ZENO: NONLINEAR FILTERING WITH ZAKAI EQUATION NEURAL OPERATORS

Aaditya Prasad
Picower Institute for Learning & Memory
Department of Brain & Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
aaprasad@mit.edu

Steven W. Flavell
Howard Hughes Medical Institute
Picower Institute for Learning & Memory
Department of Brain & Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
flavell@mit.edu

Brian DePasquale
Department of Biomedical Engineering
Boston University
Boston, MA 02215, USA
bddepasq@bu.edu

ABSTRACT

State inference and parameter learning for nonlinear filters is a core computation for scientific machine learning with varied applications in engineering and neuroscience. The unnormalized filtering distribution is governed by the Zakai equation, a stochastic partial differential equation (SPDE) that combines the forward Kolmogorov operator (known as the Fokker-Planck equation) with observed likelihood updates. Although solving these high-dimensional SPDEs is often impossible analytically and intractable numerically, advances in operator learning and physics-informed machine learning have made it feasible. Harnessing these advances, we introduce the **Z**akai **E**quation **N**eural **O**perator (ZENO), a model that optimizes a neural operator against a Zakai Equation based physics-informed loss to approximate the exact unnormalized filtering distribution for latent stochastic dynamical systems. We use this ZENO-inferred posterior distribution in an expectation-maximization (EM) framework to perform parameter learning and state inference. We demonstrate ZENO's ability to accurately infer the posterior distribution and learn its corresponding SDE parameters in three simulated stochastic dynamical systems (Double-Well bistable system, Van Der Pol (VDP) oscillator, and Lorenz system) and compare its performance against established methods for latent SDE inference and learning. We show that ZENO recovers previously observed limit cycles representing forward and backward locomotion from a whole-brain imaging dataset of immobilized *C. elegans* nematodes. ZENO is a novel approach to the problem of nonlinear filtering and dynamical inference through the lens of partial differential equations and operator theory.

1 INTRODUCTION

Estimating a latent dynamical process from a set of noisy observations is a fundamental problem in machine learning for science (Durstewitz et al., 2023). For example, in neuroscience, it is thought that neural computations are implemented by a latent dynamical system that is expressed by a neural population (Saxena & Cunningham, 2019). This problem has been well studied in signal-processing as filtering (in the online setting) or smoothing (in the offline setting) a noisy non-linear dynamical system (Yau et al., 2024). The essence of filtering as Bayesian inference is to estimate the filtering distribution $p(\mathbf{z}_t|X_{[0:t]})$ where $\mathbf{z}_t \in \mathbb{R}^n$ is an unobserved latent variable and $\mathbf{x}_t \in \mathbb{R}^m$ $(m \geq n)$ are observations at each point $t \in [0, T]$, while for smoothing the goal is to estimate the posterior

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

distribution $p(\mathbf{z}_t|X_{[0:T]})$ (Sarkka, 2013). These distributions are referred to as the filtering and smoothing distributions respectively.

For continuous time systems, the dynamics of the filtering distribution has been well studied as the Kushner-Stratonovich (KS) equation (Stratonovich, 1960; Kushner, 1964), a stochastic partial differential equation (SPDE). A more tractable analog of the KS equation is the Zakai equation (Zakai, 1969), a linear SPDE governing the unnormalized posterior density. While solving the KS or Zakai equation provides the exact posterior distribution (post-normalization for the Zakai equation), solving these equations is often impossible analytically and intractable numerically for high-dimensional systems (Sabbar & Nisar, 2025). Alternative methods such as state-space models (Bishop, 2006), (switching) linear dynamical systems (Linderman et al., 2017), and deep learning-based latent stochastic differential equations (SDE) (Li et al., 2020) offer a more tractable approach to inference and parameter learning by making assumptions or approximations about the posterior distribution. For example, the Kalman-Bucy filter (Kalman & Bucy, 1961) provides the exact solution to the KS equation under the assumption that dynamics are linear and observation noise is Gaussian. Thus, the need remains for powerful methods that can approximate the filtering posterior without restrictive assumptions while remaining computationally tractable.

Two recent approaches for solving PDEs with neural networks offer a route around these barriers. **Physics-informed neural networks (PINNs)** supervise the fit with the governing equation rather than labeled solutions, constraining the network's autodifferentiated derivatives to satisfy the equation at a set of collocation points (Raissi et al., 2019; Karniadakis et al., 2021). PINNS are applicable when the solution is never observed, as in filtering, and because no parametric form is imposed, they can represent non-Gaussian posteriors governing nonlinear dynamics.

However, PINNs are fit to a single instance of a PDE and must be retrained for each new one, whereas in filtering every observation path $X_{[0,t]}$ drives a distinct instance of the Zakai equation. **Neural operators** learn the solution operator itself, mapping a PDE's defining data to its solution and generalizing across instances without retraining (Lu et al., 2021; Azizzadenesheli et al., 2024), but are trained on labeled solutions which filtering never provides. **Physics-informed neural operators (PINOs)** resolve both obstacles, using the equation's residual when observations are unavailable to supervise operator learning across a distribution of instances (Wang et al., 2021b; Li et al., 2024).

This is the paradigm we build on to solve the Zakai equation for nonlinear filtering. We present the **Z**akai **E**quation **N**eural **O**perator (ZENO): A physics-informed neural operator for nonlinear filtering that learns the solution to the Zakai equation. ZENO represents the unnormalized filtering distribution by optimizing a physics informed loss which minimizes the residual between the PINO's posterior derivatives and the Fokker Planck operator while incorporating likelihood updates at observation times. Unlike current approaches, which use numerical solvers (Leng et al., 2026) for forecasting or a deep-splitting approach (Beck et al., 2021; Bågmark et al., 2023b;a; Beck et al., 2026), we leverage neural operators to learn a global solution over the entire observation history. Furthermore, we make no assumptions about the form of the dynamics model and instead estimate the dynamical parameters and observation model in an expectation-maximization (Dempster et al., 1977) like setup.

We show that ZENO is able to achieve similar latent inference accuracy to optimal nonlinear filters in simulated dynamical systems. We then show that ZENO performs dynamical parameter estimation at the same level as other state-of-the-art (SOTA) baselines. Finally, we evaluate our model on a real world neuroscience dataset and find that ZENO is able to recapitulate previously observed latent dynamical structure.

2 METHODS

2.1 BACKGROUND

2.1.1 PROBLEM FORMULATION

The problem of nonlinear filtering involves estimating the posterior distribution $p(\mathbf{z}_t \mid X_{[0,t]})$, where $\mathbf{z}_t \in \mathbb{R}^n$ is an unobserved latent state and $X_{[0,t]} = \{\mathbf{x}_{t_k} : t_k \leq t\}$ are noisy m-dimensional observations of $\mathbf{z}$ taken at discrete times $t_k = k\Delta t$. The latent state evolves as a stochastic differential

equation

$$d\mathbf{z}_t = f(\mathbf{z}_t, t)dt + g(\mathbf{z}_t, t)d\mathbf{W}_t \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is the deterministic drift function, $g(\mathbf{z}, t)$ is an $n \times k$ diffusion coefficient, and $\mathbf{W}_t$ is a k dimensional standard Wiener process. The latent state is observed through a noisy readouts:

$$d\mathbf{x}_t = h(\mathbf{z}_t, t)dt + d\mathbf{V}_t \tag{2}$$

in the continuous case or

$$\mathbf{x}_{t_k} = h(\mathbf{z}_{t_k}) + \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, \mathbf{R}) \tag{3}$$

in the discrete case where $h : \mathbb{R}^n \to \mathbb{R}^m$ is the observation readout function, $\mathbf{V}_t$ is an m-dimensional independent standard Wiener processes, and $\mathbf{R}$ is the observation noise respectively Yau et al. (2024).

2.1.2 PDEs AND SPDEs: FOKKER-PLANCK, KUSHNER-STRATONOVICH, AND ZAKAI EQUATIONS

The **Fokker-Planck** equation (FPE) describes the time evolution of the state's unconditional density, $p(\mathbf{z}, t)$, in the absence of any observations, under the dynamics of Equation 1:

$$\frac{\partial p}{\partial t} = \mathcal{L}^* p, \qquad \mathcal{L}^* p = -\nabla \cdot (fp) + \nabla\nabla : (Dp), \tag{4}$$

where $D = \frac{1}{2} g(\mathbf{z}, t) g^\top(\mathbf{z}, t)$ is the diffusion tensor and $\mathcal{L}^*$ is the forward Kolmogorov operator. The term $\nabla\nabla : (Dp) = \sum_{i,j} \partial_i \partial_j (D_{ij} p)$ is a generalized (anisotropic) Laplacian, reducing to the familiar $D\nabla^2 p$ when g (and hence D) is scalar.

When one only has access to corrupted observations of the state, the conditional PDF of the state $p(\mathbf{z}_t | X_{[0,t]})$ is governed by the **Kushner Stratonovich** stochastic PDE (SPDE):

$$dp = \mathcal{L}^* p dt + p\big(h(\mathbf{z}, t) - \hat{h}(\mathbf{z}, t)\big)^\top d\mathbf{I}_t \tag{5}$$

where $\hat{h}$ is the conditional expectation $\hat{h}(\mathbf{z}, t) = \mathbb{E}[h(\mathbf{z}, t) | X_{[0,t]}]$ and dI_t is the innovation process, $dI_t = dx_t - \hat{h}(z, t)dt$. For continuous observations (Eq. 2), analyzing instead the *unnormalized* probability density function $\rho(\mathbf{z}_t \mid X_{[0,t]})$ yields a more tractable analog of the Kushner-Stratonovich equation — the linear **Zakai** equation:

$$d\rho = \mathcal{L}^* \rho dt + \rho h(\mathbf{z}, t) d\mathbf{x}_t. \tag{6}$$

With the discrete observations (Eq. 3), the Zakai equation reduces to the Fokker–Planck equation between observation times and to a multiplicative likelihood update at each observation, giving the recursion

$$\rho(\mathbf{z}_t | X_{[0,t-\Delta t]}) = (e^{\Delta t \mathcal{L}^*} \rho(\cdot | X_{[0,t-\Delta t]}))(\mathbf{z}) \tag{7}$$

$$\rho(\mathbf{z}_t | X_{[0,t]}) = p(\mathbf{x}_t | \mathbf{z}_t) \rho(\mathbf{z}_t | X_{[0,t-\Delta t]}) \tag{8}$$

Solving the Zakai equation provides the unnormalized filtering distribution but in practice requires numerical methods such as finite-differences which scale poorly due to their grid dependency.

2.1.3 PHYSICS INFORMED NEURAL OPERATORS (PINOs)

In functional analysis, an operator is a function-to-function mapping. Differential equations are naturally expressed in terms of operators: the equation itself defines one operator acting on the unknown function, and its solution — when it exists — defines another, the operator that maps the equation's data to that solution. For example, for the Fokker-Planck equation, the operator of interest is $\mathcal{G} : p_0 \mapsto p(\cdot, t)$, mapping an initial density to the solution at later times.

Neural operators are the generalization of neural networks from functions between finite dimensional vector spaces to infinite dimensional functional spaces (Appendix A). They are commonly used to learn the solution operators to differential equations, especially PDEs. In our setting, we seek to learn an operator over the Zakai equation, $\mathcal{G}_\theta : X_{[0,t]} \mapsto \rho_\theta(\cdot, t)$, mapping an observation path directly to the corresponding estimated unnormalized filtering density.

