CABLE: CONVEX AFFINE BLENDS OF LATENT EXPERTS FOR PARAMETRIC REDUCED-ORDER MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-intrusive reduced-order models often learn low-dimensional latent dynamics. However, existing approaches typically trade explicit structure for the flexibility needed to capture parametric and temporal variation. Equation-based models are efficient and analyzable, but can be too rigid for strongly varying dynamics. Neural dynamical models are more expressive but hide the learned vector field inside a network and can be costly to integrate. We introduce CABLE (Convex Affine Blends of Latent Experts), which represents the latent dynamics as a convex mixture of shared affine ordinary differential equation (ODE) experts. A neural gate varies the mixture weights with the parameters, latent state, or time, allowing a single model to adapt across dynamical regimes while retaining explicit operators. These operators enable rollout-error bounds, sufficient conditions for contraction imposed directly on the experts, and a dynamics-based adaptive sampling criterion. Across several parametric systems, CABLE produces accurate and efficient predictions. On a two-parameter Vlasov–Fokker–Planck two-stream instability, uniformly sampled CABLE matches or improves upon parameterized neural ODE, Latent Space Dynamics Identification, U-Net, and Fourier neural operator baselines in typical-case accuracy while providing lower upper-tail errors and the fastest inference. Adaptive sampling further improves upper-tail accuracy, approximately halving the maximum error relative to the best uniformly sampled baseline.

1 INTRODUCTION

High-fidelity numerical simulations are indispensable across science and engineering (Calder et al., 2018; Cummings et al., 2015; Jones et al., 2020; Thijssen, 2007; Vasileska et al., 2017; Noble, 2002), but their cost often prevents the repeated evaluations needed for parameter studies, uncertainty quantification, and optimization. Reduced-order models (ROMs) reduce this cost by replacing the full-order simulation with a cheaper surrogate that can be evaluated efficiently at new conditions. Neural operators provide another global approach by learning a direct map from problem inputs to solution fields (Kovachki et al., 2023; Li et al., 2020), but can require many full-order training trajectories to cover a parameter domain. When simulations are expensive, a common non-intrusive approach is to compress the full-order state into a low-dimensional latent space with an autoencoder, and model the ordinary differential equations (ODEs) governing the latent evolution (Champion et al., 2019; Fries et al., 2022; Lee & Parish, 2021; de Jong et al., 2026). A central challenge is choosing how to model the latent dynamics.

Neural ordinary differential equations (NODE) (Chen et al., 2018), and their parameterized variant PNODE (Lee & Parish, 2021), learn a neural network which directly models the latent dynamics. This approach learns a single, extremely expressive model over the entire parameter space. However, the latent dynamics are then entirely determined by a neural network, making it difficult to characterize or inspect properties of the learned dynamics model.

To obtain more explicit models, several authors instead learn equation-based models. For example, Champion et al. (2019) apply Sparse Identification of Nonlinear Dynamics (SINDy) (Brunton et al., 2016) to learn explicit ODEs for the latent space dynamics. These models retain explicit structure, are inexpensive to evaluate, and are easier to analyze, but a single learned equation is often too restrictive for complex parameter-dependent behavior. Fries et al. (2022) address this issue with Latent Space Dynamics Identification (LaSDI), which learns a local ODE for each training input parameter and

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

interpolates coefficients for unseen input parameters. Many followup LaSDI variants still use this interpolation-based approach (He et al., 2023; Bonneville et al., 2024; Tran et al., 2024; Chung et al., 2025; Stephany et al., 2026). Coefficient interpolation relies on a potentially dubious assumption that dynamics at a testing parameter can be expressed as a simple interpolation of learned local dynamics, and is unstable for higher-order dynamics (Anderson et al., 2025). Moreover, interpolation of stable systems does not guarantee stability, and is unnatural for parameters which vary within a trajectory, such as prescribed forcing.

Compression to a latent space also changes the goal of dynamics identification. Latent coordinates generally have no physical meaning, so learned equations are hard to interpret, and even an exact latent equation says little about the full-order system because the compressed trajectory runs back through a neural decoder. Instead, what we often desire is a latent model that accurately approximates the evolution while remaining inexpensive and structured enough to analyze or constrain.

This observation raises a natural question: must global expressivity come at the expense of explicit dynamical structure? Switching Linear Dynamical Systems (SLDS) (Linderman et al., 2016; Ackerson & Fu, 1970) and Piecewise-Affine Dynamical (PWA) Systems (Pavlov et al., 2007) suggest that introducing time-dependent switches can be expressive while providing explicit dynamical structure. We build on this observation to introduce **CABLE: Convex Affine Blends of Latent Experts**, which models the latent dynamics as a convex combination of learned affine ODEs whose weights depend on time, the input parameters, and, when needed, the latent state. Following standard mixture-of-experts terminology (Jacobs et al., 1991; Jordan & Jacobs, 1994; Yuksel et al., 2012), we call each component ODE an *expert* and the network which assigns mixture weights the *gate*. The experts are shared across the parameter domain and fixed after training, while the gate varies along each trajectory. Thus, the active latent operator changes during a rollout but is always an explicit affine map. Unlike switching systems with hard or discretely inferred switches, our gate is continuous, differentiable, and trained end to end with the experts. The central design choice is that the network performs operator selection while the experts perform operator representation.

Our contributions. We introduce CABLE as a parametric reduced-order modeling approach, with the following benefits:

1. **A global parametric latent dynamics model** that learns a fixed library of affine latent ODEs and a neural gate producing their time-varying convex mixture.
2. **Analysis and constraints from explicit operators**, including an architecture-specific rollout error bound separating amplification by the experts from that introduced by state-dependent gating, and the ability to impose structural conditions on the experts such as stability.
3. **Dynamics-based adaptive sampling**, which finds parameter regions the training set represents poorly using only the learned latent model.

Related Works The individual ingredients used by CABLE connect to several established areas. Linear parameter-varying (LPV) systems, in which the coefficients of a linear model depend on a scheduling signal, have been studied extensively in control together with associated common-Lyapunov stability conditions (Boyd et al., 1994; Apkarian et al., 1995; Wu et al., 1996; Mohammadpour & Scherer, 2012; Shamma, 2012; Hoffmann & Werner, 2014). Mixtures of experts have a similarly long history in machine learning and are widely used to scale neural networks (Shazeer et al., 2017; Riquelme et al., 2021; Fedus et al., 2022; Zhou et al., 2022). Among learned latent-dynamics models, decomposed linear dynamical systems (dLDS) (Mudrik et al., 2024) and Deep Variational Bayes Filters with locally linear transitions (DVBF-LL) (Karl et al., 2017) are closest to CABLE. dLDS represents latent dynamics as a sparse, per-timestep combination of a learned linear dictionary, but does not learn a parametric surrogate for prediction at unseen input parameters. DVBF-LL forms discrete-time locally linear transitions from state and control-dependent combinations of shared matrices within a variational state-space model. However, it does not consider generalization across a physical parameter domain. CABLE instead learns a global, continuous-time parametric surrogate based on convex combinations of shared affine ODE operators. This structure allows us to analyze rollout error, impose stability conditions directly on the experts, and construct a dynamics-based adaptive sampling criterion.

2 Problem Setting

We consider non-intrusive reduced-order modeling for parameterized dynamical systems. All methods discussed here share a common structure where full-order snapshots are compressed into a low-dimensional latent space, a dynamical model is learned for the latent variables, and the trajectories are decoded back to the original state space. Prediction at a new parameter then requires only encoding the initial condition, integrating the learned latent dynamics, and decoding the result. The methods differ primarily in how they represent and learn the latent dynamics.

Throughout this manuscript, $\|\cdot\|$ denotes the Euclidean norm for vectors and the Frobenius norm for matrices unless otherwise specified, and $\|\cdot\|_1$ represents the ℓ_1-norm.

Governing Equations. We consider parameterized dynamical systems of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t;\boldsymbol{\mu}) = \mathbf{g}\big(\mathbf{u}(t;\boldsymbol{\mu}), t;\boldsymbol{\mu}\big), \qquad \mathbf{u}(0;\boldsymbol{\mu}) = \mathbf{u}_0(\boldsymbol{\mu}), \tag{1}$$

where $\mathbf{u}(t;\boldsymbol{\mu}) \in \mathbb{R}^{N_u}$ is the full-order state, $\mathbf{g} : \mathbb{R}^{N_u} \times [0, t_f] \times \mathcal{D} \to \mathbb{R}^{N_u}$ is the governing vector field, and $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^{d_\mu}$ denotes the input parameters. The parameters may affect the initial condition or physics of the simulation, and the explicit time dependence in (1) admits nonautonomous behavior such as time-dependent forcing.

We assume that $\mathbf{g}$ is unavailable and that the training data consist only of snapshots of the state vector. Let $0 = t_0 < t_1 < \cdots < t_{N_t} = t_f$ denote the snapshot times which, for simplicity, are assumed to be uniformly spaced with $\Delta t = t_{j+1} - t_j$. Given N_μ training parameters $\{\boldsymbol{\mu}^{(i)}\}_{i=1}^{N_\mu}$, we write $\mathbf{u}_j^{(i)} := \mathbf{u}(t_j;\boldsymbol{\mu}^{(i)})$ for the jth snapshot of the ith trajectory. Our objective is to construct a computationally inexpensive surrogate that predicts the state $\mathbf{u}(t;\boldsymbol{\mu}^\star)$ for an unseen parameter $\boldsymbol{\mu}^\star \in \mathcal{D}$.

Data compression. The full-order dimension N_u is often too large for efficient dynamics learning and time integration. We therefore compress each state snapshot using an autoencoder (Hinton & Salakhutdinov, 2006), following common practice in latent-space ROMs such as LaSDI (Fries et al., 2022) and PNODE (Lee & Parish, 2021). These autoencoders consist of an encoder map which compresses the data, $\mathcal{G}_{\text{enc}} : \mathbb{R}^{N_u} \to \mathbb{R}^{N_z}$, and a decoder which learns the map $\mathcal{G}_{\text{dec}} : \mathbb{R}^{N_z} \to \mathbb{R}^{N_u}$, where $N_z \ll N_u$. Each snapshot then has a latent representation $\mathbf{z}_j^{(i)} := \mathcal{G}_{\text{enc}}\big(\mathbf{u}_j^{(i)}\big) \in \mathbb{R}^{N_z}$. We denote the usual mean-squared autoencoder reconstruction loss by $\mathcal{L}_{\text{AE}}$ and train the autoencoder jointly with the latent dynamics below.

3 Convex affine blends of latent experts

We assume that the latent space satisfies its own dynamical system

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{z}(t;\boldsymbol{\mu}) = \mathbf{f}\big(\mathbf{z}(t;\boldsymbol{\mu}), t;\boldsymbol{\mu}\big), \qquad \mathbf{z}(0;\boldsymbol{\mu}) = \mathcal{G}_{\text{enc}}(\mathbf{u}_0(\boldsymbol{\mu})), \tag{2}$$

and we aim to model the latent vector field $\mathbf{f} : \mathbb{R}^{N_z} \times [0, t_f] \times \mathcal{D} \to \mathbb{R}^{N_z}$ with a learned approximation $\hat{\mathbf{f}}$. CABLE separates two roles in the latent dynamics model. A shared library of explicit affine ODE experts represents the latent operators, while a neural gate selects their convex combination along each trajectory. The model is therefore global across the parameter domain, but its active operator can vary with the parameters, time, prescribed inputs, and, when needed, the latent state.

