

Sparse Koopman Autoencoders Identify Local Dynamical Regimes in Multibasin Systems

Aidan Li^{1,2,3} Uday Kiran Reddy Tadipatri⁴ Mahan Fathi⁵
Sarath Chandar^{1,2,6,7} Ross Goroshin^{2,3}

¹Chandar Research Lab ²Mila – Quebec AI Institute ³Université de Montréal ⁴University of Pennsylvania

⁵NVIDIA ⁶Polytechnique Montréal ⁷Canada CIFAR AI Chair

NeurIPS 2026 · github.com/aidanmrli/skae

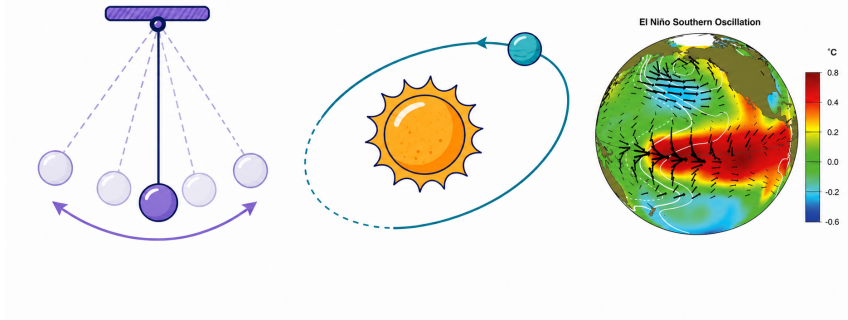


Roadmap

1. **Motivation:** linearizing nonlinear dynamics
2. **Problem:** why multibasin systems break it
3. **Idea:** sparse supports as regime variables
4. **Model:** SKAE and its training
5. **Experiments:** forecasting, interventions, basins
6. **Discussion & outlook**

Motivation: Koopman learning

What is a dynamical system?



A **state** that changes over time according to a **rule**.

$$x_t \xrightarrow{\text{dynamics}} x_{t+1} \xrightarrow{\text{dynamics}} x_{t+2} \cdots$$

Objective

Embed nonlinear dynamics into coordinates where the evolution is **linear**, or at least locally linear.

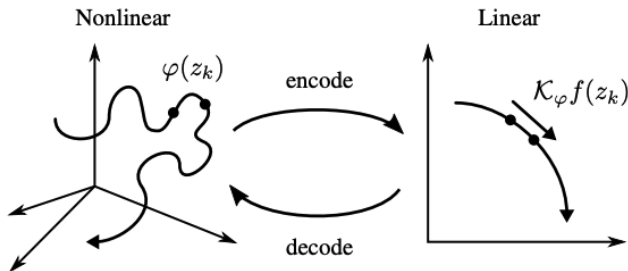


Figure from Azencot et al. (ICML 2020).

Motivation for linear coordinates

- Linear models make prediction, estimation, and control easier.
- A suitable change of coordinates can substantially simplify analysis.

Existence results for linearizing coordinates

- Local linearization near hyperbolic fixed points (Hartman–Grobman)
- Koopman eigenfunctions linearize on a basin of attraction. . .

but only *existentially*.

Good linear coordinates can exist, but how to build them from finite data?

Grobman 1959 · Hartman 1960 · Lan & Mezić 2013 · Kvalheim & Revzen 2021

A data-driven approach

Rather than hand-designing observables,
learn the embedding directly from trajectory data.

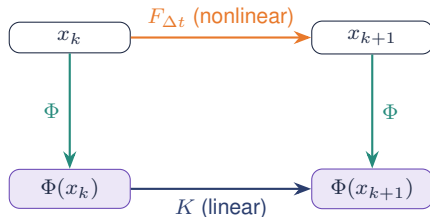
DMD $\rightarrow$ EDMD $\rightarrow$ **Koopman autoencoders**

Schmid 2010 · Williams et al. 2015 · Lusch et al. 2018 · Azencot et al. 2020 · Fathi et al. 2024

Koopman operator theory

The **Koopman operator** $\mathcal{K}$ acts *linearly* on scalar observables $g : \mathcal{X} \rightarrow \mathbb{R}$:

$$(\mathcal{K}g)(x) = g\left(F_{\Delta t}(x)\right)$$



Stack d_z such observables into a vector lift $\Phi = (g_1, \dots, g_{d_z})$; a finite approximation seeks a matrix K with $\Phi(F_{\Delta t}(x)) \approx K \Phi(x)$, keeping enough to reconstruct x .