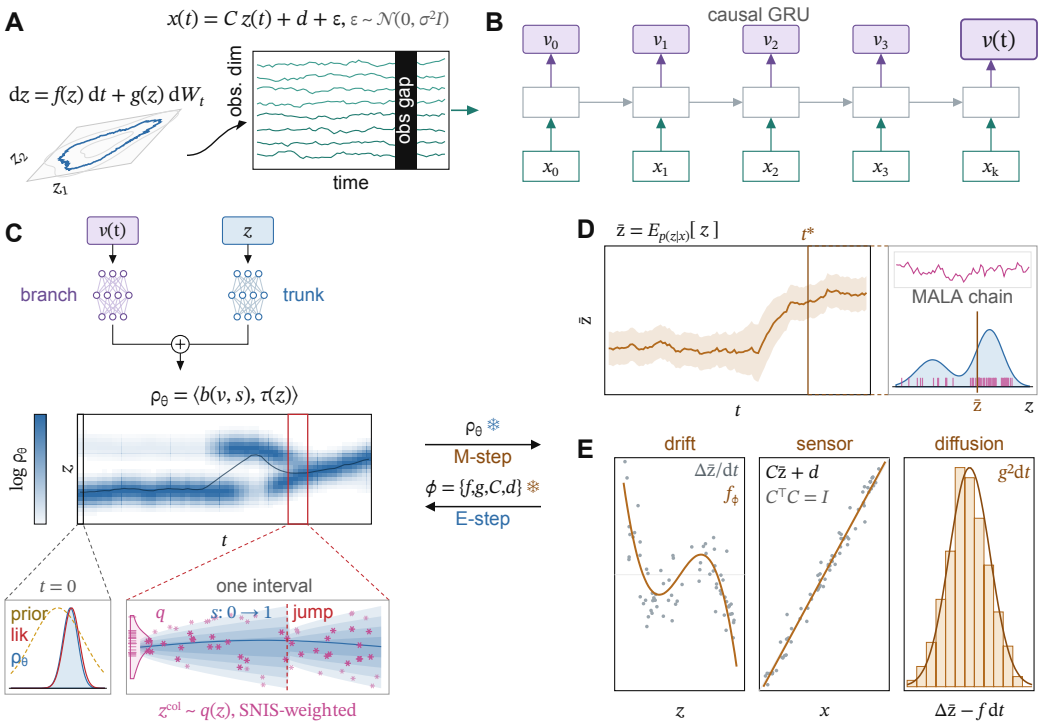

Figure 1: **Architecture Diagram. A)** Generative Model. **B)** Causal GRU encodes observations into context trajectory. **C)** Context trajectory is fed into the branch of a Deep Operator Network along with substeps to produce basis functions. Trunk network can be queried at any $\mathbf{z}$ and outputs $\log \rho_\theta$, the unnormalized log density. IC loss between the learned posterior and an IC prior $\mathcal{N}(0, \mathbf{I})$ combined with Fokker Planck Residual Loss between observations and Jump Loss at observations **D)** Posterior Expectation $\bar{z} = \mathbb{E}_{p(z|x)}[z]$ computed through MALA based MCMC sampling. **E)** Dynamics and Observation Parameters fit by regressing against $\bar{\mathbf{z}}$

While neural operators are data-driven, **Physics Informed Neural Operators** (PINOs) extend them by incorporating the governing differential equation of the system whose solution is being learned (Appendix B). This is enforced by ensuring

$$\frac{\partial \rho_\theta}{\partial t} \approx \mathcal{L}^* \rho_\theta, \tag{9}$$

which is satisfied if ρ_θ solves the Zakai equation. This enables faster and more stable learning: because Eq. 9 enforces the governing equation, the resulting solutions are consistent with the known dynamics and training remains possible even when no labeled solutions exist, since for SPDEs the true filtering posterior is never observed. These models function by sampling collocation points $\{\mathbf{z}^{(k)}\}_{k=1}^K$ on the inferred domain, computing the derivatives of the network at those points through autodifferentiation, and minimizing the discrepancy between the two sides of Eq. 9 at each point.

2.2 Model Architecture

We use a unidirectional GRU (Cho et al., 2014; Krishnan et al., 2015) to encode the observation history into a causal context vector (Fig 1. A, B) $\mathbf{v}_t \in \mathbb{R}^{d_v}$:

$$\mathbf{h}_t = \mathrm{GRU}\left(\mathbf{h}_{t-1}, [\mathbf{x}_t \odot m_t,\ m_t]\right), \qquad \mathbf{v}_t = \mathbf{W}_\mathbf{v}\mathbf{h}_t + \mathbf{b}_\mathbf{v}, \tag{10}$$

where $m_t \in \{0, 1\}$ indicates whether $\mathbf{x}_t$ is observed. By construction, $\mathbf{v}_t$ depends only on $X_{[0,t]}$. For each observation interval $[t, t + \Delta t]$, the causal context $\mathbf{v}_t$ is fed into a Deep Operator Network (DeepONet; App. A.2) (Lu et al., 2021). The DeepONet outputs the unnormalized log-density at

any latent point $\mathbf{z}$ and any time within the interval (Bågmark et al., 2023b; Hu et al., 2024b):

$$\log \rho_\theta(\mathbf{z}, t + s\Delta t \mid X_{[0,t]}) = \beta + b(\mathbf{v}_t, s)^\top \psi(\mathbf{z}), \qquad s \in [0, 1], \tag{11}$$

where s is the normalized time since the observation at t (i.e., $s = 0$ is the posterior after incorporating $\mathbf{x}_t$ and $s = 1$ is the prediction for $t + \Delta t$). The branch $b : \mathbb{R}^{d_v+1} \to \mathbb{R}^p$ is an MLP with one hidden layer (tanh) while the trunk $\psi : \mathbb{R}^n \to \mathbb{R}^p$ is an MLP with three hidden layers (softplus). $\beta \in \mathbb{R}$ is a learned scalar bias. Because the trunk network can be queried at any $\mathbf{z}$ the model is mesh-free (Fig 1. C).

We parameterize the drift $f_\phi(\mathbf{z})$ of equation 1 as an MLP. We define the diffusion as a shared parameter g_ϕ. For the readout, h_ϕ we use a Stiefel observation map (Lee et al., 2026) with learned diagonal observation noise,

$$h_\phi(\mathbf{z}_t) = \mathbf{C}_\phi \mathbf{z}_t + \mathbf{d}_\phi + \epsilon \sim \mathcal{N}(0, \mathbf{R}_\phi), \qquad \mathbf{R}_\phi = \mathrm{diag}(r_1, \ldots, r_m), \tag{12}$$

where $\mathbf{C}_\phi \in \mathbb{R}^{m \times n}$ has orthonormal columns ($\mathbf{C}_\phi^\top \mathbf{C}_\phi = \mathbf{I}_n$), i.e. lies on the Stiefel manifold (Bendokat et al., 2024), $\mathbf{d}_\phi$ is a bias vector, and $\mathbf{R}_\phi$ is the observation noise covariance (Fig 1D).

Initialization. We initialize $\mathbf{C}_\phi$, f_ϕ and $\hat{\mathbf{z}}$ by subspace identification (Van Overschee & De Moor, 1996; Qin, 2006) before training the operator (Appendix C.3) to start in a good initial estimate for learning.

2.3 Fitting Procedure

We fit the model with an approach inspired by expectation-maximization (Ghahramani & Roweis, 1998), alternating between learning $\log \rho_\theta$ with the dynamical and observation parameters $\phi = \{f, g, \mathbf{C}, \mathbf{d}, \mathbf{R}\}$ held fixed (E step) and refitting ϕ with ρ_θ frozen (M step). Full details are given in Appendix C.

E step. We update θ by stochastic gradient descent on

$$l_E = w_{\text{res}}\, l_{\text{res}} + l_{\text{jump}} + l_{\text{ic}}, \tag{13}$$

where each step draws a fresh batch of mesh-free collocation points from a defensive Gaussian mixture (Hesterberg, 1995), half around the projection $\hat{\mathbf{z}}_t = \mathbf{C}_\phi^\top(\mathbf{x}_t - \mathbf{d}_\phi)$ and half from a broad Gaussian at the origin. The residual loss l_{res} is the scale-normalized squared residual of the Fokker–Planck equation (Eq. 9), written for $\log \rho_\theta$, between observation times. The jump loss l_{jump} is a cross entropy that matches the posterior after each observation to the predicted density multiplied by the likelihood $p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1})$, i.e., the Zakai measurement update. The initial condition loss l_{ic} matches $p_\theta(\mathbf{z}_0 \mid \mathbf{x}_0)$ to the posterior under a $\mathcal{N}(0, \mathbf{I})$ prior. Densities are normalized locally over the collocation points by self-normalized importance sampling (Owen, 2013)(Appendix C.1). After N E-step updates, we freeze ρ_θ and compute the posterior means $\bar{\mathbf{z}}_t = \mathbb{E}_{p_\theta(\mathbf{z}_t \mid X_{[0,t]})}[\mathbf{z}_t]$ with the Metropolis-adjusted Langevin algorithm (MALA) (Roberts & Tweedie, 1996)

M step After we have an estimate of $\bar{\mathbf{z}}$, we then fit ϕ in the following order:

1. Fit the Stiefel readout $(\mathbf{C}_\phi, \mathbf{d}_\phi)$ by orthogonal Procrustes analysis (Schönemann, 1966)
2. Regress the drift network f_ϕ on the increments of $\bar{\mathbf{z}}_t$
3. Compute the scalar diffusion g_ϕ from the remaining increment residuals
4. Estimate $\mathbf{R}_\phi$ from the observation directions orthogonal to $\mathrm{span}(\mathbf{C}_\phi)$

3 Results

3.1 Simulated Data

We first benchmark our model on three dynamical systems with known latents by collecting observations $\mathbf{x}_t \sim \mathcal{N}(\mathbf{C}^* \mathbf{z}_t + \mathbf{d}, \sigma_{obs}\mathbf{I})$ where $\mathbf{C}^*$ is a random linear projection with $\mathbf{C}^*{}_{ij} \sim \mathcal{N}(0, 1)$. We evaluate our model on its latent and dynamical RMSE as well as its ability to reconstruct observations

3.1.1 LATENT INFERENCE

We first tested how well ZENO could infer the filtering posterior for a double well potential. This system is used in neuroscience as a simple model of decision making and is of interest because the posterior distribution can be bimodal (Duncker et al., 2019). Because the state is low dimensional and we have access to the exact drift and diffusion, it is tractable to numerically solve the Zakai equation to access the Bayes optimal filter for comparison. In general, this would not be possible as numerical solvers scale exponentially with the dimension of the PDE they are solving. We find that ZENO reaches the ceiling set by the optimal filter for the double well equation (Fig.2 A.) and Van Der Pol oscillator but gaps the optimal filter by about 0.005 on the Lorenz system Lorenz (1963) (Fig. 2D).

Fig 2A shows that we are able to represent the exact filtering distribution and its mean estimate while Fig 2B and C show that our model is able to scale to a higher dimensional system and accurately reconstruct the VDP oscillator and Lorenz system.

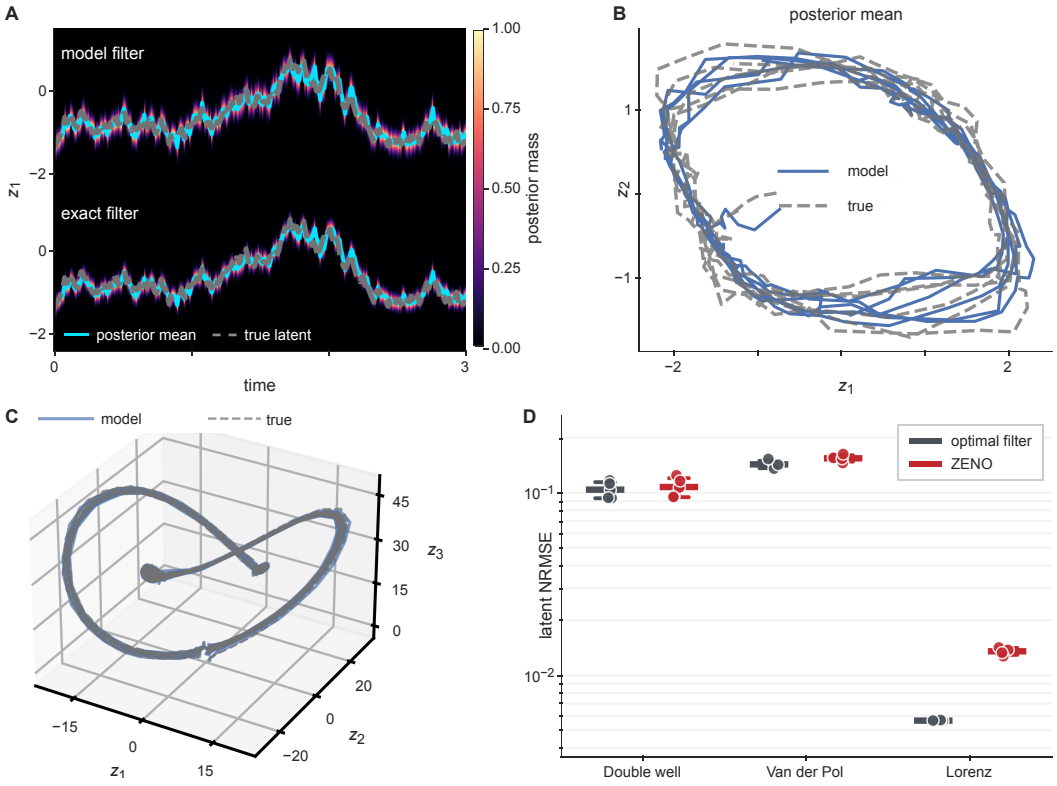

Figure 2: **Latent Inference of Simulated Dynamical Systems. A)** Exact filtering distribution for the double well system vs filtering posterior distribution estimated by ZENO. **B)** Latent Posterior Mean Estimate for the Van Der Pol System. **C)** Latent Trajectory Inference for the Lorenz Attractor. **D)** Comparison of ZENO's normalized root mean squared error with the exact optimal filter for each system.

3.1.2 DYNAMICS PARAMETER LEARNING

After we validated that latent inference by ZENO was accurate, we asked how well it learns system parameters. Fig 3A shows that ZENO learns the double well potential's cubic drift. On Van Der Pol, we show that ZENO reproduces the limit cycle but the velocity field is slower at the edges of the domain (Fig 3B). Finally, on Lorenz we find that ZENO reconstructs the dynamics but misestimates some directions and speeds of the flow field (Fig 3C).