Dynamics models. We can succinctly summarize the difference between CABLE, global black-box latent dynamics models, and local equation-based latent dynamics models. Global black-box dynamics models such as PNODE predict the latent derivative with a neural network,

$$\dot{\mathbf{z}} \approx \hat{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{z}, t;\boldsymbol{\mu}), \tag{3}$$

where $\hat{\mathbf{f}}$ is a network with learned parameters $\boldsymbol{\theta}$. Local equation-based models such as LaSDI identify a separate affine system at each training parameter $\boldsymbol{\mu}^{(i)}$, and interpolate the ODE coefficients

$$\dot{\mathbf{z}}(t;\boldsymbol{\mu}^{(i)}) \approx A^{(i)}\mathbf{z} + \mathbf{b}^{(i)}. \tag{4}$$

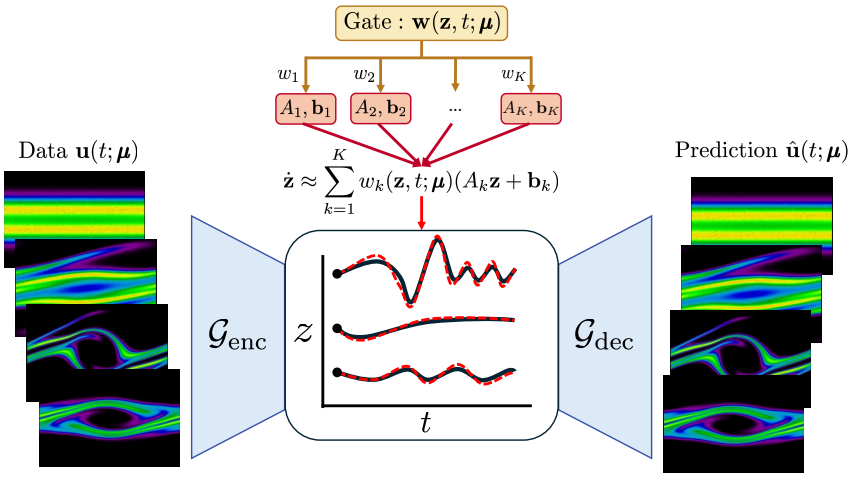

Figure 1: CABLE schema. Full-order data are spatially compressed with an autoencoder, and true latent trajectories (solid black curves) are approximated by evolving a convex mixture of K affine experts (dashed red curves). The experts are fixed after training, and the neural gate $\mathbf{w}$ decides how they combine along each trajectory. To predict at unseen input parameters $\boldsymbol{\mu}$ we encode the initial condition, evolve in the latent space, and decode the trajectory.

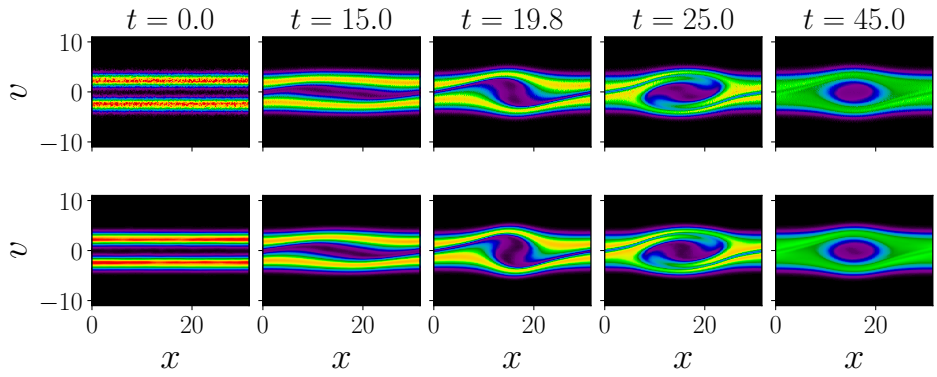

Figure 2: Example adaptive CABLE prediction for the Vlasov–Fokker–Planck equation (17), at $(\alpha, T) = (0.005, 0.58)$, comparing (top row) the full-order solution with (bottom row) CABLE. Peak error occurs at $t = 19.8$, where the vortex forms slightly early.

CABLE instead learns a shared library of K affine ODEs and a neural gate that determines their convex weights,

$$\dot{\mathbf{z}} \approx \sum_{k=1}^{K} w_k(\mathbf{z}, t; \boldsymbol{\mu}) \left(A_k\mathbf{z} + \mathbf{b}_k\right), \qquad w_k \geq 0, \quad \sum_{k=1}^{K} w_k = 1. \tag{5}$$

The network therefore selects among explicit operators rather than predicting the time derivative directly. Figure 1 shows a schematic of CABLE, and we make these statements more precise below.

3.1 Mixtures of affine experts

The *experts* are coefficient matrices $\Xi_k := (\mathbf{b}_k \;\; A_k)^\top$ with $A_k \in \mathbb{R}^{N_z \times N_z}$ and $\mathbf{b}_k \in \mathbb{R}^{N_z}$. Experts are shared across all trajectories and fixed after training. A neural *gate* produces weights in the probability simplex $\mathbf{w}(\mathbf{z}, t; \boldsymbol{\mu}) \in \mathcal{W}^K := \{\mathbf{w} \in \mathbb{R}^K : w_k \geq 0, \sum_{k=1}^{K} w_k = 1\}$. Here, we take the neural gate $\mathbf{w}$ to be a 2-hidden-layer multi-layer perceptron with Tanh activation function on the hidden layers, and a Softmax activation on the final layer. In general, we need not restrict the library

to affine terms, but this simplifying assumption will provide both error bounds and accurate results for our examples considered here.

For the analysis below, it is useful to define the convex mixtures

$$\Xi(\mathbf{w}) := \sum_{k=1}^{K} w_k \Xi_k, \qquad A(\mathbf{w}) := \sum_{k=1}^{K} w_k A_k, \qquad \mathbf{b}(\mathbf{w}) := \sum_{k=1}^{K} w_k \mathbf{b}_k. \tag{6}$$

With the affine feature library $\Theta(\mathbf{z}) = (1\ \mathbf{z}^\top)^\top$, the CABLE dynamics (5) may equivalently be written as

$$\dot{\mathbf{z}} = \Xi\big(\mathbf{w}(\mathbf{z}, t; \boldsymbol{\mu})\big)^\top \Theta(\mathbf{z}) = A(\mathbf{w})\mathbf{z} + \mathbf{b}(\mathbf{w}), \tag{7}$$

which is closer to notation traditionally used for SINDy-based models. Although each expert is affine, varying the mixture weights along a trajectory produces a richer class of dynamics than a single fixed affine system.

We train the experts and gate by matching the latent trajectory derivatives,

$$\mathcal{L}_{\mathrm{DI}} := \frac{1}{N_\mu (N_t+1) N_z} \sum_{i=1}^{N_\mu} \sum_{j=0}^{N_t} \left\| \dot{\mathbf{z}}_j^{(i)} - \Xi\big(\mathbf{w}(\mathbf{z}_j^{(i)}, t_j; \boldsymbol{\mu}^{(i)})\big)^\top \Theta(\mathbf{z}_j^{(i)}) \right\|^2, \tag{8}$$

where we approximate the latent derivatives with second-order centered differences, and first-order differences at the endpoints. The autoencoder, experts, and gate are then trained jointly by minimizing

$$\mathcal{L}_{\mathrm{CABLE}} = \mathcal{L}_{\mathrm{AE}} + \beta_1 \mathcal{L}_{\mathrm{DI}} + \beta_2 \frac{1}{K(N_z+1)N_z} \sum_{k=1}^{K} \|\Xi_k\|^2. \tag{9}$$

Here, β_1 and β_2 are hyperparameters. Additional objectives, including rollout and steady-state losses, are introduced as needed in the numerical examples.

We discuss several important implementation choices here. To encourage sparsity and stabilize rollouts, we periodically threshold small coefficients below a user-specified threshold to zero during training. In the spirit of the sequential thresholding used with SINDy (Brunton et al., 2016), thresholded coefficients are masked and receive no further gradient information during training. We also initialize our model with a large number of initial experts. Training cost is generally dominated by the autoencoder, and the mixed coefficients (6) are computed in parallel across trajectories, snapshots, and experts. This means that using more experts has negligible effect on training cost (Table 3). Remaining details, including a short rollout fine-tuning stage, are in Appendix A.

3.2 Gate inputs

The gate controls how our active latent operator changes across the parameter domain and along each trajectory. Any quantity used by the gate during prediction must either be known in advance or available while integrating the latent ODE. We consider two gate formulations depending on whether the latent state is included as an input.

Scheduling gate. A *scheduling gate* depends on time and other known quantities, but not on the evolving latent state, $\dot{\mathbf{z}} = \sum_k w_k(t; \boldsymbol{\mu})(A_k \mathbf{z} + \mathbf{b}_k)$. Due to the fact that the weights do not depend on the latent state $\mathbf{z}$, the mixed coefficient matrices $\Xi(\mathbf{w})$ and learned operators $A(\mathbf{w})$, $\mathbf{b}(\mathbf{w})$ can be determined over the entire prediction interval before time integration of the latent dynamics. This makes the weights easy to inspect and allows them to be precomputed.

State-dependent gate. A *state-dependent gate* also observes the evolving latent state, $\dot{\mathbf{z}} = \sum_k w_k(\mathbf{z}(t), t; \boldsymbol{\mu})(A_k \mathbf{z} + \mathbf{b}_k)$. This produces a nonlinear vector field and increases expressive power, but the weights can no longer be precomputed and rollout error may become coupled with error of the mixture weights. In both cases, the network predicts only the mixture weights while the operators stay explicit, which lets us analyze CABLE directly.

4 Exploiting explicit experts

A benefit of the explicit operators is that they allow us to derive a rigorous rollout error bound, and potentially impose structural conditions on the experts. These results help inform our design choices.

Rollout error bound. Let the prediction $\hat{\mathbf{z}}$ solve the CABLE dynamics from initial condition $\hat{\mathbf{z}}(0) = \mathbf{z}(0)$, with decoded prediction $\hat{\mathbf{u}} = \mathcal{G}_{\text{dec}}(\hat{\mathbf{z}})$. We define the autoencoder error, dynamics residual, and consider the Euclidean logarithmic norm,

$$\varepsilon_{\text{AE}} := \sup_t \|\mathbf{u}(t) - \mathcal{G}_{\text{dec}}(\mathbf{z}(t))\|, \quad \mathbf{r}(t) := \dot{\mathbf{z}}(t) - \hat{\mathbf{f}}(\mathbf{z}(t), t), \quad \omega_2(A) := \lambda_{\max}\big(\tfrac{1}{2}(A + A^\top)\big), \tag{10}$$

with maximum dynamics error $\varepsilon_{\text{Dyn}} := \sup_t \|\mathbf{r}(t)\|$.

The simplex constraint on our weights provides two useful properties. First, a convex mixture cannot exceed the worst expert in either matrix norm or logarithmic norm. Second, changes in the mixed operator can be bounded by the spread of the expert library. We formalize these observations in Lemma 1 (Appendix C) and use them to prove a rollout error bound.

Theorem 1 (Rollout error bound for CABLE)**.** *Let $\Omega \subset \mathbb{R}^{N_z}$ be a bounded set containing $\mathbf{z}([0, t_f]) \cup \hat{\mathbf{z}}([0, t_f])$. Assume the decoder is Lipschitz on Ω with constant L_D, and that the gate is Lipschitz in the state uniformly in t and in any suppressed constant arguments so that $\|\mathbf{w}(\mathbf{z}_1, t) - \mathbf{w}(\mathbf{z}_2, t)\|_1 \leq L_w \|\mathbf{z}_1 - \mathbf{z}_2\|$ for $\mathbf{z}_1, \mathbf{z}_2 \in \Omega$. A scheduling gate corresponds to $L_w = 0$. Let Ξ_k collect the coefficients of expert k and Θ be the affine feature map of Section 3. Define the constants*

$$M_\Theta := \sup_{\mathbf{z} \in \Omega} \|\Theta(\mathbf{z})\|, \qquad R_\Xi := \min_C \max_{1 \leq k \leq K} \|\Xi_k - C\|, \qquad \nu := \max_{1 \leq k \leq K} \omega_2(A_k), \tag{11}$$

and the effective growth rate

$$\nu_{\text{eff}} := \nu + M_\Theta R_\Xi L_w. \tag{12}$$

Then for every $t \in [0, t_f]$, our CABLE prediction error is bounded as

$$\|\mathbf{u}(t) - \mathcal{G}_{\text{dec}}\left(\hat{\mathbf{z}}(t)\right)\| \leq \varepsilon_{\text{AE}} + L_D \int_0^t e^{\nu_{\text{eff}}(t-s)} \|\mathbf{r}(s)\| \, \mathrm{d}s \leq \varepsilon_{\text{AE}} + L_D \, \varepsilon_{\text{Dyn}} \, \varphi_{\nu_{\text{eff}}}(t), \tag{13}$$

where $\varphi_\nu(t) = (e^{\nu t} - 1)/\nu$ for $\nu \neq 0$ and $\varphi_0(t) = t$.

The proof is given in Appendix C.2.

The bound (13) separates two mechanisms of error growth. The first is growth of the experts, measured by ν, which is controlled by the worst expert. The second mechanism, $M_\Theta R_\Xi L_w$, only appears when using a state-dependent gate. Here, M_Θ is fixed by the affine library, so we cannot do much to control it. If all experts coincide then $R_\Xi = 0$ and the gate can be arbitrarily sensitive at no cost, but a wide parameter space calls for diverse experts, which makes R_Ξ large. A scheduling gate avoids this problem, as it is independent of the state and thus $L_w = 0$. State dependence can still pay off by reducing dynamics model error ε_{Dyn}, but we prefer a scheduling gate whenever the problem allows one due to the lower growth rate and precomputable mixture weights.

