MARKED INDUCING POINT CASCADED SDES FOR NEURAL MANIFOLD LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The manifold hypothesis suggests that high-dimensional neural time series lie on a low-dimensional manifold shaped by simpler underlying dynamics. To uncover this structure, latent dynamical variable models such as state-space models, recurrent neural networks, neural ordinary differential equations, and Gaussian process latent variable models are widely used. We propose MIP-CSDE (Marked Inducing Point Cascaded SDE), a novel cascaded stochastic differential equation model that balances computational efficiency with interpretability and addresses key limitations of existing approaches. Our model assumes that a sparse set of trajectory samples suffices to reconstruct the underlying smooth manifold. The manifold dynamic is modeled using a set of Brownian bridge SEDs, with points–specified in both time and value–drawn from a multivariate marked point process. These Brownian bridges define the drift of a second set of SDEs, where their trajectories are mapped to the observed data. This yields a continuous, differentiable latent process capable of modeling arbitrarily complex time series as the number of inducing points increases. For MIP-CSDE, we derive efficient training and inference procedures, demonstrating that its computational complexity of inference per iteration scales as $\mathcal{O}(P \cdot N)$, exhibiting linear dependence on the observation data length N, where P is the number of particles. We then show in both synthetic data and neural recordings that our proposed model can accurately recovers the underlying manifold structure and scales effectively with data dimensionality.

1 INTRODUCTION

The manifold hypothesis proposes that high-dimensional neural time series lie on a low-dimensional manifold shaped by simpler latent dynamics (Whiteley et al., 2024). Evidence for such structure appears in auditory cortex activity (Bondanelli et al., 2021) and in speech signals constrained by vocal tract mechanics (Gonzalez-Castillo et al., 2023). Methods for uncovering latent manifolds include state-space models (SSMs) (Särkkä and Svensson, 2023), dynamical autoencoders (Girin et al., 2020), switching SSMs (Ghahramani and Hinton, 2000), Gaussian and Dirichlet processes (Fox et al., 2008; Eleftheriadis et al., 2017; Wang et al., 2005), t-SNE and UMAP (Van der Maaten and Hinton, 2008; McInnes et al., 2018), and Latent Neural ODEs (Rubanova et al., 2019).

In this paper, we focus on SSMs for high-dimensional neural time series. Classical models include Linear Gaussian SSMs (Kitagawa and Gersch, 1996) and Hidden Markov Models (Rabiner, 2002), while modern variants include Deep SSMs (Rangapuram et al., 2018), Deep Kalman Filters (Krishnan et al., 2015a), GPDM (Wang et al., 2005), GPSSMs (Eleftheriadis et al., 2017), and nonlinear latent models such as LFADS (Sussillo et al., 2016) and GPFA (Yu et al., 2008). Recent approaches such as SING improve inference for latent SDEs (Hu et al., 2025). Despite these advances, limitations remain: some models fail to capture oscillatory dynamics, others require structural constraints, and DNN-based approaches are data-hungry. Sequence models like RNNs and LSTMs (Chang et al., 2024) capture nonlinear dependencies but pose interpretability and training challenges (Glorot and Bengio, 2010).

To address these limitations, we propose MIP-CSDE (Marked Inducing Point Cascaded SDE), which balances interpretability and expressive power. Inspired by findings that neural manifolds evolve smoothly along low-dimensional trajectories (Cunningham and Yu, 2014; Gosztolai et al., 2023), MIP-CSDE assumes that a sparse set of trajectory samples suffices to reconstruct the manifold. The first layer models trajectories with Brownian bridge SDEs, using inducing points from a multivariate marked point process (Daley and Vere-Jones, 2006; Oksendal, 2013). These trajectories define the drift for a second SDE layer, whose outputs map to observed data. This cascaded structure yields

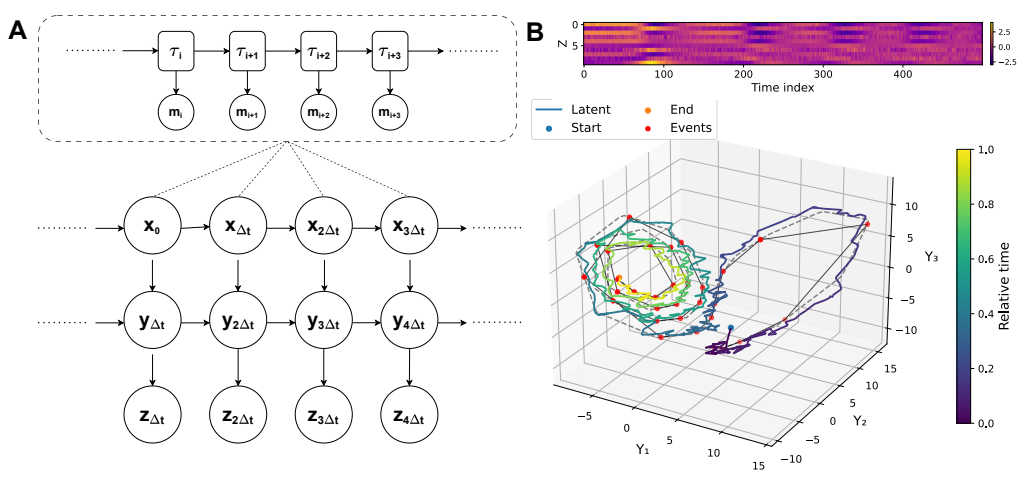

Figure 1: **Cascade SDE Model Structure and its Application in Decoding Lorenz System Trajectories:** (A) Shows the graphical representation of our proposed model, where inducing points (τ_i,m_i) go through two cascade SDEs (X_t,Y_t) followed by projection to the observation domain (Z_k). (B) Top row shows the observed data used as input to our proposed model. This data is generated by projecting the Lorenz trajectory through a 10-dimensional linear mapping followed by adding multivalent gaussian noise. The bottom row shows the underlying Lorenz trajectory and its estimation by our model, along with the timing of the inducing points. Here, MIP-CSDE simultaneously learns the mapping from the manifold to the observed space while inferring the nonlinear and complex dynamics of the Lorenz attractor. The timing and values of the inducing points reflect their adaptive behavior in capturing both fast and slow transitions along the trajectory.

a continuous, differentiable latent process capable of modeling arbitrarily complex dynamics as inducing points increase. We derive efficient inference and training procedures, with complexity scaling linearly with data length, and show that MIP-CSDE recovers latent manifolds accurately in both synthetic and neural datasets. Figure 1 illustrates the graphical model and a Lorenz-projected time series fit.

The paper is organized as follows. Section 2 presents MIP-CSDE, its properties, and inference procedure. Section 3 applies the model to simulated and neural data. Section 4 provides discussion, and Section 5 concludes.

2 Materials & Methods

In this section, we first define the components of our model, including the generation of inducing points and the SDEs that map these points to the observed time series data. We establish the universal approximation properties of the proposed model and then develop its training and inference procedures.

2.1 Cascade SDE Framework

Figure 1A illustrates the model structure. The inducing points are a set of event-value pairs that sample the underlying manifold in both time and value space. Each event is characterized by a timestamp t_i and an associated mark vector $\vec{m}_i$. The model considers an arbitrary finite sequence of these pairs, represented as the set $\mathcal{I} = \{(\vec{m}_i, t_i);\ i = 1, 2, \dots\}$. The joint probability distribution over a period T for L events is given by:

$$p\Big(\{(t_i, \vec{m}_i)\}_{i=1}^{L}, L, T\Big) = \exp\left(-\int_0^T \lambda(t \mid \mathcal{H}_t)\, dt\right) \prod_{i=1}^{L} \lambda(t_i \mid \mathcal{H}_{t_i})\, p(\vec{m}_i \mid t_i, \mathcal{H}_{t_i}) \qquad (1)$$

where $\lambda(t_i \mid \mathcal{H}_{t_i})$ is the event occurrence rate conditioned on the history of previous events $\mathcal{H}_i$, and $p(\vec{m}_i \mid t_i, \mathcal{H}_{t_i})$ defines the mark distribution conditioned on the event time t_i and the history $\mathcal{H}_i$ (Jacobsen, 2006). The sequence of event can also be described using waiting times, defined as $\tau_i = t_i - t_{i-1}$, which transforms the process into a renewal marked point process (Daley and Vere-Jones, 2006).

In the specific case, we can assume that the waiting times τ_i are independent of the previous events. Similarly, we can assume that $\vec{m}_i$ is independent of the previous events and current waiting time. Under these assumptions, the waiting times can be modeled using a Gamma distribution, and the values $\vec{m}_i$ can be modeled using a multivariate normal distribution.

