. To evaluate π-SSM on real-world data, we utilize the Michigan NCLT dataset, which consists of navigation data collected by a Segway robot. At each time step, the latent state $\mathbf{x}_t \in \mathbb{R}^4$ comprises the robot's 2D position and velocity, while the observation $\mathbf{y}_t \in \mathbb{R}^2$ corresponds to noisy GPS measurements. The goal is to accurately estimate the robot's true position given only the corrupted GPS observations. For this experiment, we sampled 4,280 usable time steps from the robot's trajectory. We employ a piece-wise constant velocity motion model, where the robot maintains constant velocity within segments but switches discrete heading modes when changing direction (e.g., north, northwest, south, etc.). This structure allows the system to jointly infer the robot's continuous state (posi-

tion and velocity magnitude) and its discrete heading mode from noisy observations. For evaluation, we employ cross-validation on temporal segments of the trajectory (see Appendix G.3 for details). In addition, we present MSE results in Appendix Table 7 for competitive methods. A visual comparison of inferred states against ground truth is in Appendix Figure 10. Overall, this experiment demonstrates that the π-SSM generalizes well to real-world localization tasks.

Navier-Stokes.

The Navier–Stokes equations (Acheson, 1990) govern incompressible, viscous fluid flow and underlie phenomena such as ocean currents, weather, and turbulence. To test whether π-SSM can capture nonlinear PDE-driven dynamics, we design a 2D Navier–Stokes experiment in vorticity form on the unit torus (see Appendix G.4 for details). We simulate particles evolving under true fluid dynamics and generate 10K trajectories. Each latent state $\mathbf{x}_t \in \mathbb{R}^5$ includes position, velocity, and pressure $(x_t, y_t, u_t, v_t, p_t)$, while observations $\mathbf{y}_t \in \mathbb{R}^2$ contain only noisy spatial positions. The model is trained to infer latent states using Gaussian predictive likelihood. At test time, given an initial (x_0, y_0), it recursively infers velocity and pressure to generate trajectories consistent with the underlying field, allowing assessment of PDE recovery from partial observations (Table 1, with examples in link). Because PINNs and structured variants (e.g., SPINNs) are tailored for PDEs, we include them as baselines alongside general sequence models. Generated trajectories are shown in Figure 1, with further numerical results in Appendix H.3.

Stability Handling. Table 2 reports the log-likelihood values and standard deviations for the Pong experiment, comparing different strategies for addressing training instability. We evaluate the conventional Gradient Clipping (GC), using threshold δ, against our proposed ISS projection strategy, which ensures the RNN satisfies the condition in Theorem 2. In our approach, the positive buffer ϵ defines the radius of the ℓ_1-ball onto which each row of U is projected, enforcing a strict ISS margin. As shown in Table 2, gradient clipping struggles to train under large thresholds, resulting in unstable behavior or divergence. In contrast, our ISS-based approach yields consistently higher log-likelihoods and lower perplexity across all settings, offering improved robustness and stability during training. Figure 8 in Appendix illustrates this effect: without ISS, $\|U\|_\infty$ frequently exceeds the admissible radius, leading to sharp jumps and oscillations in the loss. With ISS, both $\|U\|_\infty$ and the loss remain well controlled. We provide further analysis of the role of ϵ, including tuning strategies, sensitivity, and full results of table 2 in Appendix G.5.

8 CONCLUSION

This paper introduced the π-SSM framework for modeling dynamic systems via hybrid state-space models. Our approach leverages approximate message passing for inference while using a structured RNN-based architecture to realize recursive data flow. The π-SSM jointly trains both discrete and continuous latent variables using a hybrid optimization strategy, and incorporates a stability scheme based on ISS to ensure robust RNN updates. Although not explicitly designed for PDE systems, π-SSM demonstrates strong performance on such problems, highlighting its generality and opening new directions for future research. Additional extensions, such as backward smoothing via backward message passing, can further enhance the model's inference capabilities.

ACKNOWLEDGMENTS

I thank Steffen Staab for useful discussions and reviews. Also I want to thank Wan Choi as the original idea was generated when working with him in Seoul National University, while matured later. Research for this paper was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - EXC 2075 - 390740016 under Germany's Excellence Strategy (all authors). The authors acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

Bibliography

David J Acheson. *Elementary fluid dynamics*. Oxford University Press, 1990.

Abdul Fatir Ansari, Konstantinos Benidis, Richard Kurle, Ali Caner Turkmen, Harold Soh, Alexander J Smola, Bernie Wang, and Tim Januschowski. Deep explicit duration switching models for time series. *Advances in Neural Information Processing Systems*, 34:29949–29961, 2021.

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Ilze Amanda Auzina, Çağatay Yıldız, Sara Magliacane, Matthias Bethge, and Efstratios Gavves. Modulated neural odes. *Advances in Neural Information Processing Systems*, 36, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexandra Baier, Decky Aspandi, and Steffen Staab. Relinet: Stable and explainable multistep prediction

with recurrent linear parameter varying networks. In *IJCAI*, pages 3461–3469, 2023.

Shivani Bathla and Vinita Vasudevan. Approximate inference of marginals using the ibia framework. *Advances in Neural Information Processing Systems*, 36:72679–72691, 2023.

Philipp Becker, Harit Pandya, Gregor Gebhardt, Cheng Zhao, C James Taylor, and Gerhard Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In *International Conference on Machine Learning*, pages 544–552. PMLR, 2019.

Philip Becker-Ehmck, Jan Peters, and Patrick Van Der Smagt. Switching linear dynamics for variational bayes filtering. In *International conference on machine learning*, pages 553–562. PMLR, 2019.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016.

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

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Zhe Dong, Bryan Seybold, Kevin Murphy, and Hung Bui. Collapsed amortized variational inference for switching nonlinear dynamical systems. In *International Conference on Machine Learning*, pages 2638–2647. PMLR, 2020.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *arXiv preprint arXiv:1710.05741*, 2017.

Tal Friedman and Guy Van den Broeck. Approximate knowledge compilation by online collapsed importance sampling. *Advances in neural information processing systems*, 31, 2018.

Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in gaussian process state-space models with particle mcmc. *Advances in neural information processing systems*, 26, 2013.

Victor Garcia Satorras, Zeynep Akata, and Max Welling. Combining generative and discriminative models for hybrid inference. *Advances in Neural Information Processing Systems*, 32, 2019.

Victor Geadah, Jonathan W Pillow, et al. Parsing neural dynamics with infinite recurrent switching linear dynamical systems. In *The Twelfth International Conference on Learning Representations*, 2024.

Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

Amir Ghalamzan, Kiyanoush Nazari, Hamidreza Hashempour, and Fangxun Zhong. Deep-lfd: deep robot learning from demonstrations. *Software Impacts*, 9:100087, 2021.

P Gilabert, Gabriel Montoro, and E Bertran. On the wiener and hammerstein models for power amplifier predistortion. In *2005 Asia-Pacific Microwave Conference Proceedings*, volume 2, pages 4–pp. IEEE, 2005.

Chenghua Guo, Han Yu, Jiaxin Liu, Chao Chen, Qi Li, Sihong Xie, and Xi Zhang. Linear uncertainty quantification of graphical model inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Hamidreza Hashempoor and Wan Choi. Deep learning based data-assisted channel estimation and detection. *IEEE Transactions on Machine Learning in Communications and Networking*, 2025.

Hamidreza Hashempoor and Yu Dong Hwang. Fasttracker: Real-time and accurate visual tracking. *arXiv preprint arXiv:2508.14370*, 2025.

Hamidreza Hashempoor, Rosemary Koikara, and Yu Dong Hwang. Featuresort: A robust tracker with optimized feature integration. *Pattern Recognition*, page 113148, 2026.

Hamidreza Hashempoorikderi and Wan Choi. Gated inference network: Inference and learning state-space models. *Advances in Neural Information Processing Systems*, 37:39036–39073, 2024.

Hamidreza Hashempour, Kiyanoush Nazari, Fangxun Zhong, et al. A data-set of piercing needle through deformable objects for deep learning from demonstrations. *arXiv preprint arXiv:2012.02458*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29, 2016.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Sekitoshi Kanai, Yasuhiro Fujiwara, and Sotetsu Iwamura. Preventing gradient explosions in gated recurrent units. *Advances in neural information processing systems*, 30, 2017.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alexej Klushyn, Richard Kurle, Maximilian Soelch, Botond Cseke, and Patrick van der Smagt. Latent matters: Learning deep state-space models. *Advances in Neural Information Processing Systems*, 34, 2021.

Jonathan Ko and Dieter Fox. Learning gp-bayesfilters via gaussian process latent variable models. *Autonomous Robots*, 30(1):3–23, 2011.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Rahul Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Rui Li, ST John, and Arno Solin. Improving hyperparameter learning under approximate inference in gaussian process models. In *International Conference on Machine Learning*, pages 19595–19615. PMLR, 2023.

Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922. PMLR, 2017.

Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

Lennart Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24 (1):36–50, 1979.

Qi Lou, Rina Dechter, and Alexander Ihler. Interleave variational optimization with monte carlo sampling: A tale of two approximate inference paradigms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7900–7907, 2019.

Radu Marinescu, Akihiro Kishimoto, Adi Botea, Rina Dechter, and Alexander Ihler. Anytime recursive best-first search for bounding marginal map. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7924–7932, 2019.

Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In *International conference on artificial intelligence and statistics*, pages 968–977. PMLR, 2018.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2 (11):2, 2011.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A

deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *arXiv preprint arXiv:2107.10043*, 2021.

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.

David Ruhe and Patrick Forré. Self-supervised inference in state-space models. *arXiv preprint arXiv:2107.13349*, 2021.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pages 19388–19405. PMLR, 2022.

Maarten Schoukens and Koen Tiels. Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica*, 85:272–292, 2017.

Vaisakh Shaj, Dieter Buchler, Rohit Sonker, Philipp Becker, and Gerhard Neumann. Hidden parameter recurrent state space models for changing dynamics scenarios. *arXiv preprint arXiv:2206.14697*, 2022.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Francesco Tonolini, Nikolaos Aletras, Yunlong Jiao, and Gabriella Kazai. Robust weak supervision with variational auto-encoders. In *International Conference on Machine Learning*, pages 34394–34408. PMLR, 2023.

Niklas Wahlström, Thomas B Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Robert Wilson and Leif Finkel. A neural implementation of the kalman filter. *Advances in neural information processing systems*, 22:2062–2070, 2009.

Harrison Zhu, Carles Balsells-Rodas, and Yingzhen Li. Markovian gaussian process variational autoencoders. In *International Conference on Machine Learning*, pages 42938–42961. PMLR, 2023.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]

(c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [No Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

Contents of Appendix

A Background

A.1 Forward Backward Algorithm

In hybrid state-space models, such as SLDS, the latent structure typically involves both a continuous latent state sequence $\mathbf{x}_{1:T}$ and a discrete mode sequence $\mathbf{z}_{1:T}$, which governs the dynamic regime of the system Hashempoor and Hwang (2025); Hashempoor and Choi (2025); Ghalamzan et al. (2021); Hashempour et al. (2020); Hashempoor et al. (2026) . Given observations $\mathbf{y}_{1:T}$, a standard inference task, particularly within Gibbs sampling frameworks, is to compute the posterior over discrete states $p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:T})$, assuming the continuous latent trajectory $\mathbf{x}_{1:T}$ is fixed.

The model structure assumes a Markovian prior over discrete states, specified by a transition distribution $p(z_t \mid z_{t-1})$, and a likelihood model that factors as $p(\mathbf{y}_t \mid \mathbf{x}_t, z_t)$ for each time step. Additionally, the initial discrete state is governed by a prior $p(z_1)$.

The forward-backward algorithm provides an efficient dynamic programming method to compute the posterior marginals of $\mathbf{z}_{1:T}$ under this model structure. The forward pass recursively computes the joint probability of the observed data up to time t and the discrete state z_t, conditioned on the fixed continuous states. The forward recursion is initialized as

$$\alpha_1(z_1) = p(\mathbf{y}_1|z_1, \mathbf{x}_1).p(z_1) \tag{11}$$

Then, for $t = 2, \ldots, T$, the recursion proceeds as

$$\alpha_t(z_t) = p(\mathbf{y}_t|z_t, \mathbf{x}_t) \sum_{z_{t-1}} \alpha_{t-1}(z_{t-1}) p(z_t|z_{t-1}) \tag{12}$$

The backward pass propagates information from future observations. It is initialized with

$$\beta_T = 1 \tag{13}$$

and for $t = T-1, \ldots, 1$, the backward recursion is given by

$$\beta_t(z_t) = \sum_{z_{t+1}} \alpha_{t-1}(z_{t-1}) p(z_{t+1}|z_t) p(\mathbf{y}_{t+1}|z_{t+1}, \mathbf{x}_{t+1}) \beta_{t+1}(z_{t+1}) \tag{14}$$

Having computed the forward and backward messages, the posterior marginals over discrete states are proportional to their product:

$$p(z_t|\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) \propto \alpha_t(z_t).\beta_t(z_t) \tag{15}$$

Sampling a discrete trajectory $\mathbf{z}_{1:T}$ can then be performed sequentially, using the computed marginals at each time step.

The forward-backward algorithm thus enables exact posterior inference over discrete latent sequences in models with coupled discrete-continuous structures, conditioned on fixed continuous states, and serves as a key inference building block in hybrid dynamical systems.

A.2 Filtering and Smoothing Parameterization

In a SLDS, the generative process over continuous latent states $\mathbf{x}_{1:T}$, discrete switching variables $\mathbf{z}_{1:T}$, and observations $\mathbf{y}_{1:T}$ factorizes as

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{y}_{1:T}) = p(z_1)p(\mathbf{x}_1|z_1)p(\mathbf{y}_1|\mathbf{x}_1, z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}).p(\mathbf{x}_t|\mathbf{x}_{t-1}, z_t).p(\mathbf{y}_t|\mathbf{x}_t, z_t) \tag{16}$$

Inference in this model typically involves estimating the joint posterior $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{y}_{1:T})$ which is computationally intractable in general due to the coupling between discrete and continuous variables across time. In the Gibbs

sampling framework, we alternate between updating the discrete sequence $\mathbf{z}_{1:T}$ and the continuous latent states $\mathbf{x}_{1:T}$. After fixing $\mathbf{z}_{1:T}$, the conditional model over $\mathbf{x}_{1:T}$ and $\mathbf{y}_{1:T}$ reduces to a time-varying linear Gaussian state-space model, where the dynamics and emission parameters are determined by the known discrete sequence. In the following, we describe the classical Kalman filtering and smoothing procedures adapted for this setting.

In the Gibbs sampling framework, after fixing the discrete state sequence $\mathbf{z}_{1:T}$, we update the continuous latent variables $\mathbf{x}_{1:T}$ conditioned on the fixed $\mathbf{z}_{1:T}$ and observations $\mathbf{y}_{1:T}$. The following describes the classical Kalman filtering and smoothing procedures adapted for this setting.

The Kalman filter operates by alternating between two main steps: prediction and correction. In the prediction step, prior state information is used to estimate the state at the next time step. The correction step refines this estimate by incorporating newly acquired observations. Assuming that the process and observation noise are Gaussian, and that the transition and emission models are determined by the discrete mode z_{t+1}, the filter can execute these operations efficiently. During the prediction phase, the transition matrix $\mathbf{A}_{z_{t+1}}$ associated with the discrete mode z_{t+1} is used to compute the prior distribution

$$p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t}, \mathbf{z}_{1:t}) = (\mu_{t+1|t}, \mathbf{\Sigma}_{t+1|t}) \tag{17}$$

where

$$\mu_{t+1|t} = \mathbf{A}_{z_{t+1}}\mu_{t|t}, \quad \mathbf{\Sigma}_{t+1|t} = \mathbf{A}_{z_{t+1}}\mathbf{\Sigma}_{t|t}\mathbf{A}_{z_{t+1}}^T + \mathbf{Q}_{z_{t+1}}, \tag{18}$$

and $\mathbf{Q}_{z_{t+1}}$ denotes the process noise covariance determined by the mode z_t.

Upon receiving a new observation $\mathbf{y}_{t+1}$, the Kalman filter proceeds to the update (correction) phase, refining the prior estimate by incorporating the emission matrix $\mathbf{C}_{z_{t+1}}$ corresponding to the discrete mode z_{t+1}. The Kalman gain is given by

$$\mathbf{K}_{t+1} = \mathbf{\Sigma}_{t+1|t}\mathbf{C}_{z_{t+1}}^T\left(\mathbf{C}_{z_{t+1}}\mathbf{\Sigma}_{t+1|t}\mathbf{C}_{z_{t+1}}^T + \mathbf{R}_{z_{t+1}}\right)^{-1}, \tag{19}$$

where $\mathbf{R}_{z_{t+1}}$ is the observation noise covariance.

The state mean and covariance are then updated as

$$\mu_{t+1|t+1} = \mu_{t+1|t} + \mathbf{K}_{t+1}\left(\mathbf{y}_{t+1} - \mathbf{C}_{z_{t+1}}\mu_{t+1|t}\right), \tag{20}$$

$$\mathbf{\Sigma}_{t+1|t+1} = \mathbf{\Sigma}_{t+1|t} - \mathbf{K}_{t+1}\left(\mathbf{C}_{z_{t+1}}\mathbf{\Sigma}_{t+1|t}\mathbf{C}_{z_{t+1}}^T + \mathbf{R}_{z_{t+1}}\right)\mathbf{K}_{t+1}^T. \tag{21}$$

This observation update step can be interpreted as a weighted average between the predicted prior (from the state transition) and the newly acquired observation, where the relative weight is determined by the process and observation noise covariances $\mathbf{Q}_{z_{t+1}}$ and $\mathbf{R}_{z_{t+1}}$.

To derive the smoothing equations, we utilize the Markov property, which asserts that $\mathbf{x}_t$ is conditionally independent of future observations $\mathbf{y}_{t+1:T}$ given $\mathbf{x}_{t+1}$. Although $\mathbf{x}_{t+1}$ is not directly observed, it is available through its posterior distribution. By conditioning on $\mathbf{x}_{t+1}$ and subsequently marginalizing, the smoothing distribution for $\mathbf{x}_t$ can be expressed as

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) &= \int p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:T}, \mathbf{z}_{1:T})p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}, \mathbf{z}_{1:T})d\mathbf{x}_{t+1} \\ &= \int p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{1:t}, \cancel{\mathbf{y}_{t+1:T}}, \cancel{\mathbf{z}_{t+1:T}})p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}, \mathbf{z}_{1:T})d\mathbf{x}_{t+1} \end{aligned} \tag{22}$$

By using induction and and smoothed distribution for $t+1$:

$$p_{\gamma_{1:T}}(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) = \mathcal{N}(\mu_{t+1|T}, \mathbf{\Sigma}_{t+1|T}) \tag{23}$$

and applying standard Gaussian identities, the two-time-slice filtered distribution is given by

$$p_{\gamma_{1:t}}(\mathbf{x}_t, \mathbf{x}_{t+1}|\mathbf{y}_{1:t}, \mathbf{z}_{1:t}) = \mathcal{N}\left(\begin{pmatrix}\mu_{t|t} \\ \mu_{t+1|t}\end{pmatrix}, \begin{pmatrix}\mathbf{\Sigma}_{t|t} & \mathbf{\Sigma}_{t|t}\mathbf{A}_{z_{t+1}}^T \\ \mathbf{A}_{z_{t+1}}\mathbf{\Sigma}_{t|t} & \mathbf{\Sigma}_{t+1|t}\end{pmatrix}\right) \tag{24}$$

Conditioning this joint Gaussian on $\mathbf{x}_{t+1}$ yields

$$p_{\gamma_{1:t}}(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{1:t}) = \mathcal{N}(\mu_{t|t} + \mathbf{J}_t\big(\mathbf{x}_{t+1} - \mathbf{A}_{z_{t+1}}\mu_{t|t}\big), \mathbf{\Sigma}_{t|t} - \mathbf{J}_t\mathbf{\Sigma}_{t+1|t}\mathbf{J}_t^T) \tag{25}$$

where the smoothing gain matrix is defined as

$$\mathbf{J}_t = \mathbf{\Sigma}_{t|t}\mathbf{A}_{z_{t+1}}[\mathbf{\Sigma}_{t+1|t}]^{-1} \tag{26}$$

Using the rules of iterated expectation and covariance, the smoothed mean and covariance are then given by

$$\begin{aligned}
\mu_{t|T} &= \mathbb{E}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:T}, \mathbf{z}_{1:T}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mathbb{E}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{1:t}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mathbb{E}\big[\mu_{t|t} + \mathbf{J}_t(\mathbf{x}_{t+1} - \mathbf{A}_{z_{t+1}}\mu_{t|t})\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mu_{t|t} + \mathbf{J}_t(\mu_{t+1|T} - \mathbf{A}_{z_{t+1}}\mu_{t|t})
\end{aligned} \tag{27}$$

$$\begin{aligned}
\mathbf{\Sigma}_{t|T} &= \mathrm{cov}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:T}, \mathbf{z}_{1:T}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] + \mathbb{E}\big[\mathrm{cov}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:T}, \mathbf{z}_{1:T}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mathrm{cov}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{1:t}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] + \mathbb{E}\big[\mathrm{cov}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{1:t}]\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mathrm{cov}\big[\mu_{t|t} + \mathbf{J}_t(\mathbf{x}_{t+1} - \mathbf{A}_{z_{t+1}}\mu_{t|t})\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] + \mathbb{E}\big[\mathbf{\Sigma}_{t|t} - \mathbf{J}_t\mathbf{\Sigma}_{t+1|t}\mathbf{J}_t^T\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big] \\
&= \mathbf{J}_t\mathrm{cov}\big[\mathbf{x}_{t+1} - \mathbf{A}_{z_{t+1}}\mu_{t|t}\ |\mathbf{y}_{1:T}, \mathbf{z}_{1:T}\big]\mathbf{J}_t^T + \mathbf{\Sigma}_{t|t} - \mathbf{J}_t\mathbf{\Sigma}_{t+1|t}\mathbf{J}_t^T \\
&= \mathbf{J}_t\mathbf{\Sigma}_{t+1|T}\mathbf{J}_t^T + \mathbf{\Sigma}_{t|t} - \mathbf{J}_t\mathbf{\Sigma}_{t+1|t}\mathbf{J}_t^T \\
&= \mathbf{\Sigma}_{t|t} + \mathbf{J}_t\big(\mathbf{\Sigma}_{t+1|T} - \mathbf{\Sigma}_{t+1|t}\big)\mathbf{J}_t^T.
\end{aligned} \tag{28}$$

A.3 GRU Cell Review.