Like most Gronwall estimates, the bound (13) can become vacuous over long horizons when $\nu_{\text{eff}} > 0$, since it assumes worst-case scalar growth rates. For scheduling gates, having the explicit operator $A(\mathbf{w})$ before rollouts permits tighter, path-wise bounds based on the coefficient mixture and its finite-time state-transition matrix, which we derive and evaluate in Appendix C.3.

Enforcing structure on experts. The explicit operators also allow us to impose structural conditions directly on the experts. For instance, sharing a common quadratic Lyapunov function across the library makes every convex mixture contractive in a shared norm (Boyd et al., 1994; Shamma, 2012). We can enforce this property by construction through learning a shared metric $P := WW^\top + \epsilon I$ and setting $A_k = P^{-1}(J_k - R_k)$ with J_k skew-symmetric and $R_k = V_k V_k^\top + \eta P$. We do not impose this by default, since many systems need transient growth or sustained oscillation, and use it only when a system is expected to relax to a steady state. The construction, a uniform-in-time bound, and a demonstration are given in Appendix D. We may also impose other conditions on the latent evolution, such as ensuring that it conserves mass $\mathbf{m}(t)$ by enforcing $\mathbf{m}^\top A_k = \mathbf{m}^\top \mathbf{b}_k = \mathbf{0}$.

5 Dynamics-based adaptive sampling

Full-order simulations are expensive, so new training parameters should be chosen where they are expected to most improve the model. In general, we would prefer residual-based criteria based on the governing equations (He et al., 2023), but we do not assume access to the governing equations which

generated our data. Instead, we desire an approach based purely on our model and the data. For example, Gaussian Process-based LaSDI (GPLaSDI) samples based on uncertainty in the interpolated ODE coefficients (Bonneville et al., 2024). For CABLE, we leverage the fact that our model defines a latent vector field $\hat{\mathbf{f}}_{\boldsymbol{\mu}}(\mathbf{z}, t) = \Xi(\mathbf{z}, t; \boldsymbol{\mu})^\top \Theta(\mathbf{z})$ at every training and candidate input parameter. So, we can ask whether a candidate's predicted dynamics are already represented by the training set. Letting $\hat{\mathbf{z}}_{\boldsymbol{\mu}'}$ be the rollout at a candidate $\boldsymbol{\mu}'$, we compare it against a sampled parameter $\boldsymbol{\mu}$ by evaluating both vector fields along the candidate trajectory and integrating,

$$d_f(\boldsymbol{\mu}', \boldsymbol{\mu}) := \int_0^{t_f} \left\| \left[\Xi(\hat{\mathbf{z}}_{\boldsymbol{\mu}'}(t), t; \boldsymbol{\mu}') - \Xi(\hat{\mathbf{z}}_{\boldsymbol{\mu}'}(t), t; \boldsymbol{\mu})\right]^\top \Theta\big(\hat{\mathbf{z}}_{\boldsymbol{\mu}'}(t)\big) \right\| \mathrm{d}t. \tag{14}$$

This is large when the two parameters give different latent derivatives at states the candidate visits. Given the sampled set S and candidate set D, we score each candidate by its distance to the nearest training parameter and take the worst,

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}' \in D} \min_{\boldsymbol{\mu} \in S} d_f(\boldsymbol{\mu}', \boldsymbol{\mu}). \tag{15}$$

At each round of adaptive sampling we roll out every candidate, evaluate (15), and add the input parameter to the training set before continuing to train.

6 Numerical results

We now apply CABLE to several different problems. We begin with simpler demonstrations which isolate certain model properties. Our flagship result is a difficult Vlasov–Fokker–Planck two-stream instability, where CABLE is competitive with or beats several reduced-order modeling baselines. We report the relative ℓ_2-error between a trajectory $\mathbf{u}$ and the approximation $\hat{\mathbf{u}}$. Full data-generation, architecture, and training details for every example are in Appendix B.

6.1 Isolating CABLE model properties

K-ablation. As a proof-of-concept example, we first consider a one-dimensional heat equation driven by low-frequency, time-dependent forcing. Table 3 (Appendix B.1) shows that errors generally decrease as the initial number of experts K grows, while per-epoch training time remains constant at 12 ms because the expert computations are parallelized. With $K = 100$ experts, the maximum in-distribution and out-of-distribution prediction errors are 6.70% and 2.64%. This motivates our choice to over-provision the number of initial experts.

Enforcing stability. In Appendix D.2 we revisit the heat equation with a forcing function that decays to zero, comparing CABLE with a common quadratic Lyapunov function (CQLF) against base CABLE and PNODE. With a shared Lyapunov metric, CABLE is guaranteed to contract, whereas base CABLE and PNODE drift or diverge beyond the training window (Figure 10). This shows the value of imposing structure on the experts when a system is expected to relax.

Time extrapolation and expressivity. The Van der Pol oscillator

$$\dot{x} = y, \qquad \dot{y} = \mu(1 - x^2)y - x, \qquad (x(0), y(0)) = (2, 0), \tag{16}$$

is autonomous and nonlinear, with a μ-dependent limit cycle. The system is two-dimensional, so we drop the autoencoder and learn directly in (x, y), removing the nonlinear change of coordinates the autoencoder usually provides from our joint training (9) and making this a direct test of CABLE's expressiveness. We train on seven equally spaced values of $\mu \in [1, 4]$ and test on the six midpoints, with $K = 10$ experts. Since the dynamics are determined by the state rather than by time, a scheduling gate struggles here. We instead use a state-dependent gate $\mathbf{w}(x, y; \mu)$, which also allows us to extrapolate in time. Figure 5 (Appendix B.3) shows that CABLE captures the limit cycle at held-out μ and extrapolates accurately from $t = 20$ to $t = 40$, reconstructing the cubic nonlinearity.

Adaptive sampling validation. In Appendix B.4 we test our adaptive sampling scheme on a viscous Burgers problem whose initial-condition amplitude has a narrow peak around $\mu = 0.5$. Near this peak the solution advects significantly, whereas elsewhere it is primarily diffusion dominated. We freeze an autoencoder pretrained on all training trajectories so that the adaptive and uniform strategies differ only in the trajectories used to train the dynamics. Figure 6 (Appendix B.4) shows that adaptive sampling concentrates near the narrow nonlinear regime, reducing both mean and peak errors.

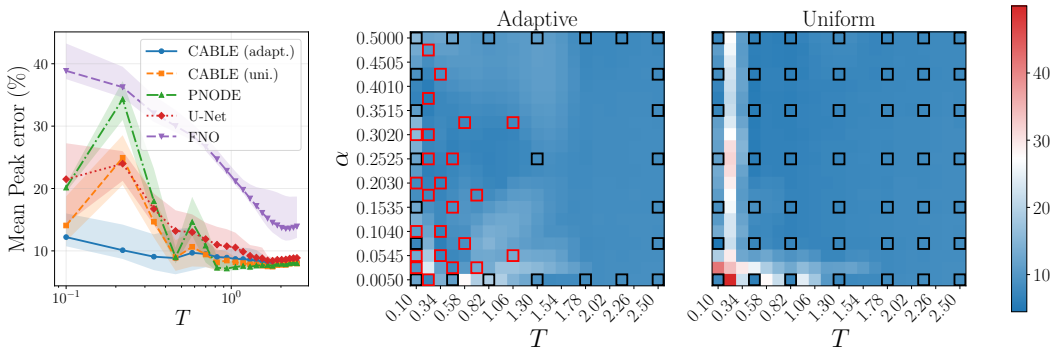

Figure 3: (Left) Peak errors for each temperature of the Vlasov example, averaged over all α values for that temperature. Shaded region indicates min/max value attained over 5 seeds. (center) CABLE peak error across the parameter domain for adaptive and (right) uniform sampling. Black boxes mark initial training parameters and red boxes adaptively added ones.

6.2 Vlasov–Fokker–Planck two-stream instability

Our main test is the 1D–1V Vlasov–Fokker–Planck two-stream instability,

$$\begin{cases} \partial_t f + \partial_x(vf) + \partial_v\left(\frac{\mathrm{d}\Phi}{\mathrm{d}x} f\right) = \gamma\, \mathcal{Q}(f), \qquad \frac{\mathrm{d}^2\Phi}{\mathrm{d}x^2} = \int_{\mathbb{R}} f\,\mathrm{d}v - 1, \\ f(0,x,v;\boldsymbol{\mu}) = \frac{1}{2\sqrt{2\pi T}}\left[1+\alpha\cos(0.2x)\right]\left[e^{-(v-2.4)^2/2T} + e^{-(v+2.4)^2/2T}\right], \end{cases} \tag{17}$$

on $t \in (0, 50]$, $x \in [0, 10\pi]$, $v \in \mathbb{R}$, with collision frequency $\gamma = 0.001$ and a Fokker–Planck operator $\mathcal{Q}$ given in Appendix B.5. The parameters are $\boldsymbol{\mu} = (\alpha, T)$ with $\alpha \in [0.005, 0.5]$ and $T \in [0.1, 2.5]$. The temperature T sets the initial width of the two beams and α controls strength of the nonlinear perturbation. The dynamics vary significantly across the parameter space (Figure 7, Appendix B.5), where cold beams lead to formation of a vortex, while hot beams are steady for low α-values. Increasing α leads to much stronger oscillations and filamentation. This problem initially motivated CABLE, because the dynamics change along even a single trajectory. The beams first evolve slowly, then the instability develops to form a vortex, and finally the solution settles into an oscillatory quasi-steady state. From this perspective, an equation-based method with one autonomous affine latent ODE to cover even a single trajectory is a strong restriction.

Data and baselines. We generate 441 full-order model simulations with an explicit particle-in-cell method (Yoo et al., 2025; Ricketson & Hu, 2025) and interpolate onto a 200×200 phase-space grid, for a uniform 21×21 grid over (α, T). A single full-order run costs 1000–1600 seconds, depending on the initial condition. We compare CABLE against PNODE and GPLaSDI as ROM baselines, which are all trained jointly with the same CNN autoencoder architecture. We also compare against a U-Net (Ronneberger et al., 2015) and an FNO (Li et al., 2020) as direct operator baselines. Every model uses a training budget of 49 trajectories, where uniformly sampled models use a fixed 7×7 subgrid, while adaptive CABLE starts from 25 samples and adaptively adds 24 more samples using the acquisition criteria (15). Both CABLE models use a scheduling gate $\mathbf{w}(t, \alpha, T)$ with $K = 100$ initial experts, and we observe that 4–12 experts remain nonzero after thresholding, depending on the initial seed. Additionally, in Table 5 (Appendix B.5), we compare our results against CABLE with a gate that has a linear last layer and unconstrained weights (motivated by comparing to a parametric extension of dLDS), showing that our convex assumption on the weights improves prediction accuracy and generalization. Each model is trained on 5 different seeds.

Accuracy. Table 1 shows that uniform CABLE produces the lowest 90th–99th percentile errors among all uniformly sampled models, while remaining comparable on mean and median accuracy. To separate the contribution of the learned dynamics from that of the autoencoder, we note that all ROM baselines use the same CNN autoencoder architecture, and that uniform CABLE, PNODE, and GPLaSDI reach comparable autoencoder error. Thus, differences among these three models primarily reflect the dynamics model, where uniform CABLE provides the lowest tail and mean

Table 1: Two-stream instability (17) relative f-error (%) and inference cost over all 441 parameter cells. "Peak" is the maximum error within a trajectory, then aggregated over all cells. All models use the same 49-trajectory budget and results are calculated over 5 seeds, with $\pm$ denoting standard deviation over the seeds. Best results in **bold**.

	CABLE		PNODE	GPLaSDI	U-Net	FNO
	Adaptive	Uniform				
Peak: maximum	**26** ± 3	50 ± 9	68 ± 9	116 ± 2	61 ± 2	50 ± 2
Peak: 99th pct.	**20** ± 5	31 ± 4	42 ± 6	113 ± 2	42 ± 4	44 ± 1
Peak: 95th pct.	**14** ± 3	21 ± 2	26 ± 3	95 ± 3	25 ± 4	37 ± 3
Peak: 90th pct.	**12** ± 2	15 ± 3	18 ± 2	83 ± 4	19 ± 4	34 ± 2
Peak: median	8.1 ± 0.3	**7.7** ± 0.2	**7.8** ± 0.2	47.4 ± 4.8	8.7 ± 0.4	19.6 ± 2.8
Overall mean	**5.4** ± 0.6	**5.6** ± 0.5	6.1 ± 0.4	22.2 ± 3.1	**5.4** ± 0.3	13.1 ± 0.9
Overall median	5.1 ± 0.5	**4.7** ± 0.3	**4.6** ± 0.4	20.8 ± 3.6	**4.5** ± 0.2	12.1 ± 1.0
Autoencoder peak[a]	**8.8** ± 0.1	27.9 ± 1.5	27.5 ± 4.1	25.8 ± 6.4	–	–
Inference time (s)	**0.039**	**0.039**	0.117	0.061	0.345	0.536

[a] Relative f-error of the autoencoder reconstruction alone, on train and test trajectories.

errors. Uniform CABLE has greater variation in its maximum error than the U-Net, although this variation is primarily toward lower errors. Its poorer seeds are comparable to the U-Net, while several seeds perform substantially better, e.g. the best uniform CABLE run had a peak error of 38.51%.

