Decomposed Linear Dynamical Systems (dLDS) for learning the latent components of neural dynamics

Noga Mudrik*   NMUDRIK1@JHU.EDU
Department of Biomedical Engineering
Center for Imaging Science
Kavli NDI
Johns Hopkins University
Baltimore, MD 21218.

Yenho Chen*   YENHO@GATECH.EDU
Department of Biomedical Engineering
Georgia Institute of Technology
Atlanta, GA 30332.

Eva Yezerets   EYEZERE1@JHU.EDU
Department of Biomedical Engineering
Center for Imaging Science
Johns Hopkins University
Baltimore, MD 21218.

Christopher J. Rozell   CROZELL@GATECH.EDU
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332.

Adam S. Charles   ADAMSC@JHU.EDU
Department of Biomedical Engineering
Center for Imaging Science
Mathematical Institute for Data Science
Kavli NDI
Johns Hopkins University
Baltimore, MD 21218.

Abstract

Learning interpretable representations of neural dynamics at a population level is a crucial first step to understanding how observed neural activity relates to perception and behavior. Models of neural dynamics often focus on either low-dimensional projections of neural activity or on learning dynamical systems that explicitly relate to the neural state over time.

---

*. Equal Contribution

We discuss how these two approaches are interrelated by considering dynamical systems as representative of flows on a low-dimensional manifold. Building on this concept, we propose a new decomposed dynamical system model that represents complex non-stationary and nonlinear dynamics of time series data as a sparse combination of simpler, more interpretable components. Our model is trained through a dictionary learning procedure, where we leverage recent results in tracking sparse vectors over time. The decomposed nature of the dynamics is more expressive than previous switched approaches for a given number of parameters and enables modeling of overlapping and non-stationary dynamics. In both continuous-time and discrete-time instructional examples, we demonstrate that our model effectively approximates the original system, learns efficient representations, and captures smooth transitions between dynamical modes. Furthermore, we highlight our model's ability to efficiently capture and demix population dynamics generated from multiple independent subnetworks, a task that is computationally impractical for switched models. Finally, we apply our model to neural "full brain" recordings of *C. elegans* data, illustrating a diversity of dynamics that is obscured when classified into discrete states.

1. Introduction

The past decade has seen rapid growth in neuroscience driven by the emergence of new technologies that enable the recording of neural population activity, such as large-scale optical imaging (Demas et al., 2021) and electrophysiology (Steinmetz et al., 2021). As a result, neural data analysis has moved beyond the characterization of single neurons to the modeling of entire large neuronal populations (Saxena and Cunningham, 2019). For large simultaneous recordings, a significant challenge lies in understanding the intricate correlated patterns of neural activity on a single trial basis. Currently, the quantitative language describing these observations has primarily leveraged two conceptual frameworks: 1) dimensionality reduction and 2) dynamical systems modeling.

Dimensionality reduction primarily addresses the identification of a small number of degrees of freedom that characterize a time series recording. Often these methods treat activity patterns at individual time points independently and seek to discover a representative geometry that underlies the data. Traditional linear methods, including PCA, ICA, POD (Berkooz et al., 1993), and their variants (Yu et al., 2008; Wu et al., 2017) are often employed to reduce high-dimensional time-series data and can be viewed as decomposing time signals into a linear summation of components with time-changing coefficients. However, these methods do not yet capture the fundamental nature of dynamical systems as they do not directly model the recursive changes in a system state as a function of it's previous values.

These methods have recently given way to more flexible descriptions of dimensionality reduction. Many recent nonlinear dimensionality reduction methods rely on the manifold hypothesis (Wu et al., 2017; Nieh et al., 2021; Gallego et al., 2017; Cunningham and Yu, 2014; Benisty et al., 2021; Mishne et al., 2016), including local embeddings (Balasubramanian and Schwartz, 2002; Roweis and Saul, 2000) and variational auto-encoders (Han et al., 2019). This *neural manifold* assumption (i.e., that instantaneous neural activity patterns lie on a low-dimensional manifold) removes the assumption of linearity in the low-dimensional neural representation, and enables the identification of correlated activity that corresponds to a potentially much lower-dimensional geometric structure. Similar to the linear dimensionality

reduction methods, however, nonlinear methods also do not explicitly capture the temporal nature of the data.

By contrast, dynamical systems models focus primarily on capturing the temporal relationships within neural activity. However, these models often treat activity patterns over time as arising from individual latent states without regard to any geometric structure being modeled by these low-dimensional latent variables. Proposed dynamical systems models include concise linear dynamical systems (LDS) (Churchland et al., 2012), switched systems that capture abrupt non-stationarities (Linderman et al., 2017; Glaser et al., 2020; Nassar et al., 2018), GLMs that learn sets of filters to describe temporal conditional probabilities (Pillow et al., 2008), and more recently arbitrary function approximations in the form of general recurrent neural networks (RNNs) (Pandarinath et al., 2018; Keshtkaran and Pandarinath, 2019). This collection of approaches reflect a fundamental trade-off between model complexity and interpretability. On one side are simple, regularized, often linear models that directly expose latent relationships in the data, but are limited in their ability to accurately model complex time-varying structures. On the other side are modern black-box deep learning methods, which are highly expressive but obfuscate learned relationships and are thus difficult to interpret (Schulz et al., 2020). While dynamical systems models remain a promising avenue for capturing the temporal dynamics of neural activity, they often lack explicit constraints on the structure of the latent variables or on the underlying neural manifold.

Accordingly, there is a critical need to develop methods that remain interpretable while maintaining a high level of expressivity (i.e., capturing rich nonlinear structure that arises when studying neural population activity). While both traditional dimensionality reduction methods and dynamical systems models seek to represent low-dimensional structures, we currently lack methods that integrate the manifold hypothesis directly into a dynamical systems model, thus maintaining both model expressivity and interpretability. This gap in modeling limits our ability to study important problems in neuroscience. As a prime example, one task that remains challenging is the identification and characterization of multiple subnetworks within a neural population. Population recordings often consist of distinct subnetworks that exhibit different functional roles or represent different aspects of information processing. Understanding the dynamics of each distinct subnetwork is crucial for revealing the underlying mechanisms of neural computation. While each subnetwork may have its own dynamics, connectivity patterns, functional properties, and interactions with other subnetworks, the multivariate time series in neural recordings represent all subnetworks' combined activity and must be unmixed to discover the true underlying structure.

To address this need, we introduce a decomposed Linear Dynamical Systems (dLDS) model that describes high-dimensional neural activity as dynamical flows on a low-dimensional manifold. Transitions between consecutive time points are decomposed as a time-varying mixture of linear dynamics systems (LDSs). In our approach, each LDS element captures a canonical movement along the neural manifold, which are linearly combined to model the overall dynamics in the low-dimensional data geometry. Specifically, we constrain the linear combination at each time point to be sparse (i.e., only a few dynamics are combined at any time point). This parsimony eases the task of interpreting the individual LDS systems in light of global dynamical structure or external variables. The dLDS model exhibits increased expressiveness per model parameter compared to existing switched Linear Dynamical Systems

(SLDS), and can capture richer dynamics while maintaining interpretability due to the use of simple LDS primitives that highlight unique dynamical patterns under different conditions. Our implementation of dLDS encompasses both continuous- and discrete-time variants, each with distinct advantages under different settings (e.g., scalability and invariance to system speed, respectively). Our main contributions in this paper are 1) introducing the dLDS model to illustrate its foundational properties, capabilities, and strengths; 2) demonstrating key advantages of this model over current approaches in multiple scenarios; and 3) illustrating its use in the analysis of population neural activity from calcium imaging in *C. elegans*.

2. Background and Related Work

Dynamical systems models of neural data. Dynamical systems models explicitly seek to identify the relationships of neural activity patterns over time. Early work leverages the learning of LDSs to approximate the progression of activity in neural state space (Sani et al., 2021a; Golub et al., 2013; Churchland et al., 2012). These approaches leverage theory from traditional linear stationary system identification literature, such as Kalman filtering, etc. (Haykin, 1996). Applications include characterization of neural populations, controlling of brain-computer interfaces, and relating brain activity to external variables (Yu et al., 2007). Due to the nonlinear and non-stationary dynamics present in neural activity, two primary approaches have been taken to extend dynamical systems models beyond the linear and stationary regimes: 1) nonlinear recurrent neural networks (Pandarinath et al., 2018; Keshtkaran and Pandarinath, 2019; Sussillo et al., 2015; Sani et al., 2021b; Kleinman et al., 2021), and 2) Switched Linear Dynamical Systems (SLDS) (Ackerson and Fu, 1970; Chang and Athans, 1978; Hamilton, 1990; Bar-Shalom and Li, 1990; Ghahramani and Hinton, 1996; Murphy, 1998; Fox et al., 2008; Linderman et al., 2017). Recurrent neural network (RNN) approaches, while highly flexible, are based on training "black boxes" which make interpretability of the learned dynamics difficult. Moreover, some emerging RNN-based approaches (e.g., (Pandarinath et al., 2018; Keshtkaran and Pandarinath, 2019)) compress the dynamics into initial conditions that recapitulate the neural dynamics when input into the trained RNN.

SLDS explicitly seeks interpretable characterizations of the dynamics by modeling transitions over time between a discrete set of linear systems. SLDS variants are described as a Gauss-Markov process and a switching model that determines transitions between the linear dynamics through a discrete-time Markov process. Although SLDS can discover latent dynamics, it is a limited generative model since state durations are determined stochastically. Recurrent SLDS (rSLDS) extends SLDS by including an additional dependency between the discrete switches and the previous location in state space through a stick-breaking logistic function (Linderman et al., 2017). The rSLDS model improves interpretability by dividing the state space into K partitions with locally linear dynamics. However, rSLDS introduces challenges during inference due to its dependencies on the permutation of the discrete switches. Tree-structured recurrent SLDS is an extension of rSLDS that addresses this issue through a generalized stick-breaking procedure that enables efficient inference and the ability to represent dynamics at multiple levels of resolution (Nassar et al., 2018). However, scalability remains an issue as the optimal number of discrete systems is difficult to determine, often leading to inefficient representations of the underlying dynamics.

We note that conventional LDSs and switching LDSs models, whether treating the system as a single dynamical entity or potentially switching between LDSs, inherently lack the capability to capture multiple co-occurring processes or overlapping subsystems (e.g., simultaneous observation of slow-timescale and faster-timescale systems). To address this limitation, we choose to draw inspiration from sparse coding, offering a solution that maintains a balance between interpretability and the ability to capture concurrent processes.

Sparse coding and dictionary learning. Learning data representations is a central theme in machine learning and neuroscience. One fundamental approach is sparse coding, which assumes a form of efficiency in representation under a linear generative model (Olshausen and Field, 1996; Aharon et al., 2006). In the sparse coding representation, each data point $\boldsymbol{y}_k$ can be linearly generated from a latent vector $\boldsymbol{a}_k$ such that $\boldsymbol{y}_k = \boldsymbol{D}\boldsymbol{a}_k + \boldsymbol{\epsilon}$, where the matrix $\boldsymbol{D}$ contains representational features as its elements, and $\boldsymbol{\epsilon}$ is representational noise. Sparse coding assumes that the representation is efficient in that for any $\boldsymbol{y}_k$, only a few of the features (columns of $\boldsymbol{D}$) are required to construct the data point (i.e., the number of non-zero entries in $\boldsymbol{a}_k$ for any $\boldsymbol{y}_k$ is assumed to be small relative to the size of $\boldsymbol{a}_k$). In general, $\boldsymbol{D}$ is unknown and must be learned from data via a process called dictionary learning. Dictionary learning can be expressed as a variational approach with a delta approximation to the posterior (Barello et al., 2018) that results in a two-step optimization that iterates between inferring the sparse vector $\boldsymbol{a}_k$ for a subset of points, and updating the dictionary $\boldsymbol{D}$ by taking a gradient step over the approximated likelihood.

Transport Operators. Related to dLDS is the Transport Operators (TOs) framework for learning manifold structure in data (Culpepper and Olshausen, 2009). TOs represent a class of generative manifold models by characterizing transformations in local regions as a continuous-time-evolving LDS model. The transformation matrix is decomposed as a weighted sum of K dictionary elements called *TOs*, each individually representing movement along a particular path on the latent manifold and can be combined to reconstruct observed transformations. Operators are learned as part of the DL iterative procedure and are updated at each iteration by taking a gradient descent step to minimize the desired cost. This approach has been demonstrated to effectively approximate geometrical structures of nonlinear manifolds in a variety of settings. For complex data, TOs have been integrated into the latent space of autoencoders and VAEs which enables generative transformation paths and meaningful extrapolations (Connor and Rozell, 2020; Connor et al., 2021a). More recent advances include relaxing the requirement of transformation labels, learning local operator statistics for identity-preserving transformations, and improving the scalability of inference for higher-order models (Connor et al., 2021b; Fallah and Rozell, 2022). Despite this progress, existing work on TOs mainly focuses on discovering the geometry of latent manifolds, but fails to describe the nonlinear temporal flows on these manifolds in complex time series data.

Interpretability. Interpretability is undoubtedly an important consideration to understanding neural dynamics. However, concrete evaluation of interpretability is hampered by the variety of different definitions in the literature. For example, some approaches focus on mapping abstract concepts into domains that humans more naturally understand Montavon et al. (2018). In the context of neural signals, data is collected from arbitrary nonlinear dynamical systems which generally have no analytical solutions available, making it challenging to predict and interpret their behavior. To facilitate the ease of human reasoning, a

model must be transparent both with respect to the whole model altogether as well as for each of its individual parts separately (Lipton, 2018). Reasoning about the entire model can be achieved through simple, parsimonious representations, while reasoning about its components is accomplished through statistically transparent structures that can be easily modified and analyzed. Taken together, this suggests that an interpretable model is one that is sparse with simple building blocks.

The dLDS model takes inspiration from these guidelines and achieves transparency and parsimony through three separate mechanisms. First, dimensionality reduction finds a latent low-dimensional representation that summarizes and approximately generates the correlated high-dimensional signals. Second, the latent nonlinear system is locally approximated with linear systems. Linear components are fully characterized mathematically, making it possible to determine the behavior of the system through its eigendecomposition and gain insight into the underlying mechanisms at any point in time. Third, each transition is decomposed into a sparse combination of a dictionary of linear dynamics which improves parsimony through the reuse of the learned dictionary to efficiently represent shared dynamics over time and under different conditions.

Expressivity. Expressivity is the ability for a model to accurately represent a broad range of highly complex functions (Raghu et al., 2017). A popular approach to modeling complicated nonlinear dynamical systems is to form a piecewise approximation by transitioning between a fixed set of linear regimes throughout the trajectory of the signal (Vyas et al., 2020), such as in SLDS, rSLDS and their variants. A direct result of the fixed number of linear regimes is that it has limited capacity in the types of systems that it can well approximate. In particular, if new system behavior must be captured, then these models must increase the number of linear states (and perhaps fit the entire model from scratch).

For example, switched systems often learn approximate dynamics which have discontinuities along the boundaries of the switches. This can be well suited to capturing sudden changes in dynamics, but may not accurately represent smooth transitions between dynamical modes. Approximating smooth transitions requires increasing the number of linear states available to capture intermediate stages. Additionally, learning discrete states independently becomes prohibitively costly when estimating dynamics for a population that contains multiple independent subgroups that may switch at different times (requiring a different model for each combination of sub-population activity). However, the number of possibilities grows exponentially as the number of subgroups and dynamical modes increases, rendering this approach combinatorially impractical. Moreover, this challenge is compounded during inference where the learned systems are rigid and cannot adapt to capture similar dynamics. For example, SLDS must learn new distinct states to accurately represent subtle variations of similar dynamics such as changes in speed. For a fixed number of parameters, dLDS improves the expressivity of the switching linear approach by offering a controlled way to flexibly modify and reuse learned linear regimes.