We consider a standard GRU cell followed by a linear output layer, as described in Chung et al. (2014), reformulated here in terms of the gating architecture. At each time step, given input vector $\mathbf{u} \in \mathbb{R}^{n_u}$ and previous hidden state $\mathbf{h} \in \mathbb{R}^{n_h}$, the GRU cell computes the new hidden state $\mathbf{h}^+$ and output $\mathbf{out} \in \mathbb{R}^{n_o}$ as follows:

$$\mathbf{g} = \sigma(\mathbf{W}_g\mathbf{u} + \mathbf{U}_g\mathbf{h} + \mathbf{b}_g), \quad \text{(update gate)} \tag{29}$$

$$\mathbf{f} = \sigma(\mathbf{W}_f\mathbf{u} + \mathbf{U}_f\mathbf{h} + \mathbf{b}_f), \quad \text{(forget/reset gate)} \tag{30}$$

$$\hat{\mathbf{h}} = \phi(\mathbf{W}_r\mathbf{u} + \mathbf{U}_r(\mathbf{f} \odot \mathbf{h}) + \mathbf{b}_r), \quad \text{(candidate activation)} \tag{31}$$

$$\mathbf{h}^+ = \mathbf{g} \odot \mathbf{h} + (1 - \mathbf{g}) \odot \hat{\mathbf{h}}, \quad \text{(hidden state update)} \tag{32}$$

$$\mathbf{out} = \mathbf{U}_{\text{out}}\mathbf{h} + \mathbf{b}_{\text{out}}, \quad \text{(linear readout)} \tag{33}$$

Here, $\sigma(\cdot)$ denotes the element-wise sigmoid function and $\phi(\cdot)$ is a nonlinear activation function, typically the hyperbolic tangent. The operators $\odot$ and $(1 - \mathbf{g})$ denote element-wise multiplication and complement, respectively.

This formulation defines the update gate $\mathbf{g}$ and forget/reset gate $\mathbf{f}$ based on both current input $\mathbf{u}$ and prior hidden state $\mathbf{h}$. The intermediate candidate $\hat{\mathbf{h}}$ incorporates gated recurrence through $\mathbf{f} \odot \mathbf{h}$, and the next hidden state $\mathbf{h}^+$ blends the past state and candidate based on $\mathbf{g}$. The output vector $\mathbf{out}$ is computed as a linear transformation of the current state.

A.4 LSTM Cell Review.

We consider a standard LSTM cell (no peepholes, uncoupled gates) followed by a linear output layer, as described in Hochreiter and Schmidhuber (1997), written in gating form. At each time step, given input $\mathbf{u} \in \mathbb{R}^{n_u}$,

previous hidden state $\mathbf{h} \in \mathbb{R}^{n_h}$, and previous cell state $\mathbf{c} \in \mathbb{R}^{n_h}$, the LSTM computes the next states $(\mathbf{c}^+, \mathbf{h}^+)$ and output $\mathbf{out} \in \mathbb{R}^{n_o}$ as:

$$\mathbf{f} = \sigma(\mathbf{W}_f \mathbf{u} + \mathbf{U}_f \mathbf{h} + \mathbf{b}_f), \quad \text{(forget gate)} \tag{34}$$
$$\mathbf{i} = \sigma(\mathbf{W}_i \mathbf{u} + \mathbf{U}_i \mathbf{h} + \mathbf{b}_i), \quad \text{(input gate)} \tag{35}$$
$$\mathbf{o} = \sigma(\mathbf{W}_o \mathbf{u} + \mathbf{U}_o \mathbf{h} + \mathbf{b}_o), \quad \text{(output gate)} \tag{36}$$
$$\hat{\mathbf{c}} = \phi(\mathbf{W}_c \mathbf{u} + \mathbf{U}_c \mathbf{h} + \mathbf{b}_c), \quad \text{(candidate cell)} \tag{37}$$
$$\mathbf{c}^+ = \mathbf{f} \odot \mathbf{c} + \mathbf{i} \odot \hat{\mathbf{c}}, \quad \text{(cell state update)} \tag{38}$$
$$\mathbf{h}^+ = \mathbf{o} \odot \phi(\mathbf{c}^+), \quad \text{(hidden state update)} \tag{39}$$
$$\mathbf{out} = \mathbf{U}_{\text{out}} \mathbf{h}^+ + \mathbf{b}_{\text{out}}, \quad \text{(linear readout)} \tag{40}$$

Here, $\sigma(\cdot)$ denotes the element-wise logistic sigmoid and $\phi(\cdot)$ is a saturating nonlinearity (typically tanh). The operator $\odot$ denotes element-wise multiplication. The gates $\mathbf{f}, \mathbf{i}, \mathbf{o}$ modulate, respectively, memory retention, information write, and exposure of the cell state. The candidate $\hat{\mathbf{c}}$ proposes a content update, which is blended with the retained memory to form $\mathbf{c}^*$; the hidden state $\mathbf{h}^+$ exposes a squashed version of $\mathbf{c}^+$ through the output gate. The readout **out** is a linear function of the current hidden state.

A.5 ISS and Comparison Functions: Intuition, Properties, and Examples

We adopt the standard comparison-function notation in Definitions 1–2. Class $\mathcal{K}_\infty$ functions act as *gauge functions*: they are continuous, strictly increasing, vanish at the origin, and grow unboundedly. They allow us to rescale and compare norms while preserving order and asymptotic behavior. Class $\mathcal{KL}$ functions model *transient decay*: for fixed r, $\beta(r,t)$ decreases to 0 as $t \to \infty$; for fixed t, $r \mapsto \beta(r,t)$ behaves like a $\mathcal{K}_\infty$ function. In an ISS estimate (Definition 2), $\beta(\|h_0\|_\infty, t)$ quantifies how the effect of the initial condition vanishes, while $\gamma_u(\|u\|_{\infty,1:t})$ and $\gamma_b(\|b\|_\infty)$ capture how bounded inputs and constant biases bound the steady-state response.

Examples.

- $\mathcal{K}_\infty$: $\gamma(r) = cr$ with $c > 0$; more generally $\gamma(r) = cr^p$ for any $c > 0$, $p \geq 1$; or $\gamma(r) = c(e^{ar} - 1)$ for $a, c > 0$.
- $\mathcal{KL}$: $\beta(r,t) = re^{-ct}$ for $c > 0$ (continuous-time) and $\beta(r,t) = \lambda^t r$ with $\lambda \in (0,1)$ (discrete-time). Mixed forms are common: $\beta(r,t) = \alpha(t)\,\tilde{\gamma}(r)$ with $\alpha(t) \downarrow 0$ and $\tilde{\gamma} \in \mathcal{K}_\infty$.

Why $\mathcal{K}_\infty/\mathcal{KL}$? These classes are closed under the manipulations that appear in small-gain, Lyapunov, and comparison arguments: sums, compositions, and norm changes. They provide a *coordinate-free* way to state stability and input gains: the specific norm and constants are absorbed into comparison functions.

Useful calculus (closure properties). Let $\gamma, \gamma_1, \gamma_2 \in \mathcal{K}_\infty$ and $\beta, \beta_1, \beta_2 \in \mathcal{KL}$. Then:

1. $c\,\gamma \in \mathcal{K}_\infty$ for all $c > 0$; $\gamma_1 + \gamma_2 \in \mathcal{K}_\infty$; $\gamma_1 \circ \gamma_2 \in \mathcal{K}_\infty$; $r \mapsto \max\{\gamma_1(r), \gamma_2(r)\} \in \mathcal{K}_\infty$.
2. $\min\{\beta_1, \beta_2\} \in \mathcal{KL}$; $r \mapsto \gamma(\beta(r,t)) \in \mathcal{KL}$; $(r,t) \mapsto \beta(\gamma(r), t) \in \mathcal{KL}$.
3. **Norm equivalence:** for any two norms $\|\cdot\|_a, \|\cdot\|_b$ on $\mathbb{R}^n$, there exist $m, M > 0$ with $m\|x\|_a \leq \|x\|_b \leq M\|x\|_a$. Hence $\|x\|_b \leq \gamma(\|x\|_a)$ for $\gamma(r) = Mr \in \mathcal{K}_\infty$, and conversely. This lets us change norms inside ISS inequalities by adjusting the comparison functions.

ISS vs. BIBO. Bounded-input–bounded-output (BIBO) guarantees $\sup_t \|h_t\|$ is finite for bounded inputs but is silent about the *transient* from h_0. ISS strengthens BIBO by requiring a *decaying* $\mathcal{KL}$ term for the initial condition and *gain functions* γ_u, γ_b for inputs/biases. Thus, as $t \to \infty$ and for bounded inputs, the state satisfies $\|h_t\|_\infty \leq \gamma_u(\|u\|_{\infty,1:\infty}) + \gamma_b(\|b\|_\infty)$, with the initial-condition effect vanishing.

Linear discrete-time example. Consider $x_{t+1} = Ax_t + Bu_t + b$ with a matrix norm $\|\cdot\|$ s.t. $\|A\| \le \alpha < 1$. Then by induction,

$$\|x_t\| \le \alpha^t \|x_0\| + \sum_{\tau=0}^{t-1} \alpha^{t-1-\tau} \big(\|B\|\,\|u_\tau\| + \|b\|\big) \le \alpha^t \|x_0\| + \frac{1-\alpha^t}{1-\alpha}\,\|B\|\,\|u\|_{\infty,1:t} + \frac{1-\alpha^t}{1-\alpha}\,\|b\|.$$

Hence ISS holds with

$$\beta(r,t) = \alpha^t r \in \mathcal{KL}, \quad \gamma_u(s) = \tfrac{\|B\|}{1-\alpha}\, s \in \mathcal{K}_\infty, \quad \gamma_b(s) = \tfrac{1}{1-\alpha}\, s \in \mathcal{K}_\infty.$$

This template underlies our RNN bounds: show the state update is a contraction (after gating/nonlinearity), then bound the input and bias channels linearly (or via Lipschitz gains) to obtain explicit $\beta, \gamma_u, \gamma_b$.

ISS via Lyapunov comparison (discrete-time sketch). A function $V : \mathbb{R}^{n_h} \to \mathbb{R}_{\ge 0}$ is an ISS–Lyapunov function if there exist $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$ and $\sigma \in \mathcal{K}_\infty$ such that

$$\alpha_1(\|h\|) \le V(h) \le \alpha_2(\|h\|), \qquad V(h^+) - V(h) \le -\alpha_3(\|h\|) + \sigma(\|u\|) + \sigma(\|b\|).$$

Standard comparison lemmas then yield an ISS estimate of the form in Definition 2 with some $\beta \in \mathcal{KL}$ and $\gamma_u, \gamma_b \in \mathcal{K}_\infty$. In our gated architectures, $V(h) = \|h\|_\infty$ or a weighted ℓ_1/ℓ_∞ norm often suffices once gate and recurrent blocks satisfy the norm constraints derived in the main text.

How this interfaces with gated RNNs. For GRU/LSTM cells, the gate blocks determine an effective contraction factor on the hidden-state channel, while the input and bias channels contribute additive gains. Our ISS conditions enforce that the induced hidden-state Lipschitz constant is strictly < 1, yielding a geometric $\beta(\cdot, t)$, and that the input/candidate paths have bounded operator norms, yielding linear γ-gains.

B Notation Summary

The summary of all variables used in the paper categorized with discrete and continuous parameters with description is provided in table 3.

Table 3: Summary of key notations used in the paper. Parameters are categorized into discrete (θ) and continuous (ϕ) sets where applicable.

Symbol	Description
Latent and Observed Variables	
$\mathbf{x}_t \in \mathbb{R}^M$	Continuous latent state at time t
$\mathbf{x}_{1:t} \in \mathbb{R}^{tM}$	Continuous latent states from time 1 to time t
$z_t \in \{1, \dots, K\}$	Discrete latent mode at time t
$\mathbf{z}_{1:t} \in \{1, \dots, K\}^t$	Discrete latent modes from time 1 tp time t
$\mathbf{y}_t \in \mathbb{R}^N$	Observation at time t
$\mathbf{y}_{1:t} \in \mathbb{R}^{tN}$	Observations from time 1 to time t
Variational Filtering Parameters	
$\boldsymbol{\mu}_{t\|t} \in \mathbb{R}^M$, $\boldsymbol{\Sigma}_{t\|t} \in \mathbb{R}^{M\times M}$	Mean and covariance of filtered posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$
$\boldsymbol{\mu}_{t\|t-1} \in \mathbb{R}^M$, $\boldsymbol{\Sigma}_{t\|t-1} \in \mathbb{R}^{M\times M}$	Mean and covariance of predictive prior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$
$\hat{\boldsymbol{\mu}}_{t\|t} \in \mathbb{R}^M$, $\hat{\boldsymbol{\Sigma}}_{t\|t} \in \mathbb{R}^{M\times M}$	Mean and covariance of approximated filtered posterior $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$
$\hat{\boldsymbol{\mu}}_{t\|t-1} \in \mathbb{R}^M$, $\hat{\boldsymbol{\Sigma}}_{t\|t-1} \in \mathbb{R}^{M\times M}$	Mean and covariance of approximated predictive prior $q(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$
Dynamics and Emission Models	
$\mathbf{A}_{z_t} \in \mathbb{R}^{M\times M}$	Transition matrix selected by z_t (continuous, part of ϕ)
$\mathbf{C}_{z_t} \in \mathbb{R}^{N\times M}$	Emission matrix selected by z_t (continuous, part of ϕ)
$\mathbf{Q}_t \in \mathbb{R}^{M\times M}$	Process noise covariance (mode-independent)
$\mathbf{R}_t \in \mathbb{R}^{N\times N}$	Observation noise covariance (mode-independent)
GRU and Stability Terms	
$\mathbf{u}_t \in \mathbb{R}^{n_u}$	Input to GRU at time t
$\mathbf{h}_t \in \mathbb{R}^{n_h}$	Hidden state of GRU
$\hat{\mathbf{h}}_t$	GRU candidate update
$\mathbf{g}_t$, $\mathbf{f}_t$	GRU update and reset gates
$\mathbf{U}_r$	Weight matrix in GRU recurrent term (ISS constrained)
Optimization Variables	
θ	Parameters related to discrete inference (e.g., $q(z_t \mid \mathbf{x}_{t-1})$ modeled by NN)
ϕ	Parameters related to continuous states (e.g., GRU weights, $\mathbf{A}_{z_t}$, $\mathbf{C}_{z_t}$)

C Proofs

C.1 Nested Message Passing Details Derivation

We provide a proof that the standard message passing updates under the factorization in equation (1) induce the stated nested inference structure.

Belief at $\mathbf{x}_t$. The belief at $\mathbf{x}_t$ is given by the product of incoming messages from the dynamics and observation factors:

$$\text{Belief}(\mathbf{x}_t) \propto m_{f_{\text{dyn}}\to\mathbf{x}_t} \times m_{f_{\text{obs}}\to\mathbf{x}_t}, \tag{41}$$

where the incoming messages are:

$$m_{f_{\text{dyn}} \to \mathbf{x}_t} = \sum_{z_t} \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, p(z_t \mid \mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}, \tag{42}$$

$$m_{f_{\text{obs}} \to \mathbf{x}_t} = p(\mathbf{y}_t \mid \mathbf{x}_t). \tag{43}$$

Substituting these into the belief expression yields:

$$\text{Belief}(\mathbf{x}_t) \propto p(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{z_t} \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, p(z_t \mid \mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}. \tag{44}$$

Belief at z_t. The belief at z_t is given by the product of incoming messages from the dynamics and mode factors:

$$\text{Belief}(z_t) \propto m_{f_{\text{dyn}} \to z_t} \times m_{f_{\text{mode}} \to z_t}, \tag{45}$$

where the messages are:

$$m_{f_{\text{dyn}} \to z_t} = \iint p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, \text{Belief}(\mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_t)\, d\mathbf{x}_{t-1}\, d\mathbf{x}_t, \tag{46}$$

$$m_{f_{\text{mode}} \to z_t} = \int p(z_t \mid \mathbf{x}_{t-1})\, \text{Belief}(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}. \tag{47}$$

Since $\text{Belief}(\mathbf{x}_t)$ already incorporates the likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t)$, we expand $m_{f_{\text{dyn}} \to z_t}$ as:

$$m_{f_{\text{dyn}} \to z_t} \propto \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})} \left[\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)} \left[p(\mathbf{y}_t \mid \mathbf{x}_t) \right] \right], \tag{48}$$

where the expectation over $\mathbf{x}_t$ reflects the probabilistic propagation and local likelihood weighting.

Similarly, the mode message can be expressed as:

$$m_{f_{\text{mode}} \to z_t} = \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})} \left[p(z_t \mid \mathbf{x}_{t-1}) \right]. \tag{49}$$

Posterior inference for $\mathbf{x}_t$ and z_t. The approximate posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ can thus be written as:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t \mid \mathbf{x}_t)\, \mathbb{E}_{\mathbf{x}_{t-1}, z_t} \left[p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t) p(z_t \mid \mathbf{x}_{t-1}) \right], \tag{50}$$

where the expectation is over the predictive prior $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ and switching distribution $p(z_t \mid \mathbf{x}_{t-1})$.

Similarly, the approximate posterior $p(z_t \mid \mathbf{y}_{1:t})$ is given by:

$$p(z_t \mid \mathbf{y}_{1:t}) \propto \mathbb{E}_{\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})} \left[p(z_t \mid \mathbf{x}_{t-1}) \times \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)} \left[p(\mathbf{y}_t \mid \mathbf{x}_t) \right] \right], \tag{51}$$

matching the nested structure described in equation (6).