Koopman autoencoders (KAEs)

$$\hat{x}_{k+h} = D_{\phi}(K^h E_{\theta}(x_k))$$

Learn encoder E_{θ} , decoder D_{ϕ} , and one latent transition K jointly, from trajectory windows.

A key modeling constraint

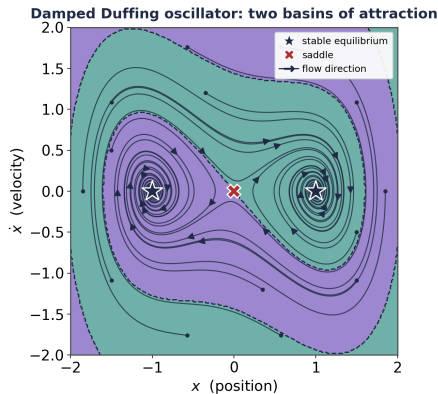
The same transition matrix K is applied at *every* rollout step.

Any state-dependent structure required for prediction must reside in the learned representation z_k .

Problem: multibasin systems

Many systems are multistable

Different initial conditions settle into **different** long-run behaviors.



No single global linearization

Multibasin systems *cannot generally* admit a single finite-dimensional *global* linearization with a continuous encoder–decoder pair.

Brunton et al. 2016 · Pan & Duraisamy 2024 · Liu, Ozay & Sontag 2025 · Fathi et al. 2024

Reframing the objective

Don't force all basins through a *single* global linearization.

Find distinct *local* linearizations.

Idea: sparse supports as regime variables

Two quantities the representation must encode

(i) **Which** local linearization is active

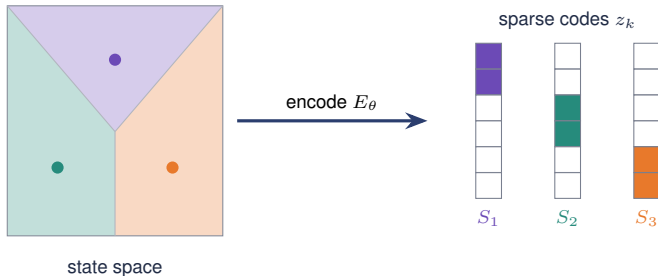
a *discrete* regime label

(ii) **Coordinates** that linearize dynamics within it

a *continuous* local chart

A sparse code encodes both quantities

the **support** (which coordinates are active) $\Rightarrow$ *which* regime
the **nonzero values** $\Rightarrow$ local Koopman coordinates



Each region of state space activates its own *support*: the colored (nonzero) cells; the code is zero elsewhere. Nearby states share a support, giving a union-of-subspaces structure.

Sparse coding

Given an overcomplete dictionary D , find a code with **few nonzeros**:

$$z^*(x) = \arg \min_z \frac{1}{2} \|x - Dz\|_2^2 + \lambda_{\text{sc}} \|z\|_1$$

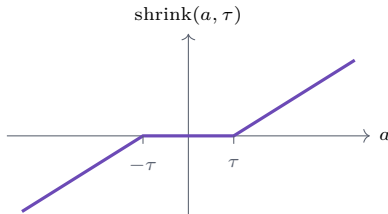
The ℓ_1 penalty drives most coordinates to *exactly* zero.