With the inducing points generated, we now introduce the remaining components of the model that map these points to the observed time-series data. Let $Z_k \in \mathbb{R}^M$ denote the observed data at discrete time points $k = 1, \ldots, K$, which are modeled as functions of an underlying continuous latent process $Y_t \in \mathbb{R}^D$, where $D \ll M$. Y_t evolves according to another latent process X_t, which has the same dimension as Y_t. The latent process $X_t = \{x_t^d\}_{d=1}^D$ is modeled using a set of SDEs, where the process is constrained to reach the mark values at times specified by the inducing time points - i.e, a Brownian bridge SDE (Pitman and Yor, 1999). Evolution of state process in each dimension $d = 1, \ldots, D$ is defined by:

$$dx_t^d = \mu_t^d \, dt + \sigma_t^d \, dw_t^d, \tag{2}$$

where w_t^d is a standard Wiener process, μ_t^d is the drift term, and σ_t^d is the time-dependent diffusion coefficient. For $t \in [t_i, t_{i+1})$, which corresponds to the waiting period τ_{i+1}, the drift and diffusion terms are defined as:

$$\mu_t^d = \frac{m_{i+1}^d - x_t^d}{t_{i+1} - t}, \quad \sigma_t^d = \sqrt{\frac{(t_{i+1} - t)(t - t_i)}{t_{i+1} - t_i}}, \tag{3}$$

where m_{i+1}^d is the d-th component of the mark vector $\vec{m}_{i+1}$, the value that the process must reach at the event time t_{i+1}. The latent process $Y_t = \{y_t^d\}_{d=1}^D$ evolves according to:

$$dy_t^d = x_t^d \, dt + \sigma_y^d \, d\nu_t^d, \tag{4}$$

where the drift term for y_t^d is defined by x_t^d, ν_t^d is a standard Wiener process, and σ_y^d is the diffusion coefficient for the d-th component. Finally, the observations Z_k are defined as:

$$Z_k = W \begin{pmatrix} y_{k\Delta t}^1 \\ \vdots \\ y_{k\Delta t}^D \end{pmatrix} + \varepsilon_k, \tag{5}$$

where Δt denotes the sampling interval at which the observations Z_k are collected, $W \in \mathbb{R}^{M \times D}$ is a linear projection matrix, and $\varepsilon_k \sim \mathcal{N}(0, R)$ represents Gaussian noise with covariance $R \in \mathbb{R}^{M \times M}$. Here, we assume a linear noisy projection; more generally, the framework can accommodate other types of mappings. For instance, for the hippocampus data analyzed in Section 3.4, the mapping between Y_t and Z_k characterizes the rate function for a point process observation.

2.2 Model Properties

In this section, we discuss two key attributes of MIP-CSDE: its universal approximation capability and its computational cost. Other aspects of the model, including its nonparametric nature and strategies for managing the growth of inducing points, are discussed in the Appendix A.1.

2.2.1 Universal Approximation Property

When the process is deterministic and the inducing points are equally spaced, we can rely on the sampling theorem which suggests that a signal can be completely reconstructed from its samples (Shannon, 1949). In simple terms, any continuous function can be reconstructed from properly sampled data points. Here, we extend a similar idea to cascade SDEs using the inducing points.

Theorem: Let $f \in C([0, T])$ be a continuous function and let $\varepsilon > 0$ be arbitrary. Then there exists a choice of inducing points such that the expected one-dimensional component of the process, $\mathbb{E}[s_t]$, uniformly approximates the integral

$$g(t) = \int_0^t f(s) \, ds \tag{6}$$

within error ε, in the sense that

$$\sup_{t \in [0,T]} |\mathbb{E}[s_t] - g(t)| < \varepsilon. \tag{7}$$

Here, s_t denotes the d-th component y_t^d of the process $Y_t = \{y_t^d\}_{d=1}^D$.

Proof: In our model, each component is given by $s_t = \int_0^t x_s^d \, ds + \sigma W_t^d$, where x_t^d is a Brownian bridge, W_t^d is a standard Brownian motion, and σ is the noise variance.

Let $\{t_i\}_{i=1}^N$ be uniformly spaced inducing points, each associated with a mark m_i. Define $x_t^{d,N}$ by piecewise linear interpolation of these inducing points. Since piecewise linear functions are dense in $C([0,T])$, we can choose $\{m_i\}$ such that $x_t^{d,N} \to f(t)$ uniformly on $[0,T]$. Define $s_t^N = \int_0^t x_s^{d,N} \, ds$. Because integration preserves uniform convergence, it follows that

$$\mathbb{E}[s_t^N] \to g(t) = \int_0^t f(s) \, ds \quad \text{uniformly on } [0,T] \tag{8}$$

The noise term satisfies $\mathbb{E}[\sigma W_t^d] = 0$ and $\mathrm{Var}(\sigma W_t^d) = \sigma^2 t$. Applying Chebyshevs inequality, we have $P\big(|s_t - \mathbb{E}[s_t]| \geq \eta\big) \leq \frac{\sigma^2 T}{\eta^2}$, so for sufficiently small σ, the process $s_t]$ concentrates around its expectation. In Appendix A.2, we establish that as $N \to \infty$, the spacing of the inducing points converges to T/N. Thus, by selecting appropriate inducing points $\{(t_i, m_i)\}$, we ensure

$$\sup_{t \in [0,T]} |\mathbb{E}[s_t] - g(t)| < \varepsilon \tag{9}$$

Corollary: Given a proper set of inducing points, any multidimensional time series can be characterized through our proposed model.

Proof: With the above theorem, we have shown that each dimension of Y_t can be properly characterized using a finite set of inducing points. We can assume that each dimension of Y_t is independent of the others; thus, we only need to adjust the corresponding inducing point values to capture each specific dynamic. Note that as the number of sample points increases, convergence to the bound is achieved using the same set of inducing point times across dimensions.

This proof does not specify the manifold representation but shows that any such representation can be constructed with suitable dimensions and a sparse set of inducing points. In practice, marks and times are adjusted from the observations, creating dependencies across dimensions and among the inducing points.

2.2.2 Computational Cost

A key advantage of the proposed model lies in its favorable computational complexity compared to other non-parametric models such as GPs. Standard GPs require inversion of an $N \times N$ covariance matrix in their prediction step, which results in a computational complexity of $\mathcal{O}(N^3)$ (where N is the number of samples or time points) (Seeger, 2004). This scaling substantially restricts the applicability of Gaussian Processes for long temporal sequences or high-frequency data. In contrast, as we show in the next section, inference for the discrete representation of our model can be performed using a sequential Monte Carlo (SMC) (Doucet et al., 2001) approach. When using particle-based methods such as Particle Marginal Metropolis-Hastings (PMMH) Andrieu et al. (2010), the computational cost of each trajectory inference run scales as $\mathcal{O}(P \cdot N)$, where P is the number of particles and N again denotes the number of time points. This linear scaling with respect to N enables our proposed model to handle long trajectories more efficiently, making it well suited for characterization of high-resolution neural data. Moreover, the computational cost of inference is independent of the number of inducing points, since neither their number nor their values affect the SMC procedure. While generating inducing points incurs additional computational cost, their sampling rate is adapted to the underlying dynamics and scales as $O(L)$, where L is the maximum number of inducing points.

2.3 Model Training and Inference

The training objective is to maximize the marginal likelihood (or evidence) of the observed data $\{Z_k\}_{k=0}^K$. This involves updating several sets of parameters, including the event-value posterior distribution or parameters. Additionally, we must infer the trajectories of X_t and Y_t over $t = [0,T]$. For the training process, we use an Expectation-Maximization (EM) algorithm, which can deal with latent process (Brown and Kass, 2018). It is also possible to use a variational inference approach Blei et al. (2017), which is discussed in Appendix A.3.

To develop the EM algorithm, we first derive a discrete-time representation of our model. To accomplish this, we use the renewal waiting time process discussed in section 2.1. Under this assumption, we can break the non-Markovian dependence of X_t, which enables the use of recursive inference methods such as SMC. Simply put, at time t, we already know when the next inducing point occurs and its mark value, which breaks the dependence on the future trajectory of X_t. It is worth pointing out that inducing points are in continuous space, thus, we can reconstruct X_t and Y_t at any discrete resolution. Appendix A.4 discusses the discrete representation of SDEs. Given this representation, Algorithm 1 outlines the custom particle filtering algorithm developed for our model, which corresponds to inferring the states (X_t, Y_t, and the inducing points) when the model parameters are known. A key component of this inference procedure is the proper sampling of event–value pairs, which is addressed within the algorithm. A more detailed explanation of this approach is provided in Appendix A.5.