Approximate factorization of $p(\mathbf{x}_t, z_t \mid \mathbf{y}_{1:t})$. Due to the model structure, where z_t depends primarily on $\mathbf{x}_t$ and future z_{t+1} depends only on $\mathbf{x}_{t+1}$, the posterior at each time step approximately factorizes as:

$$p(\mathbf{x}_t, z_t \mid \mathbf{y}_{1:t}) \approx p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \times p(z_t \mid \mathbf{y}_{1:t}), \tag{52}$$

where the approximation relies on the weak dependency between $\mathbf{x}_t$ and z_t after local updates.

Thus, standard local message passing induces a nested sampling structure consistent with the proposed inference framework.

This posterior factorization is a structured mean-field approximation that arises directly from the local message-passing design of our model. At each time step t, inference is restricted to the latent variables at that step, without referencing future or nonlocal latents. This choice enables strictly online filtering and scalable, per-step gradient estimation (as formalized in Theorem 6).

More expressive posterior families could be obtained by introducing higher-order factors such as $p(x_t, x_{t+1}, z_t)$. However, such terms would require backward message passing across time, thereby breaking the localized factorization that underpins our inference framework. This would induce nonlocal dependencies—e.g., marginalization over x_{t+1}—and complicate both the message schedule and gradient flow, increasing computational and memory cost.

From the perspective of expressivity, modeling richer joint posteriors (e.g., over (x_t, x_{t+1}, z_t)) could capture longer-range temporal correlations. Yet the practical benefits of this added complexity remain uncertain. Our empirical results indicate that the proposed approximation is already sufficiently expressive to capture relevant dynamics across diverse benchmarks, while retaining the advantages of online and scalable inference. We leave the exploration of such extensions to future work.

Recursive Factorization (Forward Path). Since the message passing algorithm proceeds forward in time and updates latent variables $\mathbf{x}_t$, z_t based only on $\mathbf{y}_{1:t}$, we may write:

$$p(\mathbf{x}_{1:t'}, z_{1:t'} \mid \mathbf{y}_{1:t'}) \approx \prod_{t=1}^{t'} p(\mathbf{x}_t, z_t \mid \mathbf{y}_{1:t}). \tag{53}$$

This corresponds to a structured filtering assumption in which each latent pair $(\mathbf{x}_t, z_t)$ is inferred independently conditioned on previous beliefs, without backward smoothing refinement.

□

C.2 Complexity Analysis: RNN-Based Updates vs. LG

We analyze the computational advantage of the proposed π-SSM parameterization over traditional Linear Gaussian State Space Models (LGs) during inference.

In LGs, the posterior state update involves matrix inversions of the state covariance matrix, which incurs a computational cost of $\mathcal{O}(M^3)$ per time step, where M is the dimensionality of the latent continuous state $\mathbf{x}_t$.

In contrast, our π-SSM uses a RNN-based update to approximate the Kalman-style correction. A standard RNN cells such as GRU or LSTM with input size n_u and hidden state size n_h has a forward-pass complexity of $\mathcal{O}(3n_h(n_h + n_u + 3))$ Chung et al. (2014). In our case, the RNN processes the flattened covariance matrix input $\hat{\mathbf{\Sigma}}_{t|t-1} \in \mathbb{R}^{M \times M}$ along with additional features, yielding an effective input size $n_u = M^2$. Thus, the per-step complexity of the RNN becomes $\mathcal{O}(3n_h M^2)$, which scales linearly with the input dimension.

When $M \gg n_h$, this leads to a significant computational saving compared to the cubic cost of LG updates. In particular, π-SSM offers a speedup factor on the order of $\frac{M}{3n_h}$, making it well-suited for high-dimensional latent state models.

C.3 Proof of Theorem 1

We aim to relate the true predictive likelihood $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ and the surrogate likelihood $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ defined as:

$$q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) = \mathbb{E}_{q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})} \left[\mathbb{E}_{q(\mathbf{z}_t \mid \mathbf{x}_{t-1})} \left[\mathbb{E}_{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{z}_t)} \left[p(\mathbf{y}_t \mid \mathbf{x}_t) \right] \right] \right]. \tag{54}$$

Now consider the KL divergence between the true joint posterior and the variational approximation:

$$\log p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) = \log \mathbb{E}_{p(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{x}_t \mid \mathbf{y}_{1:t-1})} \left[p(\mathbf{y}_t \mid \mathbf{x}_t) \right] \tag{55}$$

$$\geq \mathbb{E}_{q(\mathbf{x}_{t-1}, \mathbf{z}_t)} \left[\log \mathbb{E}_{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{z}_t)} \left[p(\mathbf{y}_t \mid \mathbf{x}_t) \right] \right] - \mathrm{KL} \left(q(\mathbf{x}_{t-1}, \mathbf{z}_t) \| p(\mathbf{x}_{t-1}, \mathbf{z}_t \mid \mathbf{y}_{1:t-1}) \right), \tag{56}$$

where the inequality follows from the variational bound and Jensen's inequality.

Using the factorization:

$$q(\mathbf{x}_{t-1}, \mathbf{z}_t) = q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\, q(\mathbf{z}_t \mid \mathbf{x}_{t-1}), \quad p(\mathbf{x}_{t-1}, \mathbf{z}_t \mid \mathbf{y}_{1:t-1}) = p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\, p(\mathbf{z}_t \mid \mathbf{x}_{t-1}),$$

we expand the KL divergence:

$$\mathrm{KL}(q(\mathbf{x}_{t-1}, \mathbf{z}_t) \| p(\mathbf{x}_{t-1}, \mathbf{z}_t \mid \mathbf{y}_{1:t-1})) = \mathrm{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \| p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})) \tag{57}$$

$$+ \mathbb{E}_{q(\mathbf{x}_{t-1})} \left[\mathrm{KL}(q(\mathbf{z}_t \mid \mathbf{x}_{t-1}) \| p(\mathbf{z}_t \mid \mathbf{x}_{t-1})) \right]. \tag{58}$$

Thus we obtain the lower bound:

$$\log p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) \geq \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) + \mathcal{E}_t, \tag{59}$$

with:

$$\mathcal{E}_t = -\mathrm{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \| p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})) - \mathbb{E}_{q(\mathbf{x}_{t-1})} \left[\mathrm{KL}(q(\mathbf{z}_t \mid \mathbf{x}_{t-1}) \| p(\mathbf{z}_t \mid \mathbf{x}_{t-1})) \right],$$

completing the proof.

C.4 Gradient Estimation Details

To optimize the surrogate predictive log-likelihood objective $\mathcal{L}(\mathbf{y}_{1:T}) = \sum_{t=1}^{T} \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$, we compute gradients with respect to both continuous and discrete inference parameters. These gradient expressions arise from the use of standard backpropagation for continuous variables and REINFORCE for discrete ones.

Continuous Gradient Derivation. For the continuous parameters ϕ of the inference model $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, the predictive surrogate is differentiable. Thus, we directly compute:

$$\nabla_\phi \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) = \nabla_\phi \log \mathbb{E}_{\mathbf{x}_{t-1}, z_t, \mathbf{x}_t} \left[p(\mathbf{y}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)\, q(z_t \mid \mathbf{x}_{t-1})\, q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \right], \tag{60}$$

which can be approximated via Monte Carlo sampling through the reparameterizable distributions. Gradients are propagated using standard backpropagation through the computational graph.

Discrete Gradient Derivation (REINFORCE). For the discrete parameters θ of the categorical distribution $q(z_t \mid \mathbf{x}_{t-1})$, we cannot backpropagate through sampling. We instead use the REINFORCE estimator Williams (1992), which applies the score function trick:

$$\nabla_\theta \log \mathbb{E}_{z_t}[f(z_t)] = \mathbb{E}_{z_t \sim q(z_t \mid \mathbf{x}_{t-1})} \left[f(z_t)\, \nabla_\theta \log q(z_t \mid \mathbf{x}_{t-1}) \right], \tag{61}$$

where we take $f(z_t) = \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ as a reward signal. To reduce variance, we use a control variate b_t (e.g., an exponential moving average of past rewards), yielding the final gradient estimate:

$$\nabla_\theta \mathcal{L}(\mathbf{y}_{1:T}) \approx \sum_{t=1}^{T} \mathbb{E}_{z_t \sim q(z_t \mid \mathbf{x}_{t-1})} \left[(\log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) - b_t)\, \nabla_\theta \log q(z_t \mid \mathbf{x}_{t-1}) \right]. \tag{62}$$

Note: The reward term $\log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ is treated as a constant during gradient computation with respect to θ. In practice, this is implemented using `detach()` in automatic differentiation frameworks.

C.5 Proof of Theorem 6

Proof. In standard SLDS models with Markovian discrete dynamics, the discrete latent variable z_t depends on the full history of past discrete variables via a chain of conditional dependencies, i.e., $p(z_t \mid z_{1:t-1}) = \prod_{\tau=1}^{t} p(z_\tau \mid z_{\tau-1})$. As a result, computing the gradient of the marginal log-likelihood with respect to parameters of z_t involves backward propagation through all previous time steps, leading to cumulative complexity $\sum_{t=1}^{T} \mathcal{O}(t) = \mathcal{O}(T^2)$.

In contrast, our model factorizes the discrete latent distribution as $p(z_t \mid x_t)$, which depends only on the local continuous latent state x_t inferred from observations up to time t. As a consequence, each $\nabla_\theta \log q(z_t \mid x_t)$ term—used in REINFORCE gradient estimates—only requires computing a local reward signal $\log q(y_t \mid y_{1:t-1})$ and does not involve backward dependencies through $z_{1:t-1}$. This allows each discrete latent parameter θ to be updated once per time step with cost $\mathcal{O}(1)$, leading to total complexity $\mathcal{O}(T)$.

Therefore, the nested local structure of $p(z_t \mid x_t)$ yields a linearly scalable training procedure in contrast to the quadratic cost in standard SLDS inference. □

C.6 A General ISS Scheme for Gated, Saturating RNN Modules

We show that a broad family of recurrent modules admits an ISS bound under a simple *effective recurrent gain* condition. The scheme is stated in terms of the standard ISS definition with comparison functions in $\mathcal{K}_\infty$ and $\mathcal{KL}$.

Definition 3 (Gated, Saturating RNN (GSRNN))**.** Let $u_t \in \mathbb{R}^{n_u}$ and $h_t \in \mathbb{R}^{n_h}$. Consider

$$h_{t+1} = g(h_t, u_t) \odot h_t \; + \; \big(1 - g(h_t, u_t)\big) \odot \phi\big(W u_t + U\, \Psi(h_t, u_t) + b\big), \tag{63}$$

where g, Ψ act componentwise, $\odot$ denotes the Hadamard product, $W \in \mathbb{R}^{n_h \times n_u}$, $U \in \mathbb{R}^{n_h \times n_h}$, $b \in \mathbb{R}^{n_h}$ are parameters, and $\phi : \mathbb{R} \to [-1, 1]$ is a saturating nonlinearity (e.g., tanh or hard-tanh). We use the induced ∞-norm for vectors and matrices; for a matrix A, $\|A\|_\infty := \max_i \sum_j |a_{ij}|$.

Standing assumptions.

(A1) (Bounded inputs) $\|u_t\|_\infty \le U_{\max}$ for all t (can be enforced by normalization).

(A2) (Saturating candidate) ϕ is monotone, L_ϕ-Lipschitz with $L_\phi \le 1$, and $\phi(\mathbb{R}) \subseteq [-1, 1]$.

(A3) (Gates in $(0, 1)$ and bounded mixing) There exist constants

$$0 \le \underline{g} < \bar{g} < 1, \qquad 0 \le \hat{\psi} < 1$$

such that for all (h, u) in the operating domain

$$\underline{g} \le \|g(h, u)\|_\infty \le \bar{g}, \qquad \|\Psi(h, u)\|_\infty \le \hat{\psi}\, \|h\|_\infty.$$

Example. If $g = \sigma(W_g u + U_g h + b_g)$ and $\Psi(h, u) = \sigma(W_\psi u + U_\psi h + b_\psi) \odot h$, then by monotonicity of σ and $\|u\|_\infty \le U_{\max}$ we can take

$$\bar{g} = \sigma\big(\|[W_g\ U_g\ b_g]\|_\infty\ \max\{U_{\max}, 1\}\big), \quad \hat{\psi} = \sigma\big(\|[W_\psi\ U_\psi\ b_\psi]\|_\infty\ \max\{U_{\max}, 1\}\big),$$

and $\underline{g} = \sigma(-M_g) = 1 - \bar{g}$ if a symmetric bound M_g is known. Here $\|[W\ U\ b]\|_\infty$ denotes the ∞-norm of the block-row concatenation.

Theorem 3 (ISS for GSRNN)**.** *Under* (A1)–(A3)*, define the effective recurrent gain*

$$c := L_\phi\, \|U\|_\infty\, \hat{\psi}. \tag{64}$$

If

$$c < 1, \tag{65}$$

then the system (63) *is Input-to-State Stable (ISS) in the $\|\cdot\|_\infty$ norm. In particular, with*

$$\delta := (1 - \bar{g})\,(1 - c) \in (0, 1), \tag{66}$$

an admissible ISS bound is

$$\|h_t\|_\infty \;\le\; (1 - \delta)^t \|h_0\|_\infty \;+\; \frac{1 - \underline{g}}{\delta}\, L_\phi \big(\|W\|_\infty\, \|u\|_{\infty,1:t} + \|b\|_\infty\big), \tag{67}$$

where $\|u\|_{\infty,1:t} := \max_{1 \le \tau \le t} \|u_\tau\|_\infty$. Thus the comparison functions

$$\beta(r, t) = (1 - \delta)^t r \in \mathcal{KL}, \quad \gamma_u(s) = \frac{1 - \underline{g}}{\delta}\, L_\phi\, \|W\|_\infty\, s \in \mathcal{K}_\infty, \quad \gamma_b(s) = \frac{1 - \underline{g}}{\delta}\, L_\phi\, s \in \mathcal{K}_\infty$$

witness ISS in the sense of Definition 2.

Proof. Fix a component j. By (A2) and sub-multiplicativity,

$$\begin{aligned}\big|\phi(Wu + U\,\Psi(h,u) + b)\big|_j &\le L_\phi\big(\|W\|_\infty\|u\|_\infty + \|U\|_\infty\|\Psi(h,u)\|_\infty + \|b\|_\infty\big)\\ &\le L_\phi\big(\|W\|_\infty\|u\|_\infty + \|U\|_\infty\hat{\psi}\|h\|_\infty + \|b\|_\infty\big).\end{aligned}$$

Let $a_u := L_\phi\|W\|_\infty\|u\|_\infty$, $a_b := L_\phi\|b\|_\infty$, and $c := L_\phi\|U\|_\infty\hat{\psi}$. From (63) we get

$$|h_{t+1,j}| \le \big(g_j + (1-g_j)c\big)\|h_t\|_\infty + (1-g_j)(a_u + a_b).$$

Since $c < 1$, the map $x \mapsto x + (1-x)c$ is increasing; hence with $g_j \le \bar{g}$,

$$g_j + (1-g_j)c \le \bar{g} + (1-\bar{g})c = 1 - \delta, \quad \delta := (1-\bar{g})(1-c).$$

Also $1 - g_j \le 1 - \underline{g}$. Taking $\max_j$,

$$\|h_{t+1}\|_\infty \le (1-\delta)\|h_t\|_\infty + (1-\underline{g})(a_u + a_b).$$

Unrolling the linear recursion yields (67). □

Remark 2 (Globalization via entrance into the saturating region)**.** If the bounds $\underline{g}, \bar{g}, \hat{\psi}$ are guaranteed only when h_t lies in $\mathcal{H} := [-1,1]^{n_h}$, the saturation of ϕ implies the standard entrance property: starting from any h_0, $\|h_t\|_\infty$ strictly decreases while $\|h_t\|_\infty > 1$ and hits $\mathcal{H}$ in finite time; subsequently Theorem 3 applies. The finite transient can be absorbed into a multiplicative factor on β.

We now illustrate how the general scheme (63) covers widely used gated RNN modules.

GRU. The standard GRU cell system detailed in equations (29)–(33) is exactly of the form (63) by taking

$$g(h_t, u_t) = z_t, \quad \phi(\cdot) = \tanh(\cdot), \quad \Psi(h_t, u_t) = f_t \odot h_t, \quad W = W_r,\ U = U_r,\ b = b_r.$$

Thus the sufficient ISS condition (65) specializes to

$$\boxed{\|U_r\|_\infty\,\hat{\sigma}_f < 1} \tag{68}$$

with

$$\hat{\sigma}_f := \sigma\big(\|[W_f \;\; U_f \;\; b_f]\|_\infty\big), \qquad \bar{g} := \sigma\big(\|[W_g \;\; U_g \;\; b_g]\|_\infty\big).$$

We provide two schemes for proof (i) ISS inequality and (ii) Lyapunov function in Appendix C.7.

LSTM (CIFG variant). The Coupled Input–Forget Gate (CIFG) LSTM reads

$$f_t = \sigma(W_f u_t + U_f h_{t-1} + b_f), \tag{69}$$
$$i_t = 1 - f_t, \tag{70}$$
$$\tilde{c}_t = \tanh(W_c u_t + U_c h_{t-1} + b_c), \tag{71}$$
$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot \tilde{c}_t, \tag{72}$$
$$h_t = o_t \odot \tanh(c_t), \quad o_t = \sigma(W_o u_t + U_o h_{t-1} + b_o). \tag{73}$$

By identifying the cell state c_t with the hidden state h_t in (63), we see that this is of the general form with

$$g(h_t, u_t) = f_t, \quad \phi(\cdot) = \tanh(\cdot), \quad \Psi(h_t, u_t) = h_t, \quad W = W_c,\ U = U_c,\ b = b_c. \tag{74}$$

The sufficient ISS condition becomes

$$\boxed{\|U_c\|_\infty\,\bar{\sigma}_i < 1,} \tag{75}$$

with $\bar{\sigma}_i := \sigma(\|[W_i\ U_i\ b_i]\|_\infty)$ and decay margin

$$\boxed{\delta = (1 - \bar{\sigma}_f)\,\big(1 - \bar{\sigma}_i\|U_c\|_\infty\big)}, \qquad \boxed{\bar{\sigma}_f = \sigma(\|[W_f\ U_f\ b_f]\|_\infty).} \tag{76}$$

The output relation $h_t = o_t \odot \tanh(c_t)$ is a bounded Lipschitz readout with $\|h_t\|_\infty \le \bar{\sigma}_o\|c_t\|_\infty$ where $\bar{\sigma}_o := \sigma(\|[W_o\ U_o\ b_o]\|_\infty)$, and thus preserves ISS.

LSTM (standard decoupled f_t, i_t). For the standard LSTM with independent input and forget gates,

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c u_t + U_c h_{t-1} + b_c), \tag{77}$$
$$h_t = o_t \odot \tanh(c_t), \tag{78}$$

we can view the update as a mild extension of (63) where the second multiplier is $i_t \in (0,1)$ rather than $1 - g_t$. The ISS proof scheme remains identical by replacing $(1-g)$ with an upper bound on i_t, so that the sufficient conditions are

$$\boxed{(1+\bar{\sigma}_o)\,\bar{\sigma}_f < 1, \qquad (1+\bar{\sigma}_o)\,\bar{\sigma}_i\,\|U_c\|_\infty < 1,} \tag{79}$$

which matches standard ISS/IOS bounds for LSTMs.

C.7 Proof of ISS schemes for GRU cells

To prove this theorem, we first state a few assumptions and lemmas.

Assumption 4 (Unity-bounded input)**.** The input sequence $\mathbf{u}_t \in \mathbb{R}^{n_u}$ is unity-bounded:

$$\mathbf{u}_t \in \mathcal{U} \subseteq [-1,1]^{n_u} \quad \Longleftrightarrow \quad \|\mathbf{u}_t\|_\infty \le 1. \tag{80}$$

This can be enforced via input normalization.

We recall the standard properties of the activation functions used in GRU cells:

$$\sigma(\cdot) \in (0,1), \qquad \text{Lipschitz with } L_\sigma = \tfrac{1}{4}, \tag{81}$$
$$\phi(\cdot) \in (-1,1), \qquad \text{Lipschitz with } L_\phi = 1, \tag{82}$$

where both Lipschitz constants hold for componentwise application under $\|\cdot\|_\infty$.