3. Decomposed linear dynamical systems

We begin our model description with the observation model. Let $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_T] \in \mathbb{R}^{k \times T}$ be a sequence of T k-dimensional observation vectors $\boldsymbol{y}_1, ..., \boldsymbol{y}_T$. A loading matrix $\boldsymbol{D} \in \mathbb{R}^{k \times p}$ links each observation vector $\boldsymbol{y}_t$ to its underlying latent state, $\boldsymbol{x}_t \in \mathbb{R}^p$, such that:
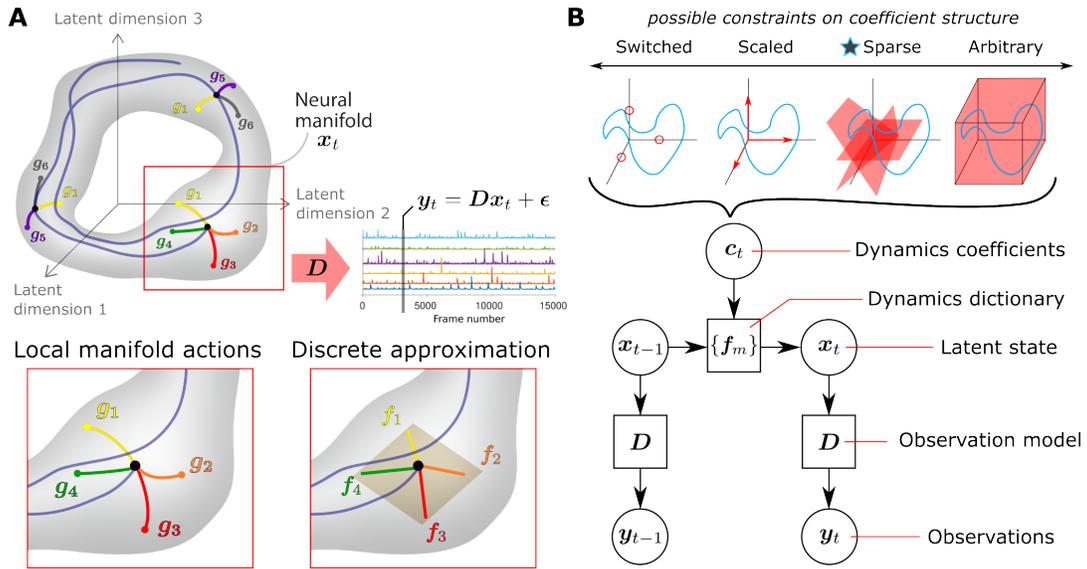
Figure 1: **Decomposed dynamical system model. A:** Trajectories along the manifold are guided by local DOs. In neuroscience, we indirectly observe the latent manifold state $\boldsymbol{x}_t$ through the observation model $\boldsymbol{D}$. The space of transports $\{\boldsymbol{g}_l\}_{l=1:L}$ can be learned directly or through a discretized approximation $\{\boldsymbol{f}_m\}_{m=1:M}$. **B:** The decomposed linear dynamical systems includes an observation model, a dynamics model, and hierarchical variables $\boldsymbol{c}_t$ that control the non-stationarity in the dynamics. These dynamics coefficients can be structured (top), e.g., one fixed active coefficient at a time results in switching between discrete states, whereas enabling flexibility in the coefficient's value can enable scaled dynamics, sparsely structured dynamics, or even more arbitrarily distributed dynamics.

$$\boldsymbol{y}_t = \boldsymbol{D}\boldsymbol{x}_t + \boldsymbol{\epsilon}_t, \tag{1}$$

where $\boldsymbol{\epsilon}_t$ is measurement noise from an isotropic Gaussian distribution.

Critical to dLDS is modeling the temporal evolution of the latent states along the underlying geometry. We begin with the common assumption that $\boldsymbol{x}_t$ lies on a d-dimensional manifold $\mathcal{M} \subset \mathbb{R}^p$. To stay on the manifold over time, the flows guiding the latent state $\boldsymbol{x}_t$ must move along the manifold, according to its tangent space at each point, as depicted in 1. An appropriate continuous-time model that describes the movement along the manifold can be expressed as $\dot{\boldsymbol{x}}_t = \boldsymbol{G_t}\boldsymbol{x}_t + \boldsymbol{\Sigma}^{1/2}\dot{\boldsymbol{W}}_t$ where $\boldsymbol{G_t}$ represents a local transformation at any point in time that maps the point $\boldsymbol{x}_t$ onto nearby points on the manifold and $\boldsymbol{W}_t$ is standard Wiener process.

Taking a finite step integrates a first-order linear differential equation whose solution involves the matrix exponential. Moreover, as the tangent space is a subspace, the operator $\boldsymbol{G}_t$ can be decomposed at each time point into a linear combination of dictionary elements $\boldsymbol{g}_l$ which we refer to as dynamic operators (DOs) to emphasize that they are specifically

designed to represent dynamic processes rather than static geometry. The set of DOs span the space of possible local motions at different points on the manifold (i.e., $\boldsymbol{g}_l \boldsymbol{x}_t$ spans the tangent space at $\boldsymbol{x}_t$) and are weighted by the coefficients $\tilde{\boldsymbol{c}}_t = [\tilde{c}_{1t}, ..., \tilde{c}_{Lt}]^T$ which encode the representation of the dynamical trajectory through $\boldsymbol{x}_t$. Any trajectory between two points $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+\tau}$ on the manifold, can be written as

$$\boldsymbol{G}_t = \sum_{l=1}^{L} \boldsymbol{g}_l \tilde{c}_{lt} \tag{2}$$

$$\boldsymbol{x}_{t+\tau} \approx \text{expm}\left(\boldsymbol{G}_t \tau\right) \boldsymbol{x}_t + \boldsymbol{\nu}_t, \tag{3}$$

where $\boldsymbol{\nu}_t \sim N(0, \boldsymbol{\Sigma}\tau)$ is Gaussian noise.

As a result, the latent dynamics, $\boldsymbol{x}_t \in \mathbb{R}^p$, evolves according to the non-stationary dynamical systems matrix $(\boldsymbol{G}_t)$ defined by the evolution of $\tilde{\boldsymbol{c}}_t$ over time. The continuous-time differential formulation in Equation (3) can hence flexibly define dynamics. However, it requires computing a matrix exponential, which can be computationally expensive in higher dimensions. Thus, for a fixed sampling rate, we can set $\tau = 1$ and approximate the action of the DO dictionary over a local area as $\exp(\sum_{l=1}^{L} \boldsymbol{g}_l \tilde{c}_{lt}) \approx \boldsymbol{F}_t$ (see Appendix A for discussion) for each time point $t = 1, ...T$, which can be expanded into its own basis as

$$\boldsymbol{F}_t = \sum_{m=1}^{M} \boldsymbol{f}_m c_{mt}, \tag{4}$$

where $\boldsymbol{f}_m \in \mathbb{R}^{p \times p}$ for all $m = 1 \ldots M$, the dynamics coefficients for the discrete time form are denoted by $\boldsymbol{c}_t = [c_{1t}, ..., c_{Mt}]^T$, and the dictionary is composed of a set of M linear dynamical systems represented by $\{\boldsymbol{f}_m\}_{m=1:M}$. Note that in the discrete formulation above, we have normalized the time interval between samples to $\tau = 1$ without loss of generality. This choice simplifies the model description, ensuring that the model evolves at fixed intervals of τ in the discrete state.

The discrete-time dynamics model is defined as

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-1} + \boldsymbol{\nu}_t = \sum_{m=1}^{M} \boldsymbol{f}_m c_{mt} \boldsymbol{x}_{t-1} + \boldsymbol{\nu}_t. \tag{5}$$

In dLDS, we distinguish between the two sets of unknowns: model parameters ($\boldsymbol{D}$ and either $\{\boldsymbol{g}_l\}_{l=1:L}$ or $\{\boldsymbol{f}_m\}_{m=1:M}$) and model coefficients ($\boldsymbol{x}_t$ and $\tilde{\boldsymbol{c}}_t$ for the continuous form, and $\boldsymbol{x}_t$ and $\boldsymbol{c}_t$ for the discrete form). The model parameters define the total geometry of the data that we wish to learn through example data. The model coefficients dictate specific trajectories through the learned geometry that can be inferred given the model parameters for any data coming from the same distribution. Next, we focus on defining algorithms capable of learning the model parameters from data.

Model training framework. We frame learning from example data as optimization over the following cost function,

$$\{\widehat{\boldsymbol{x}}_t, \widehat{\boldsymbol{c}}_t\}_{t=1}^T = \arg \min_{\{\boldsymbol{x}_t, \boldsymbol{c}_t\}} \left[\sum_{t=1}^T \|\boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t\|_2^2 + \sum_{t=2}^T \lambda_0 \left\| \boldsymbol{x}_t - \sum_{m=1}^M \boldsymbol{f}_m c_{mt}\boldsymbol{x}_{t-1} \right\|_2^2 \right.$$
$$\left. + \sum_{t=1}^T \left(\lambda_1 \|\boldsymbol{x}_t\|_1 + \lambda_2 \|\boldsymbol{c}_t\|_1 \right) + \sum_{t=2}^T \lambda_3 \|\boldsymbol{c}_t - \boldsymbol{c}_{t-1}\|_2^2 \right].$$

(6)

The first and second term encourages accurate reconstruction of the observed data and latent dynamics, the third and fourth terms encourage sparsity in the latent state and dynamics coefficient, while the fifth term encourages temporal smoothness of the coefficients. Their respective regularization terms are given by λ_0, λ_1, λ_2, and λ_3. For simplicity, we focus here on the discrete dynamics $\{\boldsymbol{f}_m\}_{m=1:M}$. However, we note that a similar learning algorithm can be derived for continuous dynamics by including the matrix exponential into the transition, at the cost of introducing nonconvexity to the objective function. To deal with this challenge, we leverage advances in automatic differentiation and stochastic optimization to provide estimates of the coefficients and model parameters. To optimize this objective, we follow the dictionary learning literature (Olshausen and Field, 1996) and perform coordinate descent, alternating between estimating latent variables $\boldsymbol{x}$ and $\boldsymbol{c}$, and updating model parameters $\{\boldsymbol{f}_m\}_{m=1:M}$ and $\boldsymbol{D}$.

Inference of Latent Variables $\boldsymbol{x}$ and $\boldsymbol{c}$. First, we derive an efficient process for inferring $\boldsymbol{x}_t$ and $\boldsymbol{c}_t$ given the data and an estimate of the model parameters, assuming isotropic Gaussian noise in the observation and dynamics model. The ℓ_1 sparsity regularization encourages the use of only a few dynamical systems, $\boldsymbol{f}_m$, at each time point. Similarly, in cases where we believe that the latent states are independent, ℓ_1 regularization over $\boldsymbol{x}_t$ encourages appropriate decoupling. Modifying λ_0, λ_1, and λ_2 (or setting them to zero) allows our approach to adapt to different modeling conditions by modulating the expected sparsity in both the dynamics coefficients and latent state (Fig. 1B, top). The last term, $\lambda_3 \|\boldsymbol{c}_t - \boldsymbol{c}_{t-1}\|_2^2$, encourages smoothness over the time-varying coefficients. This smoothness constraint between consecutive time points can prevent high jumps and abrupt changes in the model coefficients over time that may be due to noise, thereby enhancing its ability to capture underlying trends without being overly influenced by noise or outliers.

Solving the above for all time points can be computationally intensive, particularly due to the bilinear form induced by the products between $\boldsymbol{x}_{t-1}$ and $\boldsymbol{c}_t$. We adopt the recent Basis Pursuit De-Noising with Dynamical Filtering (BPDN-DF) (Charles et al., 2016) approach for dynamic filtering of sparse signals. BPDN-DF is used for inferring the coefficients, $\boldsymbol{c}_t$, and the latent variables, $\boldsymbol{x}_t$, at a single time point, and proceeds through all time points sequentially. By conditioning on past time point estimates, the dynamics prediction $\sum_{m=1}^M c_{mt}\boldsymbol{f}_m\widehat{\boldsymbol{x}}_{t-1}$ can be written as $\widetilde{\boldsymbol{F}_t}\boldsymbol{c}_t$ where the mth column of $\widetilde{\boldsymbol{F}} \in \mathbb{R}^{p \times M}$ is the product $\boldsymbol{f}_m\widehat{\boldsymbol{x}}_{t-1}$. Thus, the dynamics can be then rewritten as

$$\boldsymbol{x}_t = \boldsymbol{F}_t\boldsymbol{x}_{t-1} = \sum_{m=1}^M c_{mt}\boldsymbol{f}_m\boldsymbol{x}_{t-1} = \sum_{m=1}^M [\boldsymbol{f}_m\boldsymbol{x}_{t-1}]c_{mt} = \widetilde{\boldsymbol{F}_t}\boldsymbol{c}_t. \tag{7}$$

Inferring all coefficients at each time step reduces to a LASSO problem (or partial LASSO or least-squares, depending on if any λ values are set to zero),

$$\widehat{\boldsymbol{x}}_t, \widehat{\boldsymbol{c}}_t = \arg\min_{\boldsymbol{x}_t, \boldsymbol{c}_t} \left[\|\boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t\|_2^2 + \lambda_0 \left\|\boldsymbol{x}_t - \widetilde{\boldsymbol{F}}_t \boldsymbol{c}_t\right\|_2^2 + \lambda_1 \|\boldsymbol{x}_t\|_1 + \lambda_2 \|\boldsymbol{c}_t\|_1 + \lambda_3 \|\boldsymbol{c}_t - \widehat{\boldsymbol{c}}_{t-1}\|_2^2 \right]. \tag{8}$$

We note that for the continuous case, the same basic framework holds. However, the matrix exponential prevents the consolidation of the previous estimate and dynamics into a single matrix $\widetilde{\boldsymbol{F}}$, and retains the following form:

$$\widehat{\boldsymbol{x}}_t, \widehat{\widetilde{\boldsymbol{c}}}_t = \arg\min_{\boldsymbol{x}_t, \boldsymbol{c}_t} \left[\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}_t\|_2^2 + \lambda_0 \left\|\boldsymbol{x}_t - \operatorname{expm}\left(\sum_{l=1}^{L} \boldsymbol{g}_l \tilde{c}_{lt}\right) \widehat{\boldsymbol{x}}_{t-1}\right\|_2^2 \right.$$

$$\left. + \lambda_1 \|\boldsymbol{x}_t\|_1 + \lambda_2 \|\tilde{\boldsymbol{c}}_t\|_1 + \lambda_3 \|\tilde{\boldsymbol{c}}_t - \widehat{\widetilde{\boldsymbol{c}}}_t\|_2^2 \right]. \tag{9}$$

Updating the model parameters. Given the inferred model coefficients, the second step of the learning framework is a gradient step over the model parameters. For the latent state projection $\boldsymbol{D}$, the gradient can be computed as in traditional dictionary learning (Olshausen and Field, 1996):

$$\widehat{\boldsymbol{D}} \leftarrow \Pi_{C_D}(\boldsymbol{D} - \eta_D \nabla_{\boldsymbol{D}} \sum_{t=1}^{T} (\boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t)^2) = \Pi_{C_D}(\boldsymbol{D} + \eta_D \sum_{t=1}^{T} (\boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t)\boldsymbol{x}_t^T)). \tag{10}$$

As with any dictionary learning procedure, there is a fundamental ambiguity in the scale of the model parameters and coefficients. We therefore constrain each column of $\boldsymbol{D}$ to have unit-norm via the projection $\prod_{C_D}$. Note that this update rule is appropriate in both the continuous and discrete time cases.

For the dynamics we start with the discrete case. For each $\boldsymbol{f}_m$, the gradient can be computed as

$$\widehat{\boldsymbol{f}}_m \quad \leftarrow \quad \Pi_{C_{f_m}}\left(\widehat{\boldsymbol{f}}_m - \eta_f \nabla_{f_m} \sum_{t=2}^{T} \left\|\boldsymbol{x}_t - \sum_{m=1}^{M} \boldsymbol{c}_{mt} \boldsymbol{f}_m \boldsymbol{x}_{t-1}\right\|_2^2\right) \tag{11}$$

$$= \quad \Pi_{C_{f_m}}\left(\widehat{\boldsymbol{f}}_m + \eta_f \sum_{t=2}^{T} \left(\boldsymbol{c}_{mt}\left(\boldsymbol{x}_t - \widetilde{\boldsymbol{F}}_t \boldsymbol{c}_t\right) \boldsymbol{x}_{t-1}^T\right)\right). \tag{12}$$

A similar ambiguity with the latent projection $\boldsymbol{D}$ occurs between the scale of $\boldsymbol{f}_m$ and $\boldsymbol{c}_t$, so we impose the constraint that each $\boldsymbol{f}_m$ has a unit spectral radius. This constraint is applied at each optimization iteration and in practice transforms the gradient descent to a projected gradient descent.

