Supplementary Material: Inferring Latent Dynamics Underlying Neural Population Activity via Neural Differential Equations

Timothy Doyeon Kim [1] **Thomas Zhihao Luo** [1] **Jonathan W. Pillow** [1 2] **Carlos D. Brody** [1 3]

A. Details of Section 3

We develop Poisson Latent Neural Differential Equations (PLNDE), which infers the underlying nonlinear dynamics of neural population spike trains.

We maximize the marginal log-likelihood $\log p_\Theta(\boldsymbol{t}_x|\boldsymbol{u})$ by computing the variational lower bound, where $\boldsymbol{t}_x$ are the spike times of N observed neurons and $\boldsymbol{u}$ is the external stimulus given by the experimenter that affects $\boldsymbol{z}$:

$$\begin{aligned}
\log p_\Theta(\boldsymbol{t}_x|\boldsymbol{u}) &= \log \int p_\theta(\boldsymbol{t}_x|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})d\boldsymbol{z} \\
&= \log \int \frac{q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})}{q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})} p_\theta(\boldsymbol{t}_x|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})d\boldsymbol{z} \\
&\geq \mathbb{E}_q\left[\log \frac{p_\theta(\boldsymbol{t}_x|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{u})}{q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})}\right] \\
&= \mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{z},\boldsymbol{u})\right] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})||p(\boldsymbol{z}|\boldsymbol{u}))
\end{aligned}$$

Since the mapping from $\boldsymbol{z}(t)$ to $\boldsymbol{\lambda}(t)$ is defined by Equation (2) in the main text,

$$\begin{aligned}
\log p_\Theta(\boldsymbol{t}_x|\boldsymbol{u}) &\geq \mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{z},\boldsymbol{u})\right] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})||p(\boldsymbol{z}|\boldsymbol{u})) \\
&= \mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})\right] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x,\boldsymbol{u})||p(\boldsymbol{z}|\boldsymbol{u})) = \mathcal{L}
\end{aligned}$$

where

$$\mathbb{E}_q[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})] = \sum_{n=1}^{N} \mathbb{E}_q[\log p_\theta(\boldsymbol{t}_x^{(n)}|\lambda_n)].$$

Here, $\Theta = \{\theta, \phi\}$ are the parameters of our model, where θ are the generative parameters and ϕ are the variational parameters. The Poisson process likelihood $\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})$ (Palm, 1943; Duncker et al., 2019) is

$$\log p_\theta(\boldsymbol{t}_x^{(n)}|\lambda_n) = -\int_{\mathcal{T}} \lambda_n(t)dt + \sum_{i=1}^{\alpha(n)} \log \lambda_n(t_{x,i}^{(n)})$$

where $t_{x,\cdot}^{(n)}$ are the spike times and $\boldsymbol{\lambda} \in \mathbb{R}^N$ are the firing rates of the observed neurons $n = 1, 2, ..., N$. $\alpha(n)$ is the total number of spike counts of the n-th neuron.

[1]Princeton Neuroscience Institute, Princeton, New Jersey [2]Department of Psychology, Princeton University, Princeton, New Jersey [3]Howard Hughs Medical Institute, Princeton University, Princeton, New Jersey. Correspondence to: Timothy Doyeon Kim <tdkim@princeton.edu>.

A.1. $\mathbf{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})\right]$

While it is possible to compute the Poisson process likelihood $\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})$ via an ODE solver (Rubanova et al., 2019), we find that binning the spike times into k bins of width Δt and inferring the mean $\boldsymbol{\lambda}_k$ of the Poisson distribution works well in the synthetic datasets we test, and may be more efficient for our application as the number of neurons N can get large:

$$\begin{aligned}\boldsymbol{\lambda}_k &= \exp(\boldsymbol{C}\boldsymbol{z}_k + \boldsymbol{d})\\ x_{n,k} &\sim \text{Poisson}(\Delta t \lambda_{n,k})\end{aligned}$$