For notational compactness, define the candidate update

$$\hat{\mathbf{h}}_t := \phi\big(\mathbf{W}_r \mathbf{u}_t + \mathbf{U}_r\big(\mathbf{f}_t \odot \mathbf{h}_t\big) + \mathbf{b}_r\big)\,, \tag{83}$$

where the (reset/forget) gate is

$$\mathbf{f}_t := \sigma(\mathbf{W}_f \mathbf{u}_t + \mathbf{U}_f \mathbf{h}_t + \mathbf{b}_f)\,, \tag{84}$$

and the update/keep gate is explicitly

$$\mathbf{g}_t := \sigma(\mathbf{W}_g \mathbf{u}_t + \mathbf{U}_g \mathbf{h}_t + \mathbf{b}_g)\,, \tag{85}$$

so that the hidden update reads

$$\mathbf{h}_{t+1} = \mathbf{g}_t \odot \mathbf{h}_t + \big(\mathbf{1} - \mathbf{g}_t\big) \odot \hat{\mathbf{h}}_t. \tag{86}$$

Lemma 1 (Invariance of the saturating hypercube)**.** Let $\mathcal{H} := [-1,1]^{n_h}$. For any input $\mathbf{u}_t$ (no bound needed),

$$\mathbf{h}_t \in \mathcal{H} \implies \mathbf{h}_{t+1} \in \mathcal{H}.$$

Proof. Consider coordinate j. Write $\omega_j(t) := g_{t,j} \in (0,1)$ and $\eta_j(t) := \hat{h}_{t,j} \in (-1,1)$ by (81)–(82). The update is

$$h_{t+1,j} = \omega_j(t)\, h_{t,j} + \big(1 - \omega_j(t)\big)\, \eta_j(t).$$

If $h_{t,j} \in [-1,1]$ and $\eta_j(t) \in (-1,1)$, then $h_{t+1,j}$ is a convex combination of two values in $[-1,1]$, hence $h_{t+1,j} \in [-1,1]$. Since this holds for all j, $\mathbf{h}_{t+1} \in \mathcal{H}$. □

Lemma 2. For any arbitrary initial state $\mathbf{h}_0 \in \mathbb{R}^{n_h}$, the following holds:

i. If $\mathbf{h}_0 \notin \mathcal{H}$, then $\|\mathbf{h}_t\|_\infty$ is strictly decreasing until $\mathbf{h}_t \in \mathcal{H}$;

ii. The convergence happens in finite time, i.e., there exists a finite $\bar{t} \ge 0$ such that $\mathbf{h}_t \in \mathcal{H}$ for all $t \ge \bar{t}$;

iii. Each state component $h_{t,j}$ converges into its invariant set $[-1,1]$ in an exponential fashion.

Proof of Lemma 2. **Case b.** $\mathbf{h}_0 \notin \mathcal{H}$, i.e., $\|\mathbf{h}_0\|_\infty > 1$. Fix a coordinate j and discrete time $k := t$. Define

$$\omega_j(k) := g_{k,j}, \qquad \eta_j(k) := \hat{h}_{k,j}. \tag{87}$$

The component-wise GRU update reads

$$h_{k+1,j} \;=\; \omega_j(k)\, h_{k,j} \;+\; \big(1-\omega_j(k)\big)\, \eta_j(k). \tag{88}$$

Gate and candidate bounds. Using the monotonicity of σ and ϕ and $\|\mathbf{u}_k\|_\infty \le 1$, we have

$$\omega_j(k) \;\le\; \bar{\omega}(k) \;:=\; \sigma\Big(\|\mathbf{W}_g\|_\infty + \|\mathbf{U}_g\|_\infty\, \|\mathbf{h}_k\|_\infty + \|\mathbf{b}_g\|_\infty\Big) \;<\; 1, \tag{89}$$

and define the complementary lower bound on the "off-gate":

$$\underline{\delta}(k) \;:=\; 1-\bar{\omega}(k) \;\Longrightarrow\; 1-\omega_j(k) \;\ge\; \underline{\delta}(k) \;>\; 0. \tag{90}$$

For the candidate (using $\phi(\cdot) \in (-1,1)$), there exists $\epsilon(k) \in (0,1)$ such that

$$|\eta_j(k)| \;\le\; 1-\epsilon(k), \qquad \epsilon(k) \;=\; 1-\phi\Big(\|\mathbf{W}_r\|_\infty + \|\mathbf{U}_r\|_\infty\, \|\mathbf{h}_k\|_\infty + \|\mathbf{b}_r\|_\infty\Big). \tag{91}$$

One–step decrease for a component outside $[-1,1]$. Taking absolute values in (88) yields

$$|h_{k+1,j}| \;\le\; \omega_j(k)\, |h_{k,j}| \;+\; \big(1-\omega_j(k)\big)\, |\eta_j(k)|. \tag{92}$$

If $|h_{k,j}| > 1$, subtract $|h_{k,j}|$ from both sides of (92) to obtain

$$|h_{k+1,j}| - |h_{k,j}| \;\le\; \big(1-\omega_j(k)\big)\, \big(|\eta_j(k)| - |h_{k,j}|\big). \tag{93}$$

Using $|\eta_j(k)| \le 1-\epsilon(k)$ and $|h_{k,j}| > 1$ gives

$$|\eta_j(k)| - |h_{k,j}| \;\le\; (1-\epsilon(k)) - |h_{k,j}| \;\le\; -\epsilon(k), \tag{94}$$

and thus, by (90),

$$|h_{k+1,j}| - |h_{k,j}| \;\le\; -\big(1-\omega_j(k)\big)\, \epsilon(k) \;\le\; -\underline{\delta}(k)\, \epsilon(k) \;<\; 0. \tag{95}$$

Hence, whenever $|h_{k,j}| > 1$, that component strictly decreases in magnitude:

$$|h_{k+1,j}| \;<\; |h_{k,j}|. \tag{96}$$

Sup-norm strictly decreases until $\mathbf{h}_k \in \mathcal{H}$. Let

$$j^\star \;\in\; \arg\max_j |h_{k,j}|, \qquad |h_{k,j^\star}| \;=\; \|\mathbf{h}_k\|_\infty \;>\; 1. \tag{97}$$

By (95),

$$|h_{k+1,j^\star}| \;<\; \|\mathbf{h}_k\|_\infty. \tag{98}$$

For any i, using (92) and (91),

$$|h_{k+1,i}| \;\le\; \omega_i(k)\, |h_{k,i}| \;+\; \big(1-\omega_i(k)\big)\, (1-\epsilon(k)) \;\le\; \omega_i(k)\, \|\mathbf{h}_k\|_\infty \;+\; \big(1-\omega_i(k)\big)\, (1-\epsilon(k)). \tag{99}$$

Since $\|\mathbf{h}_k\|_\infty > 1$ and $1-\epsilon(k) < 1$, the convex combination in (99) is strictly less than $\|\mathbf{h}_k\|_\infty$:

$$|h_{k+1,i}| \;<\; \|\mathbf{h}_k\|_\infty \quad \text{for all } i. \tag{100}$$

Taking the maximum over i in (100) yields

$$\|\mathbf{h}_{k+1}\|_\infty \;<\; \|\mathbf{h}_k\|_\infty, \tag{101}$$

which proves that the sup-norm is strictly decreasing at every step as long as $\|\mathbf{h}_k\|_\infty > 1$. Therefore the trajectory strictly decreases until it enters $\mathcal{H} = [-1, 1]^{n_h}$.

Finite-time entrance into $\mathcal{H}$. By monotonicity of σ and ϕ, and using the sup-norm bound $\|\mathbf{h}_k\|_\infty \le \max\{\|\mathbf{h}_0\|_\infty, 1\}$, the arguments in the gate/candidate bounds are no larger at time k than at $k = 0$. Hence

$$0 \;<\; \bar{\omega}(k) \;\le\; \bar{\omega}(0) \;<\; 1, \tag{102}$$

and defining the complementary lower bound

$$\underline{\delta}(k) \;:=\; 1 - \bar{\omega}(k), \tag{103}$$

we obtain

$$0 \;<\; \underline{\delta}(0) \;=\; 1 - \bar{\omega}(0) \;\le\; \underline{\delta}(k) \;\le\; 1 - \omega_j(k) \;\le\; 1. \tag{104}$$

Similarly, for the candidate,

$$\epsilon(k) \;\ge\; \epsilon(0) \;>\; 0, \tag{105}$$

and therefore

$$|\eta_j(k)| \;\le\; 1 - \epsilon(k) \;\le\; 1 - \epsilon(0) \;=:\; \bar{\eta} \;<\; 1. \tag{106}$$

Combining (104)–(105) with the one–step difference inequality valid whenever $|h_{k,j}| > 1$,

$$|h_{k+1,j}| - |h_{k,j}| \;\le\; -\big(1 - \omega_j(k)\big)\,\epsilon(k) \;\le\; -\,\underline{\delta}(0)\,\epsilon(0) \;<\; 0, \tag{107}$$

we see that each step reduces $|h_{k,j}|$ by at least the fixed amount $\underline{\delta}(0)\,\epsilon(0)$ until it enters $[-1, 1]$. Hence the hitting time of $[-1, 1]$ for component j is bounded by

$$\bar{k}_j \;=\; \begin{cases} \left\lceil \dfrac{|h_{0,j}| - 1}{\underline{\delta}(0)\,\epsilon(0)} \right\rceil, & \text{if } |h_{0,j}| > 1, \\ 0, & \text{if } |h_{0,j}| \le 1\ , \end{cases} \tag{108}$$

and taking

$$\bar{t} \;=\; \max_{1 \le j \le n_h} \bar{k}_j \tag{109}$$

yields $\mathbf{h}_t \in \mathcal{H}$ for all $t \ge \bar{t}$.

Exponential convergence. We now show that each component converges to $[-1, 1]$ at an exponential rate. Decompose the evolution of coordinate j from (88) as

$$h_{k,j} \;=\; h_{k,\alpha j} + h_{k,\beta j}, \tag{110}$$

with

$$h_{k,\alpha j} = \Big(\prod_{t=0}^{k-1} \omega_j(t)\Big) h_{0,j}, \tag{111}$$

$$h_{k,\beta j} = \sum_{t=0}^{k-1} \Big(\prod_{h=t+1}^{k-1} \omega_j(h)\Big)\,(1 - \omega_j(t))\,\eta_j(t). \tag{112}$$

Decay of $h_{k,\alpha j}$. Since $\omega_j(t) \le \bar{\omega}(0) < 1$ for all t,

$$|h_{k,\alpha j}| = \Big|\Big(\prod_{t=0}^{k-1} \omega_j(t)\Big) h_{0,j}\Big| \;\le\; \big(\bar{\omega}(0)\big)^k\,|h_{0,j}|, \tag{113}$$

which tends to 0 exponentially as $k \to \infty$.

Decay of $h_{k,\beta j}$. By $|\eta_j(t)| \le 1 - \epsilon(0)$ and $\omega_j(\cdot) \le \bar{\omega}(0)$,

$$|h_{k,\beta j}| \;\le\; (1 - \epsilon(0)) \sum_{t=0}^{k-1} \big(\bar{\omega}(0)\big)^{\,k-1-t}\,(1 - \omega_j(t)). \tag{114}$$

Since $1-\omega_j(t) \le 1$, the sum is bounded by a finite geometric series:

$$|h_{k,\beta j}| \;\le\; (1-\epsilon(0)) \sum_{t=0}^{k-1} \big(\bar{\omega}(0)\big)^{\,k-1-t} = (1-\epsilon(0))\, \frac{1-\big(\bar{\omega}(0)\big)^{k}}{1-\bar{\omega}(0)}. \tag{115}$$

Combining the bounds. From (113) and (115),

$$\begin{aligned} |h_{k,j}| &\le |h_{k,\alpha j}| + |h_{k,\beta j}| \\ &\le \big(\bar{\omega}(0)\big)^{k} |h_{0,j}| \;+\; (1-\epsilon(0))\, \frac{1-\big(\bar{\omega}(0)\big)^{k}}{1-\bar{\omega}(0)}. \end{aligned} \tag{116}$$

Because $\bar{\omega}(0) < 1$, the first term decays exponentially, and the second term converges exponentially to a constant strictly less than 1. Therefore each component $h_{k,j}$ converges exponentially to the invariant interval $[-1, 1]$, completing the proof of Lemma 2(iii).

□

Proof of ISS condition for GRU cells via inequality. **Case a.** $\mathbf{h}_0 \in \mathcal{H}$. By Lemma 1, $\mathbf{h}_t \in \mathcal{H}$ for all $t \ge 0$, hence $\|\mathbf{h}\|_\infty \le 1$ throughout this case.

Fix a coordinate j and suppress the time index for compactness. Denote

$$g_j := g_j(\mathbf{u}, \mathbf{h}), \qquad f_j := f_j(\mathbf{u}, \mathbf{h}), \qquad \hat{h}_j := \hat{\mathbf{h}}_j(\mathbf{u}, \mathbf{h}). \tag{117}$$

From the GRU update (componentwise),

$$h_j^{+} = g_j\, h_j + (1-g_j)\, \hat{h}_j. \tag{118}$$

Taking absolute values and using $g_j \in (0,1)$ and $\hat{h}_j \in (-1,1)$,

$$|h_j^{+}| \;\le\; g_j\, |h_j| \;+\; (1-g_j)\, |\hat{h}_j|. \tag{119}$$

Bounding the reset candidate $\hat{h}_j$. By definition,

$$\hat{\mathbf{h}} = \phi\big(\mathbf{W}_r \mathbf{u} + \mathbf{U}_r(\mathbf{f} \odot \mathbf{h}) + \mathbf{b}_r\big), \qquad \phi = \tanh. \tag{120}$$

Using the Lipschitz property of ϕ with $L_\phi = 1$ and the induced ∞-norm,

$$|\hat{h}_j| \;\le\; \|\mathbf{W}_r\|_\infty \|\mathbf{u}\|_\infty \;+\; \|\mathbf{U}_r\|_\infty \|\mathbf{f} \odot \mathbf{h}\|_\infty \;+\; \|\mathbf{b}_r\|_\infty. \tag{121}$$

Moreover,

$$\|\mathbf{f} \odot \mathbf{h}\|_\infty \;\le\; \|\mathbf{f}\|_\infty \|\mathbf{h}\|_\infty \;\le\; \|\mathbf{f}\|_\infty, \tag{122}$$

so we obtain the useful bound

$$|\hat{h}_j| \;\le\; \|\mathbf{W}_r\|_\infty \|\mathbf{u}\|_\infty \;+\; \|\mathbf{U}_r\|_\infty \|\mathbf{f}\|_\infty \;+\; \|\mathbf{b}_r\|_\infty. \tag{123}$$

(Alternatively, if one prefers to keep the explicit linear dependence on $\|\mathbf{h}\|_\infty$, it is also valid to use $|\hat{h}_j| \le \|\mathbf{W}_r\|_\infty \|\mathbf{u}\|_\infty + \|\mathbf{U}_r\|_\infty \|\mathbf{f}\|_\infty \|\mathbf{h}\|_\infty + \|\mathbf{b}_r\|_\infty$.)

Bounding the reset gate f (not "forget" in GRU). From the gate equation,

$$\mathbf{f} \;=\; \sigma(\mathbf{W}_f \mathbf{u} + \mathbf{U}_f \mathbf{h} + \mathbf{b}_f), \tag{124}$$

we have for each coordinate

$$\big|(\mathbf{W}_f \mathbf{u} + \mathbf{U}_f \mathbf{h} + \mathbf{b}_f)_j\big| \;\le\; \|\mathbf{W}_f\|_\infty \|\mathbf{u}\|_\infty \;+\; \|\mathbf{U}_f\|_\infty \|\mathbf{h}\|_\infty \;+\; \|\mathbf{b}_f\|_\infty. \tag{125}$$

By monotonicity of σ,

$$\|\mathbf{f}\|_\infty \;\le\; \sigma\Big(\|\mathbf{W}_f\|_\infty \|\mathbf{u}\|_\infty \;+\; \|\mathbf{U}_f\|_\infty \|\mathbf{h}\|_\infty \;+\; \|\mathbf{b}_f\|_\infty\Big). \tag{126}$$

Under $\|\mathbf{u}\|_\infty \le 1$ and (in Case a) $\|\mathbf{h}\|_\infty \le 1$, this reduces to the convenient constant bound

$$\|\mathbf{f}\|_\infty \;\le\; \sigma\Big(\|\mathbf{W}_f\|_\infty + \|\mathbf{U}_f\|_\infty + \|\mathbf{b}_f\|_\infty\Big) \;=\; \sigma\big(\|[\mathbf{W}_f \;\; \mathbf{U}_f \;\; \mathbf{b}_f]\|_\infty\big) \;=:\; \hat{\sigma}_f \;\in (0,1), \tag{127}$$

where $\|[\mathbf{W}_f \;\; \mathbf{U}_f \;\; \mathbf{b}_f]\|_\infty$ denotes the maximum row-sum norm of the block-row concatenation.

Combining the bounds. From (123)–(127), we obtain

$$|\hat{h}_j| \;\le\; \phi\Big(\|\mathbf{W}_r\|_\infty + \|\mathbf{U}_r\|_\infty \hat{\sigma}_f \|\mathbf{h}\|_\infty + \|\mathbf{b}_r\|_\infty\Big) \;\le\; \|\mathbf{W}_r\|_\infty + \|\mathbf{U}_r\|_\infty \hat{\sigma}_f \|\mathbf{h}\|_\infty + \|\mathbf{b}_r\|_\infty. \tag{128}$$

Substituting into (119), we find

$$|h_j^+| \;\le\; g_j|h_j| + (1-g_j)|\hat{h}_j| \;\le\; \Big[g_j + (1-g_j)\,\|U_r\|_\infty \hat{\sigma}_f\Big]\|\mathbf{h}\|_\infty + (1-g_j)\Big(\|\mathbf{W}_r\|_\infty \|\mathbf{u}\|_\infty + \|\mathbf{b}_r\|_\infty\Big). \tag{129}$$

Since $c := \|U_r\|_\infty \hat{\sigma}_f < 1$, the map $g \mapsto g + (1-g)c$ is strictly increasing. Thus, for some $\delta \in (0,1)$,

$$g_j + (1-g_j)c \;\le\; 1-\delta. \tag{130}$$

Furthermore, if the update gate is bounded by $\bar{\sigma}_g \in (0,1)$ (e.g. from sigmoid preactivation bounds), then $(1-g_j) \le \bar{\sigma}_g$. Hence from (129),

$$\|\mathbf{h}^+\|_\infty \;\le\; (1-\delta)\,\|\mathbf{h}\|_\infty + \bar{\sigma}_g\Big(\|\mathbf{W}_r\|_\infty \|\mathbf{u}\|_\infty + \|\mathbf{b}_r\|_\infty\Big). \tag{131}$$

Iteration. Iterating (131) over k time steps gives

$$\|\mathbf{h}_k\|_\infty \;\le\; (1-\delta)^k \|\mathbf{h}_0\|_\infty + \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{W}_r\|_\infty \|u\|_{\infty,1:k} + \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{b}_r\|_\infty, \tag{132}$$

where $\|u\|_{\infty,1:k} := \max_{0\le t\le k} \|u_t\|_\infty$.