Parameter updates for updating the continuous operators can be similarly derived over the gradient of the term involving the matrix exponential of $\sum_l \boldsymbol{g}_l c_{lt}$. Thus the derivative must be computed through the matrix exponential (see Algorithm 2), and we find it easier

---

Algorithm 1 dLDS Model Training

    Input $\boldsymbol{Y}$, λ_1, λ_2, η, M ▷ Input observations and hyperparameters
    Initialize $\{\boldsymbol{f}_m\}_{m=1:M}$, $\boldsymbol{D}$ ▷ Randomly initialize model parameters
    while not converged **do**   ▷ Iterating until convergence
        Infer $\boldsymbol{c}_t$ and $\boldsymbol{x}_t$ via Equation 8
        Update $\boldsymbol{D}$ via Equation 10
        Update each $\boldsymbol{f}_m$ via Equation 11
        if $rMSE$ does not change **then**   ▷ Check if stuck in a local minimum
            $\boldsymbol{f}_{m,ij} \leftarrow \boldsymbol{f}_{m,ij} + \boldsymbol{\xi}_{ij}$ where $\boldsymbol{\xi}_{ij} \sim \mathcal{N}(0,\sigma^2)$ ▷ Randomly perturb all elements of $\boldsymbol{f}_m$
        end if
    end while

---

to replace the strict constraint over the spectral radius with a looser regularization over the Frobenius norm $\|\boldsymbol{g}_l\|_F^2$.

$$\widehat{\boldsymbol{g}}_l \leftarrow \Pi_{C_{g_l}} \left(\widehat{\boldsymbol{g}}_l - \eta_g \nabla_{g_l} \sum_{t=2}^{T} \left\| \boldsymbol{x}_t - \mathrm{expm}\left(\sum_{l=1}^{L} \tilde{c}_{lt} \boldsymbol{g}_l \right) \boldsymbol{x}_{t-1} \right\|_2^2 + \lambda_g \sum_{l=1}^{L} \|\boldsymbol{g}_l\|_F^2 \right), \quad (13)$$

where λ_g is the weight of the Frobenius norm regularization over the dynamics matrices $\boldsymbol{g}_l$, and $\Pi_{C_{g_l}}$ is the projection operator to the constraint set C_{g_l} on $\boldsymbol{g}_l$.

The full algorithm is presented in Algorithm 1. As a special case that can be of interest in lower-dimensional data settings, we also consider the scenario in which the observation matrix, $\boldsymbol{D}$, is fixed to be the identity matrix (i.e., $\boldsymbol{y}_t = \boldsymbol{x}_t$). In this case we have direct observations of points on the manifold, and the model learning can be reduced in computational complexity (see Appendix E for the continuous and discrete cases). In all cases, we also intermittently perturb the model parameters to prevent local minima.

3.1 Interpretability for dLDS

Each individual component of the sparse decomposition is interpretable in the context of dynamical systems. For the continuous coefficients, the scale and magnitude have intuitive meaning and can modify the dictionary to account for many of the natural variations seen in time series data. Changes in speed or frequency of oscillations can be represented by increasing the coefficient's magnitude. Larger coefficients represent faster speeds and higher frequencies while smaller magnitudes represent slow speeds or low frequency oscillations. The signs of the coefficients also represent the direction of motion in the dynamical systems. A positive sign indicates movement in the direction of the learned system while a negative sign indicates reverse-time or backwards movement.

The dictionary is also readily interpreted since linear dynamical systems have analytical solutions which fully characterize its properties mathematically. The rank and structure of the linear systems can reveal the underlying dimensionality of a particular dynamic. A system with low rank and block-diagonal structure may suggest subspace-specific transitions and enables the modeling of trajectories generated from several independent subnetworks.

In addition to the interpretability of individual model components, it is also possible to reason over the model as a whole. By enforcing the sparsity constraint over the coefficients,

we encourage the model to reduce redundancy and learn dictionary elements that are statistically independent. Moreover, sparsity reduces the complexity of the representation and facilitates a clearer understanding of the underlying dynamics by allowing only a few active components at any point in time. Furthermore, we formulate a mixture model to represent dynamics, and enable the learning of shared dynamics, where multiple processes are governed by the same underlying system. In grouping together similar processes and modeling them as a single unit, we simplify the analysis and interpretation of the data.

3.2 Expressivity for dLDS

The dLDS model improves the expressiveness of traditional switching linear models by replacing the discrete dynamical states with a sparse coding model. By relaxing the discrete switches and introducing continuous coefficients, we enable the model to compactly represent a more flexible family of linear approximations for nonlinear dynamical systems during learning and inference, as compared to switch-based models. In fact there are several types of constraints we can impose on the coefficients (Fig. 1B). In the most restrictive setting, coefficients are constrained to being 1-hot vectors, which recovers the SLDS model as a special case. Relaxing the constraint on the magnitude of the coefficients results in a model with scaled coefficients over individual systems (Fig 1B). As a result of the 1-sparse coefficients, only a single system can be active, but its magnitude can be scaled positively and negatively. Time-warped AR-HMM (Costacurta et al., 2022) is an example of a model in this family and uses a single continuous coefficient to scale existing discrete states to compactly represent learned dynamics at different speeds. The dLDS model relaxes the constraint of only having a single system active at a time and allows for n-sparse representations of the linear approximation at each point in time. Allowing multiple dynamics components to be active at a time dramatically improves the expressivity of dLDS in a way that is similar to how Factorial HMMs are able to represent an exponential number of bits given a linear increase in the number of parameters (Ghahramani and Jordan, 1995). In fact, there is a factorial version of SLDS that allows for the interaction of multiple factors via their cross product Quinn et al. (2008); dLDS instead superimposes dynamics operators for improved interpretability.

For a fixed number of parameters, a more expressive model should be able to achieve lower approximation error over a broad range of systems (Dong et al., 2020). During inference, dLDS achieves lower approximation error through its continuous sparse coefficients, which can flexibly adjust to modify and combine existing learned dynamics to accurately capture new trajectories from similar systems. By contrast, switching linear systems cannot adapt learned discrete states to unseen trajectories even in the presence of small variations. During learning, dLDS offers a rich model that can efficiently capture important dynamics such as smooth transitions between states and ramping of amplitudes. The SLDS model, on the other hand, can achieve a similar expressivity but only in the limit of having a separate discrete state for every pair of consecutive time slices. Such a representation is prohibitively expensive, scaling linearly with the length of the signal and quadratically with the number of discrete states. Moreover, increasing the number of parameters in this manner would result in severe overfitting and prevent SLDS models from effectively capturing interesting structures and learning meaningful representations. Further, allowing for a large number of

discrete states introduces discontinuities on the boundaries of the linear dynamics which can be an inappropriate assumption for signals collected, e.g., from neural systems.

4. Experiments

To demonstrate the capabilities of dLDS, we present several experiments in both continuous and discrete settings. We first showcase the model's efficiency in representing continuous data (Sections 4.1, 4.2, Fig. 2). Then we demonstrate the model's ability to reconstruct discrete dynamics and recover shared underlying operators from synthetic, low-dimensional, nonlinear dynamical systems in the case of changing stability regimes (Section 4.3, Fig. 3), smooth transitions between ground truth DOs (Section 4.4, Fig. 4), independent but simultaneously observed systems (Section 4.5, Fig. 5), and across model regularization settings and initializations (Sections 4.6, 4.7, Figs. 6, 7). (Note that in all of the simulation-based experiments, we focused on the case where D is the identity matrix). Finally, we apply the model to real-world *C. elegans* calcium imaging data to uncover underlying patterns that were previously obscured (Section 4.8, Figs. 8, 9).

4.1 Continuous-time dLDS can efficiently model dynamics at different speeds

Continuous coefficients provide an effective means for efficient representation and smooth transitions between dynamical modes. Specifically, by modulating the dynamics coefficients c_t, dLDS can accurately model the same system operating at different speeds. To demonstrate the representational efficiency, we consider a two-dimensional slowly decaying spiral that progressively increases its rotational velocity by one unit of speed every 5 units of time. The trajectory sampled from this system, as depicted in Figure 2A, undergoes four speed shifts. In our study, we compare dLDS with standard switching linear models, focusing on rSLDS (Linderman et al., 2017). To ensure fair comparisons, we approximately matched the number of parameters and fit the dynamics using $L = 4$ systems. dLDS learned to represent the entire system using a single DO and effectively eliminated unused operators (Fig. 2C left column). Furthermore, dLDS used the coefficients to modify the single DO in order to capture the changes in speed (Fig. 2B bottom). By contrast, rSLDS recovered a less efficient representation by learning four non-zero discrete states, despite only using three of them during inference (Fig. 2C right column). Although rSLDS managed to capture the original system, it was unable to identify the shared behavior and used separate states to model the same dynamics at different speeds. Lastly, it is worth noting that rSLDS encountered limitations in inference due to its rigid discrete states, which obscured when the system changes between the fastest speeds as the result of approximation errors in the dynamics (Fig. 2B top). On the other hand, dLDS exhibited adaptability by flexibly adjusting its coefficients to identify speed changes, despite approximation errors in the DO.

4.2 Continuous-time dLDS efficiently represents trajectories on a sphere

Next, we highlight the ability of dLDS to model smooth transitions in dynamics of a particle moving around an axis of a spherical manifold. At regular intervals of 5 units of time, the system undergoes a random rotation $\theta \in [0, 2\pi]$ along the x-axis. Figure 2D shows a sampled path for 100 rotations while Figure 2E illustrates the continuum of ground truth rotations
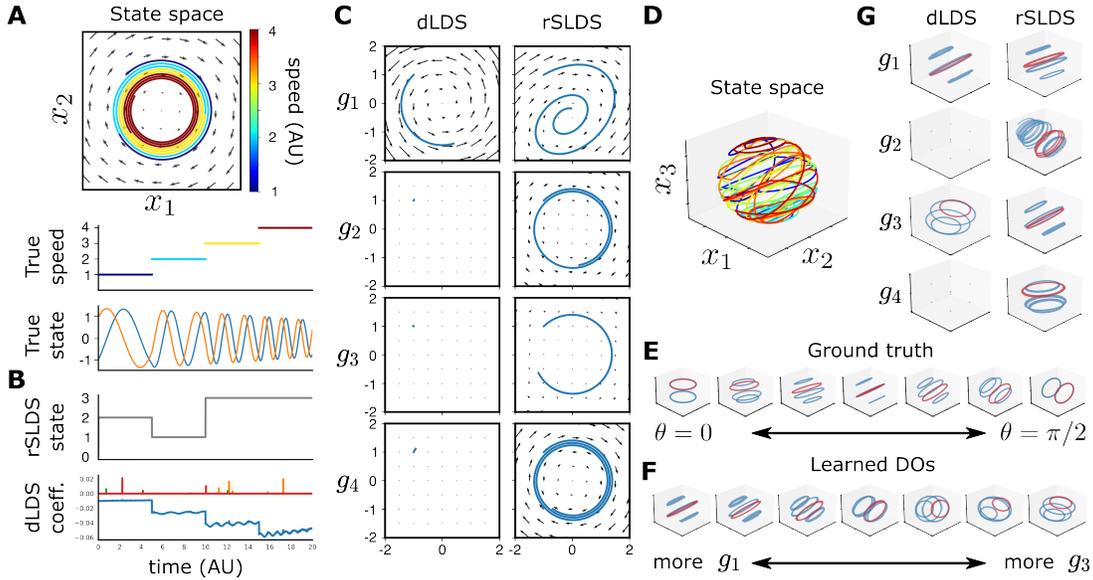
Figure 2: **Synthetic linear systems, examples of efficient representation. A:** The generated path from ground truth 2D spiral system, colored by speed in phase space (top) and unrolled over time (bottom). **B:** The inferred rSLDS discrete states (top) and the inferred dLDS coefficients (bottom). rSLDS models speed changes with three discrete states and incorrectly groups the two fastest speeds together while dLDS changes coefficients on a single dictionary element. **C:** dLDS learns a single dictionary element (left column) that can be reused while rSLDS learns redundant systems (right column). **Smooth transitions between DOs represent different paths on a spherical manifold: D:** The generated path on the ground truth 3D sphere colored to visualize progression through time. **E:** Possible ground truth rotations. Multiple traces show dynamics and a single trace is highlighted (red) for better visibility. **F:** Convex combinations of learned DOs g_1 and g_3 allow for smooth transitions along continuum of rotated systems. **G:** dLDS learns two DOs that can be combined to represent all paths on the sphere while rSLDS must learn each angle of rotation separately.

that can occur. In contrast to the previous experiment where we fit each model with the exact number of ground truth modes, we consider the situation where the models are fit with fewer systems than the total number of dynamical modes. In both switched and decomposed models, we fit the dynamics using $L = 4$ systems.

dLDS recovered two DOs, which represented the original system at two distinct angles , and shrank the remaining operators to zero (Fig. 2G, left). The two DOs offered an efficient description of the system, as they could be combined to represent the complete range of rotations (Fig. 2F). By smoothly changing the ratios of the coefficients, dLDS could smoothly transition between each DO to accurately represent intermediate trajectories. By contrast, rSLDS learned four non-zero discrete states, each capturing the original system at different

14

angles (Fig. 2G, right). Although rSLDS may provide an accurate description of the particle when its trajectory aligns with one of the learned states, trajectories that fall in between one of the learned discrete states may result in poor approximation and incorrectly inferred states. Again, rSLDS learned each discrete state separately and was unable to find the relationship between them, while dLDS learned that similar trajectories along the spherical manifold can be represented by sharing DOs.

4.3 Discrete-time dLDS flexibly models different stability regimes

By adjusting coefficients, the discrete-time dLDS model enables modeling shifts in the stability of a system (3). We consider a simple transition of a spiral system that shifts from a stable, decaying regime to an unstable, expanding regime:

$$x_t = \begin{cases} 0.99 \boldsymbol{f} \boldsymbol{x}_{t-1} & \text{if } 0 < t \leq \frac{T}{2}, \\ \frac{1}{0.99} \boldsymbol{f} \boldsymbol{x}_{t-1} & \text{if } \frac{T}{2} < t \leq T. \end{cases} \quad \boldsymbol{f} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad \text{where } \theta = \frac{\pi}{5}. \quad (14)$$

In this system, the rotational matrix $\boldsymbol{f}$ underlies the ground truth dynamics, and the matrix's coefficient $\boldsymbol{c}_t$ has a fixed value of $c_t = 0.99$ for $0 < t \leq T/2$ and then the inverse value $\boldsymbol{c}_t = 1/0.99$ for $T/2 < t \leq T$ (Fig. 3B blue). The rotational nature of $\boldsymbol{f}$ generates the spiral that initially converges exponentially and then moves outward, becoming unstable (Fig. 3C blue).

We trained both rSLDS and dLDS on a single ($M = 1$) dynamical state to emphasize the efficiency in representation. In this experiment, dLDS well approximates this original system (Fig. 3C red) by using the identified coefficients (Fig. 3B red) to adjust the single learned DO's stability. In doing so, it is able to efficiently control the direction of the rotation (inwards or outwards) while retaining the system's overall rotational behavior. Conversely, fitting rSLDS to a single discrete state conceals the presence of a shift in stability since learned dynamics are rigid during inference (Fig. 3B,C teal). In fact, we find that dLDS can recover the ground truth dynamics operator with high precision and accurately reconstruct the dynamics to a level not achievable in a switched model with a single operator.