Then,

$$\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda}) \approx \sum_k \log p_\theta(\boldsymbol{x}_k|\boldsymbol{\lambda}_k) = \sum_k \sum_n \left[x_{n,k}\log(\Delta t \lambda_{n,k}) - \Delta t \lambda_{n,k}\right]$$

We estimate $\mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})\right]$ by

$$\begin{aligned}\mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})\right] &\approx \sum_k \mathbb{E}_q\left[\log p_\theta(\boldsymbol{x}_k|\boldsymbol{\lambda}_k)\right]\\ \mathbb{E}_q\left[\log p_\theta(\boldsymbol{x}_k|\boldsymbol{\lambda}_k)\right] &\approx \log p_\theta(\boldsymbol{x}_k|\hat{\boldsymbol{\lambda}}_k)\\ \hat{\boldsymbol{\lambda}}_k &= \exp\left(\boldsymbol{C}\hat{\boldsymbol{z}}_k + \boldsymbol{d}\right)\\ \hat{\boldsymbol{z}}_k &\sim q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{u})\end{aligned}$$

A.2. $\mathbf{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{u})||p(\boldsymbol{z}|\boldsymbol{u}))$

Suppose external stimulus $\boldsymbol{u}$ is discrete, where the discrete stimulus event times are $\boldsymbol{t}_u$. Then, the posterior process $q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{t}_u)$ can be described as:

$$\begin{aligned}\dot{\boldsymbol{z}} &= \boldsymbol{f}_\psi(\boldsymbol{z}, t) + \delta_{t_u}(\boldsymbol{g}^u_\phi(\boldsymbol{z}, t) + \boldsymbol{\eta}_u)\\ \boldsymbol{\eta}_u &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}^u_\phi(\boldsymbol{z}(t), t))\end{aligned}$$

where $\boldsymbol{f}_\psi$ is a generic function that may be approximated with a feedforward neural network (FNN). $\boldsymbol{f}_\psi$ characterizes the phase portrait, with its zeros representing the fixed points. $\boldsymbol{g}^u_\phi$ is the mean update in the latent state at each event time $t_{u,j}$, and the diagonal matrix $\boldsymbol{\Sigma}^u_\phi$ represents sensory noise in the update. $\boldsymbol{\eta}_u$ models noise in the sensory input perceived by the animal, similar to Brunton et al. 2013. Because noise $\boldsymbol{\eta}_u$ is added only at stimulus event times,

$$\begin{aligned}q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{t}_u) &= q_\phi(\boldsymbol{z}(t_0), \boldsymbol{z}(t_{u,1}), ..., \boldsymbol{z}(t_{u,M})|\boldsymbol{t}_x) = q_\phi(\boldsymbol{z}(t_0)|\boldsymbol{t}_x)\prod_{j=1}^{M} q_\phi(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1}), \boldsymbol{t}_x)\\ q_\phi(\boldsymbol{z}(t_0)|\boldsymbol{t}_x) &= \mathcal{N}(\boldsymbol{\mu}^0_\phi, \boldsymbol{\Sigma}^0_\phi)\\ q_\phi(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1}), \boldsymbol{t}_x) &= \mathcal{N}(\boldsymbol{\mu}(t_{u,j}), \boldsymbol{\Sigma}^u_\phi)\\ \boldsymbol{\mu}(t_{u,j}) &= \boldsymbol{g}^u_\phi(\boldsymbol{z}(t_{u,j}), t_{u,j}) + \boldsymbol{z}(t_{u,j-1}) + \int_{t_{u,j-1}}^{t_{u,j}} f_\psi(\boldsymbol{z}(t), t)dt\end{aligned}$$

where M is the total number of jump events, t_0 is the start of the trial, and $t_{u,j}$ is the timing of the j-th event. Similarly, the prior is given by