Conclusion. Therefore, the GRU dynamics satisfies the ISS definition with

$$\beta(r,k) = (1-\delta)^k r, \qquad \gamma_u(s) = \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{W}_r\|_\infty s, \qquad \gamma_b(s) = \frac{\bar{\sigma}_g}{\delta}\,s. \tag{133}$$

Case b. $\mathbf{h}_0 \notin \mathcal{H}$. By Lemma 2, for any initial state outside $\mathcal{H} = [-1,1]^{n_h}$ the trajectory $\mathbf{h}_k$ strictly decreases in norm until it enters $\mathcal{H}$ in finite time. Thus there exists $\bar{k} \ge 0$ such that

$$\mathbf{h}_{\bar{k}} \in \mathcal{H}, \qquad \mathbf{h}_k \in \mathcal{H} \quad \text{for all } k \ge \bar{k}. \tag{134}$$

Moreover, the entrance into $\mathcal{H}$ is exponential: for any $\delta \in (0,1)$ there exists $\mu > 0$ such that

$$\|\mathbf{h}_k\|_\infty \;\le\; \mu\,(1-\delta)^k \|\mathbf{h}_0\|_\infty, \qquad 0 \le k \le \bar{k}. \tag{135}$$

Once the trajectory has entered $\mathcal{H}$ at time $\bar{k}$, the ISS bound from Case a applies. Thus for $k \ge \bar{k}$,

$$\|\mathbf{h}_k\|_\infty \;\le\; (1-\delta)^{\,k-\bar{k}} \|\mathbf{h}_{\bar{k}}\|_\infty + \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{W}_r\|_\infty \|u\|_{\infty,1:k} + \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{b}_r\|_\infty. \tag{136}$$

Conclusion. Combining (135) and (136), the GRU system is ISS with comparison functions

$$\beta(r,k) = \mu(1-\delta)^k r, \qquad \gamma_u(s) = \frac{\bar{\sigma}_g}{\delta}\,\|\mathbf{W}_r\|_\infty s, \qquad \gamma_b(s) = \frac{\bar{\sigma}_g}{\delta}\,s. \tag{137}$$

This concludes the proof of GRU cells ISS condition. □

Proof of ISS condition for GRU cells via Lyapunov function. Consider the GRU cell

$$g = \sigma(W_g u + U_g h + b_g), \qquad f = \sigma(W_f u + U_f h + b_f), \tag{138}$$

$$\hat{h} = \tanh\big(W_r u + U_r(f \odot h) + b_r\big), \qquad h^+ = g \odot h + (1-g) \odot \hat{h}, \tag{139}$$

with input $u \in \mathbb{R}^{n_u}$, hidden state $h \in \mathbb{R}^{n_h}$, and standard nonlinearities $\sigma(\cdot) \in (0,1)$, $\tanh(\cdot) \in (-1,1)$. Assume the input sequence is unity-bounded, $\|u_t\|_\infty \leq 1$.

Define the gate bounds (by monotonicity of σ and $\|u\|_\infty \leq 1$)

$$\bar{g} := \sigma\big(\|[W_g \; U_g \; b_g]\|_\infty\big), \qquad \underline{g} := \sigma\big(-\|[W_g \; U_g \; b_g]\|_\infty\big), \tag{140}$$

$$\hat{\sigma}_f := \sigma\big(\|[W_f \; U_f \; b_f]\|_\infty\big), \qquad \Rightarrow \qquad \underline{g} \leq g_j \leq \bar{g} \;\text{ and }\; \|f\|_\infty \leq \hat{\sigma}_f. \tag{141}$$

If

$$\|U_r\|_\infty \, \hat{\sigma}_f < 1, \tag{142}$$

then the GRU system is input-to-state stable (ISS).

We construct the Lyapunov candidate

$$V(h) := \|h\|_\infty, \tag{143}$$

which satisfies the sandwich bounds

$$\psi_1(\|h\|_\infty) \leq V(h) \leq \psi_2(\|h\|_\infty), \qquad \psi_1(s) = s, \;\; \psi_2(s) = s \in \mathcal{K}_\infty. \tag{144}$$

Step 1. Candidate state bound. Using $|\tanh(z)| \leq |z|$ and the induced ∞-norm,

$$|\hat{h}_j| \leq \|W_r\|_\infty \|u\|_\infty + \|U_r\|_\infty \|f\|_\infty \|h\|_\infty + \|b_r\|_\infty. \tag{145}$$

With the gate bound $\|f\|_\infty \leq \hat{\sigma}_f$, we obtain

$$|\hat{h}_j| \leq a_u \; + \; c\,\|h\|_\infty \; + \; a_b, \qquad c := \|U_r\|_\infty \hat{\sigma}_f, \;\; a_u := \|W_r\|_\infty \|u\|_\infty, \;\; a_b := \|b_r\|_\infty. \tag{146}$$

Step 2. One-step Lyapunov inequality. From the GRU update,

$$|h_j^+| \leq g_j |h_j| + (1 - g_j)\,|\hat{h}_j| \leq \big[g_j + (1 - g_j)\,c\big]\|h\|_\infty + (1 - g_j)(a_u + a_b). \tag{147}$$

Taking the maximum over j yields

$$\|h^+\|_\infty \leq \Big(\max_j \big[g_j + (1 - g_j)\,c\big]\Big)\,\|h\|_\infty \; + \; \Big(\max_j (1 - g_j)\Big)\,(a_u + a_b). \tag{148}$$

Since $c < 1$ and $g \mapsto g + (1-g)c$ is increasing, we have

$$\max_j \big[g_j + (1 - g_j)\,c\big] \leq \bar{g} + (1 - \bar{g})\,c = 1 - \underbrace{(1 - \bar{g})\,(1 - c)}_{=:\,\delta}. \tag{149}$$

Moreover,

$$\max_j (1 - g_j) \;\leq\; 1 - \underline{g}, \tag{150}$$

so (148) becomes

$$\|h^+\|_\infty \leq (1 - \delta)\,\|h\|_\infty \; + \; (1 - \underline{g})\,(a_u + a_b), \qquad \delta := (1 - \bar{g})\,(1 - c) \in (0,1). \tag{151}$$

Step 3. Contraction factor. Since $c < 1$, the map $g \mapsto g + (1-g)c$ is increasing. Therefore,

$$\max_j \big[g_j + (1 - g_j)c\big] = c + \bar{g}(1 - c) = 1 - \delta, \tag{152}$$

with

$$\delta := (1 - \bar{g})(1 - c) = (1 - \bar{g})\big(1 - \|U_r\|_\infty \hat{\sigma}_f\big) \in (0,1). \tag{153}$$

Moreover, using the logistic symmetry $\underline{g} = \sigma(-M_g) = 1 - \bar{g}$ for a symmetric preactivation bound M_g, we have

$$\max_j (1 - g_j) \;\leq\; 1 - \underline{g} \;=\; \bar{g}. \tag{154}$$

Hence,

$$\|h^+\|_\infty - \|h\|_\infty \;\leq\; -\delta\,\|h\|_\infty \; + \; \bar{g}\,\big(\|W_r\|_\infty \|u\|_\infty + \|b_r\|_\infty\big). \tag{155}$$

Step 4. Identification of comparison functions. This is the ISS–Lyapunov inequality

$$V(h^+) - V(h) \;\le\; -\psi(\|h\|_\infty) \;+\; \sigma_u(\|u\|_\infty) \;+\; \sigma_b(\|b_r\|_\infty), \tag{156}$$

with

$$\psi(s) = \delta\, s, \qquad \sigma_u(s) = \bar{g}\,\|W_r\|_\infty\, s, \qquad \sigma_b(s) = \bar{g}\, s. \tag{157}$$

Step 5. Iteration and ISS bound. Unrolling over k steps yields

$$\|h_k\|_\infty \;\le\; (1-\delta)^k \|h_0\|_\infty \;+\; \frac{\bar{g}}{\delta}\,\|W_r\|_\infty\,\|u\|_{\infty,1:k} \;+\; \frac{\bar{g}}{\delta}\,\|b_r\|_\infty, \tag{158}$$

where $\|u\|_{\infty,1:k} := \max_{0\le t\le k}\|u_t\|_\infty$. Thus, admissible comparison functions are

$$\beta(r,k) = (1-\delta)^k\, r, \qquad \gamma_u(s) = \frac{\bar{g}}{\delta}\,\|W_r\|_\infty\, s, \qquad \gamma_b(s) = \frac{\bar{g}}{\delta}\, s. \tag{159}$$

Therefore, the GRU system is ISS. □

C.8 Proof of ISS schemes for LSTM system

Proof of ISS condition for LSTM via general inequality. Consider the (decoupled) LSTM

$$\begin{aligned} &f_t = \sigma(W_f u_t + U_f h_{t-1} + b_f), \quad i_t = \sigma(W_i u_t + U_i h_{t-1} + b_i),\\ &o_t = \sigma(W_o u_t + U_o h_{t-1} + b_o), \quad \tilde{c}_t = \tanh(W_c u_t + U_c h_{t-1} + b_c),\\ &c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \qquad h_t = o_t \odot \tanh(c_t), \end{aligned} \tag{160}$$

with $\|u_t\|_\infty \le 1$, $\sigma(\cdot) \in (0,1)$, and $\tanh(\cdot) \in (-1,1)$ (Lipschitz constant 1).

Define gate bounds (by monotonicity of σ and induced ∞-norms):

$$\bar{\sigma}_f := \sigma\big(\|[W_f\ U_f\ b_f]\|_\infty\big), \quad \bar{\sigma}_i := \sigma\big(\|[W_i\ U_i\ b_i]\|_\infty\big), \quad \bar{\sigma}_o := \sigma\big(\|[W_o\ U_o\ b_o]\|_\infty\big). \tag{161}$$

We choose a Lyapunov candidate *different from the GRU one*:

$$V_t \;:=\; \|c_t\|_\infty + \|h_t\|_\infty. \tag{162}$$

Step 1. Candidate and cell bounds. Using $|\tanh(z)| \le |z|$ and submultiplicativity,

$$\|\tilde{c}_t\|_\infty \;\le\; \|W_c\|_\infty\|u_t\|_\infty + \|U_c\|_\infty\|h_{t-1}\|_\infty + \|b_c\|_\infty. \tag{163}$$

Hence the cell recursion yields

$$\|c_t\|_\infty \;\le\; \bar{\sigma}_f\|c_{t-1}\|_\infty + \bar{\sigma}_i\Big(\|W_c\|_\infty\|u_t\|_\infty + \|U_c\|_\infty\|h_{t-1}\|_\infty + \|b_c\|_\infty\Big). \tag{164}$$

Step 2. Output bound and composite Lyapunov inequality. For the output,

$$\|h_t\|_\infty \;=\; \|o_t \odot \tanh(c_t)\|_\infty \;\le\; \bar{\sigma}_o\|c_t\|_\infty. \tag{165}$$

Thus $V_t \le (1+\bar{\sigma}_o)\|c_t\|_\infty$, and combining with (164) gives

$$V_t \le (1+\bar{\sigma}_o)\,\bar{\sigma}_f\,\|c_{t-1}\|_\infty + (1+\bar{\sigma}_o)\,\bar{\sigma}_i\Big(\|W_c\|_\infty\|u_t\|_\infty + \|U_c\|_\infty\|h_{t-1}\|_\infty + \|b_c\|_\infty\Big) \tag{166}$$

$$\le (1+\bar{\sigma}_o)\,\bar{\sigma}_f\,\|c_{t-1}\|_\infty + (1+\bar{\sigma}_o)\,\bar{\sigma}_i\,\|U_c\|_\infty\,\|h_{t-1}\|_\infty + (1+\bar{\sigma}_o)\,\bar{\sigma}_i\Big(\|W_c\|_\infty\|u_t\|_\infty + \|b_c\|_\infty\Big). \tag{167}$$

Using $ax+by \le \max\{a,b\}(x+y)$ for $a,b,x,y \ge 0$ and the fact that $\|c_{t-1}\|_\infty \le V_{t-1}$, $\|h_{t-1}\|_\infty \le V_{t-1}$, we obtain

$$V_t \;\le\; \alpha\, V_{t-1} \;+\; \underbrace{(1+\bar{\sigma}_o)\,\bar{\sigma}_i\,\|W_c\|_\infty}_{=:K_u}\,\|u_t\|_\infty \;+\; \underbrace{(1+\bar{\sigma}_o)\,\bar{\sigma}_i}_{=:K_b}\,\|b_c\|_\infty, \tag{168}$$

with

$$\alpha \;:=\; (1+\bar{\sigma}_o)\,\max\{\bar{\sigma}_f,\ \bar{\sigma}_i\,\|U_c\|_\infty\}. \tag{169}$$

Step 3. Contraction and ISS. Under the stated bounds

$$(1+\bar{\sigma}_o)\,\bar{\sigma}_f < 1 \qquad \text{and} \qquad (1+\bar{\sigma}_o)\,\bar{\sigma}_i\,\|U_c\|_\infty < 1, \tag{170}$$

we have $\alpha < 1$. Writing $\delta := 1-\alpha \in (0,1)$, the one-step Lyapunov inequality (168) becomes

$$V_t \;\le\; (1-\delta)\,V_{t-1} \;+\; K_u\|u_t\|_\infty \;+\; K_b\|b_c\|_\infty, \tag{171}$$

which unrolls to the ISS estimate

$$V_t \;\le\; (1-\delta)^t V_0 \;+\; \frac{K_u}{\delta}\,\|u\|_{\infty,1:t} \;+\; \frac{K_b}{\delta}\,\|b_c\|_\infty. \tag{172}$$

Since $\|h_t\|_\infty \le V_t$ and $\|c_t\|_\infty \le V_t$, this yields ISS for the LSTM in the $\|\cdot\|_\infty$ norm with

$$\beta(r,t) = (1-\delta)^t r, \qquad \gamma_u(s) = \frac{K_u}{\delta}\,s, \qquad \gamma_b(s) = \frac{K_b}{\delta}\,s. \tag{173}$$

□

Proof of ISS condition for LSTM via Lyapunov function. Let the state be $x_t := (c_t, h_t) \in \mathbb{R}^{2n_h}$ and define the Lyapunov candidate

$$V(x_t) \;:=\; \|c_t\|_\infty + \|h_t\|_\infty. \tag{174}$$

(i) Sandwich bounds ($\psi_1, \psi_2 \in \mathcal{K}_\infty$). For $s := \|(c,h)\|_\infty$,

$$s \;\le\; V(c,h) \;\le\; 2s. \tag{175}$$

Hence we may choose

$$\psi_1(s) = s, \qquad \psi_2(s) = 2s \qquad (\in \mathcal{K}_\infty). \tag{176}$$

(ii) One–step dissipation inequality. By $|\tanh(z)| \le |z|$ and induced norms,

$$\|\tilde{c}_t\|_\infty \le \|W_c\|_\infty\|u_t\|_\infty + \|U_c\|_\infty\|h_{t-1}\|_\infty + \|b_c\|_\infty. \tag{177}$$

Thus

$$\|c_t\|_\infty \le \bar{\sigma}_f\|c_{t-1}\|_\infty + \bar{\sigma}_i\big(\|W_c\|_\infty\|u_t\|_\infty + \|U_c\|_\infty\|h_{t-1}\|_\infty + \|b_c\|_\infty\big). \tag{178}$$

Moreover $\|h_t\|_\infty = \|o_t \odot \tanh(c_t)\|_\infty \le \bar{\sigma}_o\|c_t\|_\infty$, so

$$V(x_t) = \|c_t\|_\infty + \|h_t\|_\infty \le (1+\bar{\sigma}_o)\|c_t\|_\infty \tag{179}$$

$$\le (1+\bar{\sigma}_o)\bar{\sigma}_f\,\|c_{t-1}\|_\infty + (1+\bar{\sigma}_o)\bar{\sigma}_i\,\|U_c\|_\infty\,\|h_{t-1}\|_\infty \tag{180}$$

$$+ (1+\bar{\sigma}_o)\bar{\sigma}_i\big(\|W_c\|_\infty\|u_t\|_\infty + \|b_c\|_\infty\big). \tag{181}$$

Since $\|c_{t-1}\|_\infty \le V(x_{t-1})$ and $\|h_{t-1}\|_\infty \le V(x_{t-1})$, we get

$$V(x_t) \;\le\; \alpha\,V(x_{t-1}) \;+\; K_u\|u_t\|_\infty \;+\; K_b\|b_c\|_\infty, \tag{182}$$

with

$$\alpha := (1+\bar{\sigma}_o)\max\{\bar{\sigma}_f,\ \bar{\sigma}_i\|U_c\|_\infty\}, \qquad K_u := (1+\bar{\sigma}_o)\bar{\sigma}_i\|W_c\|_\infty, \qquad K_b := (1+\bar{\sigma}_o)\bar{\sigma}_i. \tag{183}$$

By (79), $\alpha < 1$. Let $\delta := 1-\alpha \in (0,1)$.

(iii) Identification of ISS–Lyapunov data. Rewrite eq.eq:lstm-one-step as

$$V(x_t) - V(x_{t-1}) \;\le\; -\delta\,V(x_{t-1}) \;+\; K_u\|u_t\|_\infty \;+\; K_b\|b_c\|_\infty. \tag{184}$$

This is the discrete-time ISS–Lyapunov inequality in the form of

$$V(f(x,u)) - V(x) \le -\psi(\|x\|) + \sigma_u(\|u\|) + \sigma_b(\|b_c\|),$$

with the choices

$$\psi(s) = \delta s, \qquad \sigma_u(s) = K_u s, \qquad \sigma_b(s) = K_b s, \tag{185}$$

all belonging to $\mathcal{K}_\infty$.

(iv) The general form of Lyapunov function. The aggregated input magnitude be $\|v_t\| := \max\{\|u_t\|_\infty, \|b_c\|_\infty\}$. Then

$$V(x_t) - V(x_{t-1}) \;\le\; -\delta\, V(x_{t-1}) + (K_u + K_b)\, \|v_t\|. \tag{186}$$

Hence, with

$$\chi(s) := \frac{2(K_u + K_b)}{\delta}\, s \in \mathcal{K}, \qquad \alpha(s) := \frac{\delta}{2}\, s \in \mathcal{K}, \tag{187}$$

the implication

$$V(x_{t-1}) \;\ge\; \chi(\|v_t\|) \quad \Rightarrow \quad V(x_t) - V(x_{t-1}) \;\le\; -\alpha(V(x_{t-1}))$$

holds. Together with the sandwich bounds (ψ_1, ψ_2) this verifies the ISS–Lyapunov general function. □

C.9 Projection to Enforce GSRNN ISS Condition

To ensure the Input-to-State Stability (ISS) condition in Theorem 2, we require

$$L_\phi \|U\|_\infty \hat{\psi} < 1. \tag{188}$$

Introducing a small buffer $\delta > 0$, we enforce the stricter constraint

$$\|U\|_\infty \;\le\; \frac{1-\delta}{L_\phi \hat{\psi}}. \tag{189}$$

Let $\hat{U} \in \mathbb{R}^{n_h \times n_h}$ be the unconstrained matrix obtained after a gradient update. We apply a projection step to obtain U that satisfies the ISS constraint without altering other parameters W, b.

Projection Problem. For each row vector $\hat{U}_i \in \mathbb{R}^{n_h}$, solve

$$U_i = \arg\min_{u \in \mathbb{R}^{n_h}} \|u - \hat{U}_i\|_2^2 \quad \text{s.t.} \quad \|u\|_1 \le \rho, \tag{190}$$

where

$$\rho := \frac{1-\delta}{L_\phi \hat{\psi}}.$$

This corresponds to projecting onto the ℓ_1-ball

$$\mathcal{B}_1(\rho) = \{u \in \mathbb{R}^{n_h} : \|u\|_1 \le \rho\}. \tag{191}$$

Efficient Projection Algorithm. We adopt the method of Duchi et al. (2008):

1. Let $v = \hat{U}_i$ and define $\rho = \frac{1-\delta}{L_\phi \hat{\psi}}$.
2. Sort $|v|$ in descending order: $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_d$.
3. Find the smallest k such that
$$\mu_k - \frac{1}{k}\left(\sum_{j=1}^{k} \mu_j - \rho\right) > 0. \tag{192}$$
4. Set the threshold
$$\tau = \frac{1}{k}\left(\sum_{j=1}^{k} \mu_j - \rho\right). \tag{193}$$

5. Compute projection

$$U_i = \text{sign}(\hat{U}_i) \odot \max(|\hat{U}_i| - \tau, 0). \tag{194}$$

Result. The resulting U satisfies

$$L_\phi \|U\|_\infty \hat{\psi} \leq 1 - \delta < 1,$$

while minimizing the deviation from the original update $\|\hat{U}\|_F$: $\|U - \hat{U}\|_F^2$.

Remark 3 (GRU specialization)**.** For GRU cells, the general projection scheme reduces to enforcing $\|U_r\|_\infty \hat{\sigma}_f < 1$ with buffer $\delta > 0$ and $\hat{\sigma}_f = \sigma(\|\mathbf{W}_f; \mathbf{U}_f; \mathbf{b}_f\|_\infty)$, i.e. row-wise projection onto the ℓ_1-ball of radius $\rho = (1 - \delta\hat{\sigma}_f)/\hat{\sigma}_f$. This matches the procedure described in Appendix C.9.