4.4 Discrete-time dLDS efficiently reconstructs smooth transitions between ground truth dynamics

While switching models demonstrate excellent performance in capturing sudden changes in dynamics, the discrete nature of their dynamical states presents a challenge when it comes to representing seamless transitions. By contrast, discrete-time dLDS can model smooth transitions between two operational modes of a system. To illustrate this, we construct a system generated from a set of two rotational linear operators $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ (Fig. 4C top) whose coefficients $\boldsymbol{c}_{1t}$ and $\boldsymbol{c}_{2t}$ smoothly change over time following a sigmoid and a mirrored sigmoid (Fig. 4A blue), respectively: $\boldsymbol{x}_t = (\boldsymbol{c}_{1t} \boldsymbol{f}_1 + \boldsymbol{c}_{2t} \boldsymbol{f}_2) \boldsymbol{x}_{t-1}$ (Fig. 4B blue). dLDS trained with two DOs ($M = 2$) is able to both fully reconstruct the dynamics (Fig. 4B, mid row) and recover the rotational part of the ground truth operators (Fig. 4C, mid row). We observed that the differences between the lower right corner of $\boldsymbol{f}_1$ and the upper left corner of $\boldsymbol{f}_2$ in the reconstructed dLDS operators, compared to the real DOs, do not affect the reconstruction process. This is because the dynamic here ($\boldsymbol{x}$) is defined as a transition from
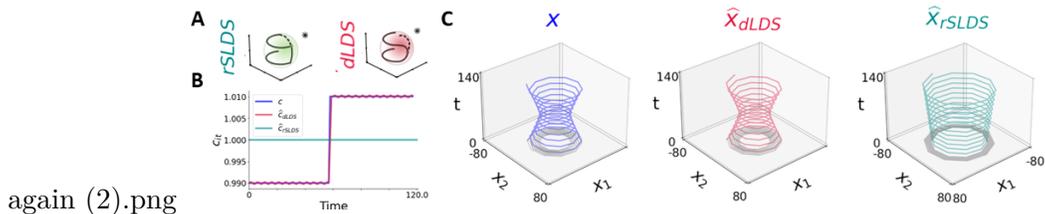
Figure 3: **dLDS captures changes in system stability.** **A:** Schematic behavior of dLDS and switching dynamics in modeling transitions between linear dynamics. Switching systems jump between dynamic states producing sharp trajectories while dLDS can smoothly change the DO coefficients to capture gradual transitions in the system. **B:** The ground truth dynamics over time (blue, top), versus the recovered dynamics by dLDS (middle, red) and rSLDS (bottom, teal). **C:** Comparison of the generated coefficients (blue) versus the dLDS recovered coefficients (red), and the rSLDS coefficients (teal).

a horizontal to a vertical spiral, resulting in $\boldsymbol{x}_3$ and $\boldsymbol{x}_1$ values that are close to zero when $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are respectively active. When trained with two discrete states, rSLDS, by definition, cannot reconstruct such dynamics well (Fig. 4B teal). Indeed, the learned discrete states in rSLDS (Fig. 4C, bottom) did not recover the ground truth systems and instead learned two similar discrete states that both incorrectly combined the dynamical modes together. As a result, rSLDS inaccurately inferred erratic jumps between the two similar systems in an attempt to capture the smooth transition (Fig. 4A, teal). In general, it is unclear how to appropriately set the number of discrete states to get an accurate representation of smooth transitions for switching models. Even if given an adequate number of discrete states, the intermediate stages can experience overfitting as a result of the low number of samples in the transition interval. dLDS addresses these issues through the continuous coefficients, which flexibly modify existing dynamics to account for minor variations and enable learning of shared structure even in the intermediate stages of a smooth transition.

4.5 Discrete-time dLDS disentangles simultaneously observed systems

One benefit of decomposing dynamics is the ability to account for multiplexed sub-systems within the same recorded data. Consider a single recording of a neural population that consists of two distinct sub-populations, denoted as a and b. The sub-population a consists of N_1 neurons, while the sub-population b consists of N_2 neurons, and the recordings of these sub-populations over time are represented by $\boldsymbol{x}_t^a \in \mathbb{R}^{N_1 \times T}$ and $\boldsymbol{x}_t^b \in \mathbb{R}^{N_2 \times T}$, respectively. To represent the full data, we concatenate these two populations' recordings vertically, resulting in $\boldsymbol{x}_t = [\boldsymbol{x}_t^a; \boldsymbol{x}_t^b] \in \mathbb{R}^{(N_1+N_2) \times T}$. If the switches of both sub-populations are not synchronized, even when both locally adhere to a switched LDS model, an SLDS model would need to transition whenever either subsystem a *or* subsystem b switches. Consequently, the SLDS model would have to consider all possible coexistence scenarios of these two systems collectively. Thus, the timescales of the unique populations are lost, and the interpretability
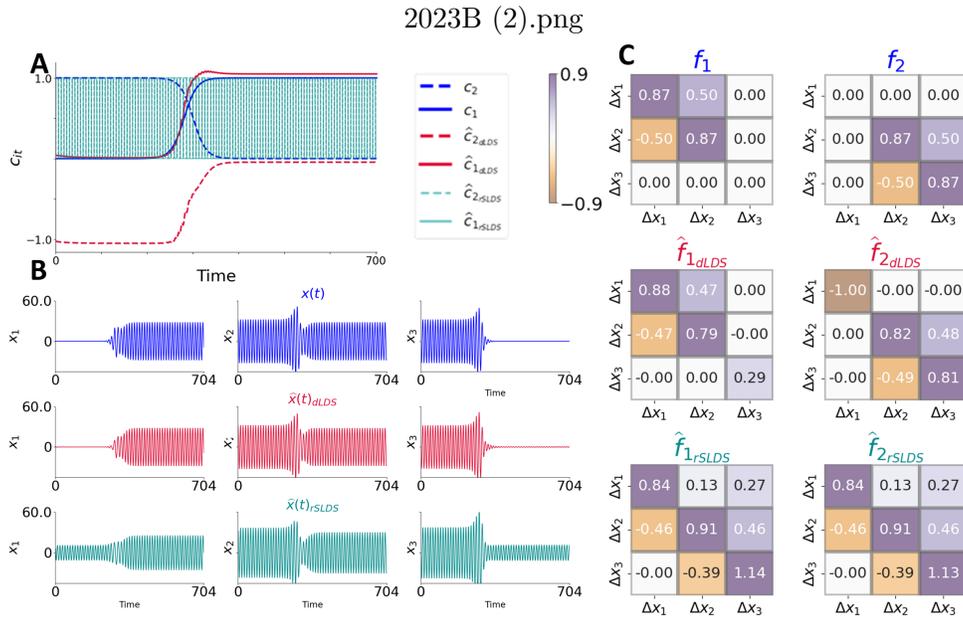
16

Figure 4: **dLDS captures rotational dynamics from 2 underlying rotational systems. A:** The ground truth dynamics (top, blue), compared to the recovered dynamics by dLDS (middle, red) and rSLDS (right, teal). Each column corresponds to a different axis. **B:** Comparison of the generated coefficients (blue), the dLDS recovered coefficients (red), and the rSLDS coefficients (teal). **C:** The ground-truth DOs (top, blue) versus the DOs recovered by dLDS (middle, red) and rSLDS (bottom, teal).

of the combined recordings becomes muddled. dLDS, however, can naturally account for such settings by summing operators that guide only the dynamics of subsets of the system.

To demonstrate this effect, we simulate a ten-dimensional state $\boldsymbol{x}_t$, where the first five elements of the vector constitute "population a" and the last five elements "population b". We then generate six ground truth DOs (Fig 5A, top row), three of which only act on population a (maroon), and the other three on population b (cyan). To simplify the setting, each population constituted an SLDS system and switched abruptly between one of its three systems, or went silent with no dynamics active (Fig. 5C blue). We repeated this process, generating 50 draws of the process with different initial states and different switch patterns.

We fit both dLDS and rSLDS to the generated data, providing both with a maximum of 15 dynamical operators ($M = 15$). The dLDS model was able to recover a basis of operators that match the ground truth synthetic operators (Fig. 5A, middle row). Specifically, the recovered dynamical operators were well localized in terms of being localized to either the upper or lower diagonal block, indicating that dLDS learned the underlying block structure of the system. Furthermore, the dynamical operators for each learned system matched one of the ground truth operators with a correlation of ≈ 1 (Fig. 5E, left), and the coefficients directly convey when each subsystem switched independent of the other subsystem (Fig. 5B,

17

red). By contrast, rSLDS learned dynamical systems that had both upper and lower blocks on the diagonal active, indicating that it was learning combined dynamics across both subsystems (Fig. 5A, bottom row) that do not convey the true underlying independence between the subsystems. In fact, when provided with more discrete states than necessary, rSLDS learns several redundant dynamics as a result of its inability to share information between its states. For instance, $\boldsymbol{f}_4$ and $\boldsymbol{f}_{14}$ resemble each other as well as $\boldsymbol{f}_9$ and $\boldsymbol{f}_{10}$, and $\boldsymbol{f}_5$ $\boldsymbol{f}_{11}$ and $\boldsymbol{f}_{12}$ (Fig. 5E, right). These redundant dynamics cause unstable inference of the discrete states and result in inappropriate jumps between states with similar dynamics in the middle of the true switching intervals (Fig. 5B, teal).

4.6 Discrete-time dLDS efficiently represents a range of simulated behaviors and aligns DO coefficients to the axes for interpretability

We will now illustrate how dLDS can effectively capture intuitive aspects of nonlinear models of the underlying dynamics of biological systems. Specifically, we consider the FitzHugh–Nagumo (FHN) model: a 2D excitable-oscillatory dynamical system model that describes the temporal behavior of a nerve membrane potential in response to a stimulus (FitzHugh, 1961) and is a simplification of the Hodgkin-Huxley model (Hodgkin and Huxley, 1952). The FHN model is defined by a pair of conjugated differential equations denoted by

$$\frac{\partial v}{\partial t} = v - \frac{v^3}{3} - w + I_{ext} \qquad \tau \frac{\partial w}{\partial t} = v + a - bw, \tag{15}$$

where the state-space (v, w) are the nerve membrane potential (v) and the voltage recovery variable (w), respectively. I_{ext} is an external stimulus, and τ, a and b are model parameters. Here we set $I_{\text{ext}} = 0.5$, $\tau = 20$, $a = 0.8$, and $b = 0.7$, and consider a sample initialized at $(v_0, w_0) = (-0.5, 0)$. This example demonstrates the effect of sparsity regularization on the model's representation of the dynamics. We ran dLDS with and without the ℓ_1 regularization over $\boldsymbol{c}_t$ to test how sparsity changes the interpretability of the system (Fig. 6) and show that including the ℓ_1 regularization term on the coefficients promoted their orientation towards the axes (Fig. 6B,G). For both the regularized and unregularized cases, the transitions of coefficients between the quadrants of the coefficients' space enable our model to well represent a diverse set of voltage behaviors (e.g., depolarization, repolarization, etc.) with only $M = 2$ operators (Fig. 6D,I). The smooth transitions using the same two operators in both forward and reverse directions cannot be captured by switched systems. This result is consistent with the $\boldsymbol{c}_t$-plane visualization in the 'arbitrary' versus 'sparse' cases (e.g., Fig. 1B).

4.7 Discrete-time dLDS recovers shared dynamics between different simulation initializations

We next demonstrate the ability of dLDS to capture the complex dynamics of the Lorenz attractor, a well-known system that exhibits chaotic behavior and is often used as a benchmark for dynamical models. The Lorenz attractor is a nonlinear chaotic system governed by

$$\frac{\partial x}{\partial t} = \alpha(y - x) \qquad \frac{\partial y}{\partial t} = x(\beta - z) - y \qquad \frac{\partial z}{\partial t} = xy - \gamma z. \tag{16}$$
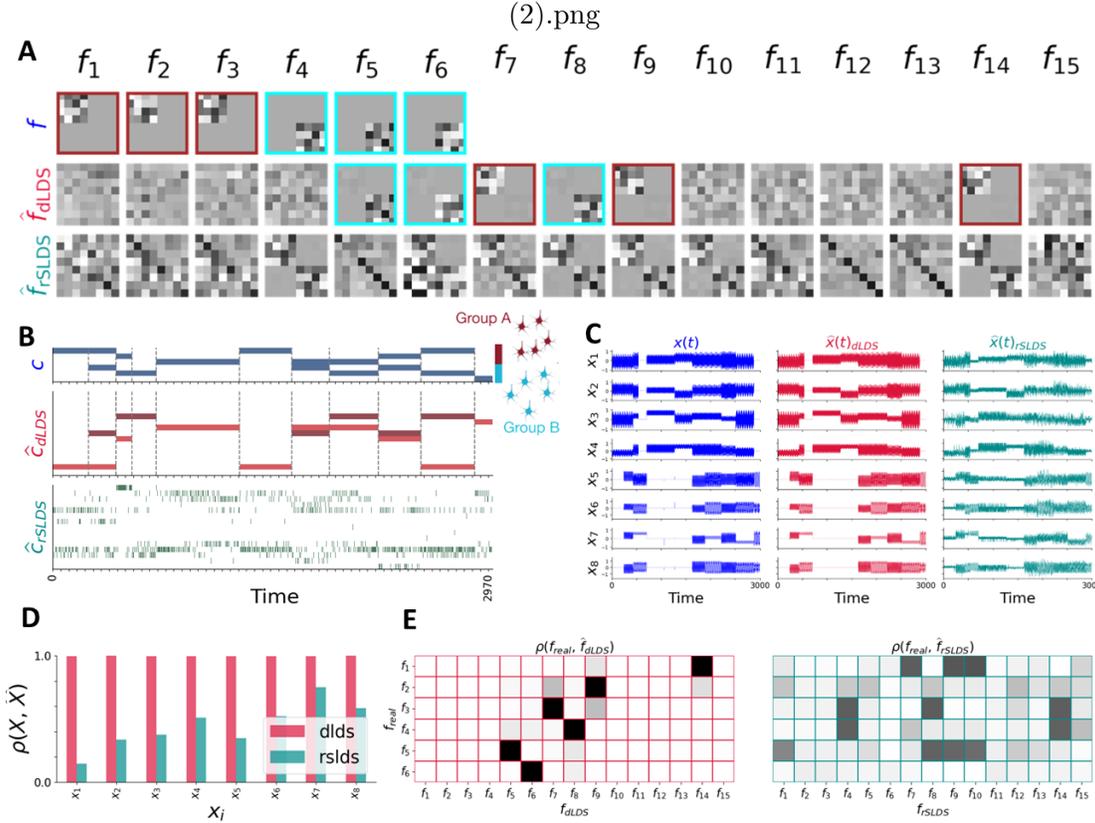
Figure 5: **dLDS identifies independently evolving groups from combined time series. A:** The ground truth DOs displaying the block structure of the data (top), DOs recovered by dLDS (middle), DOs recovered by rSLDS (bottom). Framed dLDS DOs have a perfect correlation to the "true" DOs for both populations (cyan, maroon). **B:** The ground truth coefficients (top), dLDS recovered coefficients (middle) and rSLDS recovered coefficients (bottom). dLDS can accurately recover the structure of the ground truth and nullifies redundant coefficients, while rSLDS combines dynamics across the blocks. **C:** True generated dynamics (left), dLDS reconstruction (middle), and rSLDS reconstruction (right). **D:** The data reconstruction correlations for dLDS (red) and rSLDS (teal) with the ground truth. dLDS achieves perfect reconstruction. **E:** The correlations of each true DO (rows) with each recovered DO (columns). dLDS (left) recovers all true DOs (each row presents exactly one black cell), while rSLDS tends to combine DOs across the two independent groups.

For our dLDS experiment, we set $\alpha = 10$; $\beta = 25$; $\gamma = 2.67$. Here, we demonstrate dLDS' inference capability to model unseen data of the same dynamics with different initializations, using the learned set of operators. By training dLDS on a set of Lorenz attractors with different initial conditions, and then benchmarking on unseen attractors that originated from unused initial conditions, we highlight the ability of operators learned via dLDS to generalize
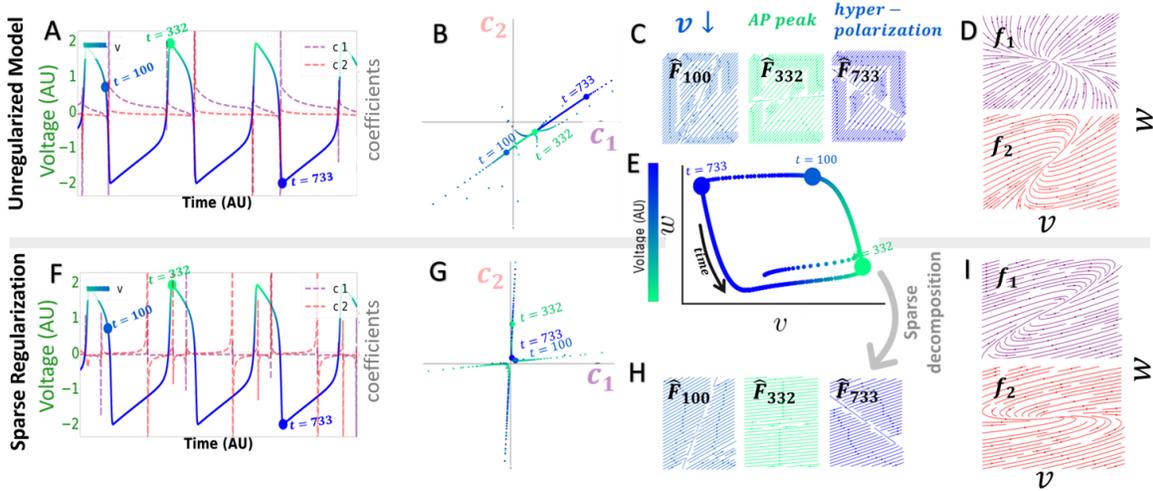
Figure 6: **Comparison between the unregularized dLDS (top row) and the sparse dLDS model (bottom row) for the FitzHugh-Nagumo oscillator. A & F:** The temporal evolution of the membrane voltage (blue-green) and of the dynamics coefficients. Three points of interest were highlighted to showcase our model's capability to capture various behaviors with only 2 basis components. These points of interest include 1) the action potential (AP) repolarization ($t = 100$), 2) the AP peak ($t = 332$), and 3) hyper-polarization ($t = 733$). **B & G:** Comparison between the coefficients' space of the unregularized and the regularized case. For the unregularized dLDS model, the model coefficients (c_t) can occupy any location in space, and need not be on the axes. By contrast, when adding regularization to the model, sparse coefficients lie near the axes. **C & H:** The learned reconstructed dynamics F_t at the time points of interest, highlighting dLDS's ability to infer more distinct phases than the number of sub-dynamics (f_i)—a capability that is not available to linear or switching models. **D & I:** Stream-plots of the basis operators learned by dLDS (f_1 and f_2). **E:** The phase-space plot of the FHN model (v-w space), with time points of interest highlighted.