Inference cost. Both CABLE variants also have the fastest prediction time, requiring only 39 ms per trajectory compared with 117 ms for PNODE, 345 ms for the U-Net, and 536 ms for the FNO. This is largely because the scheduling gate allows the mixture weights to be precomputed rather than re-evaluated at each timestep of the rollout. GPLaSDI is competitive in speed at 61 ms, but remains the least accurate model, with a seed-averaged maximum error of 116%.

Adaptive sampling. Adaptive sampling substantially reduces both the magnitude and variability of the tail errors, with seed-averaged maximum relative f-error decreasing from 50% with uniform sampling to 26% with adaptive sampling. The 99th, 95th, and 90th-percentile errors also decrease from 31%, 21%, and 15% to 20%, 14%, and 12%, while the overall mean and median errors change by less than one percentage point, demonstrating that adaptive sampling primarily reshapes the tail of the error distribution without sacrificing typical-case accuracy. Figure 3 shows errors for each temperature, demonstrating that every model except adaptive CABLE concentrates its largest errors in the low-T region, where cold beams produce the fastest-changing dynamics. GPLaSDI is excluded from the figure for visibility but follows the same pattern. Figure 3 (center) also demonstrates that the criterion (15) concentrates samples exactly in this low-T region, so the accuracy gain comes from redistributing a fixed simulation budget rather than additional samples. Figure 2 shows a representative adaptive CABLE prediction. Example latent trajectories and weights are provided in Figure 8 (Appendix B.5), showing that the dominant expert changes over time.

7 Discussion

CABLE models latent dynamics as a convex mixture of shared affine experts whose weights come from a neural gate. This provides an expressive latent dynamics model while keeping the operators explicit. We leverage this explicit structure to bound our model error, enforce structure on the learned dynamics, and develop a model-specific adaptive sampling criterion. Our results show that CABLE is far more expressive than equation-based models which use a single affine ODE, while maintaining explicit structure which is not available when applying NODE. A natural direction is to impose other structural conditions on the mixture beyond contraction, using the same explicit operators. Additionally, we have not done a comprehensive exploration of neural gate architectures.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on experimental methodology, assist with the implementation of experimental methods, and verify mathematical proofs. After theauthors formulated and proved Theorem 1, generative AI tools were used to check the proof. They also suggested the common-Lyapunov result stated as Theorem 2 and introduced us to relevant literature on stability through a common Lyapunov function. The authors independently verified the result and wrote its proof. The authors then independently verified this result and wrote its proof. We did not use generative AI tools to develop the underlying theoretical model or conceptual framework, to formulate Theorem 1 or Proposition 1, to propose or refine hypotheses, generate simulation or synthetic datasets, clean or reformat datasets, or interpret experimental results. Translation and qualitative or thematic data analysis were not applicable to this work.

Additionally, we used generative AI tools to generate and edit experimental code, draft and edit portions of the manuscript, proofread the manuscript, and identify potentially relevant literature. We reviewed all AI-assisted work. All AI-assisted manuscript text and mathematical arguments were independently checked by the authors against our derivations and the relevant literature. AI-generated code was reviewed, executed, and tested for correctness by the authors. Literature suggested by generative AI was independently located and read by the authors to verify its existence, relevance, and accurate representation. We take responsibility for the final content of this work, including all text, claims, code, results, and other artifacts produced with the aid of generative AI.

REFERENCES

G. Ackerson and K. Fu. On state estimation in switching environments. *IEEE transactions on automatic control*, 15(1):10–17, 1970.

W. Anderson, S. W. Chung, R. Stephany, and Y. Choi. mlasdi: Multi-stage latent space dynamics identification. *arXiv preprint arXiv:2506.09207*, 2025.

P. Apkarian, P. Gahinet, and G. Becker. Self-scheduled H_∞ control of linear parameter-varying systems: a design example. *Automatica*, 31(9):1251–1261, 1995.

C. Bonneville, Y. Choi, D. Ghosh, and J.L. Belof. Gplasdi: Gaussian process-based interpretable latent space dynamics identification through deep autoencoder. *Computer Methods in Applied Mechanics and Engineering*, 418:116535, 2024. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2023.116535.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

M. Calder, C. Craig, D. Culley, R. de Cani, C. A. Donnelly, R. Douglas, B. Edmonds, J. Gascoigne, N. Gilbert, C. Hargrove, F. Hinds, D. C. Lane, D. Mitchell, G. Pavey, D. Robertson, B. Rosewell, S. Sherwin, M. Walport, and A. Wilson. Computational modelling for decision-making: where, why, what, who and how. *Royal Society Open Science*, 5(6):172096, 2018. doi: 10.1098/rsos.172096.

K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

J. Cheng, R. Wang, and I. R. Manchester. Learning stable and passive neural differential equations. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pp. 7859–7864. IEEE, 2024.

S. W. Chung, C. Miller, Y. Choi, P. Tranquilli, H. K. Springer, and K. Sullivan. Latent space dynamics identification for interface tracking with application to shock-induced pore collapse. *arXiv preprint arXiv:2507.10647*, 2025.

R. M. Cummings, W. H. Mason, S. A. Morton, and D. R. McDaniel. *Applied computational aerodynamics: A modern engineering approach*, volume 53. Cambridge University Press, 2015.

E.K. de Jong, N. Gunawardena, J.E. Katona, H. Beydoun, D. Ghosh, and P. Caldwell. Data-driven reduced order modeling for warm rain microphysics. *Journal of Geophysical Research: Machine Learning and Computation*, 3(3):e2025JH001103, 2026.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

W.D. Fries, X. He, and Y. Choi. Lasdi: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 399:115436, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115436.

X. He, Y. Choi, W.D. Fries, J.L. Belof, and J. Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 489:112267, 2023. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2023.112267.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

C. Hoffmann and H. Werner. A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations. *IEEE Transactions on Control Systems Technology*, 23 (2):416–433, 2014.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020. ISSN 1755-5817. doi: https://doi.org/10.1016/j.cirpj.2020.02.002.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

M. Karl, M. Soelch, J. Bayer, and P. Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017.

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

J. Z. Kolter and G. Manek. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32, 2019.

J. Kossaifi, N. Kovachki, Z. Li, D. Pitt, M. Liu-Schiaffini, R. J. George, B. Bonev, K. Azizzadenesheli, J. Berner, V. Duruisseaux, and A. Anandkumar. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

K. Lee and E. J. Parish. Parameterized neural ordinary differential equations: applications to computational physics problems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2253):20210162, 2021. ISSN 1364-5021. doi: 10.1098/rspa.2021.0162.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

S. W. Linderman, A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

J. Mohammadpour and C. W. Scherer. *Control of linear parameter varying systems with applications*. Springer Science & Business Media, 2012.

N. Mudrik, Y. Chen, E. Yezerets, C. J. Rozell, and A. S. Charles. Decomposed linear dynamical systems (dlds) for learning the latent components of neural dynamics. *Journal of Machine Learning Research*, 25(59):1–44, 2024.

J. Nam, W. Anderson, Y. Choi, H. P. Le, M. E. Foord, B. I. Cho, H. Jeong, and M. S. Cho. Physics-informed latent space dynamics identification for time-dependent nlte atomic kinetics. *arXiv preprint arXiv:2604.16664*, 2026.

D. Noble. The rise of computational biology. *Nature Reviews Molecular Cell Biology*, 3(6):459–463, 2002.

A. Pavlov, A. Pogromsky, N. Van De Wouw, and H. Nijmeijer. On convergence properties of piecewise affine systems. *International Journal of Control*, 80(8):1233–1247, 2007.

L. F. Ricketson and J. Hu. An explicit, energy-conserving particle-in-cell scheme. *Journal of Computational Physics*, 537:114098, 2025. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2025.114098.

C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby. Scaling vision with sparse mixture of experts. *Advances in neural information processing systems*, 34:8583–8595, 2021.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

F. J. Roth, D. K. Klein, M. Kannapinn, J. Peters, and O. Weeger. Stable port-hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 38:50737–50762, 2025.

J. S. Shamma. An overview of lpv systems. *Control of linear parameter varying systems with applications*, pp. 3–26, 2012.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

R. Stephany, W. Anderson, and Y. Choi. Higher-order lasdi: Reduced order modeling with multiple time derivatives. *Computer Methods in Applied Mechanics and Engineering*, 455:118890, 2026.

J. Thijssen. *Computational physics*. Cambridge university press, 2007.

A. Tran, X. He, D. A. Messenger, Y. Choi, and D. M. Bortz. Weak-form latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 427:116998, 2024.

D. Vasileska, S. M. Goodnick, and G. Klimeck. *Computational Electronics: semiclassical and quantum device modeling and simulation*. CRC press, 2017.

F. Wu, X. H. Yang, A. Packard, and G. Becker. Induced l2-norm control for lpv systems with bounded parameter variation rates. *International Journal of Robust and Nonlinear Control*, 6 (9-10):983–998, 1996.

J. Yoo, J. Hu, and L. F. Ricketson. An explicit energy-conserving particle method for the Vlasov-Fokker-Planck equation. *arXiv preprint arXiv:2510.03960*, 2025.

S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

Y. Zhou, Y. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, and J. Laudon. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35: 7103–7114, 2022.

A Implementation details

In order to improve the performance of CABLE, there are several additional implementation details, which we discuss in detail here.

Sequential hard-thresholding. Following the sequential threshold least-squares used with SINDy (Brunton et al., 2016; Champion et al., 2019), every N_{Thresh} epochs we set expert coefficients below a threshold to zero. A truncated coefficient is masked for the rest of training and receives no further gradient information. This sparsifies the library and improves rollout stability. We apply thresholding only after loss has decreased so that the model has learned a reasonable representation of the training data.

Over-provisioning experts. Training cost is dominated by the autoencoder, and the mixed coefficients (6) are computed in parallel across trajectories, snapshots, and experts on GPU, so extra experts add expressive power at almost no cost for the small latent systems here (Table 3). After training, most experts are thresholded away or ignored by the gate, so it is safer to over-provision than to start with too few.

Expert initialization. We initialize each expert's coefficient matrix and bias by drawing samples from a normal distribution $\mathcal{N}(0, 10^{-4})$.

Dynamics-loss warmup. Training with loss (9) sometimes fails when the dynamics model cannot fit the latent trajectories of a randomly initialized autoencoder, and the autoencoder/dynamics model collapse to produce constant latent trajectories. When this happens, we ramp the dynamics weight as $\beta_1 \cdot \min(1, i/n_{\text{warmup}})$ at iteration i. It may also help to first pretrain the autoencoder (without dynamics loss) until the reconstruction is reasonably accurate, which we do for our Vlasov–Fokker–Planck problem (17).

Rollout fine-tuning. Derivative matching alone does not guarantee accurate rollouts, which is the residual ε_{Dyn} in (13). We find that training with instantaneous derivative-matching often produces model rollouts which are close to the true encoded trajectories, but may drift slightly over time. We therefore add a short fine-tuning stage after the main training run, freezing the autoencoder and training only the dynamics parameters. By reducing rollout training to only a fine-tuning stage, we avoid expensive rollout training during our minimization of the CABLE loss (9).

Let $\mathbf{\Phi}$ denote one Runge-Kutta-4 step. From an encoded state $\mathbf{z}_s^{(i)}$, we form the H-step rollout $\hat{\mathbf{z}}_{s,0}^{(i)} = \mathbf{z}_s^{(i)}$, where we define $\hat{\mathbf{z}}_{s,h+1}^{(i)} = \mathbf{\Phi}(\hat{\mathbf{z}}_{s,h}^{(i)}, t_{s+h}, \boldsymbol{\mu}^{(i)})$, and minimize

$$\mathcal{L}_{\text{roll}} = \frac{1}{N_\mu (N_t - H + 1) H} \sum_{i=1}^{N_\mu} \sum_{s=0}^{N_t - H} \sum_{h=1}^{H} \left\| \hat{\mathbf{z}}_{s,h}^{(i)} - \mathbf{z}_{s+h}^{(i)} \right\|^2. \tag{18}$$

In practice, we detach gradients between successive rollout steps, so this fine-tuning stage avoids full backpropagation through time while still exposing the model to states generated by its own dynamics. Additionally, we reduce the learning rate by a factor of 10 compared to our initial training. The final fine-tuning objective is

$$\mathcal{L}_{\text{Finetune}} = \mathcal{L}_{\text{roll}} + \beta_2 \frac{1}{K(N_z + 1)N_z} \sum_{k=1}^{K} \|\Xi_k\|^2, \tag{19}$$

where we use the same β_2 value as our original loss (9). Even short rollout fine-tuning periods can significantly reduce latent-dynamics drift, especially for oscillatory systems.