Remark 4 (LSTM specialization)**.** Let $\bar{\sigma}_g = \sigma(\|[W_g \; U_g \; b_g]\|_\infty)$ for $g \in \{f, i, o\}$ and $\text{logit}(t) = \log\left(\frac{t}{1-t}\right)$. To enforce the LSTM ISS bounds

$$(1 + \bar{\sigma}_o)\, \bar{\sigma}_f < 1, \qquad (1 + \bar{\sigma}_o)\, \bar{\sigma}_i\, \|U_c\|_\infty < 1, \tag{195}$$

we impose small buffers $\delta_f, \delta_c > 0$ and project as follows:

(i) Forget–output gate constraint. Fix the current $\bar{\sigma}_o$ and enforce $(1 + \bar{\sigma}_o)\, \bar{\sigma}_f \leq 1 - \delta_f$ by projecting the *forget-gate block* $[W_f \; U_f \; b_f]$ onto the ℓ_∞-ball of radius

$$\alpha_f \;:=\; \text{logit}\left(\frac{1 - \delta_f}{1 + \bar{\sigma}_o}\right), \tag{196}$$

i.e., elementwise clipping to $[-\alpha_f, \alpha_f]$ so that $\bar{\sigma}_f \leq \frac{1-\delta_f}{1+\bar{\sigma}_o}$. (If this target makes $\alpha_f < 0$, also shrink the *output gate* by clipping $[W_o \; U_o \; b_o]$ to radius $\alpha_o := \text{logit}(\tau_o)$ with $\tau_o := \min\{\max\{(1 - \delta_f)/\bar{\sigma}_f - 1, 0\}, 1 - \varepsilon\}$ so that $(1 + \bar{\sigma}_o)\bar{\sigma}_f \leq 1 - \delta_f$ holds.)

(ii) Input–candidate constraint. With the (possibly updated) gates, enforce $(1 + \bar{\sigma}_o)\, \bar{\sigma}_i\, \|U_c\|_\infty \leq 1 - \delta_c$ by *row-wise projection of* U_c onto the ℓ_1-ball of radius

$$\rho_c \;:=\; \frac{1 - \delta_c}{(1 + \bar{\sigma}_o)\, \bar{\sigma}_i},$$

using that $\|U_c\|_\infty = \max_i \|(U_c)_{i,:}\|_1$.

This two-step projection guarantees the buffered inequalities $(1 + \bar{\sigma}_o)\bar{\sigma}_f \leq 1 - \delta_f$ and $(1 + \bar{\sigma}_o)\bar{\sigma}_i\|U_c\|_\infty \leq 1 - \delta_c$, hence the ISS constraints for LSTM. It mirrors the GRU procedure (gate clipping for ℓ_∞ bounds; row-wise ℓ_1 projection for the recurrent matrix), cf. Appendix C.9.

D Related works, empirical running times and complexity analysis

D.1 Qualitative Comparison of the π-SSM to Recent Related Work

Table 4: Learning parameters in LG is marked by $\times/\checkmark$ because standard LGs (e.g., UKF, EKF) cannot learn parameters. In our setup, a data-driven network estimates $(\mathbf{A}, \mathbf{C})$ to make LGs comparable with π-SSM. See Appendix G.1.

Model	Learn Dyn.	Imputation	State Est.	Uncertainty	Noise Handling
LSTM Hochreiter and Schmidhuber (1997)	✓	✓	✓	×	✓
GRU Cho et al. (2014)	✓	✓	✓	×	✓
P2T Wahlström et al. (2015)	✓	×	✓	×	×
E2C Watter et al. (2015)	✓	×	×	✓	×
BB-VI Archer et al. (2015)	✓	✓	×	✓	✓
SIN Krishnan et al. (2017)	✓	✓	×	✓	✓
DVBF Karl et al. (2016)	✓	✓	×	✓	✓
VSMC Naesseth et al. (2018)	✓	✓	×	✓	✓
DSA Li and Mandt (2018)	✓	×	×	✓	×
KVAE Fraccaro et al. (2017)	✓	✓	✓	✓	✓
EKVAE Klushyn et al. (2021)	✓	✓	✓	✓	✓
MVAE Zhu et al. (2023)	✓	✓	✓	✓	✓
rSLDS Linderman et al. (2017)	×	×	✓	✓	×
irSLDS Linderman et al. (2017)	×	×	✓	✓	×
NODE Chen et al. (2018)	✓	×	✓	✓	×
MoNODE Auzina et al. (2024)	✓	×	✓	✓	×
DeepAR Salinas et al. (2020)	✓	×	✓	✓	×
DSSM Rangapuram et al. (2018)	✓	×	✓	✓	×
HybridGNN Garcia Satorras et al. (2019)	×	✓	✓	×	✓
KalmanNet Revach et al. (2021)	×	✓	✓	×	✓
SSI Ruhe and Forré (2021)	×	✓	✓	✓	✓
LG	×/✓	✓	✓	✓	✓
π-SSM	✓	✓	✓	✓	✓

SLDSs. Switching Linear Dynamical Systems (SLDS) decompose complex, nonlinear time series data into sequences of simpler, reusable dynamical modes. Fitting an SLDS to data enables learning flexible nonlinear generative models and parsing sequences into coherent discrete segments. The model proposed by Linderman et al. (2017) introduces auxiliary latent variables to switch among different linear dynamics. However, it relies on Gibbs sampling for parameter inference, which makes it impractical for large-scale datasets due to scalability limitations. Recurrent SLDS (rSLDS) Linderman et al. (2016) and infinite rSLDS (irSLDS) Geadah et al. (2024) extend the SLDS framework but still inherit many of these computational limitations. ReLiNet Baier et al. (2023) introduces a recurrent linear parameter-varying network that approximates RNN dynamics with switched linear systems to ensure exponential stability and explainable multistep predictions for dynamical systems.

Auto-Regressive State Space Models. Auto-regressive state space models (ARSSMs) are widely used in time series analysis and forecasting. These models describe the evolution of a system over time through a state representation informed by past observations. Auto-Regressive Hidden Markov Models (AR-HMMs) model temporal dependencies by mapping previous observations to the current one. For instance, Salinas et al. (2020) proposes an AR-HMM where target values are used directly as inputs. However, this dependence on ground-truth targets during training increases vulnerability to noise.

System Identification. In the domain of state-space model learning (system identification), several works Wang et al. (2007); Ko and Fox (2011); Frigola et al. (2013); Schoukens and Tiels (2017); Li et al. (2023) have developed algorithms for training Gaussian Process SSMs (GPSSMs) via maximum likelihood estimation through the Expectation-Maximization (EM) algorithm. The E-step involves filtering and smoothing using a fixed parameter set $\boldsymbol{\gamma}$, followed by the M-step that updates $\boldsymbol{\gamma}$ to maximize the resulting likelihood. For example, Frigola et al. Frigola et al. (2013) draw sample trajectories from the smoothing distribution and condition the M-step on these samples. In this context, the π-SSM bears resemblance to Hammerstein-Wiener (HW) models Gilabert et al. (2005), as it learns system parameters directly from observations while applying nonlinear mappings to the inputs and outputs. Compared to the GIN framework (Hashempoorikderi and Choi, 2024), another Markovian

framework for system identification, our π-SSM departs in several key aspects: it explicitly models discrete latent dynamics rather than relying on purely Markovian assumptions; it does not require auxiliary losses to prevent mode collapse; and it is built as a local message-passing scheme that enables scalable posterior inference over hybrid discrete–continuous states while also yielding theoretical gradient expressions. Moreover, whereas GIN's stability analysis is restricted to GRUs with zero input, we establish a general proof strategy applicable to gated RNNs under bounded, nonzero inputs.

Neural ODEs. Since the introduction of Neural ODEs Chen et al. (2018), continuous-time dynamic modeling has garnered significant interest. Extensions include combining neural ODEs with recurrent architectures Rubanova et al. (2019), enabling latent trajectories to evolve in response to observations. Other works explore dynamics governed by Hamiltonian, Lagrangian, or second-order systems, as well as structured dynamics using graph neural networks. Our approach, MoNODE Auzina et al. (2024), though developed within the latent Neural ODE framework, is applicable to many of these continuous-time modeling paradigms.

Neural PDEs. Machine learning has shown promise in approximating solutions to partial differential equations (PDEs). Notably, Physics-Informed Neural Networks (PINNs) Raissi et al. (2019) use deep learning and gradient-based optimization to solve PDEs without requiring mesh discretization, which is commonly needed in classical methods. This formulation allows simultaneous treatment of forward and inverse problems in a unified optimization framework. Leveraging automatic differentiation and modern computing power, PINNs have been successfully applied to a variety of complex PDE systems Raissi et al. (2019); Cho et al. (2024).

Table 5: Empirical running times and parameters of experiments.

Cell	Pong		Lorenz Attractor		Navier Stokes		NCLT	
	Param	T/E	Param	T/E	Param	T/E	Param	T/E
LSTM	∼18k	∼56s	∼18k	∼56s	∼18k	∼53s	∼18k	∼83s
GRU	∼18k	∼61s	∼18k	∼62s	∼18k	∼59s	∼18	∼79s
VAE	∼12k	∼50s	∼13k	∼52s	∼12k	∼49s	∼12k	∼70s
IWVAE	∼13k	∼50s	∼13k	∼54s	∼11k	∼47s	∼11k	∼75s
VAE-RNN	∼24k	∼59s	∼25k	∼61s	∼24k	∼57s	∼24k	∼89s
SVAE	∼27k	∼67s	∼27k	∼69s	∼26k	∼65s	∼27k	∼149s
KVAE	∼25k	∼95s	∼25k	∼97s	∼25k	∼94s	∼25k	∼141s
EKVAE	∼26k	∼98s	∼26k	∼99s	∼26k	∼94s	∼26k	∼145s
MKVAE	∼34k	∼112s	∼34k	∼110s	∼33k	∼105s	∼33k	∼153s
RKN	∼25k	∼57s	∼25k	∼58s	∼25k	∼56s	∼24k	∼79s
CRU	∼24k	∼55s	∼24k	∼55s	∼23k	∼54s	∼23k	∼78s
LG	∼12k	∼82s	∼12k	∼84s	∼12k	∼80s	∼12k	∼117s
π-SSM$_{\text{GRU}}$	∼18k	∼56s	∼18k	∼55s	∼18k	∼53s	∼18k	∼81s
π-SSM$_{\text{LSTM}}$	∼18k	∼57s	∼18k	∼54s	∼18k	∼56s	∼18k	∼78s

Variational Inference Approaches Variational inference (VI) has become a dominant paradigm for approximate Bayesian inference in latent variable models. Early approaches such as Variational Autoencoders (VAEs) Kingma and Welling (2013), Embed-to-Control (E2C) Watter et al. (2015), and Importance Weighted VAEs (IWVAEs) Burda et al. (2015) integrated deep learning with variational objectives. However, these methods generally lacked recurrent structures or memory, limiting their effectiveness for sequential reasoning and imputation tasks. To address these limitations, EM-inspired variational models such as Structure VAE (SVAE) Johnson et al. (2016), Kalman VAE (KVAE) Fraccaro et al. (2017), Disentangled VAE (DVAE) Li and Mandt (2018), Extended KVAE (EKVAE) Klushyn et al. (2021), Robust VAE Tonolini et al. (2023), and Markovian VAE (MVAE) Zhu et al. (2023) have embedded classical filtering and smoothing updates into deep latent variable frameworks. These approaches attempt to capture temporal dependencies more explicitly, but typically cannot directly optimize latent trajectories—a limitation noted in methods such as Recurrent Kalman Networks (RKN) Becker et al. (2019) and Continuous Recurrent Units (CRU) Schirmer et al. (2022). In contrast, memory-based architectures like LSTMs Hochreiter and Schmidhuber (1997), GRUs Cho et al. (2014), and RNNs Wilson and Finkel (2009) offer strong modeling capacity for latent dynamics but often lack principled mechanisms for uncertainty estimation and adaptation to dynamic mode transitions.

Summary Comparison. In Table 4, which is built upon Becker et al. (2019), we compare the above methods in

terms of their capabilities for handling high-dimensional observations, learning underlying dynamics, providing accurate state estimates and uncertainty quantification, dealing with noisy and missing data. Classical LGs, such as the EKF and UKF, linearize the transition and observation functions and apply Bayesian filtering on the resulting linearized systems—representing a model-based paradigm. In contrast, the π-SSM adopts a data-driven approach, leveraging learnable networks to approximate these components, thereby enabling greater flexibility and scalability.

D.2 Expressivity, Extension and Limitations

Structural difference. Assuming both SLDS and π-SSM employ equally expressive parameterizations for dynamics and emissions, the key structural distinction is the explicit Markov prior over discrete modes in SLDSs, i.e., the edge $z_t \to z_{t+1}$, which encodes mode persistence. Our graphical model removes this edge and instead infers the mode at each step from the continuous state x_t (via $x_t \to z_{t+1}$), using x_t as a sufficient signal for regime identification.

Expressivity intuition. The $z_t \to z_{t+1}$ edge in SLDS explicitly smooths mode trajectories and captures persistence in the discrete latent space. By contrast, our model relies on the continuous dynamics to reflect regime switches: when the underlying system changes mode, the induced change in the continuous transition (e.g., the local linearization A_t) is intended to be promptly visible in x_t, allowing z_{t+1} to be inferred without an explicit z-chain.

Illustrative evidence (Pong). In a simple Pong experiment, we monitored the spectrum of the learned transition matrices as the ball bounced (which changes the interaction regime). We observed sharp shifts in the eigenvalues of A_t at bounce events, indicating that x_t responds strongly to mode changes and can implicitly carry mode information—without requiring an explicit $z_t \to z_{t+1}$ prior. This evidence is qualitative, but it supports the hypothesis that x_t can act as an effective carrier of regime cues in practice.

Implications for scalability. This structural simplification enables the local message-passing inference and per-step gradient updates used in our method (cf. Theorem 6). Introducing a $z_t \to z_{t+1}$ edge would couple the discrete variables temporally, breaking the factorization that our updates exploit and necessitating forward–backward (or loopy) message schedules, with higher computational and memory cost per sequence.

Limitations and potential extensions. Our design implicitly assumes that regime changes manifest quickly in the continuous state (i.e., are promptly reflected in x_t). This is often reasonable in control and physical simulation, where dynamics shift sharply with mode changes. However, in systems with delayed or inertial effects, the absence of temporal dependence between discrete states (no $z_t \to z_{t+1}$) can hinder accurate regime smoothing. While the current model performs well empirically, richer temporal couplings could improve expressivity—for example, modeling joint posteriors over (x_t, x_{t+1}, z_t) or adding a tempered/sticky transition prior for z. Such extensions would require revisiting the factor graph and redefining messages and gradient paths, trading scalability for additional temporal structure.

D.3 Empirical analysis

We present the number of parameters for the utilized cell structures in our experiments and their corresponding empirical running times for 1 epoch in Table 5. In the first row of each model structure, we set the number of parameters approximately equal to our π-SSM to demonstrate the π-SSM's superior performance with the same parameter count. The extra running time of EM-variational approaches, like KVAE, is due to the use of classic Bayesian equations, which significantly increase running time for higher-dimensional observations. However, the π-SSM avoids this issue. The number of parameters in the π-SSM is noticeably lower than in other memory cells, such as LSTM and GRU, and EM-variational methods. This efficiency is achieved by converting high-dimensional sparse covariance matrices into lower-dimensional covariance matrices using a convolutional operator.

E Algorithms and python intuitive code

E.1 Algorithms

Algorithm Inference in π-SSM via Nested Message Passing

Input: Observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, initial posterior $q(\mathbf{x}_0) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{0|0}, \hat{\boldsymbol{\Sigma}}_{0|0})$
Output: Filtered posteriors $q(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, $q(z_t \mid \mathbf{y}_{1:t})$ for $t = 1, \ldots, T$
for $t = 1$ to T **do**
  Sample $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$
  Sample $z_t \sim q(z_t \mid \mathbf{x}_{t-1})$, forming an approximation of $p(z_t \mid \mathbf{y}_{1:t})$ *(eq.6)*
  Predict $\hat{\boldsymbol{\mu}}_{t|t-1}$, $\hat{\boldsymbol{\Sigma}}_{t|t-1}$ using transition $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, z_t)$
  Compute RNN-based gain factor: $\mathbf{L}_t = \text{RNN}([\hat{\boldsymbol{\Sigma}}_{t|t-1}, \mathbf{r}_t])$
  Compute gain matrix: $\hat{\mathbf{K}}_t = \hat{\boldsymbol{\Sigma}}_{t|t-1}\mathbf{C}_{z_t}^\top \mathbf{L}_t \mathbf{L}_t^\top$
  Compute updated posterior mean and covariance via Kalman-style rule: *(eq.7)*

$$\hat{\boldsymbol{\mu}}_{t|t} = \hat{\boldsymbol{\mu}}_{t|t-1} + \hat{\mathbf{K}}_t(\mathbf{y}_t - \mathbf{C}_{z_t}\hat{\boldsymbol{\mu}}_{t|t-1})$$

$$\hat{\boldsymbol{\Sigma}}_{t|t} = \hat{\boldsymbol{\Sigma}}_{t|t-1} + \hat{\mathbf{K}}_t(\mathbf{C}_{z_t}\hat{\boldsymbol{\Sigma}}_{t|t-1}\mathbf{C}_{z_t}^\top + \mathbf{R}_t)\hat{\mathbf{K}}_t^\top$$

  Form $q(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{t|t}, \hat{\boldsymbol{\Sigma}}_{t|t})$ *(cf. eq.5)*
end for

Algorithm Training π-SSM via Surrogate Predictive Log-Likelihood

Input: Dataset $\{\mathbf{y}_{1:T}^{(n)}\}_{n=1}^N$, initial model parameters (θ, ϕ)
for each training iteration **do**
  for each sequence $\mathbf{y}_{1:T}$ in the batch **do**
    for $t = 1$ **to** T **do**
      Sample $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ *(eq.5)*
      Sample $z_t \sim q(z_t \mid \mathbf{x}_{t-1})$ *(approx. $p(z_t \mid \mathbf{y}_{1:t})$ via eq.6)*
      Predict $\hat{\boldsymbol{\mu}}_{t|t}$, $\hat{\boldsymbol{\Sigma}}_{t|t}$ via Kalman-style update *(eq.7)*
      Compute predictive likelihood $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ *(eq of $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$)*
    end for
    Accumulate surrogate objective: $\mathcal{L} = \sum_{t=1}^T \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$
    Estimate gradients w.r.t. continuous parameters ϕ *(eq.8)*
    Estimate gradients w.r.t. discrete parameters θ using REINFORCE *(eq.9)*
  end for
  Update parameters (θ, ϕ) using gradient-based optimizer
  Project RNN weights to satisfy ISS condition *(Theorem 2)*
end for

E.2 Python intuitive code

To demonstrate the simplicity of our proposed π-SSM, we include intuitive inference and training code with Tensorflow library. The code runs with Python 3.7+. The entire code to reproduce the experiments are available in Github repository.