Algorithm 1 SMC for Joint Inference of Inducing Points and State Processes

Define:
$p(X_0), p(Y_0)$ — Processes initial distribution
$p(\tau, m \mid H_t)$ — Event-value distribution priors
Δt — Sampling interval
U — Number of particles
n_u — Number of inducing points for particle u
$\pi(X_k, Y_k \mid X_{0:k-1}, Y_{0:k-1}, Z_k, \{\tau, m\}_{0:n_u})$ — Proposal distribution
for $u = 1$ to U **do**
  Sample $X_0^u \sim p(X_0), Y_0^u \sim p(Y_0)$
  $(m_0^u, \tau_0^u) \sim p(\tau, m \mid H_0), n_u = 0$
end for
for $k = 1$ to K **do**
  for $u = 1$ to U **do**
    if $k\Delta t > \tau_{n_u}^u$ **then** ▷ Check if a new event-value pair is needed
      Sample new $(\tau^u, m^u) \sim p(\tau, m \mid H_t)$
      Update particle with new $\{\{\tau, m\}_{0:n_u}^u, \tau^u, m^u\}$ and increment n_u
    end if
    Sample $(X_k^u, Y_k^u) \sim \pi(X_k, Y_k \mid X_{0:k-1}^u, Y_{0:k-1}^u, Z_k, \{\tau, m\}_{0:n_u}^u)$
    Compute importance weight:

$$w_k^u = \frac{p(Z_k \mid X_k^u) \times p(Y_k^u \mid X_k^u) \times p(X_k^u \mid \{\tau, m\}_{0:n_u}^u)}{\pi_k(X_k^u, Y_k^u \mid X_{0:k-1}^u, Y_{0:k-1}^u, Z_k, \{\tau, m\}_{0:n_u}^u)}$$

  end for
  Normalize weights: $\hat{w}_k^u = \frac{w_k^u}{\sum_u w_k^u}$
  Resample particles using $\hat{w}_k^u$ (Each particle encompasses both processes, $X_{0:k}^u$ and $Y_{0:k}^u$, as well as the set of event-value pairs $\{\tau, m\}_{0:n_u}^u$)
end for

In the M-step, we maximize the expectation of the complete log-likelihood with respect to the model parameters. Using the particle filters from the E-step, we estimate the parameters of the event-time and mark model by maximizing the posterior. For the inducing points, we apply a MAP procedure, updating the parameters of the proposed distribution for each waiting period, so the number of parameters evolves as the model adapts to the dynamics. Observation process parameters can be updated in parallel. The M-step is computed numerically using a stochastic gradient ascent algorithm. Further details are provided in the Appendix A.6.

We defined the training and inference procedures for a single trial, but the model can be applied to multiple trials. Parameters can be shared across trials, while each trial retains its own inducing points and X_t and Y_t trajectories.

3 RESULTS

In this section, we evaluate our framework on simulated and real datasets. We first consider a one-dimensional chirp time series, then reconstruct a Lorenz system trajectory projected into a multidimensional observation space. We compare our approach to recent models for continuous latent dynamics. Finally, we apply the method to neural recordings. We decode rat hippocampal place-cell activity during navigation on a W-shaped maze and then apply our model to infer the underlying low-dimensional manifold of monkey neural activity in M1 and PMd during a center-out reach task.

3.1 CHIRP SIGNAL

For the first example, we use MIP-CSDE to reconstruct a chirp time series signal. The chirp consists of a single harmonic, with frequency changing linearly or nonlinearly over time. We consider it a good benchmark to assess whether our model can adjust inducing points timing and values to capture the dynamics. We generated 500 samples at 20 Hz, adding Gaussian noise (variance 0.1) to each sample of it. The frequency decreases linearly from 0.2 to 0.1 Hz over 25 seconds. X_t and Y_t are 1-dimensional processes ($\sigma_x = 10^{-1}$, $\sigma_y = 10^{-4}$), with waiting times following a Gamma(2,1) prior (mean 2, SD 1.41) and values with a Normal(0,1) prior.

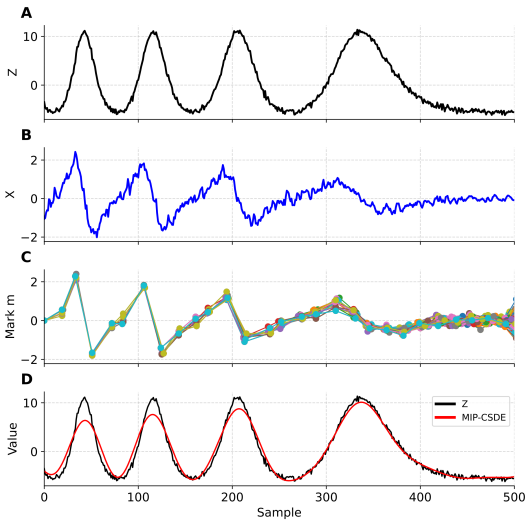

Figure 2: **Chirp Signal Reconstruction Using MIP-CSDE:** (A) Simulated chirp signal with additive Gaussian noise. (B) Mean of X_t inference derived using the SMC. (C) Inferred inducing points at the last iteration of EM. (D) Y_t trajectory generated through Cascade SDE. The generated Y_t reasonably matches the ground truth signal.

To fit the model, we ran the SMC inference algorithm (Table 1) with 1000 particles over 12 EM iterations, to update inducing point parameters while reconstructing the signal. Smaller particle numbers were tested, and ∼1000 particles provided robust inference. The mean number of inducing points increased from 13 at first iteration of SMC to 36 at the end, and their intensity, initially uniform, adapted per iteration to capture the temporal in dynamics.

Figure 2 shows the modeling results. The inferred signal in Figure 2B represents the bridge SE, where corresponding inducing points are shown in Figure 2C. Using these points, we can generate trajectories (Figure 2D) of process, which is close to the simulated chirp signal.

3.2 LORENZ SYSTEM TRAJECTORY RECONSTRUCTION

We applied our framework to simulated data generated by the Lorenz attractor system (Lorenz, 1963), a classic example of chaotic dynamics. This benchmark is ideal for assessing whether latent variable models can recover complex attractor geometry from noisy, high-dimensional observations.

The three-dimensional Lorenz trajectories were projected into a 10-dimensional observation space using a random Gaussian matrix (W in Equation 5). The projected signal was sampled at 10 Hz for 500 points, with correlated Gaussian noise added to generate Z. X_t and Y_t dimensions are set to 3, and inference was performed using our SMC–EM procedure with 20,000 particles and 25 EM iterations (noise variance for the state processes are set $\sigma_x^d = 1$ and $\sigma_y^d = 10^{-3}$ for $d = 1, 2, 3$). The model simultaneously learned W, Lorenz trajectories, and the timing and values of inducing points. We checked the likelihood growth and parameter updates, which indicated stable convergence.

Figure 1B shows the observed data (Z) and inferred manifold. The inferred trajectory closely matches the simulated one. Inducing points concentrate during lobe transitions, demonstrating adaptive allocation to rapidly changing dynamics.

3.3 COMPARATIVE ANALYSIS ON SIMULATED DATASETS

To evaluate the performance of MIP-CSDE, we benchmark it against several continuous-time baselines in terms of both predictive accuracy and computational efficiency. The baselines include models of increasing flexibility: a Linear SDE, a Gaussian Process SDE (GP-SDE) (Duncker et al., 2019), and the Gaussian Process Switching Linear Dynamical System (GP-SLDS) (Hu et al., 2024). To ensure a robust comparison, all baseline models were fitted using the highly efficient SING inference framework (Hu et al., 2025).

We report performance using the MSE in the observation space. This choice of metric is necessitated by a key property of the baseline models, where their latent spaces are only identified up to an

Table 1: Performance comparison with baseline continuous-time models. For simulated data (Chirp, Lorenz), we report MSE. (Results are reported as mean ± standard error across 5 trials.)

Dataset	MIP-CSDE	Linear SDE	GP-SDE	GP-SLDS
Chirp Signal	0.30 ± 0.02	0.48 ± 0.05	0.39 ± 0.06	0.36 ± 0.02
Lorenz System	0.18 ± 0.01	0.28 ± 0.04	0.25 ± 0.03	0.21 ± 0.03

arbitrary affine transformation. This implies that a direct comparison between their inferred latent trajectories and the ground truth is not meaningful without a post-hoc alignment procedure. The reconsructed latent along with realigned version is presented in appendix A.8.

The quantitative results in Table 1 indicate that MIP-CSDE attains the lowest error on the Lorenz system (MSE 0.18 ± 0.01), while performing comparably to GP-SDE on the chirp signal (MSE 0.30 ± 0.02 vs. 0.29 ± 0.06). On the Lorenz benchmark, our implementation required approximately $290\,\mathrm{s}$, compared with $12\,\mathrm{s}$ for Linear SDE, $38\,\mathrm{s}$ for GP-SDE, and $252\,\mathrm{s}$ for GP-SLDS. We also observed higher memory usage for GP-SLDS on longer sequences, which is consistent with the scaling behavior of GP-based kernels. For MIP-CSDE, runtime scales approximately linearly with the number of particles increasing particles improves accuracy at additional computational cost, reflecting an explicit accuracy-compute trade-off. Although implementation details and hardware choices affect absolute timings, the observed trends are consistent with the computational analysis presented in Section 2.2.2. As expected, measured runtimes scale approximately linearly with both particle count and sequence length (Appendix A.9). Overall, these results support MIP-CSDE as a more accurate and computationally competitive approach for recovering latent dynamics and manifolds in continuous time.

3.4 Rat Hippocampus: Decoding Spatial Trajectories from CA1 Spiking

We applied MIP-CSDE to hippocampal CA1 place cells' recording from a rat navigating a W-shaped maze Joo and Frank (2018). Decoding trajectories from place cells' activity is a longstanding benchmark task for SSMs. Prior work has used more flexible decoders, including Gaussian mixtures and neural models, to improve robustness Yousefi et al. (2019); Karlsson and Frank (2008); Brown et al. (1998). The dataset in this example contained spiking activity of 62 place cells sampled at every 33 ms.