Learned ISTA (LISTA): sparse coding as an encoder

Unroll the solver into a fixed-depth network: from a dense pre-code $c = f_{\theta}(x)$, apply Q learned shrinkage steps.

$$u^{(0)} = \text{shrink}(c, \tau), \quad u^{(q+1)} = \text{shrink}(S u^{(q)} + c, \tau), \quad z = u^{(Q)} = E_{\theta}(x)$$

the $u^{(q)}$ are the shrinkage iterates; the sparse code is the final iterate $z = u^{(Q)}$.



Soft-thresholding zeros out $[-\tau, \tau]$, creating the sparse support (Gregor & LeCun, 2010).

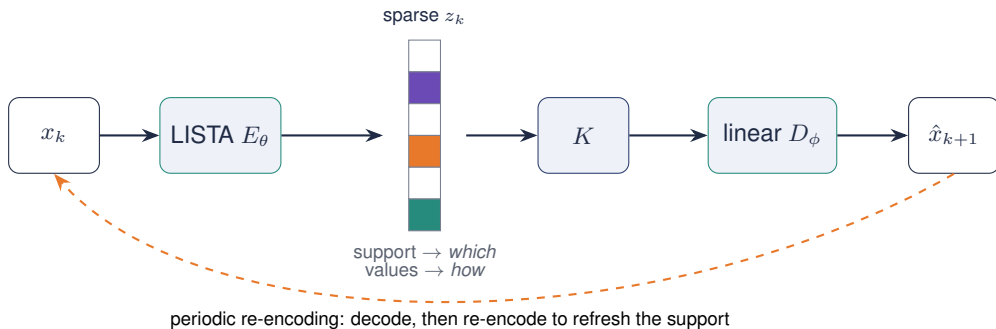
The relevant inductive bias

Thresholding makes *exact zeros*.

The encoder **explicitly selects** active coordinates,
and *learns the selection rule* from the prediction objective.

The SKAE model & training

Sparse Koopman Autoencoder (SKAE)



LISTA encodes a sparse code, one matrix K advances it, and a linear dictionary decodes it. The support selects which columns of K and the decoder act.

Two complementary objectives

- **Koopman:** values on each support evolve *linearly* under K
- **Sparsity:** different regimes use *different* active coordinates

Jointly, the discrete support selects the regime while the continuous values parameterize its local linear dynamics.

Encoders compared

- **LISTA:** sparse by construction (thresholding gives exact zeros)
- **Sparse MLP:** induced sparsity (ReLU with an ℓ_1 penalty)
- **Dense MLP:** no sparsity mechanism (control baseline)

Training objective: four terms

- **Prediction:** decoded rollout matches future states
- **Reconstruction:** code stays tied to the observed state
- **Latent linearity:** rolled-out latent matches encoded latent
- **Sparsity:** few active coordinates

$$\mathcal{L} = \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{lin}}\mathcal{L}_{\text{lin}} + \lambda_{\text{sp}}\mathcal{L}_{\text{sp}} + \mathcal{L}_{\text{struct}}$$

No basin labels enter training.

Support objects

Exact support: $S_{\text{abs}}(x) = \{i : |z_i| > 10^{-3}\}$, the set of active coordinates.

Support families F_{abs} group nearby exact supports by *Jaccard overlap* (intersection over union), merging when $J \geq 0.5$:

$$J(m, r) = \frac{|\text{active}(m) \cap \text{active}(r)|}{|\text{active}(m) \cup \text{active}(r)|}$$

We use these to test whether the active coordinates are used, and whether support families track basins.

Periodic re-encoding refreshes the support

Every m steps the predicted latent is decoded and re-encoded, so the encoder re-selects the active support as the trajectory advances.

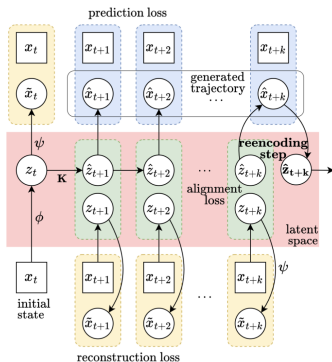


Figure from Fathi et al., ICLR 2024. Squares = ground truth, circles = model-inferred.