B Experiment details

All models use the Adam optimizer (Kingma & Ba, 2014). Table 2 summarizes the non-Vlasov experiments. Unless noted otherwise when training CABLE with loss (9), we use $\beta_1 = 0.1$, $\beta_2 = (\Delta t)^2$, a learning rate of 10^{-3}, a hard-thresholding value of 10^{-3}, and Softplus activations in

Table 2: Configuration of the non-Vlasov experiments. Network architectures list layer widths from input to output. Autoencoder architectures shown are the encoder architecture, and the decoders have the reversed architecture.

	Heat forcing (21)	Van der Pol (16)	Relaxation (56)	1D Burgers' (22)
Autoencoder	64–32–16–10	None	64–32–16–10	128–64–32–10
Gate inputs	$(t, g(t))$	$(x, y; \mu)$	$(c, q(t))$	(t, μ)
Gate MLP	2–128–128–K	3–64–64–10	2–64–64–20	2–128–128–100
Experts K	[1, 2, 5, 10, 50, 100]	10	20	100
Training epochs	10,000	10,000	10,000	10,000
Rollout fine-tuning	—	100 epochs, $H = 10$	—	—
(β_1, β_2)	$(0.1, 10^{-4})$	$(1, 0)$	$(0.1, 10^{-4})$	$(0.1, 10^{-4})$

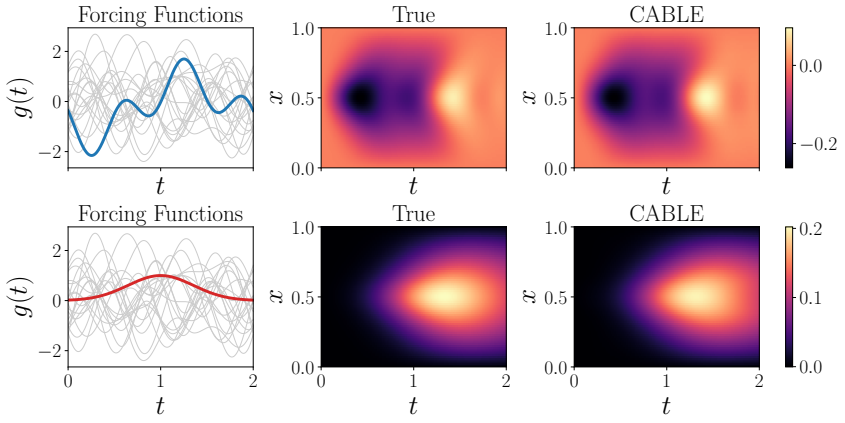

Figure 4: CABLE predictions for forced heat equation (21) with a scheduling gate $\mathbf{w}(t, g(t))$ and $K = 100$ experts. (Top row) In-distribution test forcing function. (Bottom row) Out-of-distribution forcing. Gray curves in the left column are the training forcings.

the autoencoder. We always normalize time as an input to the gate, by feeding the gate snapshots of $\tau(t) = t/t_f$. When reporting errors we use pointwise relative ℓ_2-error between a trajectory $\mathbf{u}$ and the approximation $\hat{\mathbf{u}}$

$$\varepsilon_j^i := \frac{\|\mathbf{u}(t_j; \boldsymbol{\mu}^{(i)}) - \hat{\mathbf{u}}(t_j; \boldsymbol{\mu}^{(i)})\|}{\|\mathbf{u}(t_j; \boldsymbol{\mu}^{(i)})\|} \tag{20}$$

B.1 Forced heat equation

We consider the forced heat equation

$$u_t = 0.1u_{xx} + g(t)e^{-100(x-0.5)^2}, \quad x \in (0,1),\ t \in (0,2], \quad u(x,0) = u(0,t) = u(1,t) = 0. \tag{21}$$

Forcing functions $g(t)$ are drawn from the low-frequency family

$$g^{(i)}(t) = a_0^{(i)} + \sum_{k=1}^{3} [a_k^{(i)} \sin(k\pi t) + b_k^{(i)} \cos(k\pi t)],$$

with $a_0^{(i)} \sim U(-0.5, 0.5)$ and $a_k^{(i)}, b_k^{(i)} \sim U(-1, 1)$. We generate 20 forcings for training, 10 for in-distribution testing, and additionally test on the out-of-distribution forcing $g_{\text{OOD}}(t) = e^{-4(t-1)^2}$. We discretize the PDE with 64 spatial nodes and 201 snapshots to generate data. To normalize solutions we divide the data by its standard deviation, and normalize $g(t)$ to $[0, 1]$. We threshold coefficients below 10^{-3} to zero after 8,000 epochs, and train for 2,000 additional epochs after thresholding. For this problem, the solution norm can be near zero, so we normalize pointwise errors by $\max_j \|\mathbf{u}_j^{(i)}\|$ along each trajectory. Figure 4 shows CABLE predictions for in and out-of-distribution forcing functions.

Table 3: Full ablation over the initial number of experts K for the forced heat equation (21) over 5 seeds. "Peak test max" is the worst single-snapshot relative error over all held-out trajectories, "mean test max" averages each trajectory's peak-in-time error, and "engaged" is the mean number of experts whose gate weight exceeds 10^{-3} somewhere on a test trajectory. Values report the mean and standard deviation across seeds.

K	Test mean (%)	Peak test max (%)	Mean test max (%)	OOD max (%)	Engaged	s/epoch
1	44.24 $\pm$0.29	114.72 $\pm$4.39	101.80 $\pm$1.37	93.54 $\pm$2.34	1 $\pm$0	**0.012**
2	2.92 $\pm$0.69	9.16 $\pm$2.21	6.59 $\pm$1.93	4.51 $\pm$1.93	2 $\pm$0	**0.012**
5	2.71 $\pm$0.89	9.03 $\pm$2.96	6.50 $\pm$2.35	4.69 $\pm$2.11	2 $\pm$0	**0.012**
10	2.72 $\pm$0.47	8.91 $\pm$1.89	6.05 $\pm$1.28	3.99 $\pm$0.93	2 $\pm$1	**0.012**
50	1.98 $\pm$0.57	6.94 $\pm$1.77	4.65 $\pm$1.49	2.72 $\pm$1.03	5 $\pm$3	**0.012**
100	**1.91** $\pm$0.52	**6.70** $\pm$1.60	**4.48** $\pm$1.38	**2.64** $\pm$1.15	7 $\pm$1	**0.012**

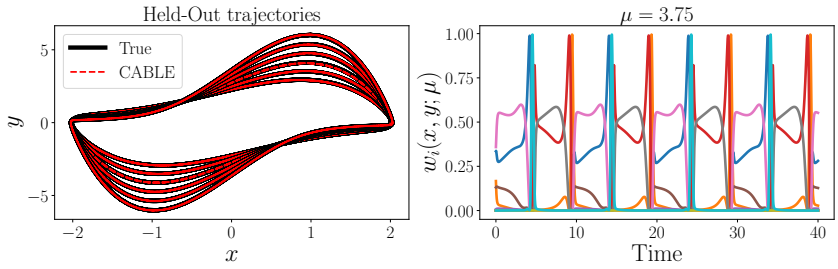

Figure 5: CABLE predictions for the Van der Pol oscillator on held-out parameters μ, using a state-dependent gate $\mathbf{w}(x, y; \mu)$ with $K = 10$ affine experts. CABLE is trained to $t = 20$ and extrapolated to $t = 40$.

B.2 Heat relaxation

The heat relaxation problem is discussed in detail with the stability criteria in Appendix D.2.

B.3 Van der Pol

Each trajectory is integrated for 20 time units with 2001 snapshots. The extrapolation experiment continues to $t = 40$ using the same timestep $\Delta t = 0.01$. For this problem there is no autoencoder, the experts directly parameterize the vector field, so shrinking their coefficients would bias the model away from the true operator. Thus, we set loss hyperparameters in (9) to $\beta_1 = 1$ and $\beta_2 = 0$.

B.4 Viscous 1D Burgers'

We consider the viscous Burgers' equation

$$u_t + u\,u_x = 0.005\,u_{xx}, \qquad x \in (0, 1), \quad t \in (0, 1], \qquad u(0, t) = u(1, t) = 0, \tag{22}$$

with parameter-dependent initial condition $u(x, 0) = A(\mu)e^{-100(x-0.5)^2}$, where the amplitude is given by $A(\mu) = 0.25 + 1.25e^{-100(\mu-0.5)^2}$.

We discretize (22) using 128 interior spatial nodes and 101 snapshots over $t \in [0, 1]$. We evaluate the sampling strategies on a candidate set of 51 uniformly spaced parameters over $\mu \in [0, 1]$, and a testing set consisting of 50 midpoints between the candidate set points. CABLE uses a scheduling gate $\mathbf{w}(t, \mu)$, with the remaining model and training settings summarized in Table 2. An autoencoder is pretrained on the 51 uniformly spaced candidate μ-values. Both uniform and adaptive sampling begin from the five parameters $\mu \in \{0, 0.25, 0.5, 0.75, 1\}$, and grow the training set to 11 samples. After the initial latent-dynamics training stage, each newly selected sample is added by warm-starting from the previous model and training for 2,000 additional epochs, with hard-thresholding of the coefficients applied midway through each update using a threshold of 10^{-3}.

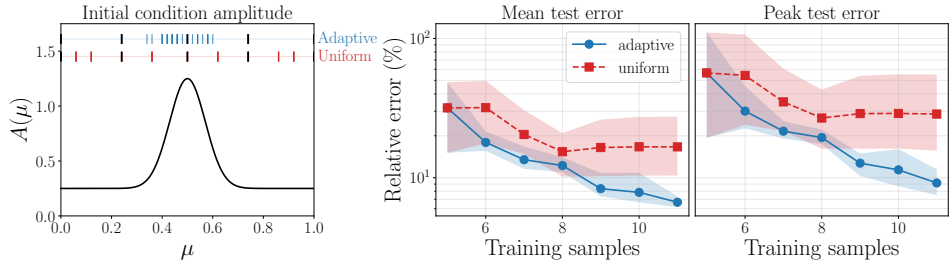

Figure 6: CABLE with adaptive versus uniform sampling for the viscous Burgers' problem (22). (Left) Initial-condition amplitude $A(\mu)$ and the parameters selected by each strategy, with black marks denoting the five shared initial samples. (Right) Mean and peak test errors over 5 seeds as the training set grows. Lines indicate mean over the seeds, and shaded regions indicate min/max values.

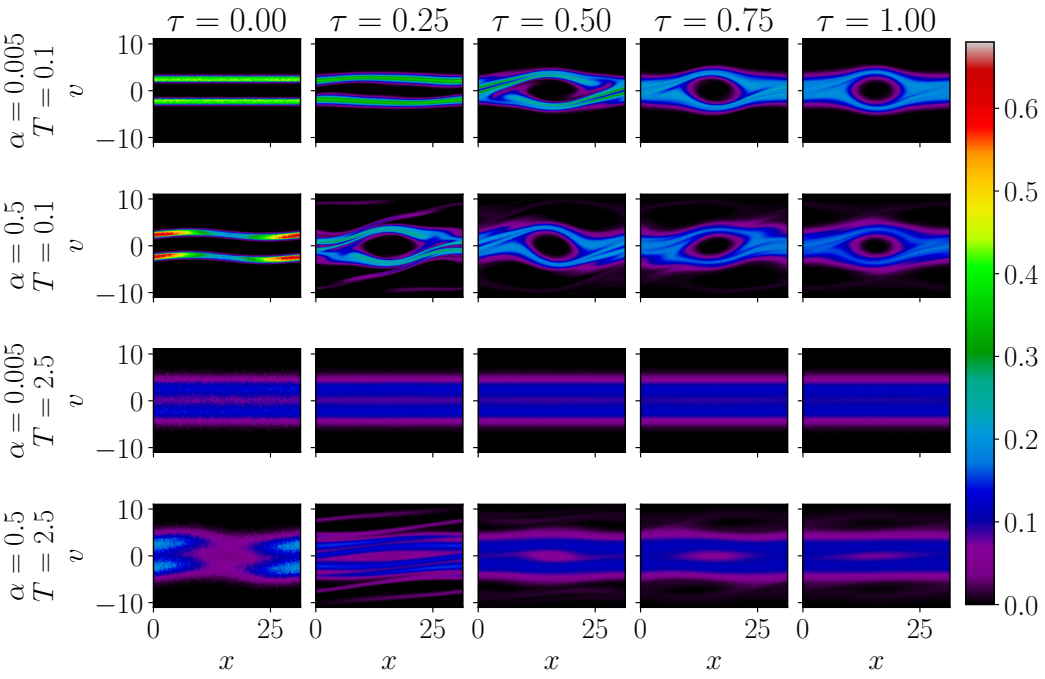

Figure 7: Full-order snapshots of the two-stream problem (17) at corners of the parameter domain, spanning nearly steady beams, vortex formation, oscillation, and filamentation. Snapshots are in normalized time τ.