X_t and Y_t dimensions are set at 2, with X_t represents a proxy of the rat velocity of rat and Y_t position. The cell spiking activity is characterized using a Poisson point-process model, where the firing rate λ_i of cell i is estimated non-parametrically using kernel-based intensity functions Yousefi et al. (2019). Given the rate function and cell spiking activity, the likelihood of Y_t is defined by

$$\begin{aligned} p(Y_t \mid z_t^i) &\propto p(z_t^i \mid \lambda_{i,t}) \\ &= (\Delta t\, \lambda_{i,t})^{z_t^i} \exp(-\lambda_{i,t} \Delta t) \end{aligned} \tag{10}$$

with $\Delta t = 33$ ms. The full likelihood given 62 cell activities is defined by the product of their corresponding likelihood across cells. We used 80% of the trajectory to estimate and build the cell rate models and decoded the remainder. Decoding was performed with 10,000 particles, initialized at the rats starting location with zero

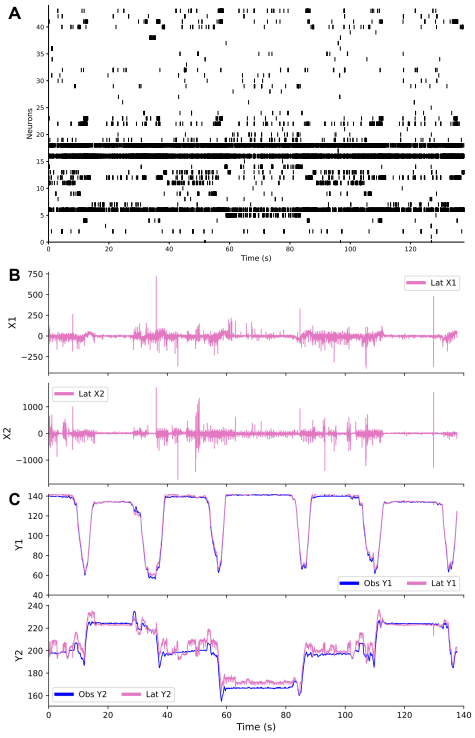

Figure 3: **MIP-CSDE Place Cell Decoder:** (A) Raster plot of 62 cells. (B) Inferred mean velocity in horizontal and vertical axes. (C) Decoded mean position vs. true trajectory. The model achieves high decoding accuracy and robustness compared to others.

velocity. Inducing-point priors were $\mathrm{Gamma}(8, 2)$ for waiting times and $\mathcal{N}\Big(\begin{bmatrix}0\\0\end{bmatrix}, 0.5\,\mathbf{I}_{2\times 2}\Big)$ for values.

Figure 3 shows the raster plot and mean of decoded trajectory, which closely follows the rats actual path traversing left, middle, right arms. Inducing points were sparse on straight arms but concentrated in the central corridor and corners, consistent with pauses, turns (Appendix Figure 7). Decoding accuracy was 6.1 ± 0.4 cm (2D MSE), outperforming state-of-the-art decoders such as SSM (17.2 cm) and Gaussian Mixture SSM (14 cm). The results and almost real-time processing speed (2 msec per time interval) suggest MIP-CSDE can be deployed as a fast and accurate decoder model.

3.5 MANIFOLD DIFFRENTIOTION DURING MONKEY REACHING TASK

We applied our framework to the Neural Latents Benchmark (NLB) MC_Maze dataset (Pei et al., 2021), which contains high-resolution electrophysiological recordings from macaque dorsal premotor (PMd) and primary motor (M1) cortices during a center-out reaching task (Churchland et al., 2012a). Previous analyses of this dataset have shown that neural activity in M1 and PMd during such stereotyped, planned movements can be predicted from the population state at movement onset. The dataset includes recordings from 182 neurons sampled at 1 ms resolution, along with behavioral covariates such as hand position, velocity, and cursor location.

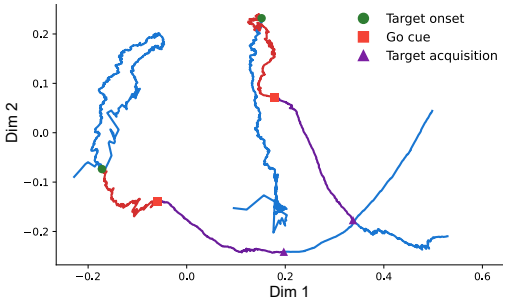

Figure 4: **Inferred Neural Manifolds During Preparation and Reach Phases.** The inferred manifolds in the preparation and reach phases show similar trajectories, with the reach phase an elongated replica of the preparatory phase. This pattern is expected, as the reach phase usually takes longer to complete.

Our aim was to infer an unsupervised manifold representation of neural activity and to examine whether the inferred dynamics during the preparation phase correlate with those in the reach phase. We hypothesized that the manifold representation in the preparatory phase would be a scaled version of that in the acquisition phase, and potentially differ across reach targets.

We assumed a two-dimensional latent space, with each neurons spiking activity modeled as a Bernoulli process, where the spiking probability depended on a linear combination of the latent state at the same moment. We ran MIP-CSDE with 10,000 particles and 30 EM steps. Figure 4 shows the inferred manifolds for two example trials. Within each trial, the trajectories during the reach and preparation phases were similar, but they differed across trials. These results are consistent with previous findings.

We also observed a higher rate of inducing points during the preparatory phase, similar to what we found in our rat hippocampus model, where the rate increased during more complex movements. This suggests more intricate neural dynamics during these periods. A critical advantage of MIP-CSDE is that it does not impose priors on the kernel or dynamics; the only prior assumption concerns the latent dimensionality. We will address how this limitation can be further refined in the Discussion section.

4 DISCUSSION

We developed the training and inference pipeline for a discretized version of our framework and applied it to simulated and neural datasets. The first example introduced the core concept of the model, showing its ability to draw proper inducing points that robustly captures temporal dynamics present in the data. The second example used high-dimensional simulated data with nonlinear and complex underlying dynamics, and highlights model capability to properly scale as the dimension of observations or latent states grow. In the third example, we showed that the model can be utilized as a robust and accurate dynamical decoder. In the fourth case, the model inference of latent processes aligns with the preparatory phase of motor task, despite there is no prior knowledge on how the neural activities encode movement. We also compared our model on simulated data against models such as Latent ODEs, SING, and DiGP. Our model demonstrates comparable or even better

prediction accuracy, while being computationally efficient and achieving similar runtime. These collective results underscoring our proposed model usefulness in tasks like manifold discovery and neural decoding. Besides, these results corroborate with the universal approximation of the model as discussed in Section 2.

It is worth to mention that with other models such as DKF or GPs (Krishnan et al., 2015b) (Casale et al., 2018) , we can infer the latent processes or reconstruct the data; however, they require large datasets, rely on strong priors on kernel or covariate choices, and involve more complex training and inference steps. Our model does not require a pre-defined kernel or extensive dataset and more importantly maintains a simple and interpretable structure. This is especially valuable in fields like neuroscience, where understanding the relationship between latent dynamics and observed data is crucial (Churchland et al., 2012b). Furthermore, our model offers dual interpretability, one where we can analyze the latent trajectories and their connection to other covariates, or we can examine the timing and values of the inducing points that shape the manifold.

We used a discretized representation of the model to derive its training and inference. However, extending to a continuous formulation similar to neural ODEs (Chen et al., 2018) could improve inference robustness and will reduce numerical errors in generating states trajectory. Noise variance in both the X_t and Y_t processes strongly influences the models behavior. High variance in the drift complicates inducing points inference, while the variance in X_t controls the flexibility of the interpolation paths, from nearly linear to highly flexible. In current development of the model, we adjust their values by checking different values for both; however, tuning these parameters can be explored as part of the model training. Training and inference in the model are performed using a custom SMC algorithm. While SMC in general can accommodate different observation types such as the point process observation model we used in neural data example, there might be more efficient algorithms for specific cases, such as the observation model defined in Equation (2). When event-value pairs are known, the model reduces to a linear state-space system, allowing use of the Kalman filter, which provides faster and more robust estimation. This motivates a hybrid approach, which we can use SMC for the event-value inference and the Kalman filter for state estimation, which will be explored in further development of this framework. Currently, we assume independence between marks and waiting times, and no dependence on past events. This simplifies the model inference and training but may reduce its ability to adjust inducing points to capture changes in data happening at different temporal scales. For example, fast transition in the latent state may require shorter waiting times and larger marks. As a result, exploring alternative distributions for mark and waiting time including those that are time or history dependent, may be necessary to improve model performance.

In our framework, the dimensionality of the latent manifold is set as a model hyperparameter. To determine an appropriate value, one can either evaluate performance across a range of candidate dimensions or adopt an automatic relevance determination (ARD) **?** strategy. In the latter approach, priors are placed on the observation model linking latent processes to observations, allowing the effective dimensionality to be learned through shrinkage.