We compare ZENO to two established approaches: 1) the SING algorithm (Hu et al., 2025), which fits a Gaussian process switching linear dynamical system (gpSLDS) (Hu et al., 2024a) to approximate the drift; and 2)SDE-matching, a simulation-free approach to variational inference of latent SDEs (Bartosh et al., 2025). We found ZENO achieves similar levels of accuracy to these baselines: matching gpSLDS on doublewell and SDE matching on VDP (Fig 3D).

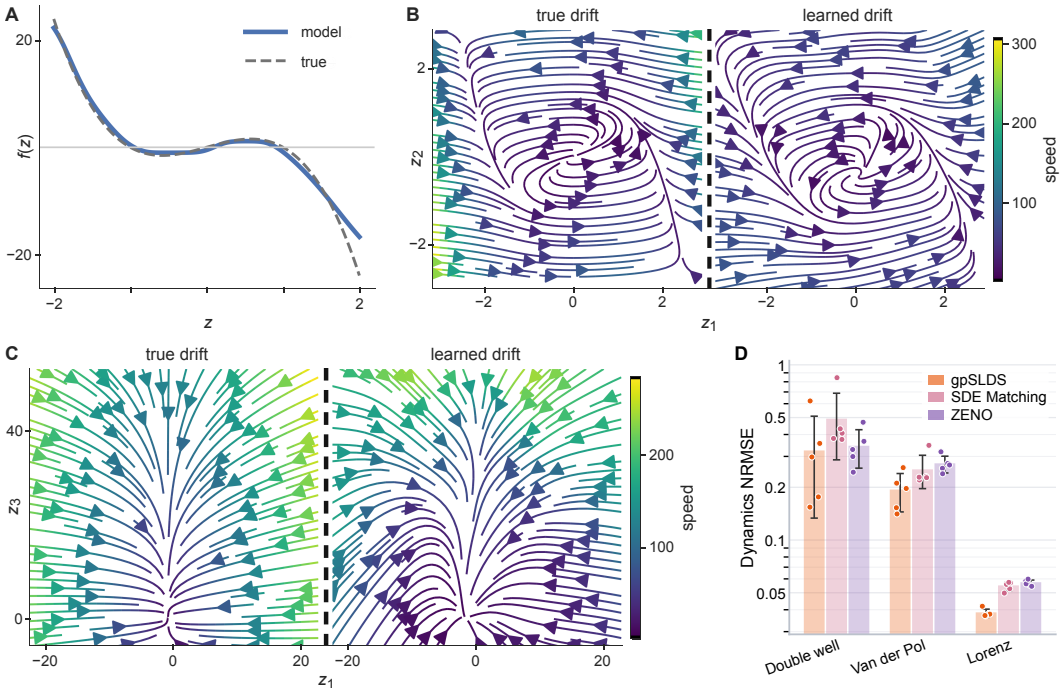

Figure 3: **ZENO Drift Estimation Performance. A)** True cubic drift of double well (dashed line) overlayed with model estimate. **B)** True Van Der Pol Vector field (Left) vs Model's infered Vector Field. **C)** Lorenz Attractor Dynamics plotted in the xz plane. **D)** Model's NRMSE performance compared to baselines. Dots represent seeds and error bars are 1 SD.

3.2 Analysis of whole-brain calcium dynamics in *C. elegans*

We used ZENO to analyze neuroscience data where the true latent state is unknown and where prior work (Linderman et al., 2019; Glaser et al., 2020; Yezerets et al., 2025) provides the opportunity to compare ZENO's latent state estimation with established methods. We tested ZENO on a dataset of whole-brain calcium dynamics in immobilized *C. elegans* (Kato et al., 2015). Fig. 4A-E shows performance on an example worm (StimWorm 1). Specifically, Fig 4A shows that ZENO accurately reconstructs neural observations from the inferred latents. The latent states inferred by ZENO recapitulate reported limit cycles (Fig. 4B,C) and the inferred latents are behaviorally relevant and consistent with human-labels (Fig. 4D,E). We find that ZENO performs comparable to established state-of-the-art methods on reconstruction R^2 and forecasting RMSE[1]. These results establish that ZENO performs comparably to established methods on real-world datasets of scientific value and can be used for scientific discovery of latent variables from neural measurments.

4 Related Work

Switching Linear Dynamical Systems. Switching linear dynamical systems are hierarchical models that approximate a nonlinear system with mixtures of linear dynamical systems. They are of-

[1]Forecasting was attempted with gpSLDS but led to inconsistent results likely due to hyperparameter tuning that we do not think accurate reflect the method.

ten fit with Bayesian methods (e.g., expectation maximization) or with variational approximation. Their mixture of linear systems structure balances interpretability and tractability against expressivity: their discrete states and linear dynamics can be interpreted for scientific discovery (Linderman et al., 2017) but their approximate (variational) posteriors, while tractable fit to data, may diverge from the true posterior. Depending on the choice of variational family these models may struggle to match posteriors that are multimodal or asymmetric. ZENO forgoes a hierarchical framework in favor of directly approximating the true posterior distribution.

Latent SDEs. Latent SDEs address a similar problem using a variational autoencoder framework where the prior and posterior processes are governed by a neural stochastic differential equation and the corresponding drift and diffusion functions are modeled as neural networks (Li et al., 2020). These models offer the flexibility and expressivity of neural networks, optimization with gradient descent, and parallelization offered by GPUs. However, balancing the data reconstruction and Kullback-Leibler (KL) loss that defines their variational objective can often be difficult, (e.g., posterior collapse, where the posterior equals the prior and becomes unusable for anything outside of reconstruction). **Simulation-based Latent SDEs** established this approach first, using the stochas-

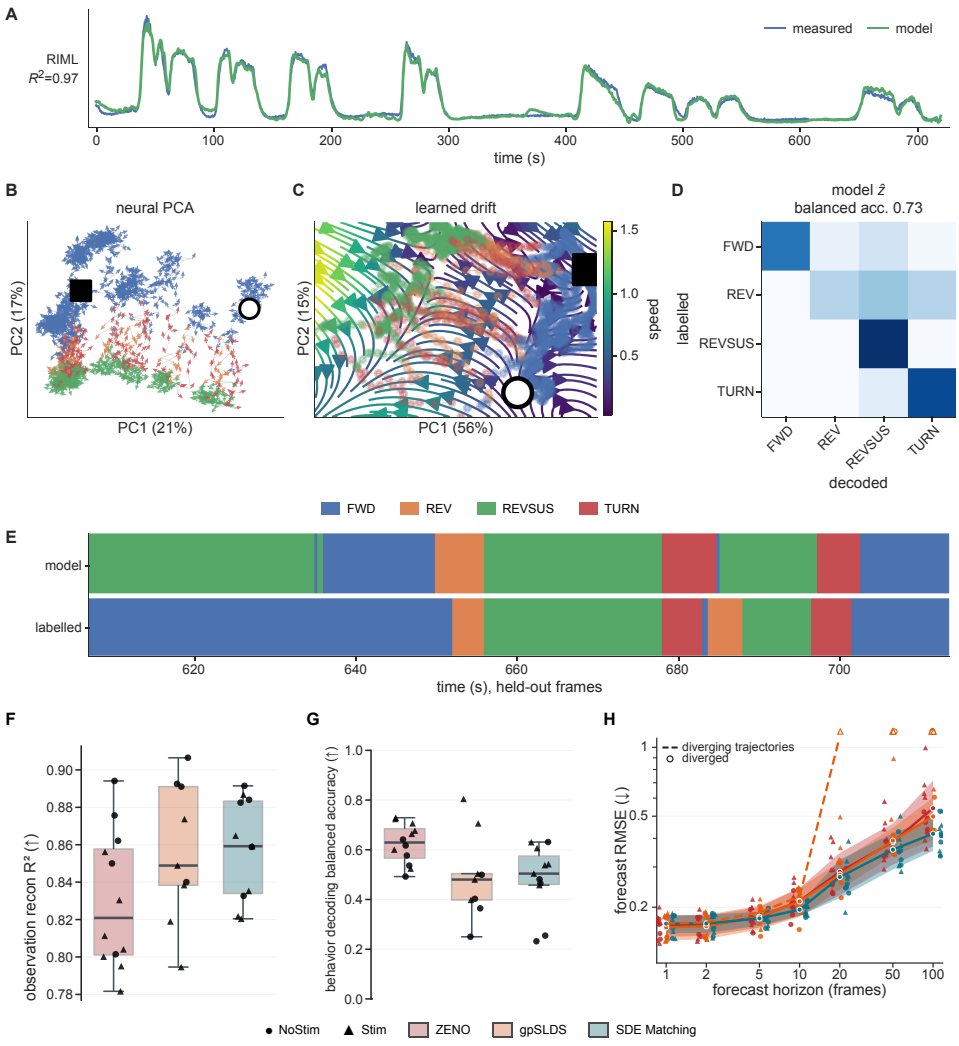

Figure 4: **Whole-brain calcium dynamics in *C. elegans* A)** Example ZENO reconstruction of RIML neuron. **B-C)** ZENO recovers limit cycles. **D-E)** Inferred latents are consistent with known behavioral labels. Model performance compared to baselines on **F)** reconstruction R^2, **G)** behavior decoding accuracy, and **H)** forecasting RMSE. Points represent individual worms, separated into NoStim and Stim conditions. Error bars are 1SD.

tic adjoint sensitivity method. This required rollouts or simulations from the variational posterior at each gradient step to estimate a path-wise KL loss, which can be expensive for long timescales and numerically sensitive. **Simulation-free Latent SDEs** extend these ideas without simulation (Course & Nair, 2023). In SDE-matching, Bartosh et al. (2025) use methods inspired by diffusion models and flow matching to learn latent dynamics through a KL divergence between the posterior and prior drift functions. In Neural Stochastic Flows, Kiyohara et al. (2025) directly model the conditional law governing the solution to the latent SDE using normalizing flows. Finally, in Neural Kolmogorov Equations, Bizzi & Fink (2025), represents the SDE in its infinite dimensional form, but does not learn parameters of a latent SDE. ZENO is a simulation-free latent-SDE model that uses a physics based loss instead of a reconstruction/ELBO, enabling learning the parameters of a latent SDE.

Physics-informed Neural Networks for the Fokker-Planck and Zakai Equations. Other works have investigated using PINNs for solving the Fokker-Planck equation (FPE) (Xu et al., 2020; Zhai et al., 2020; Chen et al., 2020; Hu et al., 2024b) and Zakai equation (Bågmark et al., 2023b; Azimov & Kim, 2026). Liu et al. (2021) solve the FPE as an inverse problem by fitting a network to the FPE solution and fitting the drift and diffusion in alternating steps. However, they use a grid based approach and only demonstrate it on one-dimensional systems. Furthermore, they assume that at least a noisy version of the PDF is accessible for training and do not handle unobserved latents. Azimov & Kim (2026) learn the FPE solution and use it for Bayesian updates to solve for the filtering distribution, but assume known drift and diffusion functions. Both methods propagate the solution step-wise. To our knowledge, our approach is the first to simultaneously perform state inference of the full FPE solution operator while learning unknown dynamics functions using EM.

5 Discussion

Here, we introduced ZENO, a physics-informed neural operator that approximates the unnormalized filtering distribution by learning the solution operator of the Zakai equation. On synthetic systems, ZENO matched or came close to optimal numerical filters for latent inference and performed on par with state-of-the-art methods for parameter learning. On whole-brain *C. elegans* recordings, it was comparable to baselines on forecasting and produced more behaviorally informative latents, at the cost of slightly lower reconstruction R^2, which we expect relaxing the Stiefel constraint on the observation model to address.

ZENO achieves similar performance to state-of-the-art methods but does not outperform them in all cases. All established methods we benchmarked against are estimating the smoothing distribution, which integrates observations both forward and backward in time, thereby providing a superior marginal distribution that can be used as a target. Future directions should investigate extending ZENO to inferring the smoothing distribution by learning the solution to the backwards stochastic PDE (Pardoux, 1982). Additionally, it may be possible to improve the objective for fitting the drift and diffusion by casting it as an inverse problem but that is outside the scope of this work. Finally, our model uses a single set of parameters to represent the entire dynamical system even in cases where there are different dynamical regimes such as in our application to *C. elegans*. Models such as decomposed linear dynamical systems (Yezerets et al., 2025) handle this through mixtures of dictionaries of linear dynamical systems. A similar idea could be combined with our model by using operator splitting methods (Serrano et al., 2026). In fact, one may be able to represent switches in regimes through jumps by generalizing the Forward Kolmogorov operator to Levy processes (Jia & Benson, 2019; Leng et al., 2026). Overall, ZENO represents a novel approach to the problem of nonlinear filtering and dynamical inference through the lens of partial differential equations and operator theory.