Experiments

Benchmarks: 25 systems

- **15 procedurally generated 2-D multibasin systems**

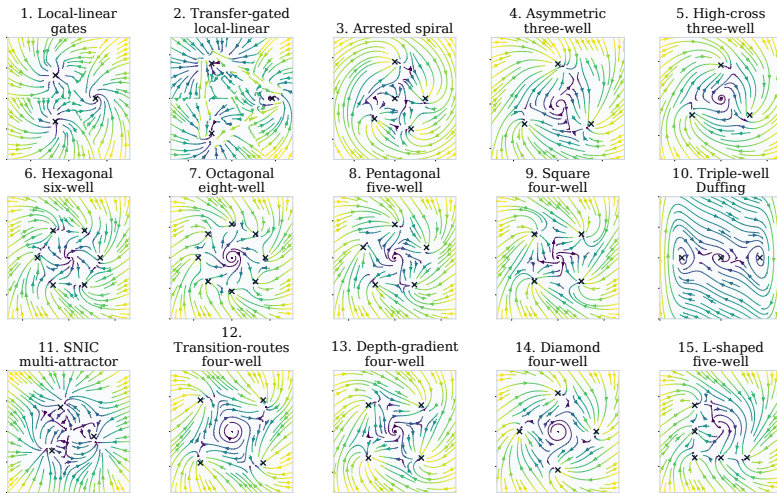
3–8 basins each, with *known* basins (used for evaluation only)

- **10 chaotic Dysts flows**

Chua, Shimizu–Morioka, and more: harder forecasting tests

Dysts: Gilpin, NeurIPS 2021.

The 15 multibasin systems



Ground-truth vector fields.

Models compared

- **Sparse LISTA encoders**

LISTA · LISTA-BD (block-diag. K) · LISTA-SB (soft-block K)

- **Sparse MLP encoders**

ReLU + ℓ_1

Sparse MLP · Sparse MLP-BD (block-diag. K)

- **Dense MLP**

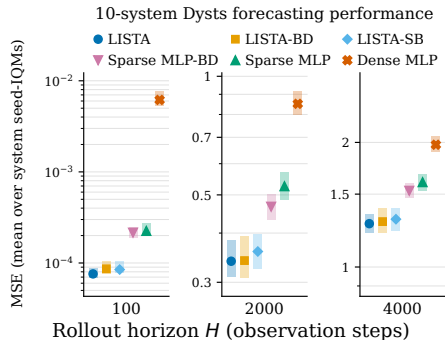
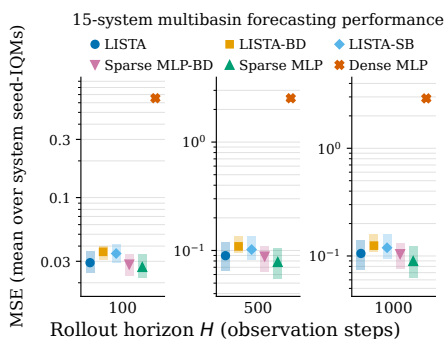
no sparsity: the control baseline

15 seeds per model and system

Experimental questions

1. Do sparse encoders lead to better *forecasting*?
2. Are the supports *functionally necessary*?
3. Do support families *align with basins*?

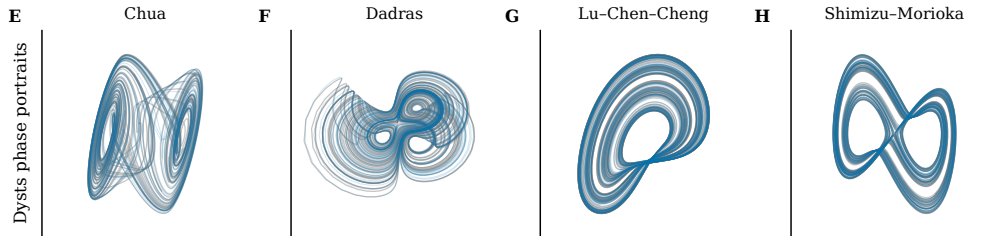
Result 1: sparse encoders forecast better



23–32 $\times$ lower error than the dense KAE on multibasin systems

Significant in 374 of 375 model–system–horizon comparisons (15 seeds, Holm-corrected Wilcoxon).