B.5 Vlasov–Fokker–Planck

Governing equations. The collision operator in (17) is

$$\mathcal{Q}(f) = \partial_v[\mathcal{T}(t,x)\,\partial_v f + (v - v_{\text{bulk}}(t,x))f],$$

where, with mass density $n = \int_{\mathbb{R}} f\,\mathrm{d}v$, the bulk velocity and temperature are $v_{\text{bulk}} = n^{-1}\int_{\mathbb{R}} fv\,\mathrm{d}v$ and $\mathcal{T} = n^{-1}\int_{\mathbb{R}} f|v - v_{\text{bulk}}|^2\,\mathrm{d}v$ (see Yoo et al. (2025)).

Data and protocol. We generate data with an explicit particle-in-cell method (Yoo et al., 2025; Ricketson & Hu, 2025) using 100 spatial cells and 5,000 particles per cell. We evolve to $t = 50$ and take snapshots every 0.2 time units. We interpolate the particles onto a uniform 200×200 phase-space grid which leaves visible particle noise in the true data (Figure 7). Trajectories are generated on a uniform 21×21 grid over (α, T), with a single run taking 1000–1600 seconds.

All three reduced-order models (CABLE, PNODE, GPLaSDI) share the same CNN autoencoder architecture, each trained with its own autoencoder, while the U-Net and FNO are standalone operators that predict the phase field directly. The CNN encoder uses five stride-2 convolutional layers with 3×3 kernels and channel widths $32, 64, 128, 256, 512$ and Softplus activations. This reduces a 200×200 FOM snapshot to a $7 \times 7 \times 512$ feature map that is flattened and passed through an MLP of shape 25088–512–20. The decoder mirrors this with transpose convolutions, and its output is passed through Softplus and rescaled to preserve the initial mass. Both CABLE models use a scheduling gate $\mathbf{w}(t, \alpha, T)$ given by a two-hidden-layer MLP of width 128 with Tanh activations, initialized with $K = 100$ experts and hard-thresholded at magnitude 10^{-3}.

PNODE learns $\dot{\mathbf{z}} = \mathbf{f}_\theta(\mathbf{z}, t, \alpha, T)$ with an MLP using architecture 23–128–128–128–20 and GELU activations, giving a comparable dynamics-parameter count to CABLE (Table 4). GPLaSDI instead fits an affine latent ODE per-trajectory and interpolates the coefficients with Gaussian processes (Bonneville et al., 2024). The U-Net uses a convolutional backbone similar to the CNN autoencoder, but with bilinear resampling, skip connections, GroupNorm, and ReLU activations. The FNO uses the neuraloperator package (Kossaifi et al., 2025; Li et al., 2020) with 16×16 retained modes, hidden width 64, and eight Fourier layers. Both operator models learn the mapping

$$\mathcal{G} : \big(\mathbf{f}_0(\alpha, T), t, \alpha, T\big) \mapsto \mathbf{f}(t; \alpha, T). \tag{23}$$

Both operator outputs are also Softplus-rescaled for non-negativity and mass preservation.

Training. Table 4 summarizes the training costs and parameter counts of all models. All models are trained on 8 AMD MI300A GPUs, with input parameters normalized to $[0, 1]$ and the training data divided by its standard deviation. Each method uses the same budget of 49 full-order trajectories. The uniformly sampled models use a fixed 7×7 subgrid of the full 21×21 parameter grid (see Figure 3 (right)). Adaptive CABLE begins from the 24 boundary points of this subgrid together with the center point and greedily selects 24 additional trajectories using the acquisition criterion (15). The fixed-grid experiments therefore compare the surrogate methods under the same simulation budget, while the adaptive-versus-uniform comparison evaluates the efficacy of our adaptive sampling criterion under the same full-order simulation budget.

The ROMs are all trained with a learning rate of 10^{-3}. We first pretrain the autoencoder for each seed so that reconstruction loss drops below 10^{-1}. Then we begin training jointly with the dynamics models, ramping the dynamics loss for CABLE and PNODE over 5,000 epochs.

CABLE is trained with loss (9). During adaptive sampling, each newly selected trajectory is incorporated by warm-starting the previous model and re-densifying the coefficient mask. We then train for $1,000$ epochs, apply hard-thresholding, and train for an additional $1,000$ epochs. Each adaptive round concludes with $1,000$ iterations of 10-step RK4 rollout fine-tuning using loss (19). For the rollout stage, the learning rate is reduced by a factor of ten to 10^{-4}. The rollout fine-tuning takes less than one minute per round and reduces the 10-step rollout error by as much as 50% in this experiment. The uniformly sampled CABLE model uses the same architecture and objective on the fixed 7×7 training grid. Hard-thresholding is only applied to uniform CABLE once at epoch 20,000. Uniform CABLE also receives 1,000 epochs of 10-step rollout finetuning.

PNODE is similarly trained with derivative-matching loss and rollout finetuning. The derivative-matching loss is also given a warmup of 5,000 epochs, and we finetune with 1,000 epochs of rollout loss after initial training.

GPLaSDI is trained for $20,000$ epochs and did not require any dynamics warmup. During training, the autoencoder is trained jointly with the trajectory-wise affine latent dynamics. We note that GPLaSDI does not get the rollout stage or SINDy warmup. This is consistent with the original algorithm (Bonneville et al., 2024), and additionally GPLaSDI coefficients are determined for each trajectory, over the entire trajectory. Thus, the discovered ODE is already fit to minimize error over the entire trajectory.

The U-Net and FNO are trained by direct regression using a learning rate of 10^{-4} and minimizing mean-squared error between the predicted and target phase fields. A larger learning rate of 10^{-3} led to unstable training.

Table 4: Model complexity and training cost for the two-stream problem (17), averaged over 5 seeds. All reduced-order models use the same CNN autoencoder and a budget of 49 full-order trajectories. Training used 8 AMD MI300A GPUs, capped at five hours of wall-clock time.

Model	Epochs	Backbone params	Dynamics params	Train time (s)
CABLE (adaptive)	$55,000$[a]	$28,872,223$	$33,081$[b]	$12,125$
CABLE (uniform)	$55,000$	$28,872,223$	$32,998$[b]	$14,531$
GPLaSDI	$20,000$	$28,872,223$	$20,580$	$14,854$
PNODE	$55,000$	$28,872,223$	$38,676$	$15,149$
U-Net	948	$31,044,097$	—	$18,000$
FNO	985	$4,803,521$	—	$18,000$

[a] Distributed across the adaptive sampling rounds, each warm-started from the previous model.
[b] Gate parameters plus the nonzero expert coefficients remaining after hard-thresholding.

Table 5: Two-stream instability (17) relative f-error (%) and inference cost over all 441 parameter cells, comparing gates with our softmax final layer (CABLE) and a linear final layer (CABLE, unconstrained). "Peak" is the maximum error within a trajectory, then aggregated over all cells. Both models use the same 49-trajectory budget and results are calculated over 5 seeds, with $\pm$ denoting standard deviation over the seeds. Best results in **bold**.

	CABLE		CABLE, unconstrained	
	Adaptive	Uniform	Adaptive	Uniform
Peak: maximum	**26** ± 3	50 ± 9	54 ± 6	66 ± 9
Peak: 99th pct.	**20** ± 5	31 ± 4	26 ± 6	49 ± 5
Peak: 95th pct.	**14** ± 3	21 ± 2	16 ± 1	25 ± 2
Peak: 90th pct.	**12** ± 2	15 ± 3	**12** ± 1	17 ± 1
Peak: median	8.1 ± 0.3	**7.7** ± 0.2	8.0 ± 0.1	**7.8** ± 0.1
Overall mean	**5.4** ± 0.6	**5.6** ± 0.5	**5.3** ± 0.2	6.1 ± 0.3
Overall median	**5.1** ± 0.5	**4.7** ± 0.3	**4.8** ± 0.2	**4.7** ± 0.3
Autoencoder peak[a]	**8.8** ± 0.1	27.9 ± 1.5	**8.8** ± 0.1	26.2 ± 4.9
Inference time (s)	**0.039**	**0.039**	**0.039**	**0.039**

[a] Relative f-error of the autoencoder reconstruction alone, evaluated on the train and test trajectories.

In Table 5 we train CABLE using the same procedure described above, but with a linear layer on the last gate instead of Softmax. These results emphasize that our assumption that the weights are convex significantly improves the tail prediction errors for this problem.

C Proofs of theoretical results

In this appendix we provide proofs for our theoretical results. We also derive tighter, path-wise bounds for the scheduling gate and show that this bound more faithfully tracks the true latent error for our Vlasov experiment.

C.1 Proof of Lemma 1

Before proving our error bound for Theorem 1, we require the following Lemma.

Lemma 1 (Convex mixtures). *Let $M_1, \dots, M_K \in \mathbb{R}^{m\times n}$, and $A_1, \dots, A_K \in \mathbb{R}^{N_z\times N_z}$. Then,*

(i) For every $\mathbf{w} \in \mathcal{W}^K$,

$$\Big\|\sum_k w_k M_k\Big\| \le \max_k \|M_k\|, \qquad \text{and} \qquad \omega_2\Big(\sum_k w_k A_k\Big) \le \max_k \omega_2(A_k). \tag{24}$$

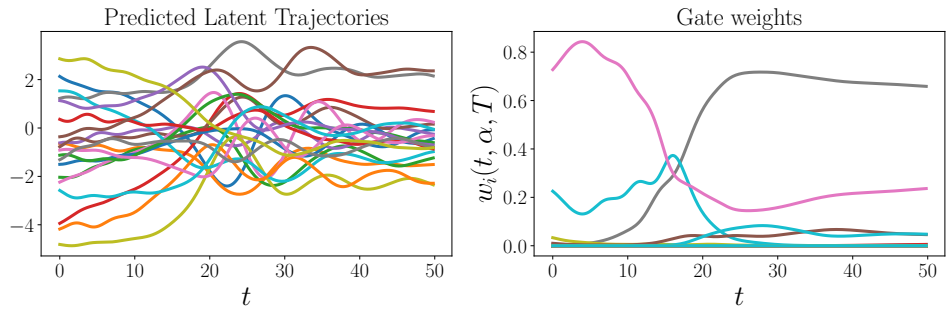

Figure 8: Predicted latent trajectories and gate weights for the adaptive parameter $(\alpha, T) = (0.005, 0.58)$. The latent trajectory turns sharply near $t \approx 15$, when the vortex starts to form, while the gate weights vary smoothly, and several experts remain engaged throughout.

(ii) For every perturbation to the weights $\Delta\mathbf{w} \in \mathbb{R}^K$ with $\sum_k \Delta w_k = 0$,

$$\Big\|\sum_k \Delta w_k M_k\Big\| \le \|\Delta\mathbf{w}\|_1 R_M, \tag{25}$$

where $\|\cdot\|_1$ is the ℓ_1-norm and $R_M := \min_C \max_k \|M_k - C\|$ is the Chebyshev radius of M_k.

Proof. (i) The first inequality follows from the triangle inequality and our simplex constraint on the weights,

$$\|\sum_k w_k M_k\| \le \sum_k w_k \|M_k\| \le \max_k \|M_k\|.$$

For the second inequality, Weyl's inequality makes ω_2 convex, so

$$\omega_2\Big(\sum_k w_k A_k\Big) \le \sum_k \omega_2(w_k A_k) = \sum_k w_k \omega_2(A_k) \le \max_k \omega_2(A_k).$$

(ii) Since the entries of $\Delta\mathbf{w}$ sum to zero, for any $C \in \mathbb{R}^{m\times n}$,

$$\Big\|\sum_{k=1}^K \Delta w_k M_k\Big\| = \Big\|\sum_{k=1}^K \Delta w_k (M_k - C)\Big\| \le \sum_{k=1}^K |\Delta w_k| \|(M_k - C)\| \le \|\Delta\mathbf{w}\|_1 \max_k \|M_k - C\|.$$

Taking the infimum over C gives the Chebyshev radius.