AI use statement

In this work, we used generative AI tools for generation of synthetic data sets, to provide feedback on research methodology or experiments, and implement methods. We have not used generative AI tools to propose or refine hypotheses, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results and formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, or assisting with translation are not applicable to this work. Additionally, we used generative AI tools for to modify scientific figures, suggest experimental parameters, create or edit software code, and brainstorming. We have reviewed

all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All code has been provided to the reviewers in an anonymized repository at `https://anonymous.4open.science/r/zeno-iclr2027/` in order to reproduce experiments. Model checkpoints, datasets and results assets can be provided upon request due to size limitations.

REFERENCES

Sherkhon Azimov and Sangil Kim. Integrating physics-informed neural networks with Bayesian inference for nonlinear filtering. *Communications in Nonlinear Science and Numerical Simulation*, 159:109837, August 2026. ISSN 1007-5704. doi: 10.1016/j.cnsns.2026.109837. URL https://www.sciencedirect.com/science/article/pii/S1007570426001978.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, May 2024. ISSN 2522-5820. doi: 10.1038/s42254-024-00712-5. URL https://www.nature.com/articles/s42254-024-00712-5.

Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. SDE Matching: Scalable and Simulation-Free Training of Latent Stochastic Differential Equations. June 2025. URL https://openreview.net/forum?id=0Hd1lh52Fi.

Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic pdes. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154, January 2021. ISSN 1095-7197. doi: 10.1137/19m1297919. URL http://dx.doi.org/10.1137/19M1297919.

Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep learning based numerical approximation algorithms for stochastic partial differential equations, 2026. URL https://arxiv.org/abs/2012.01194.

Thomas Bendokat, Ralf Zimmermann, and P.-A. Absil. A grassmann manifold handbook: basic geometry and computational aspects. *Advances in Computational Mathematics*, 50(1), January 2024. ISSN 1572-9044. doi: 10.1007/s10444-023-10090-8. URL http://dx.doi.org/10.1007/s10444-023-10090-8.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, New York, 2006. ISBN 978-0-387-31073-2.

Arthur Bizzi and Olga Fink. Towards Neural Kolmogorov Equations: Parallelizable SDE Learning with Neural PDEs. November 2025. URL https://openreview.net/forum?id=4PbIFxqosC#discussion.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Kasper Bågmark, Adam Andersson, and Stig Larsson. An energy-based deep splitting method for the nonlinear filtering problem. *Partial Differential Equations and Applications*, 4(2), March 2023a. ISSN 2662-2971. doi: 10.1007/s42985-023-00231-5. URL http://dx.doi.org/10.1007/s42985-023-00231-5.

Kasper Bågmark, Adam Andersson, and Stig Larsson. An energy-based deep splitting method for the nonlinear filtering problem. *Partial Differential Equations and Applications*, 4(2):14, April 2023b. ISSN 2662-2963, 2662-2971. doi: 10.1007/s42985-023-00231-5. URL http://arxiv.org/abs/2203.17153. arXiv:2203.17153 [stat.CO].

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

Xiaoli Chen, Liu Yang, Jinqiao Duan, and George Em Karniadakis. Solving Inverse Stochastic Problems from Discrete Particle Observations Using the Fokker-Planck Equation and Physics-informed Neural Networks, August 2020. URL http://arxiv.org/abs/2008.10653. arXiv:2008.10653 [physics.comp-ph].

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. URL https://arxiv.org/abs/1409.1259.

Kevin Course and Prasanth Nair. Amortized Reparametrization: Efficient and Scalable Variational Inference for Latent SDEs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 78296–78318. Curran Associates, Inc., 2023. doi: 10.52202/075280-3423. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f72d4fdfd5eb425cd81df9fe6272a533-Abstract-Conference.html.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAXEcosystem, 2020. URL http://github.com/google-deepmind.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 0035-9246. URL https://www.jstor.org/stable/2984875.

Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1726–1734. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/duncker19a.html.

Daniel Durstewitz, Georgia Koppe, and Max Ingo Thurm. Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nature Reviews Neuroscience*, 24 (11):693–710, November 2023. ISSN 1471-0048. doi: 10.1038/s41583-023-00740-7. URL https://www.nature.com/articles/s41583-023-00740-7.

Zoubin Ghahramani and Sam Roweis. Learning nonlinear dynamical systems using an em algorithm. In M. Kearns, S. Solla, and D. Cohn (eds.), *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1998. URL https://proceedings.neurips.cc/paper_files/paper/1998/file/0ebcc77dc72360d0eb8e9504c78d38bd-Paper.pdf.

Joshua Glaser, Matthew Whiteway, John Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14867–14878. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/aa1f5f73327ba40d47ebce155e785aaf-Paper.pdf.

Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995. doi: 10.1080/00401706.1995.10484303. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1995.10484303.

Amber Hu, David M. Zoltowski, Aditya Nair, David Anderson, Lea Duncker, and Scott Linderman. Modeling latent neural dynamics with gaussian process switching linear dynamical systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=LX1lwP90kt.

Amber Hu, Henry Smith, and Scott Linderman. SING: SDE Inference via Natural Gradients. October 2025. URL https://openreview.net/forum?id=jmnt0F21K7.

Zheyuan Hu, Zhongqiang Zhang, George Em Karniadakis, and Kenji Kawaguchi. Score-Based Physics-Informed Neural Networks for High-Dimensional Fokker-Planck Equations, February 2024b. URL http://arxiv.org/abs/2402.07465. arXiv:2402.07465 [cs.LG].

Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. In *NeurIPS*, pp. 9843–9854, 2019. URL http://papers.nips.cc/paper/9177-neural-jump-stochastic-differential-equations.

R. E. Kalman and R. S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95–108, March 1961. ISSN 0021-9223. doi: 10.1115/1.3658902. URL https://doi.org/10.1115/1.3658902.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. URL https://www.nature.com/articles/s42254-021-00314-5.

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, October 2015. ISSN 0092-8674. doi: 10.1016/j.cell.2015.09.034. URL https://www.sciencedirect.com/science/article/pii/S0092867415011964.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs.LG].

Naoki Kiyohara, Edward Johns, and Yingzhen Li. Neural stochastic flows: Solver-free modelling and inference for SDE solutions. In *Advances in Neural Information Processing Systems*, 2025.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters, 2015. URL https://arxiv.org/abs/1511.05121.

Harold J. Kushner. On the differential equations satisfied by conditional probablitity densities of markov processes, with applications. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 2(1):106–119, 1964. doi: 10.1137/0302009. URL https://doi.org/10.1137/0302009.

Hyun Dong Lee, Aditi Jha, Stephen E. Clarke, Michael P. Silvernagel, Paul Nuyujukian, and Scott W. Linderman. Stiefel manifold dynamical systems for tracking representational drift. *bioRxiv*, 2026. doi: 10.64898/2026.03.07.710319. URL https://www.biorxiv.org/content/early/2026/03/10/2026.03.07.710319.

Yan Leng, Thibaut Mastrolia, and Hao Wang. Deep zakaij: Structured filtering for jump-diffusion time series forecasting, 2026. URL https://arxiv.org/abs/2605.24548.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3870–3882. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/li20i.html.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM / IMS Journal of Data Science*, February 2024. ISSN 2831-3194. doi: 10.1145/3648506.

Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 914–922. PMLR, April 2017. URL https://proceedings.mlr.press/v54/linderman17a.html.

Scott Linderman, Annika Nichols, David Blei, Manuel Zimmer, and Liam Paninski. Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans. *bioRxiv*, 2019. doi: 10.1101/621540. URL https://www.biorxiv.org/content/early/2019/04/29/621540.

Wei Liu, Connie Khor Li Kou, Kun Hee Park, and Hwee Kuan Lee. Solving the inverse problem of time independent Fokker–Planck equation with a self supervised neural network method. *Scientific Reports*, 11(1):15540, July 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-94712-5. URL https://www.nature.com/articles/s41598-021-94712-5.

Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2): 130–141, March 1963. ISSN 0022-4928, 1520-0469. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2. URL https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL https://www.nature.com/articles/s42256-021-00302-5.

Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

E. Pardoux. Équations du filtrage non linéaire de la prédiction et du lissage. *Stochastics*, 6(3-4): 193–231, 1982. doi: 10.1080/17442508208833204. URL https://doi.org/10.1080/17442508208833204.

S. Joe Qin. An overview of subspace identification. *Computers & Chemical Engineering*, 30(10):1502–1513, September 2006. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2006.05.045. URL https://www.sciencedirect.com/science/article/pii/S009813540600158X.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045.

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998. doi: https://doi.org/10.1111/1467-9868.00123. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00123.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, December 1996. ISSN 1350-7265.

Yassine Sabbar and Kottakkaran Sooppy Nisar. A selective review of modern stochastic modeling: Sde/spde numerics, data-driven identification, and generative methods with applications in biomathematics, 2025. URL https://arxiv.org/abs/2508.11004.

Simo Sarkka. *BAYESIAN FILTERING AND SMOOTHING*. Cambridge University Press, 2013. ISBN 978-1-107-43962-7.

Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55:103–111, April 2019. ISSN 0959-4388. doi: 10.1016/j.conb.2019.02.002. URL https://www.sciencedirect.com/science/article/pii/S0959438818300990.

Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, March 1966. ISSN 1860-0980. doi: 10.1007/BF02289451.

Louis Serrano, Jiequn Han, Edouard Oyallon, Shirley Ho, and Rudy Morel. Test-time generalization for physics through neural operator splitting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gLLcjBJCXZ.

Rouslan L. Stratonovich. Conditional markov processes. *Theory of Probability and Its Applications*, 5:427–453, 1960. URL https://api.semanticscholar.org/CorpusID:121247114.

Peter Van Overschee and Bart De Moor. *Stochastic Identification*, pp. 57–93. Springer US, Boston, MA, 1996. ISBN 978-1-4613-0465-4. doi: 10.1007/978-1-4613-0465-4_3. URL https://doi.org/10.1007/978-1-4613-0465-4_3.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5): A3055–A3081, 2021a. doi: 10.1137/20M1318043. URL https://doi.org/10.1137/20M1318043.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021b. doi: 10.1126/sciadv.abi8605.

Yong Xu, Hao Zhang, Yongge Li, Kuang Zhou, Qi Liu, and Jürgen Kurths. Solving Fokker-Planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1): 013133, January 2020. ISSN 1054-1500, 1089-7682. doi: 10.1063/1.5132840. URL http://arxiv.org/abs/1910.10503. arXiv:1910.10503 [physics.comp-ph].

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/hydra-ecosystem/hydra.

Stephen S. T. Yau, Xiuqiong Chen, Xiaopei Jiao, Jiayi Kang, Zeju Sun, and Yangtianze Tao. Filtering Equations. In *Principles of Nonlinear Filtering Theory*, volume 33, pp. 141–167. Springer Nature Switzerland, Cham, 2024. ISBN 978-3-031-77683-0 978-3-031-77684-7. doi: 10.1007/978-3-031-77684-7_5. URL https://link.springer.com/10.1007/978-3-031-77684-7_5. Series Title: Algorithms and Computation in Mathematics.

Eva Yezerets, Noga Mudrik, and Adam S. Charles. Decomposed Linear Dynamical Systems (dLDS) models reveal instantaneous, context-dependent dynamic connectivity in C. elegans. *Communications Biology*, 8(1):1218, August 2025. ISSN 2399-3642. doi: 10.1038/s42003-025-08599-3. URL https://www.nature.com/articles/s42003-025-08599-3.

Moshe Zakai. On the optimal filtering of diffusion processes. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 11(3):230–243, September 1969. ISSN 1432-2064. doi: 10.1007/BF00536382. URL https://doi.org/10.1007/BF00536382.

Jiayu Zhai, Matthew Dobson, and Yao Li. A deep learning method for solving fokker-planck equations, 2020. URL https://arxiv.org/abs/2012.10696.

A PRIMER ON OPERATOR LEARNING

A.1 FORMULATION

Most deep learning methods map between finite-dimensional vector spaces, $F : \mathbb{R}^n \to \mathbb{R}^m$. Many scientific problems instead involve maps between function spaces. For example, to solve a differential equation we are interested in solving

$$\mathcal{A}u = v$$

where $\mathcal{A}$ is a differential operator: a map that takes in a function and outputs its derivative. Formally, we want to approximate an operator

$$\mathcal{G} = \mathcal{A}^{-1} : \mathcal{V} \to \mathcal{U}, \qquad v \mapsto u = \mathcal{G}(v),$$

where $\mathcal{U}$ and $\mathcal{V}$ are (typically Banach) spaces of functions defined on domains $D \subset \mathbb{R}^d$ and $D' \subset \mathbb{R}^{d'}$. Operator learning fits a parametric approximation $\mathcal{G}_\theta \approx \mathcal{G}$ from input–output pairs $\{(u_i, v_i)\}_{i=1}^N$ by minimizing an error such as $\frac{1}{N}\sum_i \|\mathcal{G}_\theta(v_i) - u_i\|_{\mathcal{U}}^2$. An example of these input-output pairs could be the evaluations of a PDE on some domain paired with the PDE solution.