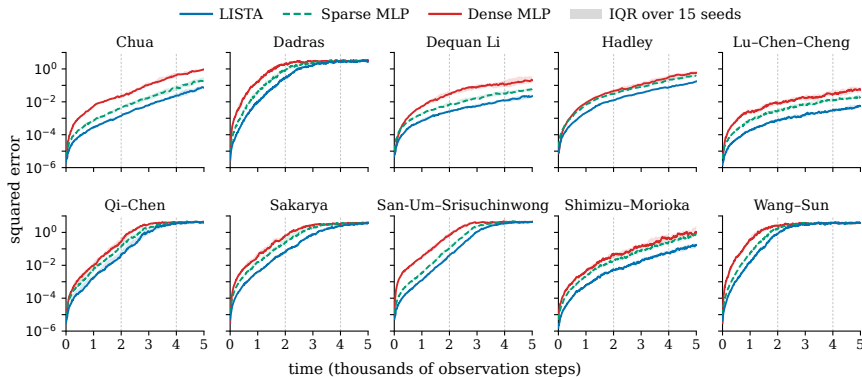
Result 1: long rollouts on chaotic flows



LISTA forecasts (blue) against held-out truth (gray): 20 test starts, 5,000 steps each.

On chaotic flows, the dense KAE has $27\text{--}81\times$ higher error at $H=100$

Result 1: forecast error over time on chaotic flows

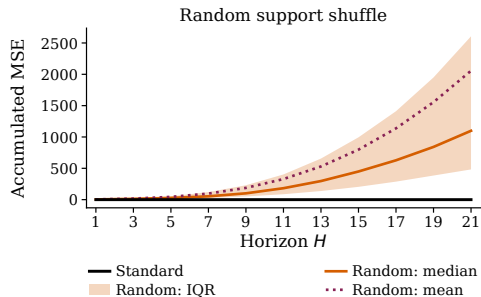
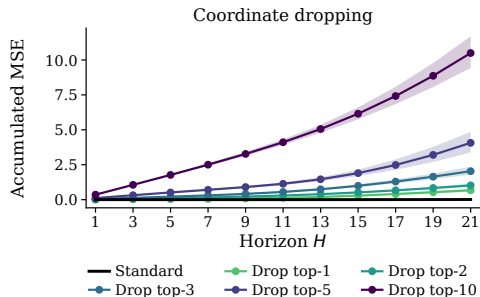


Lower error than the dense KAE in every system, steps 304 to 2,000

Median squared error over 100 test starts; lines: median over 15 seeds; bands: IQR.

Result 2: the supports are functionally necessary

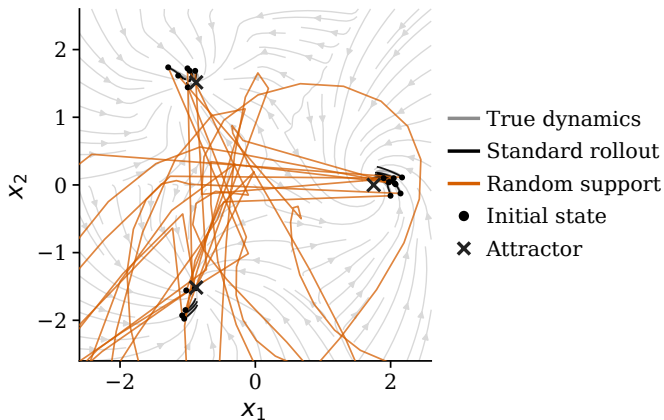
Edit *only* the active set of a trained SKAE's initial code, then forecast.