5 Conclusion

Here, we introduce a cascade SDE framework to infer the underlying latent structure and manifold present in high-dimensional time series using a sparse set of inducing points, adaptively placed in both time and value space. The model achieves a high level of expressive power, while its computational cost grows only linearly in both data dimensions and time. The comparative analyses indicate that it achieves reconstruction accuracy on par with or superior to state-of-the-art models. These results suggest that the model holds promise as an unsupervised dimensionality reduction tool and can be robustly applied as a dynamical neural decoder or adaptive feature extractor across a range of neuroscience applications.

REFERENCES

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(269):269–342, 2010.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

G. Bondanelli, T. Deneux, B. Bathellier, and S. Ostojic. Network dynamics underlying off responses in the auditory cortex. *Elife*, 10:e53151, 2021.

E. N. Brown and R. E. Kass. Estimating a state-space model from point process observations. *Unpublished Manuscript*, 2018.

E. N. Brown, L. M. Frank, D. Tang, M. C. Quirk, and M. A. Wilson. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Journal of Neuroscience*, 18(18):7411–7425, 1998.

F. P. Casale, A. V. Dalca, L. Saglietti, J. Listgarten, and N. Fusi. Gaussian process prior variational autoencoders. 31, 2018.

H. Chang, Q. Zhang, Y. Wang, Z. Qin, L. Zhao, and H. Wang. Unlocking the power of lstm for long term time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (4):4292–4300, 2024.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. 31, 2018.

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 2012a.

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012b.

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

D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer Science & Business Media, 2006.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

L. Duncker, G. Bohner, J. Boussard, and M. Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *International conference on machine learning*, pages 1726–1734. PMLR, 2019.

S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman. Identification of gaussian process state space models. *Advances in neural information processing systems*, 30, 2017.

E. Fox, E. Sudderth, M. Jordan, and A. 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.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000. doi: 10.1162/089976600300015619.

L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. *arXiv preprint arXiv:2008.12595*, 2020.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256. PMLR, 2010.

J. Gonzalez-Castillo, I. S. Fernandez, K. C. Lam, D. A. Handwerker, F. Pereira, and P. A. Bandettini. Manifold learning for fmri time-varying functional connectivity. *Frontiers in Human Neuroscience*, 17:1134012, 2023.

A. Gosztolai, R. L. Peach, A. Arnaudon, M. Barahona, and P. Vandergheynst. Interpretable statistical representations of neural population dynamics and geometry. *arXiv preprint*, 2023.

A. Hu, D. Zoltowski, A. Nair, D. Anderson, L. Duncker, and S. Linderman. Modeling latent neural dynamics with gaussian process switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 37:33805–33835, 2024.

A. Hu, H. Smith, and S. Linderman. Sing: Sde inference via natural gradients. *arXiv preprint arXiv:2506.17796*, 2025.

M. Jacobsen. *Point Process Theory and Applications: Marked Point and Piecewise Deterministic Processes*. Birkhäuser, Boston, 2006.

H. R. Joo and L. M. Frank. The hippocampal sharp wave–ripple in memory retrieval for immediate use and consolidation. *Nature Reviews Neuroscience*, 19(12):744–757, 2018.

M. P. Karlsson and L. M. Frank. Network dynamics underlying the formation of sparse, informative representations in the hippocampus. *Journal of Neuroscience*, 28(52):14271–14281, 2008. doi: 10.1523/JNEUROSCI.4261-08.2008.

G. Kitagawa and W. Gersch. Linear gaussian state space modeling. In *Smoothness priors analysis of time series*, pages 55–65. Springer, 1996.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters, 2015a.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015b.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

B. Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

F. Pei, J. Ye, D. M. Zoltowski, A. Wu, R. H. Chowdhury, H. Sohn, J. E. ODoherty, K. V. Shenoy, M. T. Kaufman, M. Churchland, M. Jazayeri, L. E. Miller, J. Pillow, I. M. Park, E. L. Dyer, and C. Pandarinath. Neural latents benchmark 21: Evaluating latent variable models of neural population activity. In *Advances in Neural Information Processing Systems.*, 2021.

J. Pitman and M. Yor. Brownian bridge and related stochastic processes. *Probability Surveys*, 1: 1–61, 1999.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 2002.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Y. Rubanova, R. T. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

S. Särkkä and L. Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

M. Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14 (02):69–106, 2004.

C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

D. Sussillo, R. Jozefowicz, L. Abbott, and C. Pandarinath. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Physical Review*, 36(5): 823–841, 1930.

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

J. Wang, A. Hertzmann, and D. J. Fleet. Gaussian process dynamical models. *Advances in neural information processing systems*, 18, 2005.

N. Whiteley, A. Gray, and P. Rubin-Delanchy. Statistical exploration of the manifold hypothesis, 2024.

A. Yousefi, A. K. Gillespie, J. A. Guidera, M. Karlsson, L. M. Frank, and U. T. Eden. Efficient decoding of multi-dimensional signals from population spiking activity using a gaussian mixture particle filter. *IEEE Transactions on Biomedical Engineering*, 66(12):3486–3498, 2019.

B. M. Yu, J. P. Cunningham, G. Santhanam, S. Ryu, K. V. Shenoy, and M. Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in neural information processing systems*, 21, 2008.

A Appendix

A.1 Nonparametric and Non-Markovian Properties

In our model, the X_t exhibits dynamics similar to that of samples from a GP with specific covariance structures, for example, an Ornstein-Uhlenbeck process corresponding to an exponential kernel (Uhlenbeck and Ornstein, 1930). Within the model, the number of inducing points is not fixed in advance and it adapts dynamically to the complexity of the observed dynamics. The nonparametric and GP-like nature of our model makes it a alternative choice for machine learning and probabilistic applications. In our proposed model, the trajectory of X_t process is dependent to both past and future event-value pairs; thus it does not satisfy the Markovian property. This might complicate both training and inference within our framework. In section 2.1 , we reformulated the inducing point distribution using a renewal marked point process, which lets us to define X_t and Y_t process with a Markovian property. In section 2.3, we leverage this reformulation of SDEs for the inference and training of the model.

A.2 Convergence of Spacing Between Adjacent Samples

Let $X_1, X_2, \ldots, X_N$ be i.i.d. random variables uniformly distributed on $[0, T]$, and let $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(N)}$ denote their order statistics. Define the adjacent spacings $d_i = X_{(i+1)} - X_{(i)}$ for $i = 1, \ldots, N-1$.

From properties of uniform order statistics, the expected value of the i-th order statistic is $\mathbb{E}[X_{(i)}] = \frac{iT}{N+1}$. It follows that

$$\mathbb{E}[d_i] = \mathbb{E}[X_{(i+1)} - X_{(i)}] = \frac{T}{N+1} \tag{11}$$

which satisfies $\mathbb{E}[d_i] \to \frac{T}{N}$ as $N \to \infty$.

Moreover, the variance of d_i satisfies $\mathrm{Var}(d_i) = O(N^{-2})$. By Chebyshevs inequality,

$$\mathbb{P}\left(|d_i - \mathbb{E}[d_i]| \geq \epsilon\right) \leq \frac{\mathrm{Var}(d_i)}{\epsilon^2} = O\left(\frac{1}{N^2}\right) \tag{12}$$

which vanishes as $N \to \infty$. Therefore, $d_i \to \frac{T}{N}$ in probability.

Furthermore, classical results on uniform spacings imply that the maximum spacing $\max_i d_i$ satisfies

$$\max_{1 \leq i \leq N-1} d_i = \frac{T}{N} + O\left(N^{-1/2}\right) \tag{13}$$

with high probability, confirming that all gaps become uniformly close to $\frac{T}{N}$ as $N \to \infty$.

A.3 Variational Inference for Model Training

To complement the EM approach, we also develop a variational inference (VI) algorithm for training our model. As in Section 2.1, we begin with the discrete-time representation of the renewal process, which breaks the non-Markovian dependence of X_t by ensuring that the next inducing point and its mark are known at any time t. This property makes recursive inference feasible, but instead of relying on exact latent sampling as in the E-step of EM, we approximate the intractable posterior using a variational family.

Specifically, we introduce the following structured mean-field approximation:

$$q_\phi\big(X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L\big) = q_X(X_{0:K}; \phi_X)\, q_Y(Y_{0:K}; \phi_Y)\, q_{\tau,m}(\{\tau_i, m_i\}_{i=1}^L; \phi_{\tau,m}), \tag{14}$$

where $\phi = \{\phi_X, \phi_Y, \phi_{\tau,m}\}$ are variational parameters. This factorization decouples states and inducing points while retaining the renewal structure. In practice, we amortize these distributions using neural networks that map observed data into variational parameters.

The training objective is the evidence lower bound (ELBO):

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi}\Big[\log p_\theta(Z_{0:K}, X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L) - \log q_\phi(X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L)\Big], \quad (15)$$

where θ denotes the generative model parameters. Maximizing $\mathcal{L}(\theta, \phi)$ yields both approximate posterior inference (via q_ϕ) and maximum likelihood estimation of θ.

We employ stochastic gradient variational Bayes (SGVB) with the reparameterization trick to obtain low-variance gradient estimates. In this formulation, sampling of event–value pairs is embedded directly into the variational distribution $q_{\tau,m}$, which is parameterized by waiting-time and mark distributions. These distributions can be chosen flexibly, e.g., Gamma and Gaussian, or replaced with neural flows for greater expressivity.