The difficulty is that functions are infinite-dimensional objects, while any computer sees them only through finite discretizations. While a high resolution discretization will lead to improved performance, generating high resolution data is expensive. A good operator-learning architecture defines a map that is meaningful independently of how inputs and outputs are sampled.

A.2 DEEP OPERATOR NETWORKS (DEEPONET)

DeepONet (Liu et al., 2021) builds on the universal approximation theorem for nonlinear operators of Chen & Chen (1995). This theorem shows that any continuous operator can be approximated arbitrarily well by a sum of products between two networks: one that encodes the input function and one that encodes the output location.

Given an input function v observed at M fixed "sensor" locations $\{\mathbf{z}_m\}_1^M$, a DeepONet evaluates the output function at a query point $\mathbf{z} \in D'$ as

$$\mathcal{G}_\theta(v)(\mathbf{z}) = \sum_{k=1}^{p} b_k\big(v(\mathbf{z}_1), \ldots, v(\mathbf{z}_M)\big)\, \psi_k(\mathbf{z}) + \beta.$$

The branch network $b : \mathbb{R}^M \to \mathbb{R}^p$ encodes the input function into p coefficients while the trunk network $\psi : \mathbb{R}^{d'} \to \mathbb{R}^p$ produces p basis functions evaluated at the query location. The solution operator is thus a linear combination of learned basis functions $\{\psi_k\}$ with input-dependent coefficients $\{b_k(v)\}$. This formulation enables the output to be mesh-free. The trunk can be queried at any $\mathbf{z}$, so outputs can be evaluated anywhere in D'. However, input is tied to a fixed discretization. The branch expects v at the same M sensors for every sample, so the input side is not discretization-invariant without further modification.

B PRIMER ON PHYSICS INFORMED NEURAL NETWORKS (PINNS)

Classical numerical solvers for partial differential equations often fail to scale to higher dimensions due to exponential scaling with the dimensionality of the domain. Instead, modern machine learning based approaches amortize the solution to a PDE by directly fitting neural networks to the derivative-solution pairs. However, these data-driven models ignore known physics and tend to need large datasets which can be expensive to generate. Physics-informed neural networks (PINNs) are a modern approach that represent the solution of a differential equation as a neural network and train it to satisfy both the observed data and the equation itself. This approach has been made practical by automatic differentiation in modern deep learning frameworks.

B.1 FORMULATION

Consider a general differential equation on a domain $\Omega \subset \mathbb{R}^d$, possibly with time $t \in [0, T]$:

$$\begin{aligned} \mathcal{A}_\lambda u &= s(z,t), && (\mathbf{z},t) \in \Omega \times (0,T], \\ \mathcal{B} u &= b(\mathbf{z},t), && (\mathbf{z},t) \in \partial\Omega \times (0,T], \\ u(\mathbf{z},0) &= u_0(\mathbf{z}). \end{aligned}$$

Here, $\mathcal{A}_\lambda$ is a differential operator with parameters λ (for example, $\mathcal{A}_\lambda = \partial_t - \mathcal{L}^*$ for the Fokker-Planck equation), and $\mathcal{B}$ is a boundary operator. A PINN approximates the solution by a network $u_\theta(\mathbf{z}, t)$, typically a multilayer perceptron with smooth activations such as $\tanh$ or sinusoids.

Because $\mathcal{A}_\lambda[u_\theta]$ can be evaluated exactly at any point by automatic differentiation of the network with respect to its inputs, one defines

$$\mathcal{A}_\lambda u_\theta(\mathbf{z}, t) \approx s(\mathbf{z}, t), \tag{14}$$

which is true if u_θ solves the differential equation.

B.2 TRAINING

Training minimizes a composite loss evaluated on sets of points sampled from the domain:

$$\mathcal{L}(\theta) = w_r \mathcal{L}_r + w_b \mathcal{L}_b + w_0 \mathcal{L}_0 + w_d \mathcal{L}_d,$$

with one term per source of information:

$$\mathcal{L}_r = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{A}_\lambda[u_\theta](\mathbf{z}_i^r, t_i^r) - s(\mathbf{z}_i^r, t_i^r) \right|^2, \qquad \mathcal{L}_b = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| \mathcal{B}[u_\theta](\mathbf{z}_i^b, t_i^b) - b(\mathbf{z}_i^b, t_i^b) \right|^2,$$

$$\mathcal{L}_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} \left| u_\theta(\mathbf{z}_i^0, 0) - u_0(\mathbf{z}_i^0) \right|^2, \qquad \mathcal{L}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| u_\theta(\mathbf{z}_i^d, t_i^d) - u_i^{\text{obs}} \right|^2.$$

$\{(\mathbf{z}_i^r, t_i^r)\}$ are called collocation points. The loss can be read as a soft penalty method: the differential equation acts as a regularizer that constrains the network wherever data are absent.

C ZENO FITTING PROCEDURE

This section gives the full E and M steps summarized in Section 2.3. We alternate between learning the unnormalized log-density $\ell_\theta = \log \rho_\theta$ with $\phi = \{f, g, \mathbf{C}, \mathbf{d}, \mathbf{R}\}$ fixed and refitting ϕ with ρ_θ frozen, in the spirit of expectation-maximization (Ghahramani & Roweis, 1998). Throughout, $m_t \in \{0, 1\}$ indicates whether $\mathbf{x}_t$ is observed, and the Gaussian log-likelihood is $\log p(\mathbf{x}_t \mid \mathbf{z}) = -\frac{1}{2}(\mathbf{x}_t - \mathbf{C}\mathbf{z} - \mathbf{d})^\top \mathbf{R}^{-1}(\mathbf{x}_t - \mathbf{C}\mathbf{z} - \mathbf{d})$ up to a constant. Hyperparameters are listed in Appendix D.2.1.

C.1 E STEP

During the E step, we hold ϕ fixed and update the operator's parameters θ by gradient descent on l_E, recomputed at each step from a freshly sampled batch of collocation points.

Collocation points. At every $(t, \text{trajectory})$ site we draw K collocation points, half from a component around the data-driven latent estimate and half from a broad "defensive" component that covers regions the first might miss (Hesterberg, 1995):

$$q_t(\mathbf{z}) = \frac{1}{2}\mathcal{N}\big(\mathbf{z};\, m_t \hat{\mathbf{z}}_t,\, \sigma_t^2 I\big) + \frac{1}{2}\mathcal{N}\big(\mathbf{z};\, 0,\, \sigma_b^2 I\big), \qquad \hat{\mathbf{z}}_t = \mathbf{C}_\phi^\top(\mathbf{z}_t - \mathbf{d}_\phi), \qquad \sigma_t = \begin{cases} \sigma_n & m_t = 1 \\ \sigma_b & m_t = 0. \end{cases} \tag{15}$$

At unobserved frames the first component is therefore also broad and centered at the origin.

Residual loss. For the isotropic diffusion g, dividing the Fokker–Planck equation $\partial_t \rho = \mathcal{L}^* \rho$ by ρ gives its log-space form $\partial_t \ell = \mathcal{A}_\phi[\ell]$ with

$$\mathcal{A}_\phi[\ell](\mathbf{z}) = -\nabla \cdot f_\phi(\mathbf{z}) - f_\phi(\mathbf{z}) \cdot \nabla \ell(\mathbf{z}) + \tfrac{1}{2} g_\phi^2 \big(\|\nabla \ell(\mathbf{z})\|^2 + \Delta \ell(\mathbf{z}) \big). \tag{16}$$

Because s is time normalized by the observation interval, $\partial_t = \Delta t^{-1} \partial_s$, and we minimize the scale-normalized residual

$$l_{\text{res}} = \frac{1}{|\mathcal{C}|} \sum_{(t,s,\mathbf{z}) \in \mathcal{C}} \frac{\big(\Delta t^{-1} \partial_s \ell_\theta - \mathcal{A}_\phi[\ell_\theta] \big)^2}{\operatorname{sg}\big[\mathcal{A}_\phi[\ell_\theta]\big]^2 + \operatorname{sg}\big[\Delta t^{-1} \partial_s \ell_\theta\big]^2 + 1}, \tag{17}$$

where $\operatorname{sg}(\cdot)$ is the stop-gradient operator and $\mathcal{C}$ contains, for each trajectory, a random subset of observation intervals t redrawn at every step, N_s evenly spaced times $s \in [0, 1]$ within each interval, and that interval's collocation points $\mathbf{z}$. The denominator normalizes the residual to prevent blow up at collocation points far from the mode of a sharp density (Wang et al., 2021a), and stop-gradienting the denominator prevents the optimizer from reducing the loss by inflating that magnitude rather than by reducing the residual. The spatial derivatives of the trunk network, the time derivative of the branch network and the divergence of the drift are computed with forward-mode automatic differentiation.

Self-normalized densities. The normalized log-posterior after the observation at t is estimated over the collocation points by self-normalized importance sampling (Owen, 2013):

$$\log p_\theta\big(\mathbf{z}_t^{(k)} \mid X_{[0,t]}\big) = \ell_\theta\big(\mathbf{z}_t^{(k)}, 0 \mid X_{[0,t]}\big) - \operatorname{LSE}_k\big(w_t^{(k)}\big) + \log K, \tag{18}$$

with importance weights $w_t^{(k)} = \ell_\theta\big(\mathbf{z}_t^{(k)}, 0 \mid X_{[0,t]}\big) - \log q_t\big(\mathbf{z}_t^{(k)}\big)$, where LSE_k is the log-sum-exp over index k, so that $\frac{1}{K} \sum_k e^{w_t^{(k)}}$ estimates the normalizer.

Jump loss. The predicted density is represented by self-normalized weights on the next interval's collocation points,

$$\log W_{\text{pred},t}^{(k)} = \operatorname{logsoftmax}_k\Big(\operatorname{sg}\Big[\ell_\theta\big(\mathbf{z}_{t+1}^{(k)}, 1 \mid X_{[0,t]}\big) - \log q_{t+1}\big(\mathbf{z}_{t+1}^{(k)}\big) \Big] \Big), \tag{19}$$

where $\ell_\theta(\mathbf{z}_{t+1}^{(k)}, 1 \mid X_{[0,t]})$ is interval t's density propagated to $s = 1$ using coefficients conditioned only on history through t, but evaluated at interval $t + 1$'s collocation points. Multiplying by the likelihood gives the target of the Zakai measurement update,

$$\log W_{\text{tgt},t}^{(k)} = \operatorname{logsoftmax}_k\Big(\log W_{\text{pred},t}^{(k)} + m_{t+1} \log p\big(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}^{(k)}\big) \Big), \tag{20}$$

so at an unobserved frame the target is the predicted density itself and the density simply propagates. We match the posterior at $t + 1$ to this target with the cross entropy

$$l_{\text{jump}} = -\frac{1}{B(T-1)} \sum_b \sum_{t=0}^{T-2} \sum_k W_{\text{tgt},t}^{(k)} \log p_\theta\big(z_{t+1}^{(k)} \mid X_{[0,t+1]}\big), \tag{21}$$

averaged over the B trajectories.

Initial condition loss. We match the first posterior to the prior $\mathcal{N}(0, I)$ multiplied by the first likelihood:

$$l_{\text{ic}} = -\frac{1}{B} \sum_b \sum_k W_{\text{ic}}^{(k)} \log p_\theta\big(\mathbf{z}_0^{(k)} \mid \mathbf{x}_0\big), \tag{22}$$

where

$$\log W_{\text{ic}}^{(k)} = \operatorname{logsoftmax}_k\Big(\log \mathcal{N}\big(\mathbf{z}_0^{(k)}; 0, I\big) + m_0 \log p\big(\mathbf{x}_0 \mid \mathbf{z}_0^{(k)}\big) - \log q_0\big(\mathbf{z}_0^{(k)}\big) \Big). \tag{23}$$