21-step error: 0.0112 $\rightarrow$ 0.670 (drop 1 coordinate) $\rightarrow$ 2060 (scramble the support)

LISTA SKAE on Local-linear gates; 100 starts inside basins; rollouts without re-encoding.

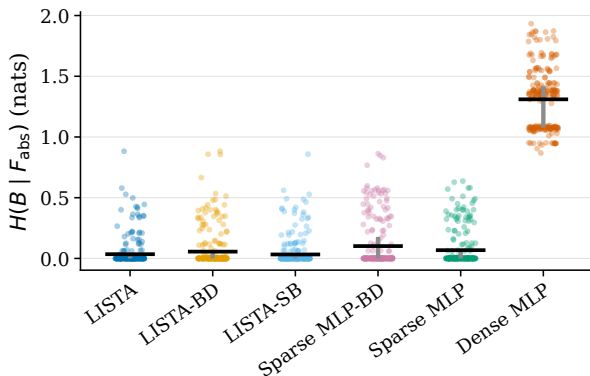
Result 2: trajectory-level effect



Scrambled supports (orange) leave the true dynamics; standard rollouts (black) track them.

Result 3: support families identify the basin

How much does a support family reveal about the withheld basin label B ?



$H(B | F_{\text{abs}})$ (nats, lower is better)

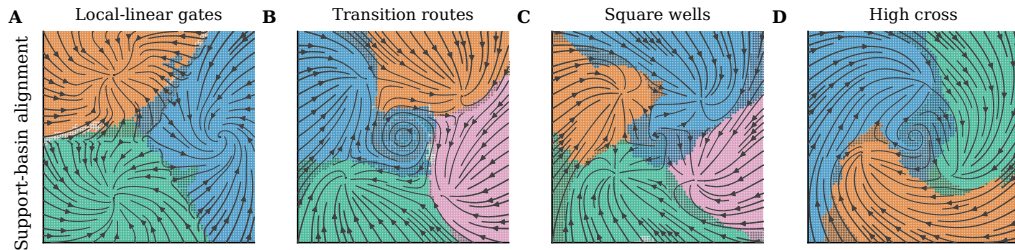
0.033–0.056 LISTA variants

0.068–0.102 Sparse MLPs

1.31 Dense MLP: one family,
no basin information

Sparse supports pin down the basin; dense latents collapse to a single family

Result 3: unsupervised basin recovery



LISTA support families, colored by their basin (black: mismatch). No basin labels in training.

One-to-one family–basin match in 175 of 225 runs (dense KAE: 0)

15 systems $\times$ 15 seeds, on held-out states inside basins.

Discussion & outlook

Takeaways

- (i) **Forecasting:** sparse latents give $23\text{--}32\times$ lower error than dense latents on multibasin systems
- (ii) **Necessity:** scrambling the active set breaks the forecasts
- (iii) **Interpretability:** support families recover the basins, without labels

A practical reconciliation

Theory: no single global finite Koopman embedding for multibasin systems in general.

Practice: a single sparse model *can* hold many local Koopman regimes, indexed by latent support.

Interpretability from the code structure

Sparse supports are an *interface* between continuous Koopman coordinates and discrete regimes:
a simple object you can inspect after training.

Periodic re-encoding may help world models

Many world models plan or learn behavior by rolling their latent dynamics forward without new observations.

Over long imagined trajectories, prediction errors can accumulate and move the latent state away from representations grounded in data.

Periodically decode and re-encode to return the rollout to **encoder-supported coordinates**.

In SKAE, this also lets the sparse support select a new local representation.

Next steps

- **Learned support families**, e.g. with a contrastive objective
- **Richer systems**: high-dimensional, noisy, partially observed
- **Control**: multi-regime forecasting and control with sparse Koopman models

Conclusion

Interpretability in dynamical representation learning
can come from the **structure of the latent code**.

Paper: [arXiv:2608.29057](https://arxiv.org/abs/2608.29057) · Code: github.com/aidanmrli/skae · aidan.li@mila.quebec