Algorithm 2 Variational Inference for Inducing Point and State Estimation

1: **Initialize:** model parameters θ, variational parameters ϕ
2: **for** each training iteration **do**
3: Sample latent variables from q_ϕ:

$$X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L \sim q_\phi$$

4: Compute stochastic ELBO estimate:

$$\widehat{\mathcal{L}} = \log p_\theta(Z_{0:K}, X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L) - \log q_\phi(X_{0:K}, Y_{0:K}, \{\tau_i, m_i\}_{i=1}^L)$$

5: Update (θ, ϕ) via gradient ascent on $\widehat{\mathcal{L}}$
6: **end for**

Unlike EM, where process noise parameters are typically fixed, VI allows them to be included in the variational family and learned directly. In practice, however, we sometimes constrain these parameters to preserve stability of the underlying SDEs.

Finally, while the description above applies to a single observed trajectory, the variational framework naturally extends to multiple trials. Each trial maintains its own approximate posterior over inducing points and latent trajectories, while global parameters θ are shared across trials. This amortized formulation enables scalable training across large experimental datasets.

A.4 Discrete Representation of Hierarchical SDE

We focus on a discrete-time formulation of the model. To construct the discrete process, we assume that X_t and Y_t are sampled at regular intervals of Δt. The discrete representation of x_t^d is defined as:

$$\begin{aligned} x_{k+1}^d &= x_k^d + \frac{m_{i+1}^d - x_k^d}{t_{i+1} - k\Delta t}\,\Delta t + \sqrt{\frac{(t_{i+1} - k\Delta t)(k\Delta t - t_i)}{t_{i+1} - t_i}\Delta t}\cdot w_k^d, \\ w_k^d &\sim \mathcal{N}(0, \sigma_x^{d^2}) \end{aligned} \quad (16)$$

where w_k^d is a Gaussian noise term. Similarly, the discrete-time evolution of y_t^d is:

$$y_{k+1}^d = y_k^d + x_k^d\,\Delta t + \sqrt{\Delta t}\cdot \nu_k^d, \quad \nu_k^d \sim \mathcal{N}(0, \sigma_y^{d^2}) \quad (17)$$

where ν_k^d is Gaussian noise. The discrete observation process is given by:

$$Z_k = WY_k + \xi_k, \quad \xi_k \sim \mathcal{N}(0, R) \quad (18)$$

where $Y_k = \begin{pmatrix} y_k^1 \\ \vdots \\ y_k^D \end{pmatrix}$, $W \in \mathbb{R}^{M \times D}$ is a projection matrix, and ξ_k is observation noise.

To ensure accuracy, Δt must be much smaller than the minimum inter-event time, i.e., $\Delta t \ll \min(\tau_i)$, so that no two inducing points fall within the same discrete time bin. This constraint can be satisfied by analyzing the posterior distribution of waiting times and adjusting Δt accordingly. In essence, we require an orderly event process—allowing at most one event per bin—which can be enforced by carefully selecting the bin size Δt.

When using the waiting time representation, at time k, we already know when the next inducing point (e.g., event 213) will occur and what its value will be. From a Markovian perspective, we can assume that the entire process is determined at time k. This assumption simplifies the inference procedure presented in Algorithm **??**.

A.5 Detailed Version of the SMC Algorithm

Here, we provide a more detailed description of the SMC algorithm introduced in the main text for inference in our model. This is presented in Algorithm 2.

A.6 Training Step: M-step

For the M-step, we assume that the SMC algorithm has been run and that we have obtained D_K^u for $u = 1, \ldots, U$. The full likelihood of the process is defined as:

$$\begin{aligned} P(Z_{1:K}, X_{0:K}, Y_{0:K}, \tau_{1:n_u}, \vec{m}_{1:n_u}; \omega, \omega_0) = P(X_0, Y_0) \prod_{k=1}^{K} p(Z_k \mid Y_k, W, R) p(Y_k \mid X_{k-1}, Y_{k-1}, \sigma_y) \\ \times p(X_k \mid X_{k-1}, \tau_{1:n_s}, \sigma_y, \vec{m}_{1:n_s}) \prod_{n=1}^{n_u} p(\tau_n) p(\vec{m}_n) p(\omega_0^n) \end{aligned} \tag{19}$$

Here, ω represents the model parameters $\{W, R, \sigma_y, \sigma_x\}$, and ω_0 is the set of hyperparameters defining the priors, as detailed in Appendix A.1. The term $p(\omega_0^n)$ appears for each evnt-mark pair because the prior is applied individually to each waiting time and mark.

In the M-step, we compute the expectation of the full log-likelihood with respect to the posterior distribution over the latent processes and variables in the model. The latent processes are denoted by X and Y, while t_i and m_i represent another set of latent variables. The Q-function, with respect to which the expectation is taken, is defined as:

$$\begin{aligned} Q &= \mathbb{E}_{p(X_{0:K}, Y_{0:K}, \tau_{1:n_u}, \vec{m}_{1:n_u} \mid Z_{1:K}, \omega)} \left[\log P(Z_{1:K}, X_{0:K}, Y_{0:K}, \tau_{1:n_u}, \vec{m}_{1:n_u}; \omega) \right] \\ &= \sum_{u=1}^{U} \log P(X_0^u, Y_0^u) + \sum_{u=1}^{U} \sum_{k=1}^{K} \log p(Z_k \mid Y_k^u, W, R) + \sum_{u=1}^{U} \sum_{k=1}^{K} \log p(Y_k^u \mid X_{k-1}^u, Y_{k-1}^u, \sigma_x) \\ &\quad + \sum_{u=1}^{U} \sum_{k=1}^{K} \log p(X_k^u \mid X_{k-1}^u, \tau_{1:n_s}^u, \sigma_y, \vec{m}_{1:n_s}^u; n_s = \min_n \tau_n^u > k) \\ &\quad + \sum_{u=1}^{U} \sum_{n=1}^{n_u} \log p(\tau_n^u) + \sum_{u=1}^{U} \sum_{n=1}^{n_u} \log p(\vec{m}_n^u) + \sum_{n=1}^{n_u} \log p(\omega_0^n) \end{aligned} \tag{20}$$

For the observation model defined in Equation 5, the estimation of parameters W and R corresponds to a multivariate linear regression fit to samples of the X trajectory. Thus, W and R can be estimated in closed form, similar to the approach used in linear regression. The waiting time distribution

Algorithm 3 SMC Algorithm For Inferring Inducing Points and State Estimation

Set Algorithm Hyperparameters:
Set number of particles U
Define initial distributions $p(x_0)$ and $p(y_0)$
Define proposal density $\pi_k(x_k, y_k \mid x_{0:k-1}, y_{0:k-1}, z_k, \tau_{0:n_u}, m_{0:n_u})$
Set hyperparameters α_0, λ_0 For τ distribution
Set hyperparameters μ_0, ξ_0 For m distribution
Initialization:
for $u = 1$ to U **do**
    Sample $x_0^u \sim p(x_0)$, $y_0^u \sim p(y_0)$
    Set $m_0^u = \vec{0}$, $\tau_0^u = 0$, $n_u = 0$
    Set initial weight $w_k^u = \frac{1}{U}$
    Initialize particle $D_0^u = \{x_0^u, y_0^u, \tau_0^u, m_0^u, n_u\}$
end for
Inference:
for $k = 1$ to K **do**
    1. Time & Mark Sampling:
    for $u = 1$ to U **do**
        if $k \cdot \Delta t > \tau^u_{\max(n_u)}$ **then**
            Sample $\tau^u_{\text{new}} \sim \Gamma(\tau; \alpha_0, \lambda_0)$
            Sample $m^u_{\text{new}} \sim \mathcal{N}(m; \mu_0, \xi_0)$
            Update $D_k^u = \{x^u_{0:k-1}, y^u_{0:k-1}, \tau^u_{0:n_u}, m^u_{0:n_u}, \tau^u_{\text{new}}, m^u_{\text{new}}, n_u + 1\}$
        end if
    end for
    2. Sampling:
    for $u = 1$ to U **do**
        Sample $(x_k^u, y_k^u) \sim \pi_k(x_k, y_k \mid x^u_{0:k-1}, y^u_{0:k-1}, z_k, \tau^u_{0:n_u}, m^u_{0:n_u})$
        Compute importance weight:

$$w_k^u = w_{k-1}^u \cdot \frac{p(z_k \mid y_k^u) \cdot p(y_k^u \mid x_{k-1}^u) \cdot p(x_k^u \mid \tau^u_{0:n_u}, m^u_{0:n_u})}{\pi_k(x_k^u, y_k^u \mid x^u_{0:k-1}, y^u_{0:k-1}, z_k, \tau^u_{0:n_u}, m^u_{0:n_u})}$$

    end for
    3. Normalization:
    for $u = 1$ to U **do**

$$\hat{w}_k^u = \frac{w_k^u}{\sum_{v=1}^U w_k^v}$$

    end for
    4. Resampling:
    Resample U particles $D_k^u = \{x^u_{0:k}, y^u_{0:k}, \tau^u_{0:n_u}, m^u_{0:n_u}, n_u\}$ from $\{D_k^u\}_{u=1}^U$ with probabilities $\hat{w}_k^u$
    for $u = 1$ to U **do**
        Reset weight: $w_k^u = \frac{1}{U}$
    end for
end for

parameters, i.e., the shape and scale, and the mark distribution parameters, i.e., mean and covariance, are estimated via MAP using optimization routines.