Algorithm 1 One gradient update to $\log \rho_\theta$

Require: operator parameters θ; fixed $\phi = \{f_\phi, g_\phi, \mathbf{C}_\phi, \mathbf{d}_\phi, \mathbf{R}_\phi\}$; training windows $X_{0:T-1}$ with observation mask $\mathbf{m}_{0:T-1}$ for B trajectories; interval Δt
Require: collocation sizes K (state), N_s (time in interval), N_t (residual intervals); proposal scales σ_n, σ_b; residual weight w_{res}
$\hat{\mathbf{z}}_t \leftarrow \mathbf{C}_\phi^\top(\mathbf{x}_t - \mathbf{d}_\phi)$ for all t, b
for each site (t, b) **do**
    draw $\mathbf{z}_t^{(k)}$: half from $\mathcal{N}(m_t \hat{\mathbf{z}}_t, \sigma_t^2 \mathbf{I})$, half from $\mathcal{N}(0, \sigma_b^2 \mathbf{I})$ ▷ Eq. 15
end for
draw a random subset $\mathcal{T}$ of N_t intervals; set $\mathcal{S} = \{0, \frac{1}{N_s - 1}, \ldots, 1\}$
compute contexts v_t with the causal GRU and $\ell_\theta(\mathbf{z}_t^{(k)}, s \mid X_{[0,t]})$ at $s \in \{0, 1\}$
$\log p_\theta(\mathbf{z}_t^{(k)} \mid X_{[0,t]}) \leftarrow$ self-normalized estimate ▷ Eq. 18
for $t = 0, \ldots, T-2$ **do**
    $\log W_{\text{pred},t}^{(k)} \leftarrow \text{logsoftmax}_k\big(\text{sg}[\ell_\theta(\mathbf{z}_{t+1}^{(k)}, 1 \mid X_{[0,t]}) - \log q_{t+1}(\mathbf{z}_{t+1}^{(k)})]\big)$ ▷ Eq. 19
    $\log W_{\text{tgt},t}^{(k)} \leftarrow \text{logsoftmax}_k\big(\log W_{\text{pred},t}^{(k)} + m_{t+1} \log p(\mathbf{z}_{t+1} \mid \mathbf{z}_{t+1}^{(k)})\big)$ ▷ Eq. 20
end for
$l_{\text{jump}} \leftarrow$ cross entropy of $\log p_\theta(\mathbf{z}_{t+1}^{(k)} \mid X_{[0,t+1]})$ under $W_{\text{tgt},t}$ ▷ Eq. 21
$l_{\text{ic}} \leftarrow$ cross entropy of $\log p_\theta(\mathbf{z}_0^{(k)} \mid \mathbf{x}_0)$ under W_{ic} ▷ Eqs. 22–23
for $(t, s, \mathbf{z}) \in \mathcal{T} \times \mathcal{S} \times \{z_t^{(k)}\}$ **do**
    forward-mode AD: $\partial_s \ell_\theta$, $\nabla \ell_\theta$, $\Delta \ell_\theta$, f_ϕ, $\nabla \cdot f_\phi$; form $\mathcal{A}_\phi[\ell_\theta]$
end for
$l_{\text{res}} \leftarrow$ scale-normalized residual ▷ Eq. 17
$l_E \leftarrow w_{\text{res}}\, l_{\text{res}} + l_{\text{jump}} + l_{\text{ic}}$ ▷ Eq. 24
$\theta \leftarrow \text{Adam}(\theta, \nabla_\theta l_E)$, accumulating $\nabla_\theta l_E$ over chunks of trajectories

Total loss. The E-step loss is

$$l_E = w_{\text{res}}\, l_{\text{res}} + l_{\text{jump}} + l_{\text{ic}}, \tag{24}$$

where $w_{\text{res}} \geq 0$ balances the residual, which spreads the posterior, against the measurement updates, which sharpen it. The jump and initial-condition terms use every interval, while the residual uses the random subset of intervals described above. We minimize l_E with Adam (Kingma & Ba, 2017) under an exponentially decaying learning rate; each update uses the gradient over all training trajectories, accumulated in chunks to bound memory, and resamples the collocation points and residual intervals, so the E step consists of many such updates rather than a single fit to convergence.

Algorithm 1 summarizes one E-step update.

Posterior means. After N gradient steps, we freeze θ and estimate $\bar{\mathbf{z}}_t = \mathbb{E}_{p_\theta(\mathbf{z}_t \mid X_{[0,t]})}[\mathbf{z}_t]$ by sampling each filtering marginal $p_\theta(\mathbf{z}_t \mid X_{[0,t]}) \propto \exp \ell_\theta(\mathbf{z}, 0 \mid X_{[0,t]})$ with the Metropolis-adjusted Langevin algorithm (MALA) (Roberts & Tweedie, 1996). For each (time, trajectory) site we run K_c parallel chains, initialized at $\hat{\mathbf{z}}_t = \mathbf{C}_\phi^\top(\mathbf{x}_t - \mathbf{d}_\phi)$ at observed frames and from $\mathcal{N}(0, \sigma_b^2 \mathbf{I})$ at unobserved frames, with the proposal

$$\mathbf{z}' = \mathbf{z} + \epsilon\, \nabla_{\mathbf{z}} \ell_\theta(\mathbf{z}, 0) + \sqrt{2\epsilon}\, \xi, \qquad \xi \sim \mathcal{N}(0, \mathbf{I}), \tag{25}$$

followed by a Metropolis–Hastings correction. During the first third of the sweeps (burn-in) the step size ϵ of each site adapts toward an acceptance rate of 0.574 (Roberts & Rosenthal, 1998), and $\bar{\mathbf{z}}_t$ is the average of the chain states over the remaining sweeps.

C.2 M STEP

Once, we have a posterior mean estimate, we refit ϕ through an M step.

Observation map. We first refit the Stiefel readout by orthogonal Procrustes analysis (Schönemann, 1966):

$$\mathbf{C}_\phi = \mathbf{U}\mathbf{V}^\top, \qquad \mathbf{d}_\phi = \bar{\mathbf{x}} - \mathbf{C}_\phi \bar{\mathbf{z}}, \tag{26}$$

where $\mathbf{U}\mathbf{V}^\top$ is the singular value decomposition of the empirical cross-covariance $M = \sum_t (\mathbf{x}_t - \bar{\mathbf{x}})(\bar{\mathbf{z}}_t - \bar{\mathbf{z}})^\top$ and $\bar{x}$, $\bar{\mathbf{z}}$ are the means of the observations and of the posterior means. We record how far the readout moved, $\delta_{\mathbf{C}_\phi} = 1 - \frac{1}{n}\sum_i \cos\vartheta_i$, where ϑ_i are the principal angles between the new and previous column spaces of $\mathbf{C}_\phi$.

Drift. Only if $\delta_{\mathbf{C}_\phi}$ is below a tolerance, so that the latent coordinates have stabilized, do we refit the drift, by minimizing with Adam, warm-started from the current network,

$$l_{\text{drift}} = \frac{1}{|\mathcal{V}|}\sum_{t\in\mathcal{V}} \left\| f_\phi(\tilde{\mathbf{z}}_t) - \frac{\bar{\mathbf{z}_{t+1}} - \bar{\mathbf{z}_t}}{\Delta t} \right\|^2 + \lambda\|\phi_f\|^2 + \alpha\, l_{\text{curv}}, \qquad \tilde{\mathbf{z}}_t = \bar{\mathbf{z}} + \tfrac{\Delta t}{2} f_\phi^{\text{old}}(\bar{\mathbf{z}}_t), \tag{27}$$

where $\mathcal{V}$ is the set of times at which both $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are observed and f_ϕ^{old} is the drift before the refit. Evaluating the drift at the second-order Runge–Kutta midpoint $\tilde{\mathbf{z}}_t$ removes the $O(\Delta t)$ bias of regressing $f_\phi(\bar{\mathbf{z}}_t)$ on the forward increment. The weight decay $\lambda\|\phi_f\|^2$ acts on all weights and biases, and the curvature penalty is a finite-difference estimate of the squared second directional derivative along a random unit direction $\mathbf{u}_t$, redrawn at each iteration,

$$l_{\text{curv}} = \frac{1}{\delta^4|\mathcal{V}|}\sum_{t\in\mathcal{V}} \left\| f_\phi(\tilde{\mathbf{z}}_t + \delta\mathbf{u}_t) - 2f_\phi(\tilde{\mathbf{z}}_t) + f_\phi(\tilde{\mathbf{z}}_t - \delta\mathbf{u}_t) \right\|^2, \tag{28}$$

with δ equal to 5% of the average standard deviation of the latent coordinates.

Diffusion. We refit the scalar diffusion from the trapezoidal increment residual under the current drift:

$$g_\phi = \max\Big\{ \Big(\frac{\Delta t}{n|\mathcal{V}|} \sum_{t\in\mathcal{V}} \|r_t\|^2 \Big)^{1/2},\, g_{\min} \Big\}, \qquad r_t = \frac{\bar{\mathbf{z}_{t+1}} - \bar{\mathbf{z}_t}}{\Delta t} - \tfrac{1}{2}\big(f_\phi(\bar{\mathbf{z}}) + f_\phi(\bar{\mathbf{z}_{t+1}}) \big). \tag{29}$$

Observation noise. Finally, we refit the diagonal observation noise $\mathbf{R}_\phi = \mathrm{diag}(r_1, \ldots, r_m)$ from the observation directions orthogonal to $\mathrm{span}(\mathbf{C}_\phi)$, which no latent state can explain:

$$r_i = \frac{1}{|\mathcal{O}|}\sum_{t\in\mathcal{O}} \frac{\big[(I - \mathbf{C}_\phi\mathbf{C}_\phi^\top)(\mathbf{x}_t - \mathbf{d}_\phi)\big]_i^2}{1 - \|\mathbf{C}_{\phi_{i,:}}\|^2}, \tag{30}$$

where $\mathcal{O}$ is the set of observed frames and $1 - \|\mathbf{C}_{\phi_{i,:}}\|^2 = (\mathbf{I} - \mathbf{C}_\phi\mathbf{C}_\phi^\top)_{ii}$.

Algorithm 2 summarizes the M step, which is run once every N E-step updates and once, in bootstrap mode, at initialization (Appendix C.3).

C.3 Initialization

To initialize the model, we take inspiration from subspace identification (Van Overschee & De Moor, 1996; Qin, 2006). We form the mean-centered future block Hankel matrix of the observations, take its top-n left singular vectors, and set $\mathbf{C}_\phi$ to the orthonormalized first block row, with $\mathbf{d}_\phi = \bar{x}$. A linear dynamics matrix $\mathbf{A} = (\mathbf{A}_{\text{disc}} - \mathbf{I})/\Delta t$ is then obtained by least squares, $\hat{\mathbf{z}_{t+1}} \approx \mathbf{A}_{\text{disc}}\hat{\mathbf{z}}_t$, on the projection $\hat{\mathbf{z}}_t = \mathbf{C}_\phi^\top(\mathbf{z}_t - \mathbf{d}_\phi)$. We warm start the drift network by regressing $f_\phi(\mathbf{z})$ onto $\mathbf{A}\mathbf{z}$ at points drawn uniformly from a box around the origin, initialize g_ϕ and $\mathbf{R}_\phi$ to fixed values, and initialize the operator randomly. Before any operator training we take one bootstrap M step that uses $\hat{\mathbf{z}}_t$ in place of the MALA posterior means; it refits $\mathbf{C}_\phi$, $\mathbf{d}_\phi$, f_ϕ and g_ϕ as above and leaves $\mathbf{R}_\phi$ at its initial value.

D Experimental Details

All experiments are trained on one GPU; Models are initialized randomly and a new simulated dataset is generated per seed. However methods using the same seed use the same dataset as input. ZENO and all baselines are implemented in Python ($\geq$ 3.10) on JAX ($\geq$ 0.6) (Bradbury et al. (2018)): ZENO uses Equinox ($\geq$ 0.13) for its networks and Optax ($\geq$ 0.2) for optimization (DeepMind et al., 2020), SDE Matching uses the same stack with Diffrax (0.7) (Kidger (2021)) for its forecast rollouts, gpSLDS uses the authors' SING implementation (Hu et al., 2025), and experiments are configured with Hydra (Yadan, 2019).