past individual trajectories. dLDS was able to reconstruct the unseen Lorenz attractors with high accuracy (Fig. 7B bottom row). Although the model was allowed to use up to seven operators, no more than 2-3 operators were active at a time (Fig. 7A bottom row), maintaining the balance between interpretability and expressivity. Additionally, the clear association between the patterns of the DOs' coefficients and the location on the attractor highlights the model's interpretability, making it a powerful tool for modeling nonlinear dynamical systems. In Figure 7B, the stars on the Lorenz attractors mark the peaks of c_1 (cyan), c_2 (green), c_5 (pink), c_7 (red), highlighting their repeated and consistent locations along the Lorenz manifold. The link between the different dimensions of the Lorenz and the c values can be easily seen (Fig. 7A,C). Figure 7C depicts the link between the Lorenz's first dimension and the coefficients representation, by coloring the three most dominant
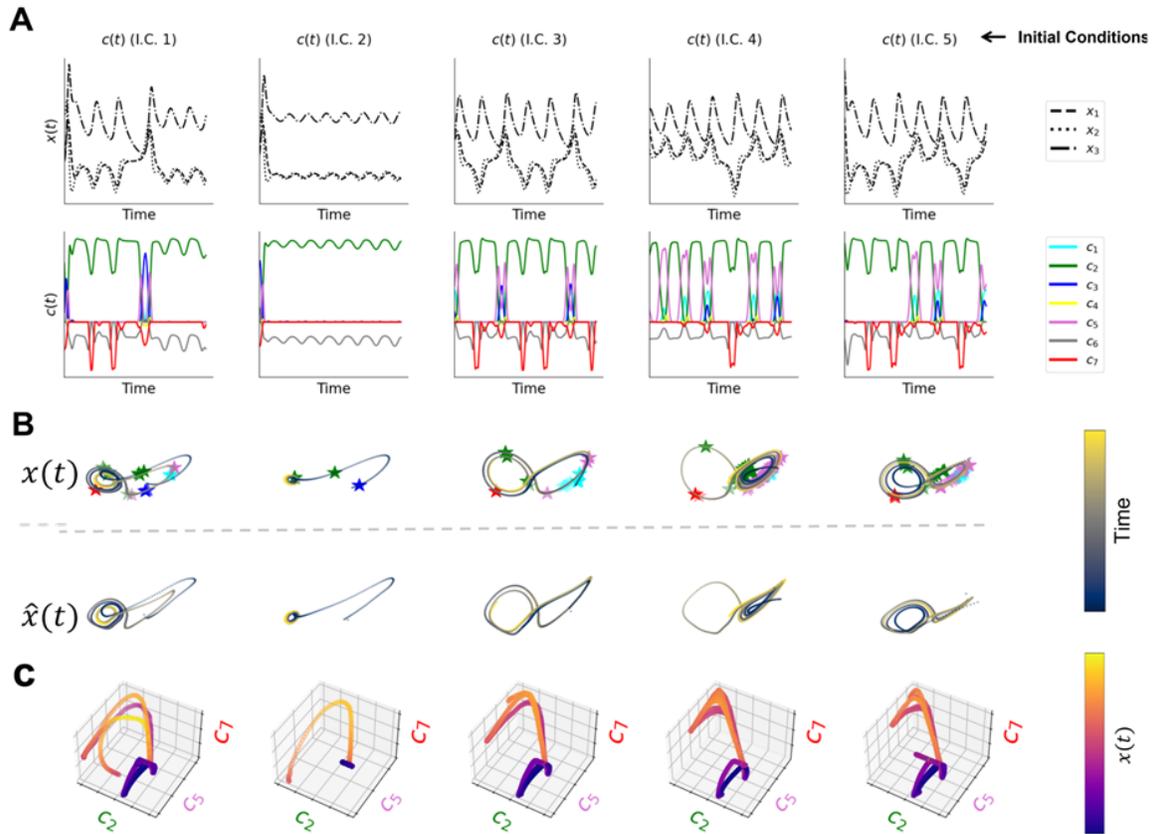
(3)B.png

Figure 7: **Demonstrating model generalizability by applying dLDS to the Lorenz attractor with unseen initial conditions (I.C.s). A:** DO coefficients (bottom) plotted against the Lorenz state values (top) show coefficient patterns associated with different sections of the Lorenz attractor. **B:** Comparison between unseen Lorenz trajectories (top row) and their dLDS reconstructions (bottom row). Stars indicate the locations of peaks of different operators' coefficients, highlighting how the underlying components localize on the attractor. **C:** The three most active dynamics coefficients are color-coded by the corresponding x_1 values. Regions visibly associated with different ranges of x_1 highlight repeated patterns dynamics in the Lorenz attractor.

operators' coefficients according to the Lorenz's first dimension values. One can see the clear change of colors in different areas of the c_t space.

4.8 Discrete-time dLDS identifies latent dynamics in *C. elegans* data

Finally, we apply dLDS to "whole brain" *C. elegans* calcium imaging recordings (Kato et al., 2015; Zimmer, 2021; Linderman et al., 2019) (Fig. 9A). We benchmarked dLDS model against rSLDS in the experiments where Kato et al. (2015) inferred the immobilized worms' pirouetting behavior under varying oxygen concentrations (four states: forward crawl, reverse crawl, sustained reverse crawl, and post-reversal turn) from the neural activity. dLDS revealed obscured differences in the neural dynamics during different behavioral states for further exploration.

First, we observed that the map D from the latent space to the neural observations was able to highlight neurons that heavily overlapped with known neurons of interest and group them together in the latent space. For example, the stronger weights in D highlighted AVAL and RIML, interneurons involved in a backward motion, and VA01, a motor neuron (Fig. 8, neurons of interest in red). AVAL and RIML were both represented in latent dimensions 1, 2, 4, 6, and 8, whereas AIBL, an interneuron that instead promotes turns, was represented in latent dimensions 1, 2, 5, 8, and 10. VA01 was most represented in latent dimensions 3, 4, 6, 8, and 10, which suggests some shared utilization of this motor neuron for backward motion and some for turning. This mapping can be used in future studies to reconstruct measures of functional connectivity from latent space dynamics back to the neural activity (ambient) space.

Second, we noticed obscured within-state evolving dynamics (Fig. 9B). This may indicate a gradual change in the worm's internal state or behavior in the middle of these discretely-labeled behavioral states; however, it is difficult to identify a behavioral or internal state correlated with the data provided. These results indicate that dLDS can be applicable to datasets with continuous descriptions of behavior as opposed to discrete behavioral state labels.

Third, while rSLDS recapitulated 1-sparse behavioral state classification with high fidelity, we observed periods where the inferred discrete states oscillated unrealistically, e.g., between the post-reversal turn (rSLDS state 4) and forward crawling (rSLDS state 1). By contrast, dLDS adjusted the dynamics coefficients without changing the usual relationships between the dynamics. The traces maintained a standard motif shape and order of purple/yellow/blue/red from top to bottom, with blue activating during periods of high oscillation in the rSLDS classification (Fig. 9B). However, not every behavioral state transition from state 4 to state 1 ("4-to-1") induced high oscillation in the rSLDS classification; we take as controls those time intervals where the rSLDS inferred states were stable, which we defined as a remaining constant for at least 10 time points in a row. Focusing on a specific example, the stable rSLDS control (Fig. 9C,E,G) and high rSLDS oscillation (Fig. 9D,F,H) intervals both appeared to reflect 4-to-1 behavioral state transitions according to the ground-truth inferred behavior labels from Kato et al. (2015). However, during the high oscillation interval, in the dLDS latent states, the yellow trace stayed on, unlike in the control interval (Fig. 9E,F). In the dLDS dynamics coefficients (Fig. 9C,D), several dynamics were active at the beginning of State 4 in the control interval but not in the high oscillation interval (264-266s vs. 367-370s). Moreover, in the high oscillation interval, the dynamic corresponding to the orange trace stayed active, unlike in the control interval. Both dLDS and rSLDS identified some differences in neural activity between these
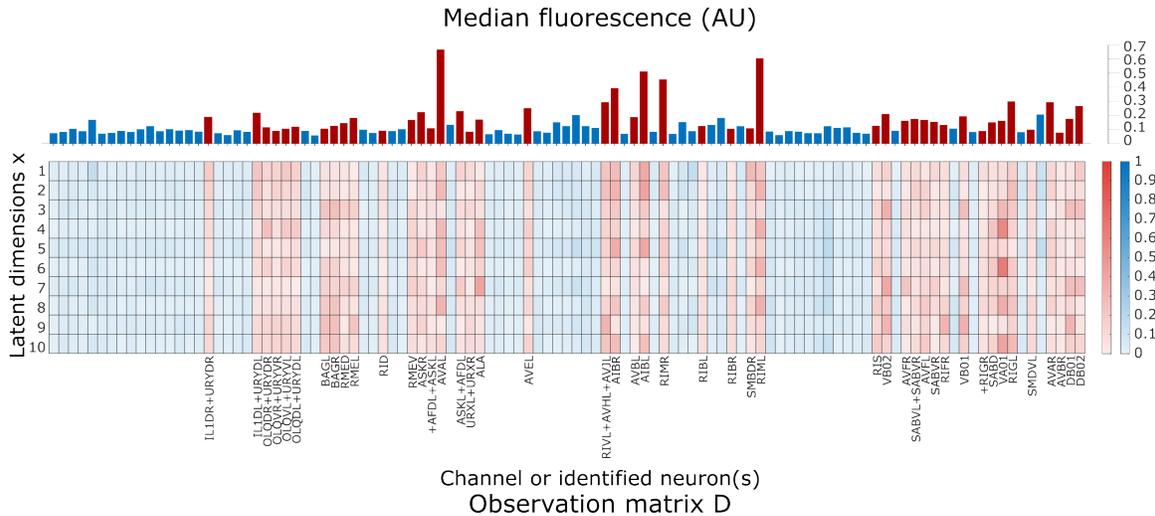
Figure 8: **dLDS identifies important neurons in *C. elegans* data via the observation matrix *D*.** Top: Median fluorescence values for each recording channel (Worm 1, stimulated case). Channels labeled by neuron names represent individual neurons identified by Kato *et al.* (Kato et al., 2015; Zimmer, 2021). Bottom: The observation matrix *D* mapping the latent states *x* to the fluorescence *y*.

two time intervals. However, dLDS offers a finer resolution lens for parsing and interpreting the dynamics that create these differences.

4.9 Additional experiments

We further tested dLDS in a higher-dimensional discrete-time setting (12x12 image patches), demonstrating the ability to recover ground truth dynamics in simple permutation tests, as well as identifying a decomposable transformation of image patches learned from natural videos (Appendix J).

5. Discussion, Limitations, and Future Work

In this work, we present a manifold-flow-inspired model of learning decomposed Linear Dynamical Systems. Our proposed model, dLDS, expands on the idea of switching linear systems to a model where linear combinations of a finite dictionary of systems can represent a richer set of dynamics. Unlike more unstructured dynamics models (Harris et al., 2021; Proctor et al., 2016; Luttinen et al., 2014), the sparsity specifically provides an intuitive sense of interpretability for the dynamics that we observe over multiple examples. We present the model in both continuous- and discrete-time settings and provide both synthetic and real-data examples of learning the dynamics dictionary in practice.

Our model-learning provides an avenue by which we can estimate dynamical systems that are locally linear at each point, but whose parameters change over time. This enables both the
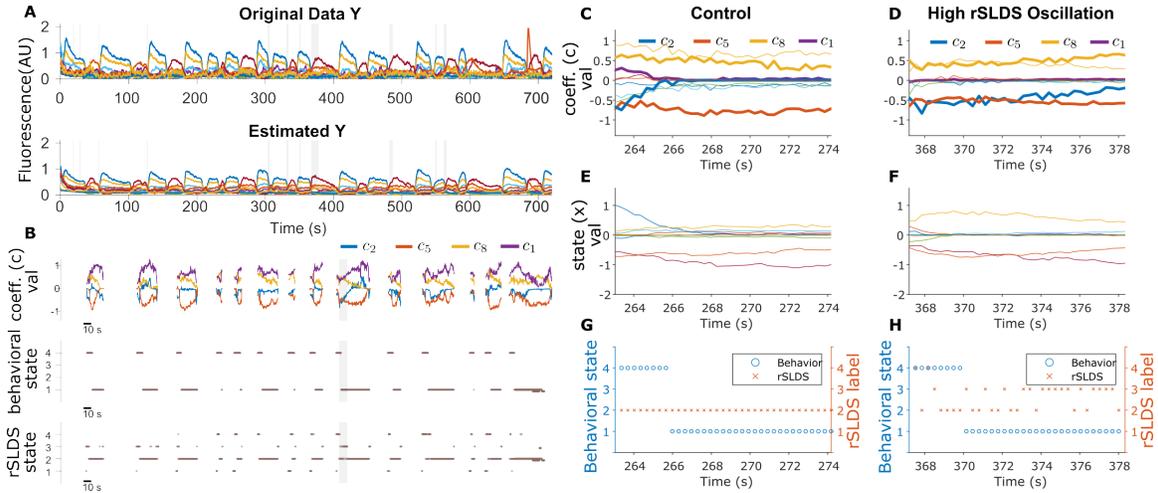
Figure 9: **Demixed dynamics in a single *C. elegans* in the stimulated experimental setting, with 4 behavioral states. A:** Neuronal activity for a subset of neurons and their reconstruction ($R^2 = 0.74$) using dLDS with 10 DOs. **B:** Regions corresponding to inferred behavioral states 4 and 1, with the top 4 dynamics coefficients for these epochs. Gray highlights indicate periods of high oscillation in the rSLDS state. By contrast, the dLDS dynamics coefficients during the 4-to-1 state transition varied more smoothly. c_1, c_2, c_5, and c_8 were highly active, with a quantitative change at the transition between states 4 and 1. **C-H:** A time interval where rSLDS exhibits state oscillations during a transition from state 4 to 1 (right column, **D, F, H**) juxtaposed with a control interval with stable rSLDS states during a similar transition (left column, **C, E, G**). Compared are the dLDS *c* traces (**C, D**), dLDS *x* traces (**E, F**), and Behavioral states (blue circles) labeled by Kato *et al.* (2015) vs. rSLDS states (red crosses, not matched) (**G, H**).

estimation and tracking of non-stationary dynamics, as well as the approximation of nonlinear dynamics by treating the nonlinearity as a temporal non-stationarity. dLDS benefits from the efficiency, sparsity, and convergence guarantees of the BPDN-DF algorithm Charles et al. (2016), which filters forward in time; however, future implementation of a smoothing solution (e.g., forward-backward, or joint local estimation of multiple states) may improve model accuracy. In addition, the learning procedure requires tuning a number of parameters, including regularization parameters and the number of latent and dynamics coefficients. Important targets of future work will be to develop approaches to automatically set the model hyperparameters. Furthermore, we aim to investigate additional aspects of the model such as its stability, its performance under different statistical assumptions (e.g., Poisson), and its predictive capability.

In this work, we primarily focused on the linear observation model (Eq. (1)) common in many neural data analysis methods. However, it is known that for manifold structured

data, nonlinear models can often further reduce data dimensionality. Thus future methods should implement nonlinear observation models to reduce the dimensionality of the space in which the dynamics dictionary is learned, e.g., learning DOs in the latent space of an auto-encoder (Connor et al., 2021b).

The examples we present highlight a series of both simple and more complex systems. We note that even the simpler systems represent plausible system behaviors in real data that are not captured well by existing techniques. For example, the evolution of a system at different speeds indeed targets a weakness of switching models. However, it is not a contrived example. In fact, there are noted examples of neural systems that have warped timescales representing elongated processing of the same computation (Wang et al., 2017).