Given the model formulation, we require running $2n_u$ (maximum number of inducing points generated by the SMC algorithm): one per waiting time and one per mark. Although both σ_y and σ_x can be learned, we typically fix σ_y to ensure meaningful propagation from Y to X. If σ_y is too large, changes in Z are mostly captured by shifting Y, which limits the propagation of observed data information to X and the inducing points. On the other hand, σ_x can be optimized, and a closed-form solution for its estimation can be derived. Similar to the waiting time and mark parameters, we can use optimization techniques for its estimation.

In Equation 19, we assume shared σ_y and σ_x across latent dimensions, in practice, these can vary per dimension.

A.7 Gamma Distribution for Waiting Times and Prior Selection for Inducing Points

To complete the Bayesian framework, we define priors for the model parameters. For the mark distribution parameters, we assume:

$$\mu_i \mid \Sigma_i \sim N(\mu_0, \lambda\Sigma_i), \quad \Sigma_i \sim \text{Inverse-Wishart}(\nu, \Psi) \tag{21}$$

where μ_0, λ, ν, and Ψ are hyperparameters. For the Gamma distribution parameters α and λ governing the waiting times τ_i, we consider the following prior options based on domain-specific knowledge, though their specific forms remain to be fully specified in this study:

- For α:
  - $\alpha \sim \text{Gamma}(a_0, b_0) = \frac{b_0^{a_0}}{\Gamma(a_0)}\alpha^{a_0-1}e^{-b_0\alpha}$,
  - $\alpha \sim \text{Exp}(\lambda_0) = \lambda_0 e^{-\lambda_0\alpha}$,
  - $\alpha \sim \text{Lognormal}(\mu_0, \sigma_0^2) = \frac{1}{\alpha\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{(\log\alpha-\mu_0)^2}{2\sigma_0^2}\right)$,
- For λ:
  - $\lambda \sim \text{Gamma}(c_0, d_0) = \frac{d_0^{c_0}}{\Gamma(c_0)}\lambda^{c_0-1}e^{-d_0\lambda}$,
  - $\lambda \sim \text{InvGamma}(\gamma_0, \delta_0) = \frac{\delta_0^{\gamma_0}}{\Gamma(\gamma_0)}\lambda^{-(\gamma_0+1)}e^{-\delta_0/\lambda}$,

where $a_0, b_0, \lambda_0, \mu_0, \sigma_0^2, c_0, d_0, \gamma_0, \delta_0$ are hyperparameters.

In our model, the waiting times τ_i and the marks $\vec{m}_i$ associated with each event are generated according to specific probabilistic distributions:

- **Waiting Times τ_i:**
  The waiting times between events are assumed to follow a Gamma distribution parameterized by a shape parameter α and a rate parameter λ. The probability density function for τ_i is given by:

  $$p(\tau_i) = \text{Gamma}(\tau_i; \alpha, \lambda) \tag{22}$$

  where the Gamma distribution is defined as:

  $$\text{Gamma}(\tau_i; \alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)}\tau_i^{\alpha-1}e^{-\lambda\tau_i}, \quad \tau_i > 0 \tag{23}$$

  and $\Gamma(\alpha)$ denotes the Gamma function evaluated at α.

  Motivation for using the Gamma distribution:
  Consider N i.i.d. samples $U_1, \ldots, U_N \sim \text{Uniform}(0, T)$, and denote their order statistics by $U_{(1)} \leq \cdots \leq U_{(N)}$. Define the gaps between consecutive order statistics as

  $$\Delta_0 = U_{(1)}, \quad \Delta_i = U_{(i+1)} - U_{(i)} \text{ for } i = 1, \ldots, N-1, \quad \Delta_N = T - U_{(N)} \tag{14}$$

  As $N \to \infty$, it is well-known that each gap satisfies $\Delta_i \xrightarrow{p} T/N$, and the rescaled gaps $N\Delta_i$ converge in distribution to an exponential random variable, that is,

  $$N\Delta_i \xrightarrow{d} \text{Exp}(1) \tag{15}$$

  Moreover, the normalized gaps $(\Delta_0/T, \ldots, \Delta_N/T)$ jointly follow a Dirichlet$(1, \ldots, 1)$ distribution. Marginally, each normalized gap Δ_i/T follows a Beta$(1, N)$ distribution. As N becomes large, the Beta$(1, N)$ distribution approximates a Gamma$(1, 1/N)$ distribution, because

  $$N \cdot (\Delta_i/T) \xrightarrow{d} \text{Exp}(1) \tag{16}$$

  which suggests that

  $$\Delta_i \approx \text{Gamma}(1, T/N) \tag{17}$$

  Thus, in the large-sample limit, the gaps between ordered uniform samples behave approximately like scaled exponential random variables.

To simulate ordered points efficiently for a finite number M of samples, we propose sampling M independent gaps

$$\Delta_i \sim \text{Gamma}(1, T/M) \tag{18}$$

and constructing ordered points via the cumulative sums

$$U_{(i)} = \sum_{j=0}^{i-1} \Delta_j, \quad i = 1, \ldots, M \tag{19}$$

This motivates our use of Gamma-distributed waiting times τ_i in the model, capturing the natural variability in the timing of events.

- **Marks $\vec{m}_i$:**
  The marks, representing additional information associated with each event, are modeled as drawn from a multivariate normal (Gaussian) distribution. Each mark vector $\vec{m}_i$ has an associated mean vector μ_i and covariance matrix Σ_i, with the distribution:

$$p(\vec{m}_i) = \mathcal{N}(\vec{m}_i; \mu_i, \Sigma_i) \tag{24}$$

  explicitly given by:

$$\mathcal{N}(\vec{m}_i; \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\vec{m}_i - \mu_i)^\top \Sigma_i^{-1}(\vec{m}_i - \mu_i)\right) \tag{25}$$

  where d is the dimensionality of the mark vector.

This modeling choice allows flexible and realistic characterization of the temporal dynamics τ_i and the event-related features $\vec{m}_i$ within the system under study.

A.8 Comparison of Latent Trajectories

In this section, we provide additional analyses of the latent trajectories inferred by the compared models. As discussed in the main text, the latent spaces of baseline continuous-time models (e.g., Linear SDE, GP-SDE, GP-SLDS) are identifiable only up to an arbitrary affine transformation. To enable meaningful comparisons, we apply a Procrustes-based alignment procedure between the inferred latent trajectories and the ground-truth latent dynamics.

Representative examples are shown in Figure 5, where we compare raw latent trajectories (top panels) with their aligned counterparts (bottom panels) across baseline models. This visualization highlights the necessity of alignment for baseline approaches, as their raw latents are not directly comparable to the true dynamics.

The generative structure of MIP-CSDE naturally constrains its latent space, yielding trajectories that are more directly interpretable without alignment. Nonetheless, for fairness, all quantitative performance metrics reported in the main text are computed in the observation space.

A.9 Runtime Scalability Experiments

This appendix examines the empirical runtime scaling of the SMC–EM procedure used in our experiments. We vary two primary factors that drive computational cost: the number of particles P in the SMC layer and the sequence length K (number of time bins). For each setting, we run the Lorenz benchmark ten times with independent random seeds and report wall-clock time averaged over runs. Following common practice for GPU timing, we insert explicit CUDA synchronizations around the timed region and use a high-resolution host timer; we discard a short warm-up to avoid one-time kernel compilation and cache effects. All experiments use the same model configuration and batch size as in the main results to isolate the effect of P and K.

Figure 6 summarizes the measurements. Panel (A) varies P at fixed K, plotting seconds per 1,000 time bins on a log scale. Panel (B) varies K at fixed $P = 20{,}000$, reporting seconds per 20,000 particles. In both regimes, ordinary least squares fits (orange) achieve $R^2 \geq 0.995$ against the measured times (blue), consistent with the expected $\mathcal{O}(P)$ and $\mathcal{O}(K)$ complexity under our implementation. Absolute times depend on hardware, kernel fusion, and memory bandwidth, but the trends align with the cost analysis in Section 2.2.2. We note that memory usage grows linearly in P and modestly in K due to buffering of particle states; for large P, gradient checkpointing and mixed precision can reduce footprint without materially affecting the observed scaling.