Python intuitive code for inference.

```
# Inference loop for PiSSM

import tensorflow.keras as k
import PredictiveStep
import KalmanUpdate
import get_mode_params
```

```
class PiSSMCell(k.layers.Layer):
    def __init__(self, initial_states):
        super().__init__()
        self.mu_tm1, self.Sigma_tm1 = initial_states
        self.filtered_states = []

    def call(self, y_seq, R_seq):
        for t, (y_t, R_t, Q_t) in enumerate(zip(y_seq, R_seq)):
            # (eq. 6) From q(x_{t-1} | y_{1:t-1})
            x_tm1 = sample(self.mu_tm1, self.Sigma_tm1)

            # (eq. 7) From q(z_t | x_{t-1}) modeled by NN
            z_t = q_z_given_x(x_tm1)

            # A_{z_t}, C_{z_t}
            A_z_t, C_z_t = get_mode_params(z_t)

            mu_t_pred, Sigma_t_pred = PredictiveStep(A_z_t, x_tm1, Q_t)

            # (eq. 8) Form q(x_{t} | y_{1:t})
            mu_t_filt, Sigma_t_filt = KalmanUpdate(mu_t_pred, Sigma_t_pred, y_t, R_t,
    C_z_t)

            self.filtered_states.append((mu_t_filt, Sigma_t_filt, z_t))
            self.mu_tm1, self.Sigma_tm1 = mu_t_filt, Sigma_t_filt
        return self.filtered_states


class PiSSM(k.models.Model):
    def __init__(self, initial_states):
        super().__init__()
        self.cell = PiSSMCell(initial_states)

    def call(self, y_seq, R_seq, Q_seq):
        filtered_seq = self.cell(y_seq, R_seq, Q_seq)
        mu_seq, Sigma_seq, z_seq = zip(*filtered_seq)
        x_seq = sample(mu_seq, Sigma_seq)
        return x_seq, z_seq
```

Python intuitive code for training.

```
# Training loop for PiSSM

import tensorflow as tf
import get_current_U_r_weights
import get_params_q_z

def project_onto_l1_ball(v, radius):
    """Projects vector v onto the l1-ball of specified radius."""
    abs_v = tf.abs(v)
    if tf.reduce_sum(abs_v) <= radius:
        return v
    sorted_v = tf.sort(abs_v, direction='DESCENDING')
    cumsum = tf.cumsum(sorted_v)
    rho = tf.where(sorted_v - (cumsum - radius) / (tf.range(1, len(v)+1, dtype=v.dtype
    )) > 0)
    if len(rho) == 0:
        return tf.zeros_like(v)
    rho = rho[-1][0] + 1  # adjust for indexing
    tau = (tf.reduce_sum(sorted_v[:rho]) - radius) / tf.cast(rho, v.dtype)
    return tf.sign(v) * tf.maximum(abs_v - tau, 0.0)

class PiSSMTrainer:
    def __init__(self, model: PiSSM, optimizer_phi, optimizer_theta):
        self.model = model
        # Continuous (GRU, Kalman flow, Mode vars A_z_t, C_z_t, etc.)
```

```
        self.opt_phi = optimizer_phi

        # Discrete (NN for q(z | x))
        self.opt_theta = optimizer_theta

        # Moving average baseline for REINFORCE
        self.baseline = 0.0

    def compute_log_likelihood(self, y_t, mu_t, Sigma_t, C_z_t):
        # log p(y_t | x_t)
        return log_prob_gaussian(y_t, mu_t, Sigma_t, C_z_t)

    def train_step(self, y_seq, R_seq, Q_seq):
        log_likelihoods = []

        with tf.GradientTape(persistent=True) as tape:
            x_seq, z_seq = self.model(y_seq, R_seq, Q_seq)

            for t, (x_t, z_t, y_t, R_t) in enumerate(zip(x_seq, z_seq, y_seq, R_seq)):
                mu_t, Sigma_t = mean_cov_from_x(x_t)
                A_z_t, C_z_t = get_mode_params(z_t)

                # (eq. 9) log predictive
                log_py_t = self.compute_log_likelihood(y_t, mu_t, Sigma_t, C_z_t)
                log_likelihoods.append(log_py_t)

                # (eq. 12) REINFORCE gradient
                log_q_z = log_prob_q_z_given_x(z_t, x_seq[t-1])
                reinforce_term =  tf.stop_gradient(log_py_t - self.baseline) * log_q_z

        # (eq. 11) Continuous update: backprop GRU, A_z_t, C_z_t, etc.
        loss_phi = -tf.reduce_sum(log_likelihoods)
        grads_phi = tape.gradient(loss_phi, self.model.trainable_variables)
        self.opt_phi.apply_gradients(zip(grads_phi, self.model.trainable_variables))

        # Discrete update: REINFORCE
        loss_theta = -tf.reduce_sum(reinforce_term)
        grads_theta = tape.gradient(loss_theta, get_params_q_z())
        self.opt_theta.apply_gradients(zip(grads_theta, get_params_q_z()))

        # Update moving baseline
        current_ll = tf.reduce_mean(log_likelihoods)
        self.baseline = 0.95 * self.baseline + 0.05 * current_ll

        # Stability Projection Steps for two cases of RNNs (GRU or LSTM)
        # Choose small buffers
        delta_f = 1e-3   # (1 + sigma_o) sigma_f < 1 - delta_f
        delta_c = 1e-3   # (1 + sigma_o) sigma_i |U_c|_inf < 1 - delta_c
        eps = 1e-6

        if self.model.rnn_type.upper() == 'GRU':
            U_r = get_current_U_r_weights()  # shape: [n_h, n_h]
            U_r_projected = []
            radius = 1. / hat_sigma_f - delta
            for row in U_r:
                row_proj = project_onto_l1_ball(row, radius)
                U_r_projected.append(row_proj)
            U_r_projected = tf.stack(U_r_projected, axis=0)
            set_projected_U_r(U_r_projected)

        elif self.model.rnn_type.upper() == 'LSTM':

            W_f, U_f, b_f = get_lstm_gate_params('f')
            W_i, U_i, b_i = get_lstm_gate_params('i')
            W_o, U_o, b_o = get_lstm_gate_params('o')
```

```
            #FIRST TERM: (1 + sigma_o) sigma_f < 1 - delta_f
            bar_o = bar_sigma_from_block(W_o, U_o, b_o)
            target_f = tf.minimum((1.0 - delta_f) / (1.0 + bar_o + eps),
               1.0 - eps)
            alpha_f  = logit(target_f, eps=eps)  # clip radius
            W_f, U_f, b_f = clip_gate_block(W_f, U_f, b_f, alpha_f)
            set_lstm_gate_params('f', W_f, U_f, b_f)

            #SECOND TERM:(1 + sigma_o)sigma_i |U_c|_inf < 1 - delta_c
            bar_i = bar_sigma_from_block(W_i, U_i, b_i)
            rho_c = (1.0 - delta_c) / ((1.0 + bar_o) * bar_i + eps)
            U_c = get_current_U_c_weights()  # [n_h, n_h]
            U_c_projected = tf.stack(
                [project_onto_l1_ball(row, rho_c) for row in
    tf.unstack(U_c, axis=0)],
                 axis=0
                                      )
            set_projected_U_c(U_c_projected)

        return current_ll.numpy()
```

F Hyperparameters and architecture

F.1 Hyperparameters and Training Details

All experiments were conducted using the Adam optimizer Kingma and Ba (2014) on an NVIDIA GeForce GTX 1050 Ti with 16GB RAM. We started by evaluating each dataset using multiple random seeds. For each seed, we conducted a grid search over the learning rate (LR) range specified in the appendix—50 values from 0.001 to 0.2 in increments of 0.002, each followed by exponential decay every 10 epochs. This process produced a performance table per dataset. We then averaged validation performance across seeds and selected the LR with the best average score. For this, we followed the recipe from optimization literature, which suggests that hyperparameters varying across random initializations should be selected based on validation performance averaged across runs. The best learning rates selected per dataset are shown below:

Dataset	Pong	Lorenz	Navier	NCLT
LR	0.011	0.011	0.011	0.007

To provide a consistent learning rate for the overall model, we set the final LR to **0.011** for all experiments.

In order to reduce the variance of the REINFORCE gradient estimator in eq. (9), we use an exponential moving average baseline for the control variate b_t. At each training step, this baseline is updated as:

$$b_t \leftarrow 0.95 \cdot b_t + 0.05 \cdot \log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}),$$

where $\log q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ is the current predictive log-likelihood. This formulation ensures that b_t tracks the typical scale of the learning signal over time while remaining stable, and helps to decorrelate the stochastic gradient direction from the reward signal, thereby reducing the variance of updates without introducing bias.

Backpropagation through time (BPTT) (Werbos, 1990) was used to compute gradients through the RNN cells. Gradient updates were applied with the stability constraint described in Theorem 2, using the projection method detailed in the main paper. We applied layer normalization (Ba et al., 2016) to stabilize the dynamics and normalize the filter responses.

To avoid poor local optima—e.g., the model overly focusing on prediction rather than learning latent dynamics—we adopted two additional training strategies:

1. We generated time-correlated noisy sequences as inputs. This forces the model to capture temporal dependencies and discourages reliance on pointwise prediction.
2. During early epochs, only the globally-shared parameters (e.g., $\mathbf{A}_{z_k}$ and $\mathbf{C}_{z_k}$) were optimized, while the parameters of the inference model $q(z_t|\mathbf{x}_{t-1})$ were frozen. After initial convergence, all parameters were

jointly optimized. This warm-up phase facilitates the learning of meaningful embeddings before introducing mode-specific modeling.

We set $K = 18$ latent modes to accommodate diverse dynamics in the latent space, with each mode representing a distinct dynamical regime. Notably, parameter tuning was not particularly sensitive: when the π-SSM has sufficient flexibility, unused modes can be effectively pruned by the learned distribution $q(z_t|\mathbf{x}_{t-1})$.

F.2 Proposed architecture

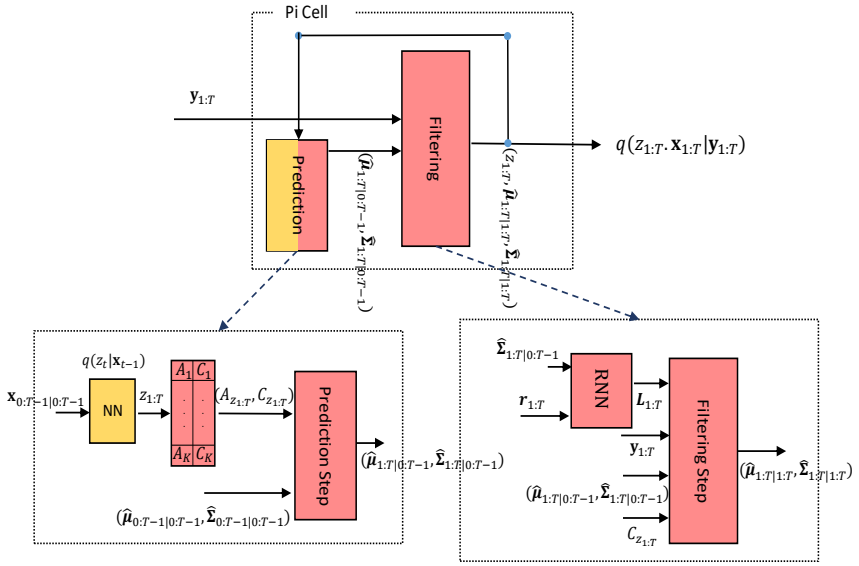


Figure 7: Proposed architecture diagram. Modules highlighted in red correspond to continuous-variable components in the parameter set ϕ, while yellow modules represent discrete-variable components in the set θ.

The proposed architecture is illustrated in Figure 7. To model $q(z_t \mid \mathbf{x}_{t-1})$, we use a multi-layer perceptron (MLP) with 10 hidden units and ReLU activation, followed by a softmax output layer producing K mode probabilities. The input to this network is the sampled latent state $\mathbf{x}_{t-1} \in \mathbb{R}^N$, where N denotes the state dimensionality.

In the state estimation tasks considered, the dimensionality N varies across experiments: 4 for Pong, 3 for Lorenz, 5 for Navier–Stokes, and 4 for NCLT.

G Experimental Systems and Formulations

G.1 LG Variant Used in Ablation Study

In our ablation studies, we include a variant of the Linear Gaussian State-Space Model (LG) to isolate the impact of discrete latent variables and neural parameterizations in the full π-SSM model. This LG configuration retains a Markovian latent structure with continuous latent states $\mathbf{x}_t \in \mathbb{R}^M$ and linear Gaussian transitions and emissions, but omits discrete latent variables z_t. That is, the generative model is defined by:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{q}_t, \quad \mathbf{q}_t \sim \mathcal{N}(0, \mathbf{Q}), \tag{197}$$

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{x}_t + \mathbf{r}_t, \quad \mathbf{r}_t \sim \mathcal{N}(0, \mathbf{R}), \tag{198}$$

where $\mathbf{A}_t \in \mathbb{R}^{M \times M}$ and $\mathbf{C}_t \in \mathbb{R}^{N \times M}$ are linear transition and emission matrices, respectively. The model applies standard Kalman filtering equations for inference, maintaining a fully analytical Gaussian belief state $p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \mathcal{N}(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})$ at each step.

To make the model expressive while retaining the linear structure, we parameterize the transition and emission matrices as convex combinations of K base matrices. Specifically, we define:

$$\mathbf{A}_t = \sum_{k=1}^{K} \alpha_t^{(k)} \mathbf{A}^{(k)}, \qquad \mathbf{C}_t = \sum_{k=1}^{K} \alpha_t^{(k)} \mathbf{C}^{(k)}, \tag{199}$$

where $\mathbf{A}^{(k)}$, $\mathbf{C}^{(k)}$ are trainable base matrices, and $\boldsymbol{\alpha}_t = \text{Softmax}(\text{MLP}(\mathbf{x}_{t-1}))$ produces mixture weights conditioned on the previous state $\mathbf{x}_{t-1}$. The use of soft attention over modes allows the system to adaptively interpolate between K locally linear dynamics, while maintaining the analytic filtering update structure of LG.

This hybrid approach allows us to compare π-SSM against a strong continuous-only baseline that leverages neural flexibility but does not involve discrete switching or nested inference. All parameters, including base matrices and the MLP for generating $\boldsymbol{\alpha}_t$, are trained end-to-end by maximizing the predictive log-likelihood of the observations.

G.2 Lorenz System Dynamics

The Lorenz system is a set of coupled nonlinear ordinary differential equations (ODEs) that describe the evolution of a particle in a chaotic 3D space. Originally derived for atmospheric convection, the system is now widely used as a benchmark for nonlinear dynamical systems. In our context, the system state at time t is denoted by $\mathbf{x}_t = [x_t, y_t, z_t]^\top$, where x_t, y_t, and z_t represent the particle's position coordinates in 3D space.

The system is governed by the following equations:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z, \tag{200}$$

where the standard parameters are set to $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$.

To generate trajectories, we numerically integrate the system eq.200 using a time step of $dt = 10^{-5}$, and then uniformly subsample with $\Delta t = 0.01$ to obtain discrete-time data for training and evaluation.

For inference and linearization purposes, we locally approximate the nonlinear dynamics at each time step using a time-varying transition matrix $\mathbf{A}_t$ such that:

$$\dot{\mathbf{x}}_t = \mathbf{A}_t \mathbf{x}_t, \quad \text{with} \quad \mathbf{A}_t = \begin{bmatrix} -10 & 10 & 0 \\ \rho - z & -1 & 0 \\ y & 0 & -\frac{8}{3} \end{bmatrix}. \tag{201}$$

To compute a discrete transition matrix $\mathbf{A}_t = \exp(\mathbf{A}_t \Delta t)$, we use the Taylor expansion of the matrix exponential truncated at degree $J = 5$:

$$\mathbf{A}_t = \exp(\mathbf{A}_t \Delta t) \approx \mathbf{I} + \sum_{j=1}^{J} \frac{(\mathbf{A}_t \Delta t)^j}{j!}, \tag{202}$$

where $\mathbf{I}$ is the identity matrix. This provides a first-order linear approximation of the nonlinear Lorenz system suitable for ground truth generation and use in filtering-based state estimation.

G.3 Movement Model Details for the NCLT Experiment

The NCLT dataset Carlevaris-Bianco et al. (2016) consists of sensor recordings from a Segway robot navigating a university campus environment. Robot localization in this context aims to estimate the true position of the robot over time using noisy GPS observations. To model the robot's multi-directional motion, we adopt a piece-wise constant velocity assumption where the continuous dynamics remain constant velocity (zero acceleration) within each heading direction, but discrete modes correspond to different heading directions (e.g., north, northwest, south, etc.). This structure naturally captures mode switches when the robot changes direction during navigation.

Under this assumption, the continuous-time dynamics are given by:

$$\frac{dp_1}{dt} = v_1, \quad \frac{dp_2}{dt} = v_2, \quad \frac{dv_1}{dt} = 0, \quad \frac{dv_2}{dt} = 0, \tag{203}$$

where p_1, p_2 represent the 2D position coordinates and v_1, v_2 their respective velocities. This yields the latent state and observation:

$$\mathbf{x}_t = [p_1, v_1, p_2, v_2]^T, \quad \mathbf{y}_t = [p_1, p_2]^T.$$

By discretizing the system with sampling interval $\Delta t = 1$ (1 Hz), the linear transition and observation models are given by:

$$\mathbf{A}_t = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{C}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{R}_t = \lambda^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{204}$$

Here, $\mathbf{A}_t$ is the state transition matrix, $\mathbf{C}_t$ is the emission matrix, and $\mathbf{R}_t$ models the observation noise with variance λ^2.

For training and evaluation, the session from January 22, 2012 is selected. After removing invalid GPS readings, the remaining 4280 time steps are partitioned into: a training set of 3600 steps (18 sequences of length $T = 200$), a validation set of 400 steps (2 sequences of length $T = 200$), and a test set of 280 steps (1 sequence of length $T = 280$).

G.4 Navier-Stokes System Setting

The incompressible Navier-Stokes equations govern the evolution of velocity fields $\mathbf{u} = [u, v] : \mathcal{X} \to \mathbb{R}^2$, where the spatial domain is $\mathcal{X} \subset \mathbb{R}^2$. The equations are expressed as:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \mu \nabla^2 \mathbf{u} - \nabla p + f, \quad \nabla \cdot \mathbf{u} = 0 \tag{205}$$

Here, μ is the kinematic viscosity, p is the scalar pressure field, and f denotes an external force — in our case, a buoyancy term. The constraint $\nabla \cdot \mathbf{u} = 0$ enforces incompressibility and ensures mass conservation. The term $-(\mathbf{u} \cdot \nabla)\mathbf{u}$ describes convection, capturing the self-advection of the velocity field. The diffusion term $\mu \nabla^2 \mathbf{u}$ models viscous dissipation, while ∇p represents internal pressure forces.

To generate ground truth, we solve the Navier-Stokes system in vorticity form on a unit torus using numerical methods. The solver produces time-resolved velocity and pressure fields that define the fluid's true dynamics.

We then simulate particle trajectories by randomly placing particles at initial spatial coordinates and integrating their motion through the time-evolving velocity field using a Newtonian update. At each time step, a particle state is recorded as $\mathbf{x}_t \in \mathbb{R}^5$, consisting of:

- Position: $(x_t, y_t) \in \mathbb{R}^2$
- Velocity: $(u_t, v_t) \in \mathbb{R}^2$
- Local pressure: $p_t \in \mathbb{R}$

To simulate realistic sensor observations, Gaussian noise is added to the trajectories. The resulting observations $\mathbf{y}_t \in \mathbb{R}^2$ are used for training and evaluation of the state-space inference models. Formally saying, we define the 5-dimensional latent state and 2-dimensional observations at time t as:

$$\mathbf{x}_t = \begin{bmatrix} u_t \\ v_t \\ x_t \\ y_t \\ p_t \end{bmatrix} \in \mathbb{R}^5, \quad \mathbf{y}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \in \mathbb{R}^2$$

The latent-state-dependent non-linear transition function $\mathbf{A}_{z_t}(\cdot)$ is derived from the discretized Navier-Stokes

equations and particle motion:

$$\mathbf{x}_{t+1} = \mathbf{A}_{z_t}(\mathbf{x}_t) = \begin{bmatrix} u_t + \Delta t \left[-(u_t\, \partial_x u_t + v_t\, \partial_y u_t) + \mu \nabla^2 u_t - \frac{1}{\rho}\, \partial_x p_t \right] \\ v_t + \Delta t \left[-(u_t\, \partial_x v_t + v_t\, \partial_y v_t) + \mu \nabla^2 v_t - \frac{1}{\rho}\, \partial_y p_t \right] \\ x_t + \Delta t \cdot u_t \\ y_t + \Delta t \cdot v_t \\ \text{PoissonSolve}\left(\frac{\rho}{\Delta t} (\partial_x u_t + \partial_y v_t) \right) \end{bmatrix} \tag{206}$$

The observation model is a linear projection of the state to the spatial coordinates:

$$\mathbf{y}_t = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{C}_{z_t}} \mathbf{x}_t \tag{207}$$

In the general formulation of π-SSM, the discrete latent variable $z_t \in \{1, \ldots, K\}$ represents a mode index that selects between different system dynamics. Each mode z_t is associated with its own transition and observation functions, denoted $\mathbf{A}_{z_t}(\cdot)$ and $\mathbf{C}_{z_t}$, allowing the model to capture complex behavior by switching between locally consistent dynamics. In the context of the Navier-Stokes experiment, the underlying physical law is uniform across the domain; however, introducing multiple modes (i.e., $K > 1$) enables the model to specialize on distinct flow regimes—such as boundary layers versus central flow, or convection-dominated versus diffusion-dominated zones—thereby improving generalization and interpretability. Therefor we used $K = 3$ in this experiment.