One nuanced point that is critical to address is that our framework can model both nonlinear and non-stationary systems. In particular, dLDS achieves this ability, over models that purely target nonlinear systems (Brunton et al., 2016), by modeling the nonlinearities as nonstationarities through the time-varying coefficients c_t. The benefit of this model is that for nonlinear systems we expect, as we have seen in the FHN and Lorenz cases, regularities in the system behavior that can be tracked by analyzing the dynamics coefficients. Nonstationarities can also be identified in a similar way by noting non-periodic changes in c_t or changes in c_t in nearby locations in state space at different times. This nuance is important, as it is often of interest if the observed changes in neural dynamics are due to changes in the system behavior or due to the intrinsic nonlinear interactions. Further work should continue to try to identify markers to better disambiguate these cases.

In our method, the coefficients (c's) and dynamics values (x's) are co-inferred to address a joint LASSO problem. While there is no overt connection between them, an implicit relationship is established through joint optimization. Specifically, the association between the data and x and c can be expressed as $\tilde{y} \approx A\tilde{x}$, where $\tilde{y} = [y; [0]_{p,1}]$, $A = [[D, [0]_{M,M}]; [-I_{p \times p}, Fx]]$, and $\tilde{x} = [x; c]$. Here, F_x denotes the concatenated columns of $f_i x_{t-1}$. The dependencies manifest in the lower part of the A matrix. However, these dependencies lack the explicit or localized nature observed in rSLDS. We have included additional discussions on this limitation in the corresponding section.

It is crucial to emphasize that dLDS is not conceived as a generative model in the sense that each sample represents a fully realistic trajectory. We draw an analogy to the total variation (TV) model for image reconstruction. Natural images exhibit low TV norms, but random samples of images with low TV norms may not necessarily be natural images. Similarly, samples from our model are not intended for trajectory generation; rather, they provide essential constraints for inference, enabling the learning of meaningful trajectories based on the data. In other words, the prior is selected to yield a robust posterior rather than being the most accurate prior. This choice facilitates a reduction in variance without introducing extraneous bias. One advantage is that inadequate data or flawed inference parameters will not yield correct-looking model outputs, adding an extra layer of error correction and validation to the interpretation of the results.

We acknowledge that other advancements in modeling may aim to establish a more explicit relationship between c and x. However, we assert that such endeavors go beyond the core model's scope, as they would introduce additional assumptions and biases that are more application-specific. Future research should certainly explore these directions.

One direction that we have not explored in this work is the inclusion of a driving term in the dynamics equation, e.g., in discrete time by adding a term $\boldsymbol{Bu}_t$ such that $\boldsymbol{x}_t = \boldsymbol{F}_t\boldsymbol{x}_t + \boldsymbol{Bu}_t$. Here $\boldsymbol{u}_t$ can represent a set of control variables, and $\boldsymbol{B}$ is the projection of those inputs into the latent space. While this term is trivial to add to our basic framework (in fact, similar update rules can be included to learn $\boldsymbol{B}$), we focused here on the most basic case to demonstrate the utility of the dynamics model alone. Future work will focus on this term, in particular as it can be used to account for system inputs such as sensory stimuli in modeling neural dynamics.

Similarly, it may be possible in future work to explore additional constraints on the latent states and dynamics coefficients such that they are explicitly, rather than implicitly, connected. For example, certain dynamics operators may be confined to certain parts of the state space by grouping them together in subsets during inference.

Limitations: In addition to the above discussion points, other current limitations of our work are that under some parameter regimes, the estimates for the dynamics coefficients can be unstable. Additionally, we note that we observe better performance when normalizing and centering the datasets. These shortcomings recommend future directions for preventing severe instability in the inference step by building in bias terms, e.g., similar to the learned offset terms in other LDS approaches to approximating nonlinear systems (Linderman et al., 2017). We note that our model is not proposed to replace rSLDS, but to reveal how the activity of the same elements smoothly evolves over time, rather than a binary 'switch' between elements. The two methods will have different strengths depending on the assumed dynamics of a given system.

6. Data and Code Availability

The code for both the discrete and the continuous models can be found at https://github.com/dLDS-Decomposed-Linear-Dynamics. The discrete code can also be pip-installed using the **dLDS-discrete** Python package, as described in `https://pypi.org/project/dLDS-discrete-2022/`. Additionally, an interactive Python notebook of the discrete model visualization is available at `https://colab.research.google.com/drive/1PgskOtYoLL83ecXz_ALXk9oLofR2AeRf?usp=sharing`.

Data from Kato et al. (2015) were obtained from the Open Science Framework site (Zimmer, 2021). The SSM Python package from the Linderman lab was used to run rSLDS (Linderman et al., 2020).

7. Acknowledgments

References

Guy A. Ackerson and K. S. Fu. On State Estimation in Switching Environments. *IEEE Transactions on Automatic Control*, AC-15(1):10–17, 1970. ISSN 15582523. doi: 10.1109/TAC.1970.1099359.

M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

Mukund Balasubramanian and Eric L Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and tracking*. Artech House, Boston, MA, 1990.

Gabriel Barello, Adam S Charles, and Jonathan W Pillow. Sparse-coding variational auto-encoders. *bioRxiv*, page 399246, 2018.

Stephen R Becker, Emmanuel J Candès, and Michael C Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical programming computation*, 3(3):165–218, 2011.

Hadas Benisty, Andrew H. Moberly, Sweyta Lohani, Daniel Barson, Ronald R. Coifman, Gal Mishne, Jessica A. Cardin, and Michael J. Higley. Rapid fluctuations in functional connectivity of cortical networks encode spontaneous behavior. *bioRxiv*, 2021. doi: 10.1101/2021.08.15.456390.

Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

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

Chaw-Bing Chang and Michael Athans. State estimation for discrete systems. *IEEE Transactions on Aerospace and Electronic Systems*, 14(3):418–425, 1978. ISSN 02755823. doi: 10.5711/1082598316323.

A. S. Charles, A. Balavoine, and C. J. Rozell. Dynamic filtering of time-varying sparse signals via ℓ_1 minimization. *IEEE Transactions on Signal Processing*, 64(21):5644–5656, 2016.

Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

Marissa Connor and Christopher Rozell. Representing closed transformation paths in encoded network latent space. *Association for the Advancement of Artificial Intelligence (AAAI)*, 34:3666–3675, Apr 2020. doi: 10.1609/aaai.v34i04.5775. URL https://ojs.aaai.org/index.php/AAAI/article/view/5775.

Marissa Connor, Gregory Canal, and Christopher Rozell. Variational autoencoder with learned latent structure. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2359–2367. PMLR, April 2021a. URL `https://proceedings.mlr.press/v130/connor21a.html`.

Marissa Connor, Kion Fallah, and Christopher Rozell. Learning identity-preserving transformations on data manifolds, 2021b. URL `https://arxiv.org/abs/2106.12096`.

Julia Costacurta, Lea Duncker, Blue Sheffer, Winthrop Gillis, Caleb Weinreb, Jeffrey Markowitz, Sandeep R Datta, Alex Williams, and Scott Linderman. Distinguishing discrete and continuous behavioral variability using warped autoregressive hmms. *Advances in Neural Information Processing Systems*, 35:23838–23850, 2022.

Benjamin Culpepper and Bruno Olshausen. Learning transport operators for image manifolds. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2009/file/a1d50185e7426cbb0acad1e6ca74b9aa-Paper.pdf`.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

Jeffrey Demas, Jason Manley, Frank Tejera, Kevin Barber, Hyewon Kim, Francisca Martínez Traub, Brandon Chen, and Alipasha Vaziri. High-speed, cortex-wide volumetric recording of neuroactivity at cellular resolution using light beads microscopy. *Nature Methods*, 18 (9):1103–1111, 2021.

Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In *International conference on machine learning*, pages 2627–2637. PMLR, 2020.

Kion Fallah and Christopher J Rozell. Variational sparse coding with learned thresholding. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6034–6058. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/fallah22a.html`.

Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

Emily Fox, Erik Sudderth, Michael Jordan, and Alan Willsky. Nonparametric bayesian learning of switching linear dynamical systems. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL `https://proceedings.neurips.cc/paper/2008/file/950a4152c2b4aa3ad78bdd6b366cc179-Paper.pdf`.

Juan A Gallego, Matthew G Perich, Lee E Miller, and Sara A Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.

Zoubin Ghahramani and Geoffrey E. Hinton. Switching state-space models. Technical report, King's College Road, Toronto M5S 3H5, 1996.

Zoubin Ghahramani and Michael Jordan. Factorial hidden markov models. *Advances in Neural Information Processing Systems*, 8, 1995.

Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

Matthew Golub, Steven Chase, and Byron Yu. Learning an internal dynamics model from control demonstration. In *International Conference on Machine Learning*, pages 606–614. PMLR, 2013.

James D. Hamilton. Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45(1-2):39–70, 1990. ISSN 03044076. doi: 10.1016/0304-4076(90)90093-9.

Kuan Han, Haiguang Wen, Junxing Shi, Kun-Han Lu, Yizhen Zhang, Di Fu, and Zhongming Liu. Variational autoencoder: An unsupervised model for encoding and decoding fmri activity in visual cortex. *NeuroImage*, 198:125–136, 2019.

Kameron Decker Harris, Aleksandr Aravkin, Rajesh Rao, and Bingni Wen Brunton. Time-varying autoregression with low-rank tensors. *SIAM Journal on Applied Dynamical Systems*, 20(4):2335–2358, 2021.

Simon Haykin. *Adaptive Filter Theory (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. ISBN 0-13-322760-X.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500, 1952.

Saul Kato, Harris S. Kaplan, Tina Schrödel, Susanne Skora, Theodore H. Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global Brain Dynamics Embed the Motor Command Sequence of Caenorhabditis elegans. *Cell*, 163(3):656–669, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.09.034.

Mohammad Reza Keshtkaran and Chethan Pandarinath. Enabling hyperparameter optimization in sequential autoencoders for spiking neural data. *Advances in Neural Information Processing Systems*, 32, 2019.

Michael Kleinman, Chandramouli Chandrasekaran, and Jonathan Kao. A mechanistic multi-area recurrent network model of decision-making. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Scott Linderman, Benjamin Antin, David Zoltowski, and Joshua Glaser. SSM: Bayesian Learning and Inference for State Space Models (version 0.0.1, 10 2020. URL `https://github.com/lindermanlab/ssm`.

Scott W. Linderman, Annika L. A. Nichols, David M. Blei, Manuel Zimmer, and Liam Paninski. Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans. *bioRxiv*, page 621540, 2019. doi: 10.1101/621540. URL `https://www.biorxiv.org/content/10.1101/621540v1.abstract?%3Fcollection=`.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Jaakko Luttinen, Tapani Raiko, and Alexander Ilin. Linear state-space model with time-varying dynamics. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14*, pages 338–353. Springer, 2014.

Gal Mishne, Ronen Talmon, Ron Meir, Jackie Schiller, Maria Lavzin, Uri Dubin, and Ronald R. Coifman. Hierarchical coupled-geometry analysis for neuronal structure and activity pattern discovery. *IEEE Journal of Selected Topics in Signal Processing*, 10(7): 1238–1253, 2016. doi: 10.1109/JSTSP.2016.2602061.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.

K. P. Murphy. Switching Kalman filters. Technical report, Technical Report, UC Berkeley, 1998.

Josue Nassar, Scott W Linderman, Monica Bugallo, and Il Memming Park. Tree-structured recurrent switching linear dynamical systems for multi-scale modeling. *arXiv preprint arXiv:1811.12386*, 2018.

Edward H Nieh, Manuel Schottdorf, Nicolas W Freeman, Ryan J Low, Sam Lewallen, Sue Ann Koay, Lucas Pinto, Jeffrey L Gauthier, Carlos D Brody, and David W Tank. Geometry of abstract learned knowledge in the hippocampus. *Nature*, 595(7865):80–84, 2021.

B. A. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(13):607–609, June 1996.

Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

John A Quinn, Christopher K I Williams, and Neil Mcintosh. Factorial switching linear dynamical systems applied to physiological condition monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017.

Matteo Ravasi and Ivan Vasconcelos. Pylops—a linear-operator python library for scalable algebra and optimization. *SoftwareX*, 11:100361, 2020.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Omid G Sani, Hamidreza Abbaspourazad, Yan T Wong, Bijan Pesaran, and Maryam M Shanechi. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, 24(1):140–149, 2021a.

Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Where is all the nonlinearity: flexible nonlinear modeling of behaviorally relevant neural dynamics using recurrent neural networks. *bioRxiv*, 2021b.

Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current opinion in neurobiology*, 55:103–111, 2019.

Marc-Andre Schulz, BT Yeo, Joshua T Vogelstein, Janaina Mourao-Miranada, Jakob N Kather, Konrad Kording, Blake Richards, and Danilo Bzdok. Different scaling of linear models and deep learning in ukbiobank brain images versus machine-learning datasets. *Nature communications*, 11(1):1–15, 2020.

Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021.

David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020. ISSN 15454126. doi: 10.1146/annurev-neuro-092619-094115.

Jing Wang, Devika Narain, Eghbal A. Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience 2017 21:1*, 21:102–110, 12 2017. ISSN 1546-1726. doi: 10.1038/s41593-017-0028-6. URL `https://www.nature.com/articles/s41593-017-0028-6`.

Anqi Wu, Nicholas A Roy, Stephen Keeley, and Jonathan W Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. *Advances in neural information processing systems*, 30, 2017.

Byron M Yu, Caleb Kemere, Gopal Santhanam, Afsheen Afshar, Stephen I Ryu, Teresa H Meng, Maneesh Sahani, and Krishna V Shenoy. Mixture of trajectory models for neural decoding of goal-directed movements. *Journal of neurophysiology*, 97(5):3763–3780, 2007.

Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in neural information processing systems*, 21, 2008.

Manuel Zimmer. Kato2015 whole brain imaging data, January 2021. URL `osf.io/2395t`.

Appendix A. Discrete-time Approximation of Continuous-time dLDS

We can derive the discrete-time model by taking the Taylor Series expansion of matrix exponential from the continuous-time latent transition (Eq. (3)),

$$
\begin{aligned}
\boldsymbol{x}_{t+1} &= \mathrm{expm}\left(\sum_{l=1}^{L} c_{ct}\boldsymbol{g}_l\right)\boldsymbol{x}_t + \boldsymbol{\nu}_t \\
&= \left[\boldsymbol{I} + \sum_{l=1}^{L} c_{ct}\boldsymbol{g}_l + \frac{1}{2}\left(\sum_{l=1}^{L} c_{ct}\boldsymbol{g}_l\right)^2 + \frac{1}{6}\left(\sum_{l=1}^{L} c_{ct}\boldsymbol{g}_l\right)^3 + \cdots\right]\boldsymbol{x}_t + \boldsymbol{\nu}_t \\
&\approx \left[\boldsymbol{I} + \sum_{l=1}^{L} c_{ct}\boldsymbol{g}_l\right]\boldsymbol{x}_t + \boldsymbol{\nu}_t,
\end{aligned}
$$

where for simplicity, we assume that samples are equally spaced such that $\tau = 1$. The last approximation comes from considering only the linear approximation in the Taylor series expansion.

To use the BPDN-DF inference procedure, we must rewrite it in the form

$$
\boldsymbol{x}_{t+1} - \boldsymbol{x}_t = \sum_{l=1}^{L} \boldsymbol{g}_l \boldsymbol{x}_t c_{ct} + \boldsymbol{\nu}_t = \widetilde{\boldsymbol{G}}_t c + \boldsymbol{\nu}_t,
$$

where the columns of $\widetilde{\boldsymbol{G}}_t$ are given by $\boldsymbol{g}_l\boldsymbol{x}_t$. However, the main challenge with working in this form is that the variance of the left hand side increases as a result of the noise,

$$
\mathrm{var}(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t) = \mathrm{var}(\boldsymbol{\nu}_{t+1} - \boldsymbol{\nu}_t) = \mathrm{var}(\boldsymbol{\nu}_{t+1}) + \mathrm{var}(\boldsymbol{\nu}_t) - 2\mathrm{cov}(\boldsymbol{\nu}_{t+1}, \boldsymbol{\nu}_t) = 2\mathrm{var}(\boldsymbol{\nu}_t),
$$

since $\mathrm{var}(\boldsymbol{\nu}_{t+1}) = \mathrm{var}(\boldsymbol{\nu}_t)$ and $\mathrm{cov}(\boldsymbol{\nu}_{t+1}, \boldsymbol{\nu}_t) = 0$ for independent random noise variables. Moreover, since $\boldsymbol{x}_{t+1}$ and $\boldsymbol{x}_t$ are correlated random variables, we expect that $\mathbb{E}[\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|]$ is significantly smaller than $\mathbb{E}[\|\boldsymbol{x}_t\|]$. Combining these two results, we see that the finite differences approach to BPDN-DF reduces the magnitude of the signal while also doubling the variance of the noise, resulting in the degraded accuracy of coefficient estimates. Instead, we define a discrete-time dynamics operator $\mathrm{expm}(\sum_{l=1}^{L} c_{lt}\boldsymbol{g}_l) \approx \boldsymbol{F}_t$ which allows us to circumvent the issue of low signal-to-noise during coefficient inference in noisy datasets.