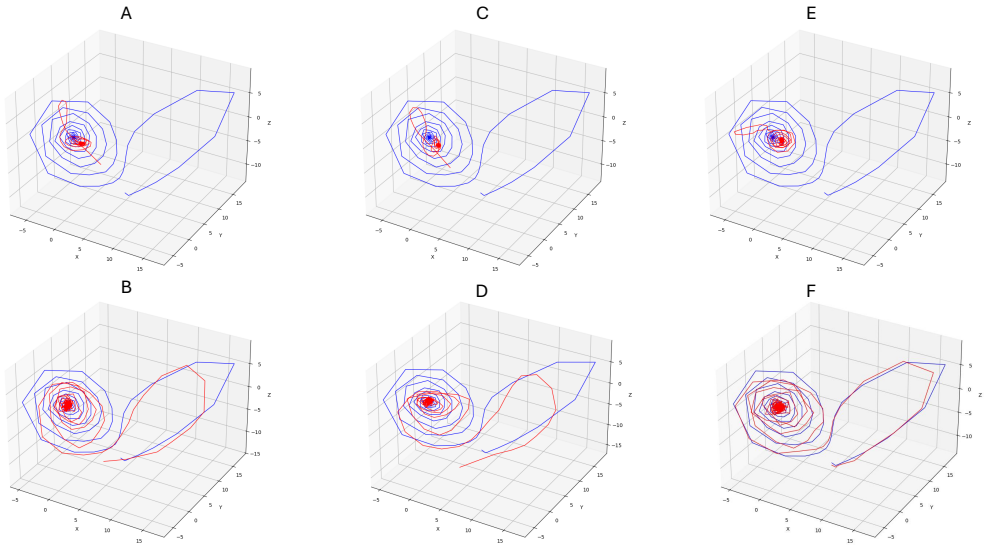

Figure 5: **Reconstruction of the Lorenz Trajectory Using Baseline Models.** Panels (A–B): Linear SDE; (C–D): GP-SDE; (E–F): GP-SLDS. For each model, the top panel shows the raw reconstructed latent trajectory, and the bottom panel shows the trajectory after Procrustes alignment to the ground truth.

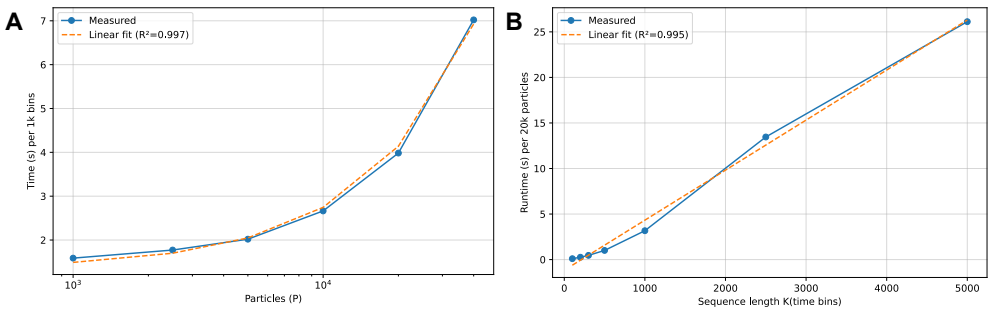

Figure 6: **Runtime of the SMC–EM Algorithm With Varying Numbers of Particles on the Lorenz Benchmark.** (A) Runtime vs. particle count P, shown on a log scale (seconds per 1k bins).(B) Runtime vs. sequence length K (seconds per 20k particles). Results are averaged over 10 runs. In both cases, measured runtime (blue) closely follow linear fits (orange, $R^2 \geq 0.995$), confirming the expected $O(P \cdot N)$ complexity discussed in Section 2.2.2. Experiments were run on an NVIDIA T4 GPU (16 GB) using PyTorch + CUDA.

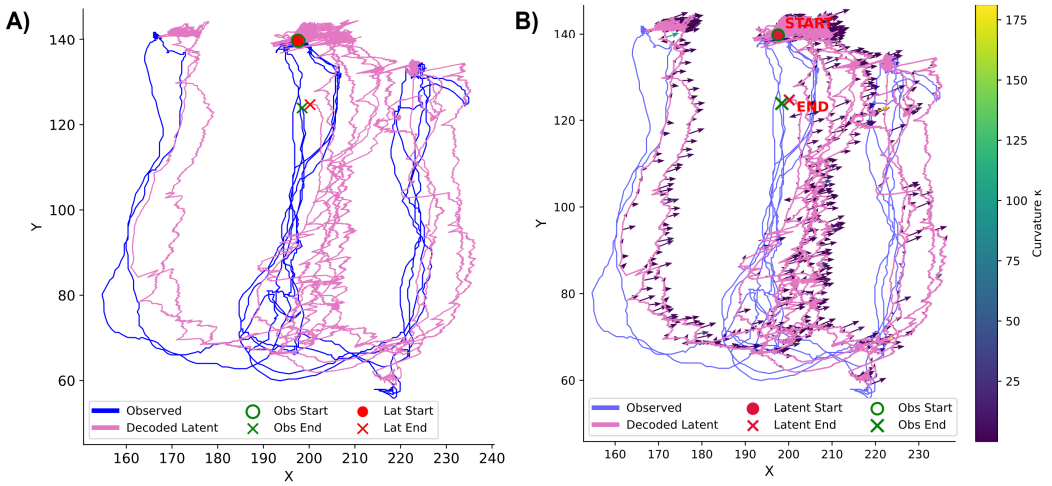

Figure 7: **Decoding of Rat Movement Trajectory Inside a W-Maze.** (A) Decoding results using MIP-CSDE closely follow the rat's movement. A small rightward shift appears due to the training session, during which the rat mostly moved toward the right side of the maze. The decoded trajectory remains inside the maze. (B) Timing of inducing points overlaid on the decoded trajectory. A higher number of inducing points occurs toward the ends of the arms, where the rat spends more time and movement patterns are less clear. Increased intensity of inducing points is also observed in the middle arm and corners of the maze, suggesting that reconstructing movement in these regions from observed spikes is more complex. Although not explicitly probed here, this may reflect aspects of the rat's decision-making and cognitive processing at these key locations

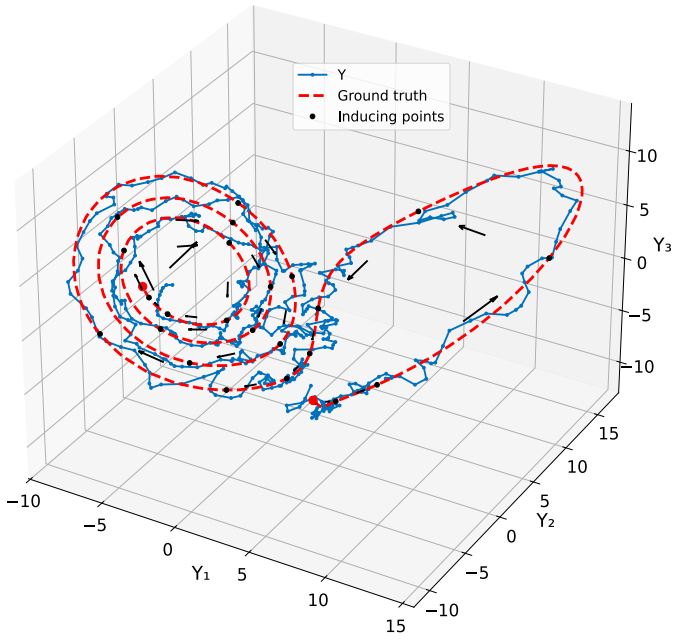

Figure 8: **Decoding of Lorenz Trajectory Using MIP-CSDE.** Here, the projection matrix from the latent process to observations and the additive noise covariance in the simulated data are known. The plot shows one inferred trajectory (Y), and dots indicate a sample set of inducing points. The decoded trajectory is aligned with the generated trajectory.

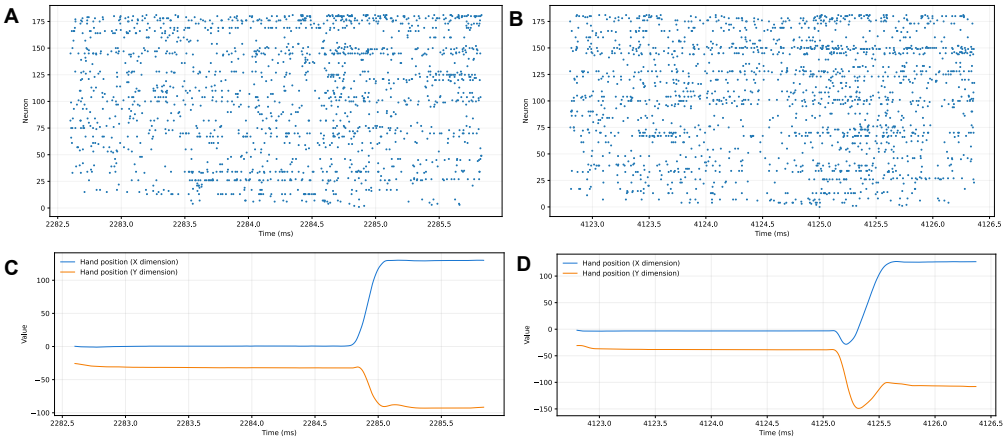

Figure 9: **Raster Plots of Monkey PDM and M1 Neurons Along with Hand Position During the Reach Task.** (A, B) Raster plots of 182 neurons from M1 and PMd, showing activity before target onset, the go cue, and target acquisition across two task trials. (C, D) Corresponding monkey hand positions during the same trials.