Algorithm 2 M step

Require: frozen operator θ; current $\phi = \{f_\phi, g_\phi, \mathbf{C}_\phi, \mathbf{d}_\phi, \mathbf{R}_\phi\}$; training windows $X_{0:T-1}$, mask $m_{0:T-1}$; interval Δt
Require: broad scale σ_b; stability tolerance τ_C; drift iterations M, weight decay λ, curvature weight α; floor $g_{\min}$; flag BOOTSTRAP
Require: MALA estimate of $\bar{\mathbf{z}}$
$\hat{\mathbf{z}}_t \leftarrow \mathbf{C}_\phi^\top \mathbf{x}_t - \mathbf{d}_\phi)$ for all t
if BOOTSTRAP **then**
    $\bar{\mathbf{z}_t} \leftarrow \hat{\mathbf{z}}_t$ ▷ no operator yet
end if
$\mathbf{C}_{\text{old}} \leftarrow \mathbf{C}_\phi$; $(\mathbf{C}_\phi, \mathbf{d}_\phi) \leftarrow$ orthogonal Procrustes fit of $\mathbf{x}$ on $\bar{\mathbf{z}}$ ▷ Eq. 26
$\delta_{\mathbf{C}} \leftarrow 1 - \frac{1}{n}\sum_i \cos\vartheta_i(\mathbf{C}_\phi, \mathbf{C}_{\text{old}})$
$\mathcal{V} \leftarrow \{t : m_t = m_{t+1} = 1\}$
if $\delta_{\mathbf{C}} < \tau_{\mathbf{C}}$ **then** ▷ refit dynamics only once the sensor is stable
    $\tilde{\mathbf{z}}_t \leftarrow \bar{\mathbf{z}_t} + \frac{\Delta t}{2} f_\phi(mat\bar{h}bfz_t)$ for $t \in \mathcal{V}$ ▷ RK2 midpoint, current drift
    for $j = 1, \ldots, M$ **do**
        draw unit directions $\mathbf{u}_t$; $f_\phi \leftarrow \text{Adam}(f_\phi, \nabla l_{\text{drift}})$ ▷ Eqs. 27–28
    end for
end if
$g_\phi \leftarrow$ trapezoidal residual estimate, floored at $g_{\min}$ ▷ Eq. 29
if not BOOTSTRAP **then**
    $r_i \leftarrow$ orthogonal-complement estimate for each dimension i ▷ Eq. 30
end if
return $\phi = \{f_\phi, g_\phi, \mathbf{C}_\phi, \mathbf{d}_\phi, \mathbf{R}_\phi\}$

D.1 DATASETS

Simulated systems. We benchmark on latent SDEs $dz = f(z)\,dt + G\,dW$ with three drifts. For the double well and the Van der Pol oscillator we follow the systems of Duncker et al. (2019); for the Lorenz system we follow the setup of Li et al. (2020); Bartosh et al. (2025); Hu et al. (2025), replacing their multiplicative noise with an additive diffusion.

$$\begin{aligned}
\text{Double well:}\quad & f(z) = a\,z\,(1 - z^2), \quad a = 4,\\
\text{Van der Pol:}\quad & f_1(\mathbf{z}) = \tau\mu\big(z_1 - \tfrac{1}{3}z_1^3 - z_2\big), \quad f_2(\mathbf{z}) = \tau z_1/\mu,\\
& \tau = 15,\ \mu = 2,\\
\text{Lorenz:}\quad & f_1(\mathbf{z}) = \sigma(z_2 - z_1), \quad f_2(\mathbf{z}) = z_1(\rho - z_3) - z_2, \quad f_3(\mathbf{z}) = z_1 z_2 - \beta z_3,\\
& (\sigma, \rho, \beta) = (10, 28, 8/3).
\end{aligned}$$

Each trajectory is simulated with an Euler–Maruyama scheme whose step is at most 1 ms and is refined so that every observation time is hit exactly. Observations are $\mathbf{x}_t \sim \mathcal{N}(\mathbf{C}\mathbf{z}_t + d, \epsilon^2\mathbf{I})$ on a regular grid. For the double well and Van der Pol, C and d are a random Gaussian sensor with elements drawn IID from the standard normal distribution and shared by all splits. the noise is added in physical units. For Lorenz, observations are $x_t \sim \mathcal{N}(\mathbf{z}_t, \epsilon^2 I)$, the clean signal is standardized with train-only constants, and the noise is added after this normalization, following the `diffrax` Kidger (2021) reference. Training, validation and test trajectories are independent draws that share the sensor. Before training, observations are standardized using training data only through a per-channel centering and one global scale set by the leading singular values of the training observations. Table 1 lists the parameters of each dataset.

Whole-brain calcium imaging in *C. elegans*. The dataset of Kato et al. (2015) contains 12 immobilized worms. Five worms (WT NoStim) were recorded spontaneously for about 18 minutes (3021–3311 frames at 2.8–3.1 Hz, 109–135 identified neurons), and seven worms (WT Stim) were recorded for about 12 minutes (2017–2201 frames, 107–151 neurons) while oxygen stimuli were delivered; we do not model the stimulus, so the Stim recordings are treated as an unconditional benchmark. We use the corrected fluorescence traces at the native frame rate with no smoothing, clipping or derivative preprocessing. Following the whole-trace protocol of Yezerets et al. (2025), we fit a separate model to each worm and fit and score every frame of the recording; reconstruction

Table 1: Simulated datasets. d is the latent dimension and D the number of observed channels. G multiplies dW. The observation noise SD ϵ is in physical units for the double well and Van der Pol and in units of the standardized clean signal for Lorenz. Trajectory counts are train / validation / test.

	Double well	Van der Pol	Lorenz
Latent dimension d	1	2	3
Drift parameters	$a = 4$	$\tau = 15,\ \mu = 2$	$(10, 28, 8/3)$
Diffusion G	1	I_2	$0.3\, I_3$
Initial state z_0	$\mathcal{N}(0, 1)$	$\mathrm{Unif}([-3, 3]^2)$	$\mathcal{N}(0, I_3)$
Duration T (s)	3	3	2
Observations per trajectory	301	301	100
Observation interval Δt (s)	0.01	0.01	$2/99 \approx 0.0202$
Simulation step (s)	10^{-3}	10^{-3}	$\leq 10^{-3}$
Sensor	Gaussian C, d	Gaussian C, d	identity
Observed channels D	15	20	3
Observation noise SD ϵ	0.5	1.5	0.01
Trajectories (train / val / test)	20 / 5 / 10	20 / 5 / 10	1024 / 128 / 128

and forecasting scores on this dataset are therefore in-sample. Recordings are cut into windows of 200 frames (training stride 100, non-overlapping scoring windows); each neuron is centered by its training mean and all neurons are divided by one global signal scale, the geometric mean of the top-10 singular values of the training frames. Every method uses a 10-dimensional latent state and an observation-noise initialization of 0.1 in standardized units, which is then learned. Human-annotated behavioral states accompany each recording (7 labeled states for NoStim, 4 for Stim; frames labeled `NOSTATE` are excluded); they are never used to fit or select a dynamical model.

D.2 Training

Every method receives the same standardized data, observation grid, dimensions and validation cadence. Gradient-based methods (ZENO, SDE Matching) train for 10,000 steps on the double well, Van der Pol and Kato datasets and 3,000 steps on Lorenz, with validation every 100 steps; gpSLDS is fit by variational EM for 50 iterations. Each method selects its checkpoint on the validation split with a criterion that does not use ground truth (Section D.2.1–D.2.3), and the test split is scored once, after selection, with 128 posterior and forecast samples. Baselines run in float64; ZENO in float32. For the simulated systems we ran six seeds (0–5) per dataset and method; the seed changes both the data draw and the initialization. For Kato we fit one model per worm (seed 0) and treat animals as the replicates.

D.2.1 ZENO

The E-step minimizes l_E with Adam (Kingma & Ba, 2017) with $\lambda = 2 \times 10^{-3}$ with an exponential decay of $\beta = 0.9998$ per step, using full-batch gradients accumulated over chunks of 16 trajectories. The Fokker–Planck residual is evaluated at 4 time-in-interval collocation points $s \in [0, 1]$ on 24 randomly chosen observation intervals per step, and at N_z mesh-free state collocation points per (interval, trajectory) drawn from the defensive mixture with $\sigma_n = 0.3$ and $\sigma_b = 1.6$. Every 2,000 E-steps we run one M-step: the filter mean is read out with 64 MALA chains of 30 sweeps per site (the first third is burn-in; the step size self-tunes to an acceptance rate of 0.574 (Roberts & Rosenthal, 1998)), the Stiefel sensor is refit by orthogonal Procrustes, the drift network is refit for 400 Adam iterations at learning rate 2×10^{-3} against the increment target evaluated at the RK2 predicted midpoint (`det_mid`), the scalar diffusion is refit from the trapezoidal increment residual, and the diagonal observation noise R is refit from the residual directions orthogonal to $\mathrm{span}(C)$. The drift is refit only when the sensor has moved little since the previous M-step. The sensor and a linear dynamics matrix are initialized by temporal subspace identification; the drift network is warm-started to those linear dynamics and one bootstrap M-step from the projection $\hat{z} = C^\top (x - d)$ precedes any operator training. The operator itself is randomly initialized with the default uniform fan-in scheme. Checkpoints are selected by the forecast RMSE on the validation windows (16

forecast samples), the same quantity that is reported on the test split. Table 2 lists the different ZENO hyperparameters used for each dataset.

Table 2: ZENO hyperparameters. Shared values apply to every dataset; the last four rows are the only per-dataset settings.

Component	Hyperparameter	Value
Context encoder	causal GRU hidden / context dimension	64 / 64
DeepONet	basis size p	64
	branch MLP hidden width	128
	trunk MLP width $\times$ depth, activation	64×3, softplus
Drift f_ϕ	MLP width $\times$ depth, activation	64×3, tanh
Diffusion	scalar g: initial value / lower bound	1.0 / 0.05
Observation model	Stiefel C, bias d, diagonal R (learned)	yes
E-step	optimizer, learning rate, decay per step	Adam, 2×10^{-3}, 0.9998
	s-collocation points, intervals per step	4, 24
	collocation proposal (σ_n, σ_b)	(0.3, 1.6)
	residual weight w_{res}	0.4
	trajectories per gradient chunk	16
M-step	E-steps between M-steps	2,000
	drift iterations, learning rate	400, 2×10^{-3}
	drift weight decay λ	3×10^{-4}
	drift target	RK2 midpoint (`det_mid`)
	MALA chains $\times$ sweeps	64×30
	sensor-stability tolerance for refitting f	0.05
Initialization	sensor / dynamics	subspace ID + bootstrap M-step
Selection	criterion (validation), forecast samples	prefix forecast RMSE, 16
		DW / VDP / Lorenz / Kato
Per dataset	latent d, observed D	1, 15 / 2, 20 / 3, 3 / 10, N
	state collocation points N_z	128 / 256 / 384 / 128
	drift curvature penalty α	10^{-3} / 10^{-3} / 10^{-3} / 0
	training steps	10k / 10k / 3k / 10k

D.2.2 GPSLDS

We use the authors' SING implementation (Hu et al., 2025) of the gpSLDS (Hu et al., 2024a) with the smoothly switching linear (SSL) kernel. Table 3 lists the settings we used; the per-dataset entries were chosen on validation data from single-seed tuning runs. Held-out inference runs SING's E-step with the learned drift frozen, starting each trial from the closed-form posterior under the Brownian prior followed by drift-free E-steps (`warm_iters`), which is what keeps inference stable at small σ; the variational inducing distribution is initialized at the prior $\mathcal{N}(0, K_{zz})$. On Kato, 3 of the 12 fits used a second configuration (a fixed prior noise $\sigma = 1$, 3 regimes, learned emission and SING's `neural` natural-gradient schedule) and the configuration reported for each animal is the one with the lower in-sample forecast RMSE. gpSLDS is therefore a per-animal best-of over hyperparameters on Kato, while ZENO and SDE Matching use one fixed configuration for all worms.

D.2.3 SDE MATCHING

We implement the diagonal-Gaussian marginal specialization of SDE Matching (Bartosh et al., 2025) in JAX. The prior is a latent SDE with a neural drift and a constant diagonal diffusion; the posterior marginals $q(z_t \mid x) = \mathcal{N}(m(t), \operatorname{diag} s(t)^2)$ are produced by a network that reads the time and the state of a backward GRU run over the observations (so the encoder is a smoother), and the posterior drift follows in closed form from dm/dt, ds/dt and the diffusion (time derivatives by forward-mode differentiation). No SDE is solved during training. The objective is the initial KL plus the path energy, estimated at 4 uniformly sampled times per trajectory and multiplied by the physical duration, plus the observation negative log-likelihood at sampled times multiplied by the number of observations; there is no KL annealing or time normalization. Forecasts integrate the learned prior from the posterior at the end of the prefix with an Euler–Maruyama step of 0.01. Table 4 lists the hyperparameters, which are identical on every dataset except for the step budget.

Table 3: gpSLDS hyperparameters. The upper block is shared; the lower block is per dataset. Kato lists the primary configuration (9 of 12 animals; the alternative used for the other three is given in the text).

Shared hyperparameter	Value
Kernel	smoothly switching linear (SSL)
Variational EM iterations	50
Natural-gradient E-steps per iteration	15
Adam M-steps per iteration (kernel hyperparameters)	50 (Kato: 10)
Drift-free warm E-steps per trial	15
Held-out inference iterations (parameters frozen)	15
Initial slope-variance prior	1.0
Regime fixed-point initialization	$\mathcal{N}(0, I)$
Emission C, d, R	closed-form updates (Kato: frozen at PCA init.)
Precision	float64

Per-dataset hyperparameter	Double well	Van der Pol	Lorenz	Kato
Linear regimes	3	3	3	4
Prior noise scale σ (fixed by SING)	1.0	1.0	0.3	0.3
M-step learning rate	10^{-2}	3×10^{-3}	10^{-2}	10^{-2}
SSL kernel expectation	1-sample MC	5-pt Gauss–Hermite	3-pt Gauss–Hermite	$2d$ sigma points
Inducing points	16 (grid)	8×8 (grid)	4^3 (grid)	64 (k-means)
Inducing-grid padding (fraction of PCA range)	0.25	0.25	0.25	–
Natural-gradient schedule ρ	synthetic	synthetic	constant 0.1	constant 0.1
Inference sub-steps per observation interval	1	1	4	1
Partition temperature τ	0.5	0.5	0.5	5
Trials per vEM iteration	20	20	16	all windows

Table 4: SDE Matching hyperparameters (identical for all datasets unless noted).