Appendix B. Demonstrating dLDS behavior, using the Lorenz attractor

As we increase the sparsity regularization in dLDS, the temporal changes in the coefficients become sharper, but still maintain smooth transitions between temporal epochs dominated by individual dynamical systems. Thus dLDS enables the learning of dynamics under many conditions from smoothly changing dynamics to sharper switching-like dynamics. The choice of sparsity level is thus a model hyperparameter that can be set by the users for specific applications.

Appendix C. Duffing Dynamics and Linearization Components Recovery

As an example of the application of our model's abilities, we demonstrate its capability in recovering the latent components of a linearization of the Duffing oscillator. The Duffing
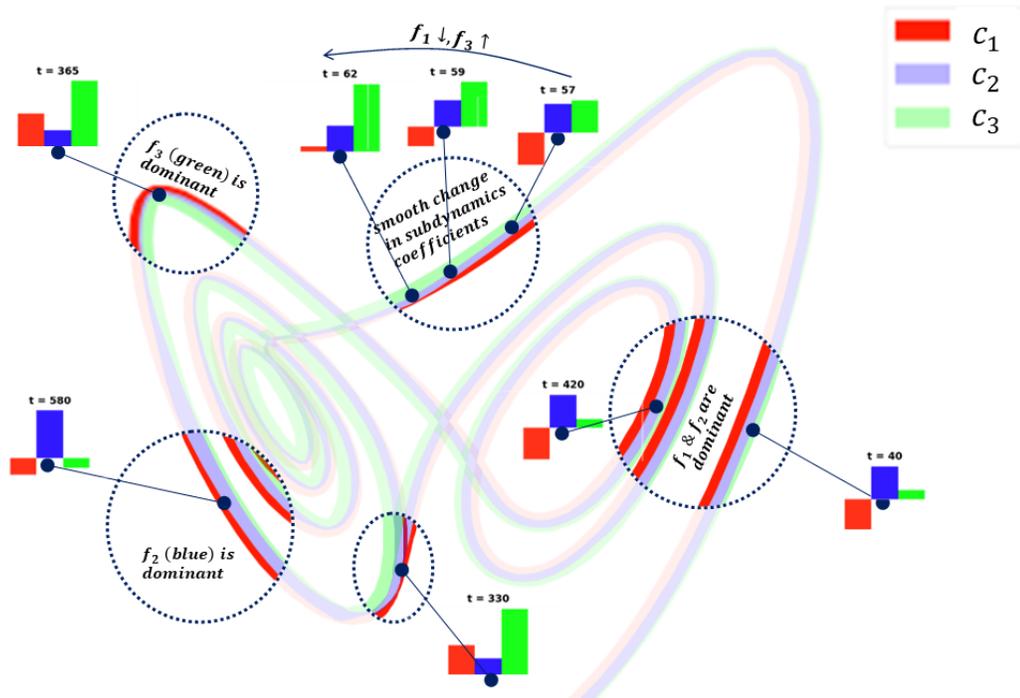
Figure 10: A 2-dimensional projection of the Lorenz reconstruction results on the x-z axes, for the unregularized case. Learning three dynamical systems to approximate the Lorenz attractor motion reveals that dLDS can track smoothly changing dynamics by titrating the contributions of separate dynamical systems. We note that even in the case where the values of c_t are not regularized e.g., data in this plot, different dynamics become more or less dominant over different areas within the attractor.
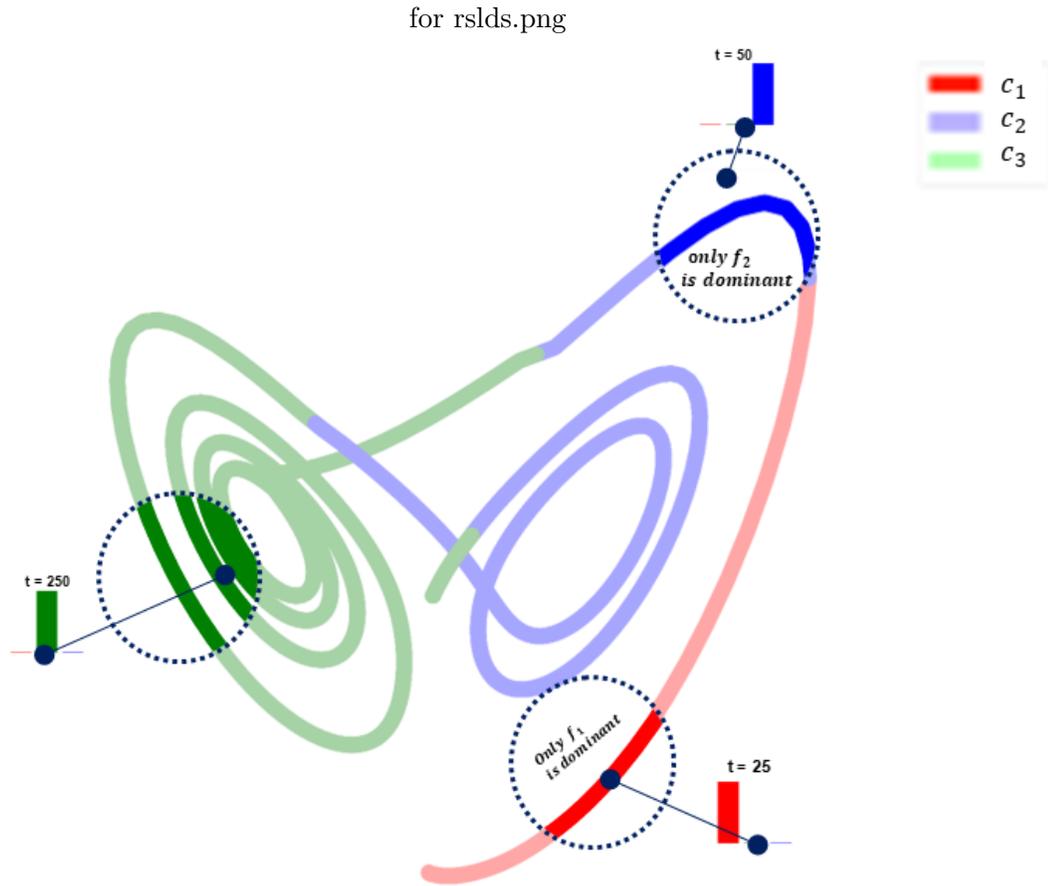
for rslds.png

Figure 11: A 2-dimensional projection of the Lorenz reconstruction results as obtained by rSLDS. In contrast to dLDS, here the transitions between different representations are sudden (via switching), resulting in only one dynamical system active at each time point.

3

oscillator is a classical example of a nonlinear, second-order differential equation that exhibits chaotic behavior. It is described by the following equation:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t), \tag{17}$$

where α, β, and γ are parameters that control the behavior of the oscillator, and $\gamma\cos(\omega t)$ represents a periodic driving force with amplitude γ and frequency ω.

To apply our model to the Duffing oscillator, we define $y = \dot{x}$. We can then rewrite the Duffing oscillator equation in terms of y as follows:

$$\dot{y} + \delta y + \alpha x + \beta x^3 = \gamma\cos(\omega t). \tag{18}$$

We can represent the Duffing oscillator using a decomposed linear dynamical systems model by discretizing the system over time. We use the state vector $[x_t, y_t]$ to represent the state of the system at time t. The discrete-time model of the Duffing oscillator is given by the following equation:

$$\begin{bmatrix} \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\alpha - \beta x_t^2 & -\delta \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma\cos(\omega t) \end{bmatrix}. \tag{19}$$

We can then apply our model to this discrete-time Duffing oscillator equation. We estimate the transition matrix A_t by fitting our model to the time series data. We can decompose A_t into two matrices, $F_t = F_{t,1} + F_{t,2}$, where $F_{t,1}$ and $F_{t,2}$ correspond to distinct physical processes.

Specifically, the matrix $F_{t,1} = \begin{bmatrix} 1 & \Delta t \\ -\alpha\Delta t & 1 - \delta\Delta t \end{bmatrix}$ describes the linear dynamics of the system, while $F_{t,2} = \begin{bmatrix} 0 & 0 \\ -\beta x_t^2\Delta t & 0 \end{bmatrix}$ describes the nonlinearity of the system.

Using these matrices, we can write the discrete-time Duffing oscillator equation in matrix multiplication form as follows:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \left(\begin{bmatrix} 1 & \Delta t \\ -\alpha\Delta t & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\beta x_t^2\Delta t & 0 \end{bmatrix} \right) \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} 0 \\ -\cos(t) \end{bmatrix}, \tag{20}$$

where $\cos(t)$ represents the cosine function evaluated at time t. By estimating the matrices $F_{t,1}$ and $F_{t,2}$ using our model, we can recover the underlying linear and nonlinear components of the Duffing oscillator dynamics.

By applying dLDS to the Duffing oscillator, we highlight the model's capacity to reconstruct nonlinear dynamics using basic linear components whose coefficients change over time and to recover the ground-truth basic linear elements that underlie these complex dynamics. These results emphasize the potential of dLDS for extracting valuable insights from intricate systems, by facilitating the reconstruction of nonlinear dynamics with basic linear components, and by retrieving the underlying basic linear elements.
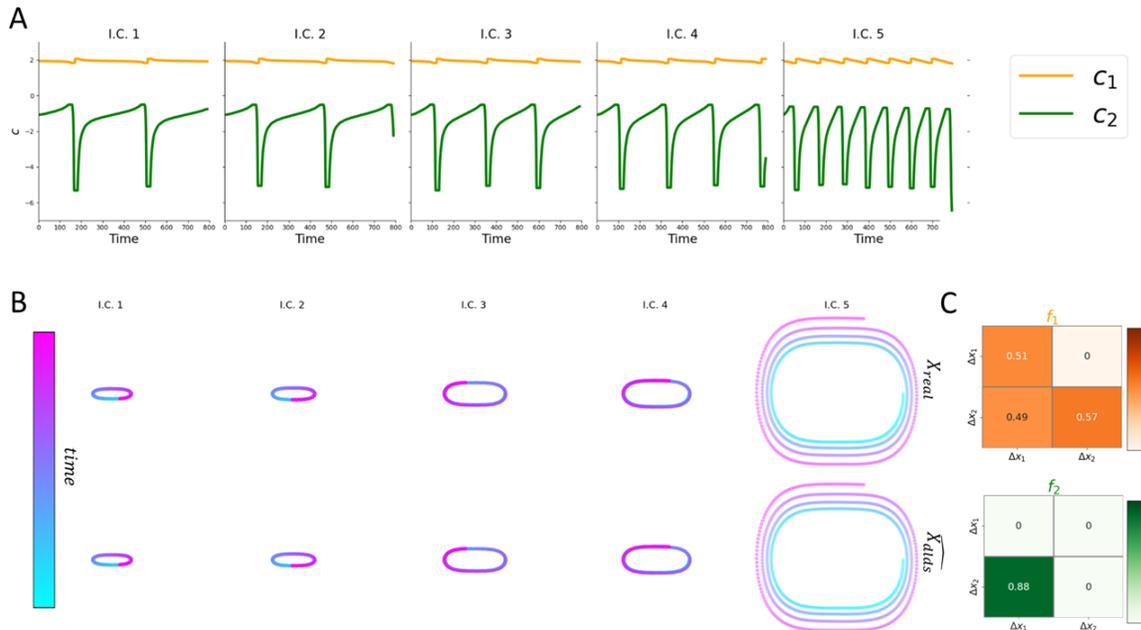
Figure 12: **Results of dLDS analysis on the Duffing oscillator.** A) Time traces (orange and green) of each dynamic operators (DO) for different initial conditions (I.C.) in separate subplots. B) Comparison between the real dynamics of the Duffing oscillator and the reconstructed dynamics by dLDS for each I.C. The color represents time. C) Heatmaps of each DO, where the first DO shows the main diagonal and other non-zero elements, as its coefficient in A is 2, while the second DO shows the lower left diagonal element and all other elements are zero.

Appendix D. Continuous-time dLDS learning

For completeness, we provide here in Algorithm 2 the dLDS learning algorithm for continuous time dynamics. In this work for continuous time we assume that $D = I$.

Appendix E. Special case of no observation model

For completeness, we provide here in Algorithm 3 the dLDS learning algorithm for discrete-time dynamics under the condition that $D = I$. This special case is less computationally intensive and enables the learning of dynamics in the native data space.

Appendix F. A note on LASSO solvers

We found that the correlations between $\boldsymbol{f}_m \widehat{\boldsymbol{x}}_t$ could be large for certain time points. These correlations meant that some ℓ_1 regularize least-squares solvers would exhibit instability during learning. We noticed this in particular for the primary LASSO functions in both MAT-

---

Algorithm 2 Continuous-time dLDS training assuming $\boldsymbol{D} = \boldsymbol{I}$

   Input Y, λ_1, λ_2, η, L ▷ Data, penalty on c, penalty on G, learning rate, n dictionary elements

   Initialize $\{\boldsymbol{G}_i\}_{l=1:L}, D$ ▷ Randomly initialize sub-dynamics, observation model
   Normalize each dynamical system by dividing by its spectral radius

   while not converged **do**   ▷ Iterating until convergence
      for t in $\{0, \ldots, T-1\}$ **do**   ▷ Run for every time-adjacent data pair
         Randomly sample $\{\hat{c}_l(t)\}_{l=1}^{L}$
         Update $\{\hat{c}_l(t)\}_{l=1}^{L} \leftarrow \arg\min_{\{c_l(t)\}} E$
         Update $\hat{G} \leftarrow \arg\min_G E$
      end for
   end while

Algorithm 3 Dynamics dictionary learning for $\boldsymbol{D} = \boldsymbol{I}$

   Input Y, λ_1, η, M ▷ Input observations and hyperparameters
   Initialize $\{\boldsymbol{f}_m\}_{m=1:M}$,   ▷ Initialize dynamics dictionary randomly
   Normalize each $\boldsymbol{f}_m$ to unit spectral radius

   while not converged **do**   ▷ Iterating until convergence
      $\widehat{c}_t = \arg\min_c \|\boldsymbol{y}_t - \sum_m \boldsymbol{f}_m c_{mt} \boldsymbol{y}_{t-1}\| + \lambda_1 \|\boldsymbol{c}\|_1$
      Update each $\boldsymbol{f}_m$ via Equation 11
      Normalize each $\boldsymbol{f}_m$ to unit operator norm
      if $rMSE$ does not change **then**   ▷ Check if the algorithm stuck in a local minimum
         $f_m \leftarrow f_m + \nu$ ▷ Add random noise to each $\{f_m\}$
      end if
   end while

LAB and Python. We found that instead, the SPGL1 solver of the pylops package (Ravasi and Vasconcelos, 2020) was more robust in Python, and the TFOCS software [1] (Becker et al., 2011) was more robust in MATLAB. While TFOCS solves the LASSO program directly, SPGL1 solves a slightly modified version:

$$\hat{c}_t = \arg\min_c \left\| x_{t+1} - \sum_{m=1}^{M} \boldsymbol{f}_m c_m(t) * x_t \right\|_2 \qquad \text{s.t.} \qquad ||c_t||_1 \leq \tau. \tag{21}$$

Importantly, despite presenting only the SPGL1 results in the above paper, the Python code enables the user to choose from a wide range of solvers (including FISTA, ISTA, Sklearn LASSO (Pedregosa et al., 2011), OMP), and the decision of which solver to use is up to the user and should depend on the data properties and the user's goals in running the model.

---

1. https://github.com/cvxr/TFOCS

Experiment	L	λ_G	λ_c	η_G	η_c	γ
Speed	4	1	1e-1	1e-1	1e-2	0.985
Rotation	4	20	8e-2	5e-3	1e-2	0.985

Table 1: **Continuous-time experiment model parameters.** L is the total number of dictionary elements initialized. λ_G is regularization placed on dictionary elements; we control this value through the weight decay. λ_c is the regularization strength placed on the coefficients. η_G and η_c are the learning rates for dictionary elements and coefficients, respectively. γ is the rate for the learning rate decay schedule.