G.5 Effect of Buffer ϵ in RNN Stability Projection

Role of ϵ. In our stability-constrained training scheme, the buffer $\epsilon > 0$ enforces a strict inequality in the ISS constraints for gated RNNs.

(i) GRU case: We require $|U_r|_\infty \hat{\sigma}_f < 1$, and implement this by projecting each row of U_r onto an ℓ_1-ball of radius $\rho = \frac{1-\epsilon}{\hat{\sigma}_f}$, ensuring $|U_r|_\infty \hat{\sigma}_f \leq 1 - \epsilon$.

(ii) LSTM case: The ISS conditions are $(1+\bar{\sigma}_o)\bar{\sigma}_f < 1$ and $(1+\bar{\sigma}_o)\bar{\sigma}_i |U_c|_\infty < 1$. The first inequality involves only gate parameters and can be satisfied by parameter initialization/normalization. The second inequality requires constraining U_c, which we enforce via row-wise ℓ_1 projection onto a ball of radius $\rho = \frac{1-\epsilon}{(1+\bar{\sigma}_o)\bar{\sigma}_i}$. Thus ϵ again serves as a buffer to guarantee strict satisfaction.

Interpretation. The buffer provides a safety margin to protect against numerical errors, approximation artifacts, and small fluctuations during training. Without it, parameters may drift close to the boundary of the ISS region, risking instability in long sequences or under high-variance inputs.

Tuning ϵ. In practice, ϵ must be chosen to balance two competing goals:

- **Stability guarantee:** Larger ϵ provides a more conservative margin, making the system more robust to exploding activations or gradients.
- **Optimization fidelity:** However, larger ϵ also leads to more aggressive projections, causing the updated U to deviate significantly from the unconstrained optimum proposed by gradient descent. This may prevent the model from converging to an optimal solution, especially in tasks that require precise temporal modeling.

Empirical behavior. In our experiments, we observed that:

- When ϵ is too small (e.g., $\epsilon < 10^{-1}$), numerical instability occasionally occurred in long sequences or chaotic systems.
- When ϵ is too large (e.g., $\epsilon \geq 4 \times 10^{-1}$), likelihood performance noticeably degrades due to overly constrained dynamics and suppressed learning capacity of the GRU.
- A moderate value of ϵ (e.g., $\epsilon \in [10^{-1}, 2 \times 10^{-1}]$) provided a good balance between stability and performance across all datasets.

Guidelines. We recommend tuning ϵ via validation likelihood. Begin with a relatively loose constraint (e.g., $\epsilon = 10^{-1}$), and gradually increase it if instability is observed. Avoid unnecessarily large ϵ, as this limits the expressiveness of the learned transition dynamics and tends to reduce final log-likelihood.

Full Comparison of our ISS stability with Gradient Clipping. Table 6 provides the stability behavior of our ISS-based scheme against conventional gradient clipping (GC) on full 4 benchmarks. The ISS approach yields **100 % success across all datasets and buffer values** $\epsilon \in [0.1, 0.25]$, demonstrating that once the sufficient stability inequality is enforced, the system remains robust regardless of sequence length or dynamics. In contrast, GC exhibits partial or complete failure: as the clipping threshold δ increases, instabilities emerge, with divergence occurring in several tasks at $\delta \geq 10$.

Another advantage of our method is the ease of parameter tuning. The ISS buffer ϵ is interpretable, bounded, and chosen within the compact interval $(0, 1)$, making it straightforward to adjust and comparable across architectures (GRU, LSTM, etc.). In contrast, the GC threshold δ must be tuned over an unbounded range $[0, \infty)$, with optimal values highly problem-dependent and unstable across tasks. This makes our ISS projection scheme not only more reliable but also far more practical for deployment.

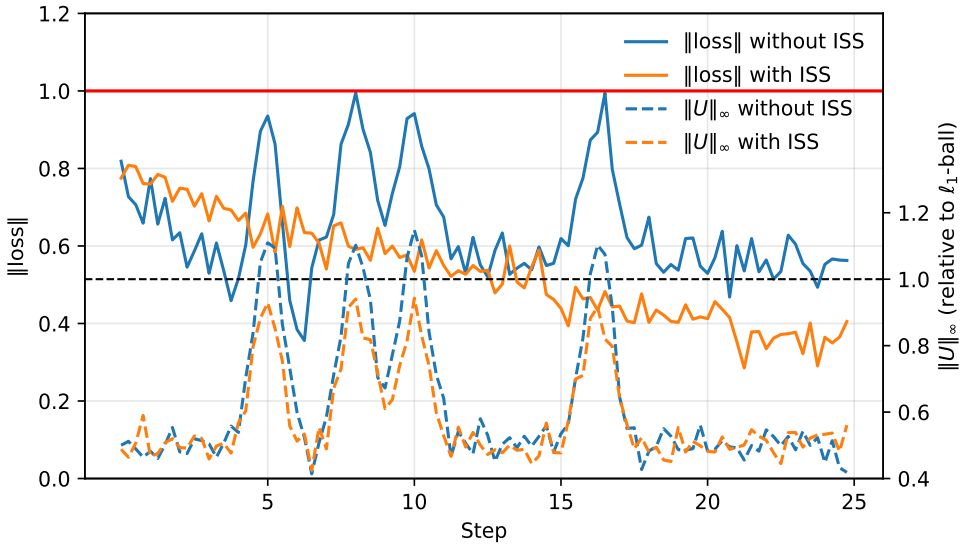


Figure 8: Training stability with and without ISS. Left axis: loss (-likelihood) norm. Right axis: $\|U\|_\infty$ relative to the ℓ_1-ball radius.

Table 6: Stability handling: comparison between GC and ISS.

		Pong		Lorenz		Navier-Stokes		NCLT	
		Objective	Success	Objective	Success	Objective	Success	Objective	Success
π-SSM$_{\text{GRU}}$(ISS)	$\epsilon = 0.1$	5.401 ± 0.197	100 %	5.856 ± 0.387	100 %	5.097 ± 0.247	100 %	-23.18 ± 1.07	100 %
	$\epsilon = 0.15$	5.231 ± 0.228	100 %	5.841± 0.528	100 %	5.026 ± 0.377	100 %	-23.44 ± 0.97	100 %
	$\epsilon = 0.2$	5.084 ± 0.281	100 %	5.624 ± 0.271	100 %	4.745 ± 0.216	100 %	-23.91 ± 1.07	100 %
	$\epsilon = 0.25$	4.888 ± 0.324	100 %	5.344 ± 0.400	100 %	4.510 ± 0.421	100 %	-24.51 ± 1.29	100 %
π-SSM$_{\text{LSTM}}$(ISS)	$\epsilon = 0.1$	5.475 ± 0.217	100 %	5.844 ± 0.292	100 %	5.137 ± 0.18	100 %	-23.25 ± 0.94	100 %
	$\epsilon = 0.15$	5.281 ± 0.174	100 %	5.827± 0.397	100 %	5.074 ± 0.333	100 %	-23.59 ± 0.94	100 %
	$\epsilon = 0.2$	5.172 ± 0.202	100 %	5.600 ± 0.311	100 %	4.816 ± 0.193	100 %	-23.97 ± 1.14	100 %
	$\epsilon = 0.25$	4.911 ± 0.215	100 %	5.344 ± 0.382	100 %	4.574 ± 0.266	100 %	-24.55 ± 0.93	100 %
π-SSM$_{\text{GRU}}$(GC)	$\delta = 5$	5.166 ± 0.446	100 %	5.511 ± 0.521	100 %	4.814 ± 0.495	100 %	-23.38 ± 1.84	100 %
	$\delta = 10$	5.249 ± 1.12	60 %	5.691 ± 1.18	50 %	4.92 ± 0.936	50 %	-23.20 ± 3.57	60 %
	$\delta = 15$	5.281 ± 2.541	30 %	N/A	0 %	N/A	0 %	-23.21 ± 9.54	20 %
	$\delta = 20$	N/A	0 %	N/A	0 %	N/A	0 %	N/A	0 %

H Additional Results

H.1 Extended Example for Pong Experiment and Imputation Strategy Explanation

To further illustrate the ability of π-SSM to capture mode-dependent dynamics, we provide a detailed example consisting of 8 representative frames from a ball trajectory. In this example, the ball undergoes 4 collisions with the enclosure walls, inducing 4 distinct mode transitions. The frames are selected to showcase the key dynamical changes before, during, and after each collision. See figure 9.

The learned transition matrices $\mathbf{A}_{z_t}$ corresponding to each mode are inspected, and their eigenvalue spectra are compared to the ground truth to verify structural alignment.

We examine the predictive distributions $q(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})$ at four critical points in time: $t = 15$, $t = 35$, $t = 55$, and $t = 75$. These represent pre- and post-collision phases and provide insight into how the model adjusts its uncertainty and dynamics based on the inferred mode. Each predictive distribution is modeled as a multivariate Gaussian, and we plot the corresponding ellipses for qualitative evaluation. These plots confirm that the model adapts to the nonlinear behavior of the ball by switching appropriately between learned dynamics and refining its state belief accordingly.

According to the inferred mode, here, each mode corresponds to a distinct linear segment of the ball's trajectory between wall bounces, making regime switches clearly observable. In this setting, the discrete states z_t can be directly interpreted and tracked, with four true modes governing the dynamics. Empirically, the inferred z_t samples capture these regime changes in a meaningful way. The behavior of z_t sequences can be characterized by two properties: (i) *uniqueness* of active modes, and (ii) *persistence* of modes across time.

Uniqueness. The number of modes K is treated as a hyperparameter. We observed a pruning effect from $q(z_t \mid x_{t-1})$: when K exceeds the true number of modes, redundant modes gradually vanish during training and are no longer sampled. In Pong, the inferred z_t concentrated on four dominant modes, consistent with the ground-truth structure (As it is evident in the Figure).

Persistence. To assess temporal persistence, we considered intervals between two consecutive bounces (T_1, T_2) where the true regime is constant. In these intervals, the inferred z_t consistently sampled the same mode index k, demonstrating coherent mode tracking over time and alignment with the actual regime (Refer to the consistent colors in each mode switch interval).

Additionally, for the imputation task, we randomly remove half of the observations from each generated trajectory and evaluate the model's ability to infer the missing values. When observations are unavailable during a temporal interval $[t{+}1, \ldots, t{+}\tau]$, the model is required to generate these values by leveraging its learned predictive structure. Specifically, at each step, the model uses the inferred latent state to recursively propagate forward through its transition dynamics and approximate the corresponding observation distributions.

As established in the main paper, the predictive likelihood at time $t + 1$ can be approximated as:

$$q(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t}) = \int p(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})\, p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, z_{t+1})\, q(z_{t+1} \mid \mathbf{x}_t)\, q(\mathbf{x}_t \mid \mathbf{y}_{1:t})\, d\mathbf{x}_t\, d\mathbf{x}_{t+1}\, dz_{t+1}. \tag{208}$$

The same logic extends recursively to future time steps $t' = t + 2, \ldots, t + \tau$, allowing the model to generate:

$$q(\mathbf{y}_{t'} \mid \mathbf{y}_{1:t}) \quad \text{for } t' > t$$

using a sequence of predictions through the latent state space. In practice, the system generates missing observations as follows:

$$q(\mathbf{y}_{t+\tau} \mid \mathbf{y}_{1:t}) = \int p(\mathbf{y}_{t+\tau} \mid \mathbf{x}_{t+\tau}) \prod_{s=1}^{\tau} \Big[p(\mathbf{x}_{t+s} \mid \mathbf{x}_{t+s-1}, z_{t+s})\, q(z_{t+s} \mid \mathbf{x}_{t+s-1}) \Big]\, q(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \\ d\mathbf{x}_t \cdots d\mathbf{x}_{t+\tau}\, dz_{t+1} \cdots dz_{t+\tau}. \tag{209}$$

This recursive structure leverages the learned dynamics of π-SSM to roll out latent trajectories and generate observations in the absence of measurements.

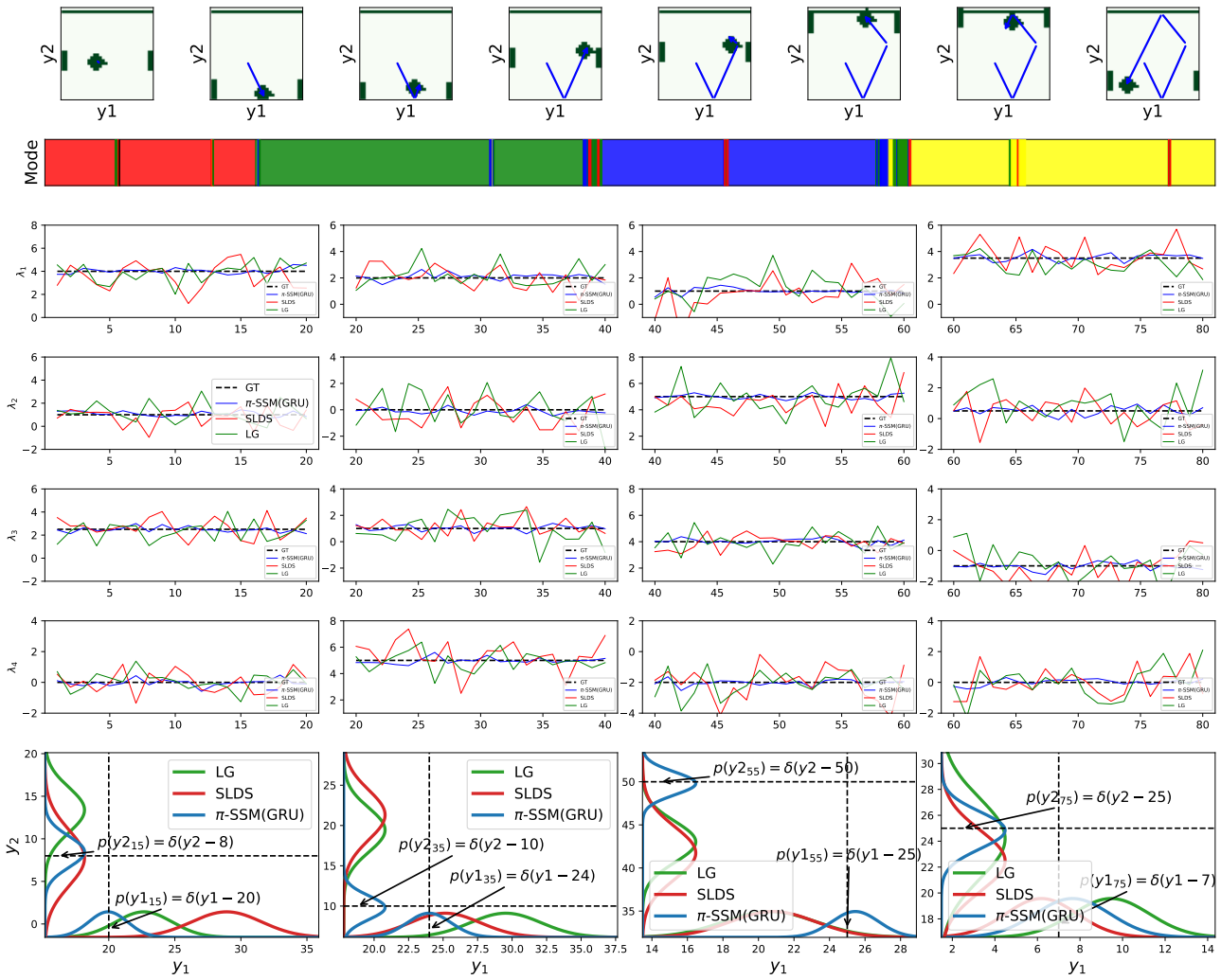


Figure 9: Predicted observation (position) at 15-th, 35-th, 55-th and 75-th time steps (last row). The first row shows the ground truth ball position in 10, 20, 30, 40, 50, 60, 70 and 80-th time steps, respectively.

H.2 NCLT Extra Results

Here we provide additional results of the NCLT experiment including the MSE result and one visualization of the inferred states.

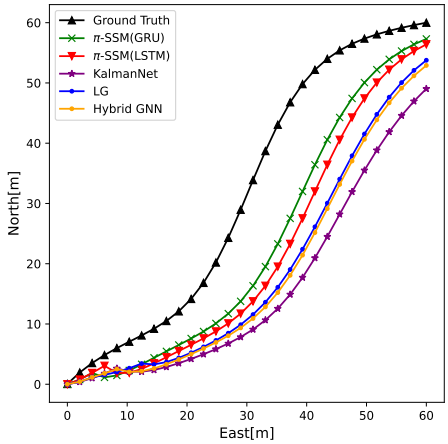


Figure 10: NCLT: Position estimation for the first 60 observations.

Table 7: MSE for NCLT experiment

Model	MSE [dB]
π-SSM$_{\text{GRU}}$	**19.50± 0.11**
π-SSM$_{\text{LSTM}}$	19.64± 0.14
LG	20.45±0.22
Hybrid GNN	20.73±0.21
KalmanNet	22.20±0.17
LSTM	22.83±0.62
GRU	22.57±0.41
Observation	25.47±0.08

H.3 MSE Results

In addition to the log-likelihood metrics reported in the main text, we include MSE results here to provide a more interpretable evaluation of predictive accuracy across tasks in Table 8. This supplementary evaluation highlights the effectiveness of π-SSM in reconstructing latent trajectories and predicting observations, particularly in the presence of noise or partial information. Furthermore, we include two PDE-specific baselines, PINN and SPINN—which are designed for physical systems governed by partial differential equations. Since these methods are not applicable to general state-space models or sequence modeling tasks, we only report their results for the Navier-Stokes experiment. For other experiments such as Pong, Lorenz, and NCLT, PINN and SPINN results are omitted.

Table 8: MSE across four benchmarks (lower is better).

Model	**Pong**	**Lorenz**	**Navier-Stokes**	**NCLT**
LSTM	0.097 ± 0.013	0.090 ± 0.015	0.360 ± 0.039	191.3 ± 10.3
GRU	0.095 ± 0.020	0.091 ± 0.014	0.356 ± 0.026	181.1 ± 7.45
SLDS	0.110 ± 0.031	0.105 ± 0.021	0.367 ± 0.019	141.5 ± 10.9
irSLSD	0.092 ± 0.027	0.083 ± 0.024	0.297 ± 0.030	130.2 ± 4.7
NODE	0.104 ± 0.024	0.095 ± 0.021	0.331 ± 0.051	155.4 ± 6.54
MoNODE	0.093 ± 0.021	0.084 ± 0.019	0.302 ± 0.039	134.5 ± 4.99
KalmanNet	0.086 ± 0.013	0.077 ± 0.009	0.291 ± 0.041	165.1 ± 6.34
GIN	0.085 ± 0.011	0.077 ± 0.004	0.288 ± 0.029	131.7 ± 5.17
Hybrid GNN	0.082 ± 0.013	0.075 ± 0.011	0.284 ± 0.033	118.2 ± 4.23
LG	0.083 ± 0.009	0.076 ± 0.011	0.260 ± 0.030	111.24 ± 2.03
PINN			0.227 ± 0.019	
SPINN			$\mathbf{0.202 \pm 0.030}$	
π-SSM$_{\text{GRU}}$	0.061 ± 0.009	$\mathbf{0.056 \pm 0.010}$	0.208 ± 0.019	$\mathbf{89.14 \pm 1.29}$
π-SSM$_{\text{LSTM}}$	$\mathbf{0.058 \pm 0.011}$	0.057 ± 0.007	0.205 ± 0.011	92.15 ± 0.98