Hyperparameter	Value
Prior drift network	MLP $d \to 64 \to 64 \to d$, tanh, last layer zero-initialized
Prior diffusion	constant diagonal, softplus-parameterized, initialized at 0.3
Encoder	backward GRU, hidden size 64
Marginal network $(m, \log s)$	MLP $(2 \cdot 64 + 1) \to 64 \to 64 \to 2d$, tanh
Initial prior $p(z_0)$	learned diagonal Gaussian
Emission C, d	affine, initialized by PCA of the training observations
Observation noise	diagonal, learned; init. at the dataset noise SD (Kato: 0.1)
Matching times per trajectory	4
Optimizer, learning rate	Adam, 10^{-3}
Gradient-norm clipping	10
Batch size (trajectories)	16
Training steps	10,000 (Lorenz: 3,000)
Validation cadence / selection	every 100 steps / validation objective (neg. ELBO)
Forecast solver step	0.01
Precision	float64

D.3 Evaluation Metrics

D.3.1 Simulated Data

Because the generating latents are known, we score the simulated systems directly, on independent test trajectories. The latent state is identifiable only up to an affine transformation, so we first fit an unrestricted affine map $z \mapsto zA + b$ by least squares from the inferred posterior means to the true latents *on the validation split*, after checkpoint selection, and freeze it before scoring the test split. This removes coordinate ambiguity without fitting away test error. No latent alignment enters the observation-space scores.

Reconstruction R^2 and RMSE. With $\hat{\mathbf{x}}_t = \mathbf{C}_\phi \bar{\mathbf{z}}_t + \mathbf{d}_\phi$ the reconstruction from the posterior mean and $\mathbf{x}_t^\star = \mathbf{C}^* \mathbf{z}_t + \mathbf{d}^*$ the noiseless signal (both in standardized units),

$$\mathrm{RMSE}_{\mathrm{rec}} = \sqrt{\tfrac{1}{TBD} \textstyle\sum_{t,b,i} (\hat{\mathbf{x}}_{t,b,i} - \mathbf{x}^\star_{t,b,i})^2}, \qquad R^2_{\mathrm{rec}} = 1 - \frac{\sum_{t,b,i} (\hat{\mathbf{x}}_{t,b,i} - \mathbf{x}^\star_{t,b,i})^2}{\sum_{t,b,i} (x^\star_{t,b,i} - \bar{\mathbf{x}}^\star_i)^2},$$

where b indexes test trajectories, i channels and $\bar{\mathbf{x}}^\star_i$ is the channel mean over the test split.

Latent NRMSE. With $\tilde{\mathbf{z}}_t = \bar{\mathbf{z}}_t \mathbf{A} + \mathbf{b}$ the aligned posterior mean and σ_j the standard deviation of latent coordinate j over the training trajectories,

$$\mathrm{NRMSE}_{\mathrm{lat}} = \sqrt{\tfrac{1}{TBd} \textstyle\sum_{t,b,j} \left((\tilde{\mathbf{z}}_{t,b,j} - \mathbf{z}_{t,b,j}) / \sigma_j \right)^2}.$$

Dynamics NRMSE. The learned drift is compared with the true drift at the same ground-truth test states: each true state $\mathbf{z}$ is mapped into the model's coordinates, $\mathbf{u} = (\mathbf{z} - \mathbf{b})\mathbf{A}^{-1}$, the learned field is evaluated there and mapped back, and

$$\mathrm{NRMSE}_{\mathrm{dyn}} = \frac{\sqrt{\tfrac{1}{TBd} \textstyle\sum_{t,b} \| f_\phi(\mathbf{u}_{t,b}) A - f(\mathbf{z}_{t,b}) \|^2}}{\sqrt{\tfrac{1}{T'B'd} \textstyle\sum_{t,b} \| f(\mathbf{z}^{\mathrm{train}}_{t,b}) \|^2}},$$

so that the denominator is the RMS true drift on the training states. Because both fields are evaluated at the true states, latent-trajectory error does not enter this metric; it evaluates the analytic drift, not finite differences of noisy paths.

Forecasting NRMSE. The second half of every test trajectory is hidden. Each method infers its posterior from the first half only, then propagates its learned prior SDE forward (128 sample paths) and we compare the mean forecast, mapped through the learned sensor, with the clean signal on the hidden half. The RMSE is divided by the pooled standard deviation of the per-channel-centered clean test signal, so that predicting the channel means scores 1.

D.3.2 Kato Dataset

There is no ground-truth latent, clean signal or drift, so the scores are observation-space and behavioral.

Reconstruction R^2. R^2 between the reconstruction $\mathbf{C}_\phi \bar{\mathbf{z}}_t + \mathbf{d}_\phi$ and the observed standardized fluorescence, pooled over neurons.

Forecasting NRMSE. The second half (100 frames, about 34 s) of every 200-frame window is hidden, the posterior is inferred from the first half, and the learned prior is propagated across the hidden half (128 samples). The RMSE of the mean forecast against the observed frames is divided by the RMSE of persistence, i.e., of repeating the last observed frame, so values below 1 beat the persistence floor of that worm.

Behavioral decoding. After the dynamical model is frozen we fit a class-balanced multinomial logistic regression from the posterior-mean latents to the annotated behavioral states. Features are standardized with training statistics; the inverse regularization strength $C \in \{10^{-3}, \ldots, 10^{2}\}$ is

selected by validation balanced accuracy (ties favor stronger regularization), and we report balanced accuracy (mean per-class recall) on held-out frames. Chance is $1/7$ for NoStim and $1/4$ for Stim worms.

D.4 Additional Results

Table 5 summarizes the simulated benchmarks behind Fig. 3D for ZENO, gpSLDS and SDE Matching. Six seeds were run per dataset and method; as in the figure, the seed with the worst dynamics NRMSE for each (dataset, method) pair is excluded and the remaining five are reported (this drops one diverged gpSLDS fit on Van der Pol, seed 4, whose reconstruction R^2 was -2.85; for the other pairs the dropped seed is an ordinary fit). ZENO reports its causal filtering posterior, whereas gpSLDS and SDE Matching report smoothing posteriors that condition on the full trajectory; the forecasting column is the one metric that is prefix-only for every method.

Table 5: Simulated benchmarks: mean $\pm$ SD over 5 seeds. Reconstruction R^2 and RMSE are against the noiseless signal in standardized units; latent, dynamics and forecasting errors are normalized as defined in Section D.3 (lower is better for all but R^2). ZENO scores are filtering estimates; the baselines are smoothers.

		Reconstruction		NRMSE ↓		
Dataset	Method	R^2 ↑	RMSE ↓	Latent	Dynamics	Forecast
Double well	ZENO	0.988 ± 0.004	0.027 ± 0.003	0.107 ± 0.013	0.342 ± 0.085	0.644 ± 0.163
	gpSLDS	0.992 ± 0.002	0.023 ± 0.003	0.090 ± 0.010	0.321 ± 0.188	0.574 ± 0.127
	SDE Matching	0.984 ± 0.008	0.030 ± 0.005	0.106 ± 0.024	0.488 ± 0.201	0.605 ± 0.141
Van der Pol	ZENO	0.978 ± 0.004	0.048 ± 0.002	0.154 ± 0.006	0.271 ± 0.030	0.655 ± 0.067
	gpSLDS	0.987 ± 0.002	0.038 ± 0.002	0.127 ± 0.005	0.192 ± 0.047	0.766 ± 0.026
	SDE Matching	0.981 ± 0.001	0.045 ± 0.001	0.139 ± 0.005	0.251 ± 0.054	0.766 ± 0.074
Lorenz	ZENO	1.000 ± 0.000	0.018 ± 0.001	0.014 ± 0.001	0.057 ± 0.002	0.029 ± 0.002
	gpSLDS	1.000 ± 0.000	0.012 ± 0.000	0.009 ± 0.000	0.038 ± 0.002	0.031 ± 0.001
	SDE Matching	1.000 ± 0.000	0.019 ± 0.001	0.010 ± 0.001	0.055 ± 0.003	0.065 ± 0.007

Table 6 lists the per-worm scores behind Fig. 4F–H. All scores follow the whole-trace protocol and are in-sample except the decoding probe, which holds out its own frames. gpSLDS forecast rollouts diverged on three animals (ratio to persistence above 10; marked "div."). Over the 12 animals, mean reconstruction R^2 is 0.830 (ZENO), 0.857 (gpSLDS) and 0.858 (SDE Matching); mean decoding balanced accuracy is 0.624, 0.524 and 0.482; and the mean forecast ratio is 0.78, 1.95 (over the nine non-diverged gpSLDS rollouts) and 0.71.

Table 6: Kato et al. (2015) per-worm results. N is the number of recorded neurons and T the recording length in seconds. For each method: reconstruction R^2 (↑), behavioral decoding balanced accuracy (↑; chance $1/7$ NoStim, $1/4$ Stim) and forecast RMSE relative to persistence (↓; below 1 beats persistence). "div." marks a diverged forecast rollout.

			ZENO			gpSLDS			SDE Matching		
Worm	N	T (s)	R^2 ↑	Decode ↑	Forecast ↓	R^2 ↑	Decode ↑	Forecast ↓	R^2 ↑	Decode ↑	Forecast ↓
NoStim 0	109	1080	0.876	0.617	0.613	0.891	0.403	0.658	0.883	0.232	0.652
NoStim 1	135	1080	0.862	0.537	0.761	0.893	0.250	0.731	0.884	0.255	0.701
NoStim 2	131	1080	0.801	0.577	0.782	0.840	0.364	0.770	0.833	0.481	0.774
NoStim 3	125	1080	0.894	0.492	0.730	0.906	0.499	0.806	0.891	0.464	0.628
NoStim 4	129	1080	0.850	0.641	0.838	0.856	0.726	div.	0.859	0.631	0.813
Stim 0	107	720	0.811	0.724	0.916	0.874	0.706	0.733	0.865	0.630	0.703
Stim 1	122	720	0.856	0.729	0.786	0.849	0.398	0.717	0.859	0.542	0.586
Stim 2	124	720	0.800	0.601	0.853	0.899	0.464	div.	0.887	0.458	0.550
Stim 3	134	720	0.782	0.664	0.742	0.819	0.688	div.	0.820	0.607	0.667
Stim 4	123	720	0.830	0.524	0.942	0.838	0.504	4.333	0.858	0.436	1.090
Stim 5	151	720	0.795	0.706	0.749	0.819	0.805	6.834	0.822	0.504	0.665
Stim 6	146	720	0.804	0.677	0.675	0.795	0.480	1.959	0.835	0.538	0.730

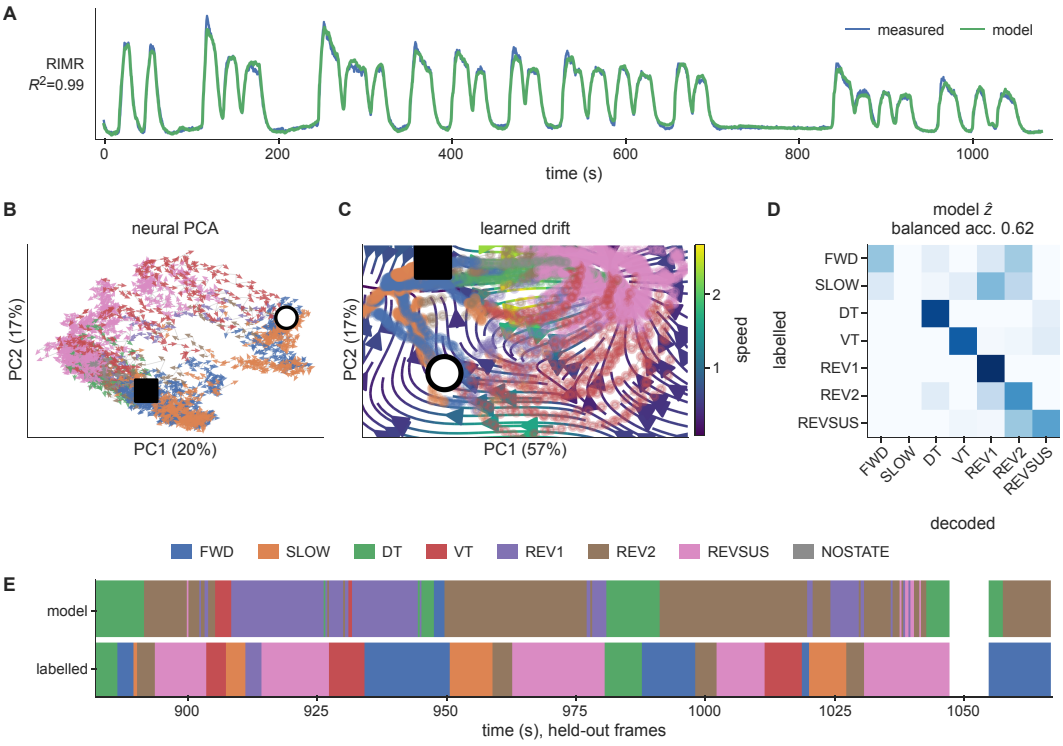

Figure 5: Example results for a NoStim worm from the Kato Dataset. See Fig 4A-E