Appendix G. Additional information on experiments

G.1 Continuous-time model parameters

Continuous-time models were implemented in PyTorch. Model parameters for each experiment are shown in Table 1.

G.2 FitzHugh-Nagumo (discrete-time model)

We used the Python discrete code for the FHN case. The iterative model ran until convergence (reconstructed error $<$ 1e-8) or until reaching a maximum of 6,000 iterations. The ground truth for the FHN dynamics was created based on (15), using 1000 samples with time intervals of 0.2 (s.t. $t_{\max} = 200$). We used $M = 2$ dictionary elements, an initial value of $\eta = 30$ (from (11)), while its decay rate over the training iterations was set to $\gamma = 0.99$; The standard-deviation of the perturbations added randomly to each $\boldsymbol{f}_m$ in case of local minimum, as described in Algorithm 1, was set to 0.1.

For the regularized dynamics case, the hyperparameters of the SPGLl solver, "iter lim", the maximum number of solver iterations in each coefficients updating step, was set to 10, and τ (from (21)) was set to 0.3.

For the unregularized case, the following pseudo-inverse was used for each the updating step of the coefficients in each time point and in each iteration:

$$
\begin{aligned}
\widetilde{\boldsymbol{F_t}} &= [\boldsymbol{f}_1\boldsymbol{x}_t, \boldsymbol{f}_2\boldsymbol{x}_t, ..., \boldsymbol{f}_M\boldsymbol{x}_t] \in \mathbb{R}^{(2 \times M)} \\
\boldsymbol{x}_{t+1} &= \widetilde{F_t}\boldsymbol{c}_t \\
\widehat{\boldsymbol{c}_t} &= \widetilde{\boldsymbol{F_t}}^{\dagger}\boldsymbol{x}_{t+1},
\end{aligned}
\tag{22}
$$

where $\dagger$ denotes the pseudo-inverse.

G.3 The Lorenz attractor (discrete-time model)

We used the Python discrete code for the Lorenz attractor case, with different options for the number of dictionary elements. In the paper, the results of our model for $M = 5$ dictionary elements are presented, along the corresponding results from the reference (rSLDS with 5 discrete states).

As in the FHN case, the dLDS iterative model ran until convergence (reconstructed error $<$ 1e-8) or until reaching a maximum of 6,000 iterations. The ground truth for the Lorenz

	rSLDS		dLDS	
Experiment	R	R^2	R	R^2
Synthetic speed	0.99	0.99	0.99	0.99
Synthetic rotation	0.99	0.99	0.99	0.99
Unregularized FHN	1.0	1.0	1.0	1.0
Regularized FHN	-	-	1.0	1.0
Unregularized Lorenz	0.99	0.99	1.0	1.0
Regularized Lorenz	-	-	0.93	0.70
C. elegans	0.96	0.92	0.86	0.74

Table 2: **Pearson correlation and R^2 values for one-step prediction for rSLDS vs dLDS, for each experiment.**

attractor was created based on Equation (16), using 1000 samples with time intervals of 0.01 (s.t. $t_{\max} = 10$). Similarly to the FHN, the initial value of η was set to 30, while its decay rate over the training iterations was set to 0.99. The standard-deviation of the perturbations added randomly to each f_i in case of local minimum, as described in Algorithm 1, was set to 0.1.

The updating step of the coefficients was performed similarly to the updating methodology presented above for the FHN case, according to which in the unregularized case, the coefficients were updated using the pseudo-inverse, and for the regularized case, the coefficients were updated using the SPGL1 solver.

With respect to the hyperparameters of the SPGLl solver in the Lorenz case, "iter lim" was set to 10, and τ (from Equation (21)) was set to 0.55. A comparison of model performance measures between dLDS and rSLDS is shown in Table 2.

G.4 Summary of model performance comparisons

Table 2 compares rSLDS and dLDS summary statistics for each experiment.

Appendix H. Comparison between dLDS and rSLDS for the FHN model

In the main text we focus on comparing dLDS for the FHN oscillator under regularized and unregularized conditions to emphasize the role of regularization over the dynamics coefficients c_t. Here we further demonstrate the comparison between dLDS to rSLDS (Fig. 13).

In contrast to the rSLDS model, for which the coefficients are binary, in our model the coefficients can take on continuous values. Hence, contrasting the observed coefficients-space spanned by rSLDS (13D) and dLDS (13E,F), in dLDS the dynamics representations are not limited to discrete locations on the axes, but can travel along them, resulting in a more flexible representation without the need to increase the number of dynamical systems learned. Specifically, we identify that while rSLDS learns slightly varying dynamical systems, while dLDS learns reorientations to different axes which more smoothly trade off with each other as the system rotates about the attractor.
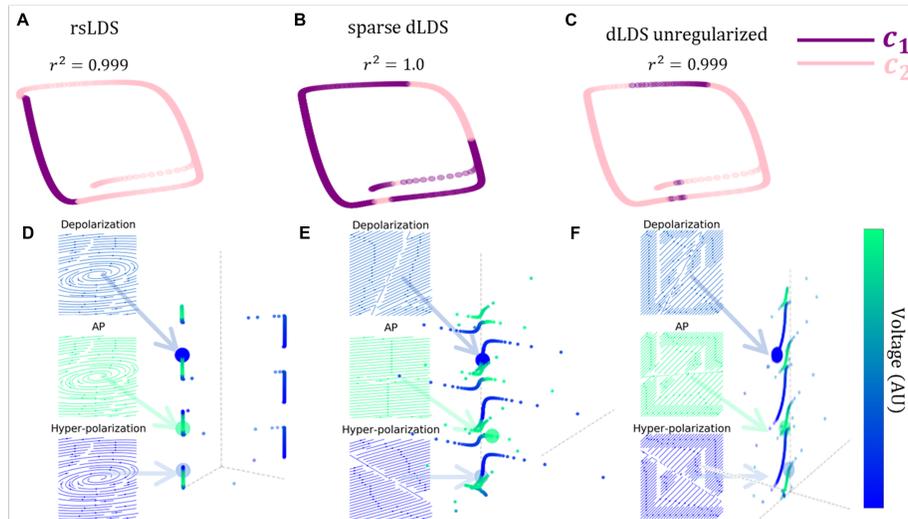
Figure 13: **Learned representations using dLDS and rSLDS for the FHN oscillator.** Note three time points of interest: repolarization, action potential peak, and hyperpolarization. With $M = 2$, dLDS can reconstruct the three distinct states, while rSLDS can only capture two reconstructed states in this case. A, B, C: All three models were able to reconstruct the FHN dynamics. D: The coefficients obtained by the rSLDS are restricted to the axes, resulting in no more than two distinct reconstructed states to describe the action potential cycle. E: Although the coefficients of the regularized dLDS tend to live on the axes due to the sparse regularization, this constraint is softer than of the rSLDS (in which living outside the axes is not possible). F: Coefficients space obtained for the unregularized dLDS model. Most coefficients do not necessarily live on the axes, since no regularization was applied.

Additionally, as the ℓ_1 regularization over $\boldsymbol{c}_t$ in dLDS increases, the coefficients become more similar to those obtained by rSDLS, namely, more restricted to the axes. Thus, modulating the regularization in our model makes possible the creation and exploration of a continuum of representations whose coefficients-space range from switched systems (high regularization) to arbitrary structured (unregularized), as described in Figure 1B.

Appendix I. Sparse video example

To test the model in a sparse higher-dimensional setting we simulate a single dynamics function is present ($M = 1$) and the sparsity dictionary as the canonical basis ($\boldsymbol{D} = \boldsymbol{I}$). This test will check if our algorithm can accomplish simple system identification as a special case. We modeled the single dynamics function as a permutation matrix concatenated with a scaling matrix, i.e., signal coefficients move around and may be scaled (Fig. 14A). The learned and true models are a very close match, differing by only a permutation and sign change (the same ambiguity present in all dictionary learning methods).
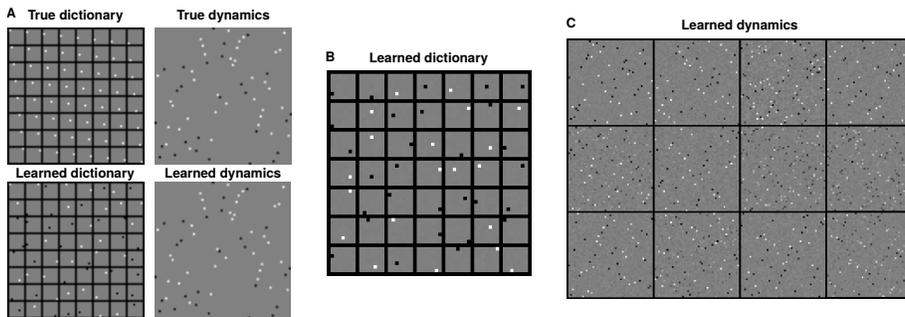
Figure 14: **Pixel permutation example.** A: Example test set consisting of a sparse number of pixels being permuted via an unknown permutation matrix. dLDS recovers in this case both the pixel-sparse dictionary as well as the ground truth permutation matrix. B: For a more complex example where multiple permutation matrices may be used (sometimes in tandem to split or merge pixels), the correct pixel-sparse dictionary is again learned. C: For the same example as B, the set of permutation matrices is learned, capturing the underlying dynamics.

In a more complex simulation, we simulate a dictionary of twelve distinct scaled permutation functions, only two of which are used at any time step (i.e. the sparsity of c_t is two). This system induces complex, highly non-stationary dynamics. Figure 14B,C depicts the results of the learning procedure, demonstrating that the sparsity dictionary is again learned up to a permutation and sign change, and the learned dynamics functions are again close matches to the true dynamics (i.e., we recover 12 scaled permutation matrices).

Appendix J. BBC video example

To test dLDS on higher-dimensional real data, we learn a dynamics dictionary for natural video sequences. For computational considerations we restricted our algorithm to learn representations of 12x12 pixel patches, and learned a 4x overcomplete sparsity dictionary concurrently with 25 576x576 DOs. As no ground truth is available for video sequences, we instead qualitatively explore learned dictionaries. First we note that the sparsity dictionary recovered the expected Gabor-like statistics for image patches (Olshausen and Field, 1996; Aharon et al., 2006) (Fig. 15A). This result matches the intuition that the spatial statistics are not qualitatively changed by including the temporal model. To assess the dynamics we note that despite the high-dimensional nature of the data, the learned dynamics were relatively low-dimensional (rank 2-10), with one exception that had almost full rank (Fig. 15B). Additionally, the top eigenvectors tend to be correlated, but not overly so. The correlations cluster around ~ 0.2 with some correlations as high as 0.8 (Fig. 15C). This indicates that the learned functions are neither independent nor identical. Thus, interactions between the dLDS DOs permit flexible nonlinear behaviors.

Three such examples are shown in Figure 15D-F. First we project a single frame forward by a combination of two overlapping dynamics, f_1 and f_5. As the weight on f_1 is reduced and that on f_5 increased, the projection changes from exaggerating the linear feature in the

B

Dyanmics function rank

C

Top two eigenvector correlations

A Spatial dictionary

D Single time-step transition

Previous frame:

More F_8 ← → More F_5

E Dynamics nonlinear effects

More F_1

More F_{12}

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Frame number

F

More F_1

More F_{23}
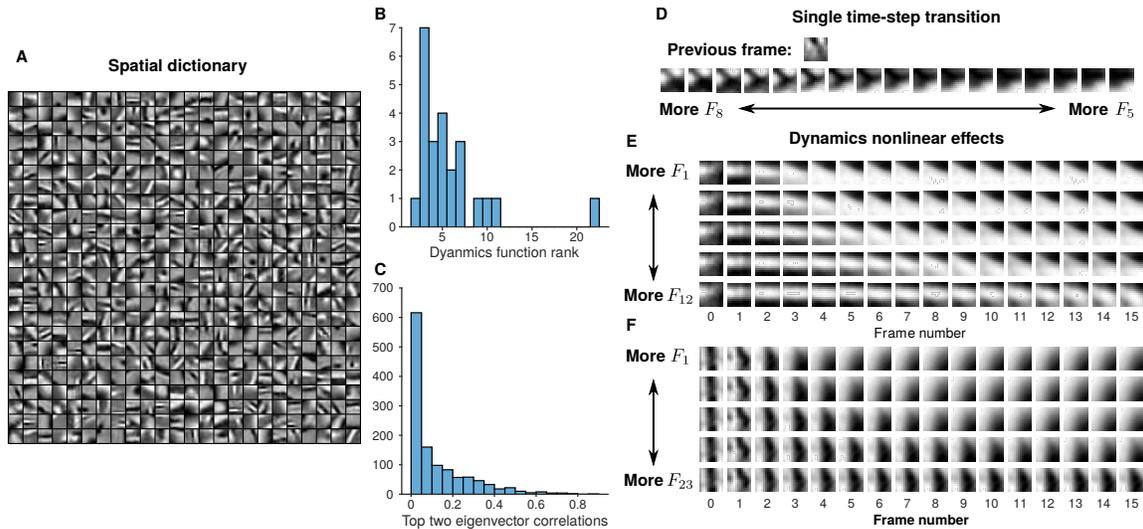
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Frame number

Figure 15: **Results of dynamics learning on natural image patches.** A: The spatial dictionary interestingly retains the Gabor-like structure seen in previous static dictionary learning algorithms. B: Learned dynamics are low rank. The typical rank of each $\boldsymbol{f}_m$ (aside from one almost full-rank function) ranges from 2-10. C: the correlations between the top two eigenvectors show that the dynamics are mostly non-aligned, yet overlap, allowing for second-order effects when combining dynamics. D: Linear combinations of dictionary elements can achieve nonlinear effects. Starting from the same frame, the next frame changes continuously between two possible next frames as the fraction of each dynamic function used is swept from completely using $\boldsymbol{f}_8$ to $\boldsymbol{f}_5$. E,F: Examples of dynamics combinations that achieve nonlinear effects. For each of changing from using more of $\boldsymbol{f}_1$ and $\boldsymbol{f}_{12}$ and $\boldsymbol{f}_1$ and $\boldsymbol{F}_{23}$, the overall effect (rotation/expansion and outward expansion respectively) happens with faster or slower speeds.

top-left to inverting the image with an emphasis on the bottom right. Similar effects appear in iterated dynamics projections, for example changing the weights on $\boldsymbol{f}_1$ and $\boldsymbol{f}_{12}$. This combination effectively rotates a bar over time, and the speed of rotation depends on the amount of $\boldsymbol{f}_1$ vs. $\boldsymbol{f}_{12}$ in the linear combination. This type of speed modulation is impossible in a switched model unless one mode for each speed is included, which is untenable for a continuum of speeds. Similarly when $\boldsymbol{f}_1$ and $\boldsymbol{f}_{23}$ are traded off, a vertical bar slowly has the bottom edge expand to encompass the bottom-right corner, again with different speeds depending on the ratio chosen.

Appendix K. Invariance of the model to transformations in the latent state

Consider the base model complete with the observation equation and decomposed dynamics,

$$\boldsymbol{y}_t = \boldsymbol{D}\boldsymbol{x}_t, \qquad \boldsymbol{x}_t 11 = \left[\sum_{m=1}^{M} \boldsymbol{f}_m c_{mt} \right] \boldsymbol{x}_{t-1}. \qquad (23)$$

For any learned model, we can always define a transformation of the latent space via an invertible matrix $\boldsymbol{U}$ such that

$$\boldsymbol{z}_t = \boldsymbol{U}^{-1}\boldsymbol{x}_t \qquad \boldsymbol{x}_t = \boldsymbol{U}\boldsymbol{z}_t. \qquad (24)$$

This transformation results in an equivalent solution

$$\boldsymbol{y}_t = \boldsymbol{D}\boldsymbol{U}\boldsymbol{z}_t, \qquad \boldsymbol{z}_t = \left[\boldsymbol{U}^{-1} \sum_{m=1}^{M} \boldsymbol{f}_m \boldsymbol{U} c_{mt} \right] \boldsymbol{z}_{t-1}, \qquad (25)$$

i.e., an equivalent set of parameters $\widetilde{\boldsymbol{D}} = \boldsymbol{D}\boldsymbol{U}$ and $\widetilde{f}_m = \boldsymbol{U}^{-1} \sum_{m=1}^{M} \boldsymbol{f}_m \boldsymbol{U}$ result in the same sequence of dynamics but in a transformed latent space. One way to prevent the rotational ambiguity is to assume structure over the latent space, such as we implement via sparsity over $\boldsymbol{x}_t$, which enables us to learn the correct representation (up to a permutation and sign-flip) of observation model (Fig. 14).