□

Applying Lemma 1(i) to the CABLE latent vector field gives $\|A(\mathbf{w})\mathbf{z} + \mathbf{b}(\mathbf{w})\| \le \max_k \|A_k\| \, \|\mathbf{z}\| + \max_k \|\mathbf{b}_k\|$, so the vector field grows at most linearly and latent trajectories exist for all time.

C.2 Proof of Theorem 1

Proof. We define the latent model error $\boldsymbol{\varepsilon}_z(t) := \mathbf{z}(t) - \hat{\mathbf{z}}(t)$. For notational convenience we also define $\mathbf{w} := \mathbf{w}(\mathbf{z}(t), t)$ and $\hat{\mathbf{w}} := \mathbf{w}(\hat{\mathbf{z}}(t), t)$. Differentiating and using the definition of the residual, we have

$$\dot{\boldsymbol{\varepsilon}}_z = \dot{\mathbf{z}} - \hat{\mathbf{f}}(\hat{\mathbf{z}}, t) = \mathbf{r}(t) + \hat{\mathbf{f}}(\mathbf{z}, t) - \hat{\mathbf{f}}(\hat{\mathbf{z}}, t).$$

Recalling that $\hat{\mathbf{f}}(\mathbf{z}, t) = \Xi(\mathbf{w})^\top \Theta(\mathbf{z}) = A(\mathbf{w})\mathbf{z} + \mathbf{b}(\mathbf{w})$, we split the vector-field difference into a part where the operator is held fixed and a part coming from the gate,

$$\begin{aligned}\hat{\mathbf{f}}(\mathbf{z}, t) - \hat{\mathbf{f}}(\hat{\mathbf{z}}, t) &= \big[\Xi(\mathbf{w})^\top \Theta(\mathbf{z}) - \Xi(\mathbf{w})^\top \Theta(\hat{\mathbf{z}})\big] + \big[\Xi(\mathbf{w}) - \Xi(\hat{\mathbf{w}})\big]^\top \Theta(\hat{\mathbf{z}}) \\ &= A(\mathbf{w})\boldsymbol{\varepsilon}_z + \big[\Xi(\mathbf{w}) - \Xi(\hat{\mathbf{w}})\big]^\top \Theta(\hat{\mathbf{z}}) \\ &= A(\mathbf{w})\boldsymbol{\varepsilon}_z + \mathbf{h},\end{aligned} \tag{26}$$

where the first bracket collapses to $A(\mathbf{w})\boldsymbol{\varepsilon}_z$ because the experts are affine and the constant parts cancel, and we define $\mathbf{h}$ as the second term. We note that $\Delta\mathbf{w} = \mathbf{w} - \hat{\mathbf{w}}$ sums to zero since both vectors of weights lie in the simplex. Thus, Lemma 1(ii) together with Lipschitz continuity of the gate gives

$$\|\mathbf{h}\| \le \|\Xi(\mathbf{w}) - \Xi(\hat{\mathbf{w}})\|\|\Theta(\hat{\mathbf{z}})\| \le \|\mathbf{w} - \hat{\mathbf{w}}\|_1 R_\Xi \|\Theta(\hat{\mathbf{z}})\| \le M_\Theta R_\Xi L_w \|\boldsymbol{\varepsilon}_z\|. \tag{27}$$

Using the logarithmic-norm inequality and the Cauchy–Schwarz inequality, we can show $\frac{\mathrm{d}}{\mathrm{d}t}\|\boldsymbol{\varepsilon}_z\| \le \omega_2(A(\mathbf{w}))\|\boldsymbol{\varepsilon}_z\| + \|\mathbf{r}\| + \|\mathbf{h}\|$. So, using Lemma 1(i) and (27),

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\boldsymbol{\varepsilon}_z\| \le \big(\nu + M_\Theta R_\Xi L_w\big)\|\boldsymbol{\varepsilon}_z\| + \|\mathbf{r}\| = \nu_{\mathrm{eff}}\|\boldsymbol{\varepsilon}_z\| + \|\mathbf{r}\|. \tag{28}$$

Since $\boldsymbol{\varepsilon}_z(0) = 0$, Gronwall's inequality gives

$$\|\boldsymbol{\varepsilon}_z(t)\| \le \int_0^t e^{\nu_{\mathrm{eff}}(t-s)}\|\mathbf{r}(s)\|\,\mathrm{d}s. \tag{29}$$

Finally,

$$\|\mathbf{u}(t) - \hat{\mathbf{u}}(t)\| \le \|\mathbf{u}(t) - \mathcal{G}_{\mathrm{dec}}(\mathbf{z}(t))\| + \|\mathcal{G}_{\mathrm{dec}}(\mathbf{z}(t)) - \mathcal{G}_{\mathrm{dec}}(\hat{\mathbf{z}}(t))\| \le \varepsilon_{\mathrm{AE}} + L_D\|\boldsymbol{\varepsilon}_z(t)\|, \tag{30}$$

and substituting the Gronwall estimate, then bounding $\|\mathbf{r}(s)\| \le \varepsilon_{\mathrm{Dyn}}$, gives our bound (13). □

Remark 1 (Nonlinear expert libraries)**.** *The proof uses affineness of the experts (specifically, that the $\mathbf{b}(\mathbf{w})$ terms cancel) at (26) to obtain a logarithmic-norm rate that may be negative. For a general library Θ with Lipschitz constant L_Θ on Ω, the same decomposition yields only $L_f \le L_\Theta \max_k \|\Xi_k\| + M_\Theta R_\Xi L_w$, which is always nonnegative, and generally may provide looser bounds, for example as in the LaSDI analysis of Anderson et al. (2025).*

C.3 Path-wise bounds for scheduling gates

Theorem 1 controls error growth using the worst expert in the library. For a scheduling gate, we can obtain tighter bounds by leveraging that the mixed operator $A(\mathbf{w})$ is independent of the rollout.

Proposition 1 (Path-wise bounds for scheduling gates)**.** *Assume the hypotheses of Theorem 1 and that CABLE uses a scheduling gate. Define the realized operator and its state-transition matrix by*

$$A(t) := \sum_{k=1}^{K} w_k(t;\boldsymbol{\mu})A_k, \qquad \frac{\partial}{\partial t}\Phi(t,s) = A(t)\Phi(t,s), \qquad \Phi(s,s) = I. \tag{31}$$

Then the CABLE prediction error satisfies

$$\|\mathbf{u}(t) - \mathcal{G}_{\mathrm{dec}}\left(\hat{\mathbf{z}}(t)\right)\| \le \varepsilon_{\mathrm{AE}} + L_D \int_0^t \|\Phi(t,s)\mathbf{r}(s)\|\ \mathrm{d}s \tag{32}$$

$$\le \varepsilon_{\mathrm{AE}} + L_D \int_0^t \exp\left(\int_s^t \omega_2\left(A(\tau)\right)\ \mathrm{d}\tau\right)\|\mathbf{r}(s)\|\,\mathrm{d}s \tag{33}$$

$$\le \varepsilon_{\mathrm{AE}} + L_D \int_0^t e^{\nu(t-s)}\|\mathbf{r}(s)\|\,\mathrm{d}s. \tag{34}$$

Proof. For a scheduling gate, the weights depend on time and the fixed input parameters, but not on the evolving latent state. Writing $\mathbf{b}(t) := \sum_{k=1}^{K} w_k(t;\boldsymbol{\mu})\mathbf{b}_k$, by the definition of the dynamics residual we have

$$\dot{\mathbf{z}}(t) = A(t)\mathbf{z}(t) + \mathbf{b}(t) + \mathbf{r}(t),$$

while the predicted trajectory satisfies

$$\dot{\hat{\mathbf{z}}}(t) = A(t)\hat{\mathbf{z}}(t) + \mathbf{b}(t).$$

Subtracting these equations and defining $\boldsymbol{\varepsilon}_z(t) := \mathbf{z}(t) - \hat{\mathbf{z}}(t)$ gives

$$\dot{\boldsymbol{\varepsilon}}_z(t) = A(t)\boldsymbol{\varepsilon}_z(t) + \mathbf{r}(t), \qquad \boldsymbol{\varepsilon}_z(0) = \mathbf{0}. \tag{35}$$

The affine terms cancel because their weights are identical along the two trajectories, which is the key simplification provided by an affine expert library. Variation of parameters applied to (35) gives

$$\boldsymbol{\varepsilon}_z(t) = \Phi(t,0)\boldsymbol{\varepsilon}_z(0) + \int_0^t \Phi(t,s)\mathbf{r}(s)\,\mathrm{d}s = \int_0^t \Phi(t,s)\mathbf{r}(s)\,\mathrm{d}s. \tag{36}$$

Taking norms and applying the triangle inequality yields

$$\|\boldsymbol{\varepsilon}_z(t)\| \le \int_0^t \|\Phi(t,s)\mathbf{r}(s)\|\,\mathrm{d}s. \tag{37}$$

The decoder Lipschitz condition and the definition of $\varepsilon_{\mathrm{AE}}$ then give

$$\begin{aligned}\|\mathbf{u}(t) - \mathcal{G}_{\mathrm{dec}}(\hat{\mathbf{z}}(t))\| &\le \|\mathbf{u}(t) - \mathcal{G}_{\mathrm{dec}}(\mathbf{z}(t))\| + \|\mathcal{G}_{\mathrm{dec}}(\mathbf{z}(t)) - \mathcal{G}_{\mathrm{dec}}(\hat{\mathbf{z}}(t))\| \\ &\le \varepsilon_{\mathrm{AE}} + L_D\|\boldsymbol{\varepsilon}_z(t)\| \\ &\le \varepsilon_{\mathrm{AE}} + L_D\int_0^t \|\Phi(t,s)\mathbf{r}(s)\|\,\mathrm{d}s,\end{aligned} \tag{38}$$

which proves (32).

We next bound the state-transition matrix using the mixed operator $A(\mathbf{w})$. For fixed s and any $\mathbf{x}_s \in \mathbb{R}^{N_z}$, let $\mathbf{x}(t) := \Phi(t,s)\mathbf{x}_s$. Then $\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t)$, and the logarithmic-norm inequality gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{x}(t)\| \le \omega_2(A(t))\,\|\mathbf{x}(t)\|. \tag{39}$$

Applying Gronwall's inequality over $[s,t]$ gives

$$\|\Phi(t,s)\mathbf{x}_s\| \le \exp\left(\int_s^t \omega_2(A(\tau))\,\mathrm{d}\tau\right)\|\mathbf{x}_s\|, \tag{40}$$

Taking the supremum over all $\mathbf{x}_s$ with $\|\mathbf{x}_s\| = 1$ yields

$$\|\Phi(t,s)\|_2 \le \exp\left(\int_s^t \omega_2(A(\tau))\,\mathrm{d}\tau\right), \tag{41}$$

where $\|\cdot\|_2$ is the induced spectral norm. Consequently,

$$\|\Phi(t,s)\mathbf{r}(s)\| \le \|\Phi(t,s)\|_2\|\mathbf{r}(s)\| \le \exp\left(\int_s^t \omega_2(A(\tau))\,\mathrm{d}\tau\right)\|\mathbf{r}(s)\|. \tag{42}$$

Substitution into (32) proves (33).

Finally, $A(t)$ is a convex mixture of the expert matrices. Lemma 1(i) therefore gives

$$\omega_2(A(t)) = \omega_2\left(\sum_{k=1}^K w_k(t;\boldsymbol{\mu})A_k\right) \le \max_k \omega_2(A_k) = \nu. \tag{43}$$

It follows that

$$\exp\left(\int_s^t \omega_2(A(\tau))\,\mathrm{d}\tau\right) \le e^{\nu(t-s)}. \tag{44}$$

Substitution into (33) proves (34). □

The final expression in Proposition 1 is the bound used by Theorem 1 when $L_w = 0$. The gate-path bound (33) uses the mixed operator instead of globally assuming the worst-case growth rate of all experts. The state-transition bound (32) additionally accounts for how each residual is rotated and amplified by the finite-time dynamics.

We evaluate the state-transition bound using the same Runge-Kutta-4 (RK4) scheme used for the latent rollout. Applying RK4 to the affine dynamics produces a one-step map

$$\hat{\mathbf{z}}_{n+1} = F_n\hat{\mathbf{z}}_n + \mathbf{c}_n, \tag{45}$$

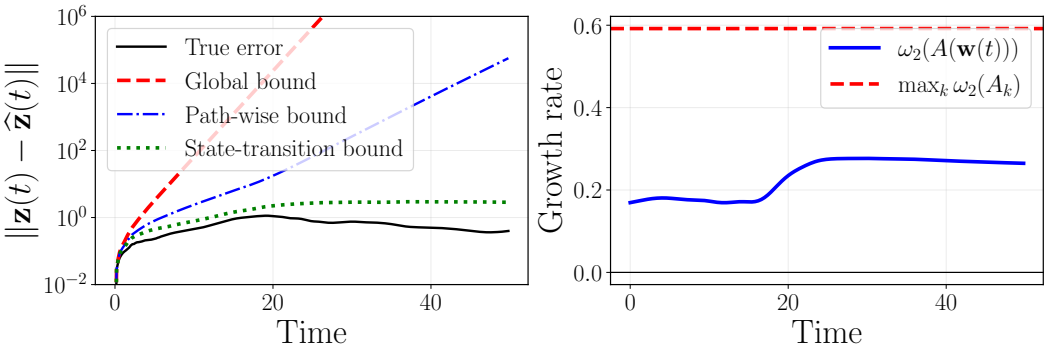

Figure 9: Path-wise latent error bounds for a held-out Vlasov–Fokker–Planck trajectory with $(\alpha, T) = (0.005, 0.58)$. (Left) True latent error compared to the global logarithmic-norm bound (13), gate-path logarithmic-norm bound (33), and the RK4-consistent state-transition bound (47). (Right) Logarithmic growth rate of the realized mixed operator compared with the worst expert.

where F_n is the numerical state-transition matrix over the step and $\mathbf{c}_n$ contains the affine contribution. We compute the one-step residual and longer state transitions as

$$\boldsymbol{\delta}_n := \mathbf{z}_{n+1} - (F_n \mathbf{z}_n + \mathbf{c}_n), \qquad \Phi_{n,j} := F_{n-1} \cdots F_j, \qquad \Phi_{j,j} := I. \tag{46}$$

The corresponding discrete state-transition bound is

$$\|\boldsymbol{\varepsilon}_n\| \le \|\Phi_{n,0}\boldsymbol{\varepsilon}_0\| + \sum_{j=0}^{n-1} \|\Phi_{n,j+1}\boldsymbol{\delta}_j\|. \tag{47}$$

This calculation is inexpensive because the latent dimension is small and the required mixed operators are already evaluated during rollout.

Figure 9 compares the resulting bounds for a held-out Vlasov–Fokker–Planck (17) trajectory in the latent space. Although the instantaneous logarithmic growth rate remains positive, the state-transition bound remains within approximately one order of magnitude of the observed error. In contrast, both logarithmic-norm bounds grow exponentially over the prediction interval. Most of their conservatism therefore comes from assuming persistent alignment with the instantaneously most expansive direction at any given time.

D Stability by construction

D.1 Stability through a common Lyapunov function

Having access to the latent operators allows us to impose conditions on the experts for CABLE. In particular, if the experts share a quadratic Lyapunov function, then any mixture is guaranteed to be uniformly bounded in time. This is a well-known result in control theory (Boyd et al., 1994; Shamma, 2012), which we re-state and re-prove here in the context of CABLE.

Theorem 2 (Stability under a common Lyapunov function)**.** *Assume the hypotheses of Theorem 1 and that there exist a symmetric positive definite $P \in \mathbb{R}^{N_z \times N_z}$ and scalar $\eta > 0$ satisfying*

$$\mathbf{x}^\top \left(A_k^\top P + P A_k\right) \mathbf{x} \le -2\eta \mathbf{x}^\top P \mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^{N_z}, \qquad k = 1, \dots, K. \tag{48}$$

Define the condition number $\kappa(P) := \lambda_{\max}(P)/\lambda_{\min}(P)$ and assume that η is large enough to satisfy

$$\eta_{\text{eff}} := \eta - \sqrt{\kappa(P)}\, M_\Theta\, R_\Xi\, L_w > 0. \tag{49}$$

Then for every $t \in [0, t_f]$, the CABLE prediction error is bounded uniformly in time as

$$\|\mathbf{u}(t) - \mathcal{G}_{\text{dec}}(\hat{\mathbf{z}}(t))\| \le \varepsilon_{\text{AE}} + L_D \sqrt{\kappa(P)}\, \frac{1 - e^{-\eta_{\text{eff}} t}}{\eta_{\text{eff}}}\, \varepsilon_{\text{Dyn}} \le \varepsilon_{\text{AE}} + L_D \sqrt{\kappa(P)}\, \frac{\varepsilon_{\text{Dyn}}}{\eta_{\text{eff}}}. \tag{50}$$

Proof. We work in the norm $\|\mathbf{x}\|_P := (\mathbf{x}^\top P\mathbf{x})^{1/2}$, which satisfies $\lambda_{\min}(P)^{1/2}\|\mathbf{x}\| \le \|\mathbf{x}\|_P \le \lambda_{\max}(P)^{1/2}\|\mathbf{x}\|$. The hypothesis on the experts gives that, for every $\mathbf{w} \in \mathcal{W}^K$, we have $\mathbf{x}^\top(A(\mathbf{w})^\top P + PA(\mathbf{w}))\mathbf{x} \le -2\eta\,\mathbf{x}^\top P\mathbf{x}$, since the condition is preserved under convex combinations. Hence, the logarithmic norm of $A(\mathbf{w})$ induced by $\|\cdot\|_P$ is at most $-\eta$. Repeating the calculation (26) in the P-norm yields,

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\boldsymbol{\varepsilon}_z\|_P \le -\eta\|\boldsymbol{\varepsilon}_z\|_P + \|\mathbf{r}\|_P + \|\mathbf{h}\|_P. \tag{51}$$

Using norm equivalence and essentially repeating the computation in (27),

$$\|\mathbf{h}\|_P \le \lambda_{\max}(P)^{1/2}\|\mathbf{h}\| \le \lambda_{\max}(P)^{1/2} M_\Theta R_\Xi L_w \|\boldsymbol{\varepsilon}_z\| \le \sqrt{\kappa(P)}\, M_\Theta R_\Xi L_w \|\boldsymbol{\varepsilon}_z\|_P, \tag{52}$$

so that $\frac{\mathrm{d}}{\mathrm{d}t}\|\boldsymbol{\varepsilon}_z\|_P \le -\eta_{\text{eff}}\|\boldsymbol{\varepsilon}_z\|_P + \|\mathbf{r}\|_P$. Gronwall's inequality with $\boldsymbol{\varepsilon}_z(0) = 0$ gives

$$\|\boldsymbol{\varepsilon}_z(t)\|_P \le \int_0^t e^{-\eta_{\text{eff}}(t-s)}\|\mathbf{r}(s)\|_P\,\mathrm{d}s. \tag{53}$$

Therefore,

$$\begin{aligned}\|\mathbf{u}(t) - \mathcal{G}_{\text{dec}}\left(\hat{\mathbf{z}}(t)\right)\| \le \varepsilon_{\text{AE}} + L_D\|\boldsymbol{\varepsilon}_z(t)\| &\le \varepsilon_{\text{AE}} + \frac{L_D}{\sqrt{\lambda_{\min}(P)}}\|\boldsymbol{\varepsilon}_z(t)\|_P \\ &\le \varepsilon_{\text{AE}} + L_D\sqrt{\kappa(P)}\int_0^t e^{-\eta_{\text{eff}}(t-s)}\|\mathbf{r}(s)\|\,\mathrm{d}s \\ &\le \varepsilon_{\text{AE}} + L_D\sqrt{\kappa(P)}\,\varepsilon_{\text{Dyn}}\,\frac{1-e^{-\eta_{\text{eff}}t}}{\eta_{\text{eff}}}.\end{aligned} \tag{54}$$

Since $\eta_{\text{eff}} > 0$, the right-hand side is bounded uniformly in t by $\varepsilon_{\text{AE}} + L_D\sqrt{\kappa(P)}\,\varepsilon_{\text{Dyn}}/\eta_{\text{eff}}$. □

For a scheduling gate $L_w = 0$, so $\eta_{\text{eff}} = \eta > 0$ automatically. For a state-dependent gate the contraction rate must dominate the gate-sensitivity term. We can in fact ensure that the hypotheses of Theorem 2 are satisfied by construction, which we describe in the following Remark.

Remark 2 (Lyapunov structure by construction)**.** *Let the experts share a learned symmetric positive definite metric $P := WW^\top + \epsilon I$ and set*

$$A_k = P^{-1}(J_k - R_k), \qquad J_k = U_k - U_k^\top, \qquad R_k = V_kV_k^\top + \eta P. \tag{55}$$

Through skew-symmetry of J_k and positive semi-definiteness of $V_kV_k^\top$, the hypothesis of Theorem 2 holds automatically. During training, we learn matrices W, U_k, and V_k together with the expert biases b_k and the gate parameters. The use of a shared P is the standard common-quadratic-Lyapunov construction for LPV systems (Boyd et al., 1994; Shamma, 2012). Related parameterizations have also been used in stability-preserving learned dynamics (Kolter & Manek, 2019; Cheng et al., 2024; Roth et al., 2025). Here we apply this construction to a learned convex library of affine experts.

As mentioned in the main text, we do not impose this contraction structure by default when applying CABLE. Many systems require transient growth or sustained oscillation, and forcing all experts to share a dissipative metric significantly restricts the model. We use the common-Lyapunov construction only when the system is expected to decay to a steady solution, as in the following example.

D.2 Steady-State Relaxation

This experiment considers a forced diffusion problem

$$u_t = 0.05u_{xx} - 0.5u + c\,s_\infty(x) + q(t)\,s_{\text{tr}}(x), \qquad x \in (0,1), \tag{56}$$

with zero initial and boundary conditions, $s_\infty(x) = \sin(\pi x)$, $s_{\text{tr}}(x) = e^{-100(x-0.5)^2}$, and $q(t) = 5e^{-((t-2.5)/1.5)^2}$. The transient forcing $q(t)$ decays, so each trajectory approaches a c-dependent steady state independent of the path taken. We consider base CABLE (without thresholding), CABLE with a common quadratic Lyapunov function (CABLE CQLF), and PNODE with different weight decay values λ as baselines. We train on $t \in [0, 10]$ and integrate to $t = 100$ without further training. For training we use nine equally spaced values of $c \in [0.5, 1.5]$ and test on six midpoints of the

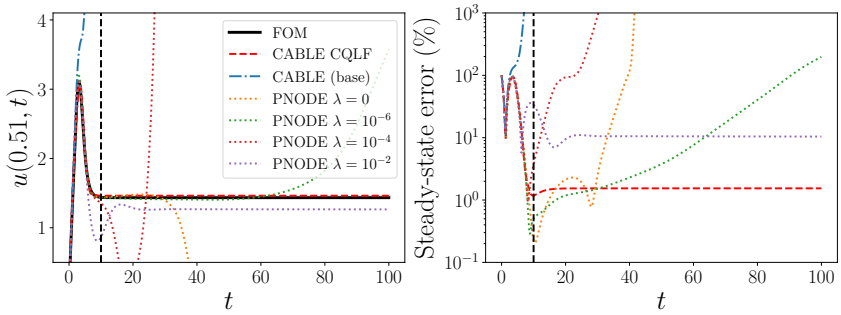

Figure 10: Long-time predictions of the relaxation problem (56) for held-out $c = 1.425$. We compare CABLE under the Lyapunov parameterization of Remark 2 against base CABLE and PNODE at several weight decays λ. (Left) solution at $x = 0.51$ and (right) relative error against the true steady state, with the dashed line at $t = 10$ ending the training window. All models get the same steady-state supervision, but only CABLE with a shared Lyapunov function is guaranteed to contract toward its equilibrium.

training parameters. The autoencoder is pretrained on reconstruction loss alone and then frozen so all models see identical latent trajectories. Our CABLE models both use a scheduling gate $\mathbf{w}(c, q(t))$, which omits explicit time so that the model can run past the training window, and CABLE CQLF uses the parameterization of Remark 2 with $\eta = 0.1$.

As $q(t) \to 0$ the gate tends to $\mathbf{w}(c, 0)$ and the dynamics approach a constant affine system with unique equilibrium $\mathbf{z}^\star(c) = -A(\mathbf{w}(c, 0))^{-1}\mathbf{b}(\mathbf{w}(c, 0))$. Contraction guarantees trajectories reach this equilibrium but not that it sits at the true steady state, so we add a penalty $\|A(\mathbf{w}(c, 0))\mathbf{z}_\infty(c) + \mathbf{b}(\mathbf{w}(c, 0))\|^2$, anchoring it to the encoded steady state (Nam et al., 2026), and give PNODE the analogous penalty.

Figure 10 shows a representative held-out trajectory at $c = 1.425$. CABLE CQLF stays near the true steady state throughout extrapolation, while base CABLE and PNODE drift or diverge. Using $\lambda = 10^{-2}$ gives the best PNODE equilibrium error, but the solution begins to degrade before the end of the training window. The largest logarithmic norm among the CABLE experts is $\max_k \omega_2(A_k) = 10.78$, so contraction holds in the learned P-norm even though the experts are non-contractive in the Euclidean norm. We observed similar stability results over all test trajectories.