$$\begin{aligned}p(\boldsymbol{z}|\boldsymbol{t}_u) &= p(\boldsymbol{z}(t_0), \boldsymbol{z}(t_{u,1}), ..., \boldsymbol{z}(t_{u,M})) = p(\boldsymbol{z}(t_0))\prod_{j=1}^{M} p(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1}))\\ p(\boldsymbol{z}(t_0)) &= \mathcal{N}(\boldsymbol{\mu}^0_{\text{prior}}, \boldsymbol{\Sigma}^0_{\text{prior}})\\ p(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1})) &= \mathcal{N}(\boldsymbol{\mu}(t_{u,j}), \boldsymbol{\Sigma}^u_{\text{prior}})\\ \boldsymbol{\mu}(t_{u,j}) &= \boldsymbol{g}^u_{\text{prior}}(\boldsymbol{z}(t_{u,j}), t_{u,j}) + \boldsymbol{z}(t_{u,j-1}) + \int_{t_{u,j-1}}^{t_{u,j}} f_\psi(\boldsymbol{z}(t), t)dt\end{aligned}$$

The Kullback-Leibler (KL) divergence then becomes

$$\begin{aligned}
D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{t}_u)||p(\boldsymbol{z}|\boldsymbol{t}_u)) &= D_{KL}(q_\phi(\boldsymbol{z}(t_0)|\boldsymbol{t}_x)||p(\boldsymbol{z}(t_0))) + \sum_{j=1}^{M} D_{KL}(q_\phi(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1}), \boldsymbol{t}_x)||p(\boldsymbol{z}(t_{u,j})|\boldsymbol{z}(t_{u,j-1}))) \\
&\approx \frac{1}{2}\Big[\log\frac{|\boldsymbol{\Sigma}^0_{\text{prior}}|}{|\boldsymbol{\Sigma}^0_\phi|} - d + \text{tr}\{\boldsymbol{\Sigma}^{0^{-1}}_{\text{prior}}\boldsymbol{\Sigma}^0_\phi\} + (\boldsymbol{\mu}^0_\phi - \boldsymbol{\mu}^0_{\text{prior}})^\top \boldsymbol{\Sigma}^{0^{-1}}_{\text{prior}}(\boldsymbol{\mu}^0_\phi - \boldsymbol{\mu}^0_{\text{prior}})\Big] \\
&+ \sum_{j=1}^{M}\frac{1}{2}\Big[\log\frac{|\boldsymbol{\Sigma}^u_{\text{prior}}|}{|\boldsymbol{\Sigma}^u_\phi|} - d + \text{tr}\{\boldsymbol{\Sigma}^{u^{-1}}_{\text{prior}}\boldsymbol{\Sigma}^u_\phi\} \\
&+ (\boldsymbol{g}^u_\phi(\hat{\boldsymbol{z}}(t_{u,j}), t_{u,j}) - \boldsymbol{g}^u_{\text{prior}}(\hat{\boldsymbol{z}}(t_{u,j}), t_{u,j}))^\top \boldsymbol{\Sigma}^{u^{-1}}_{\text{prior}}(\boldsymbol{g}^u_\phi(\hat{\boldsymbol{z}}(t_{u,j}), t_{u,j}) - \boldsymbol{g}^u_{\text{prior}}(\hat{\boldsymbol{z}}(t_{u,j}), t_{u,j}))\Big]
\end{aligned}$$

where $\hat{\boldsymbol{z}}$ is a single trajectory sampled from the posterior process. If $\boldsymbol{g}^u_{\text{prior}}$ and $\boldsymbol{g}^u_\phi$ are not functions of $\boldsymbol{z}$, we have an equivalence as in the main text. Therefore, we have an estimate of $\mathcal{L} = \mathbb{E}_q\left[\log p_\theta(\boldsymbol{t}_x|\boldsymbol{\lambda})\right] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{t}_x, \boldsymbol{u})||p(\boldsymbol{z}|\boldsymbol{u}))$. We find parameters Θ that maximize $\mathcal{L}$ via a first-order method such as ADAM.

A.3. Computing the Adjoints with Stochastic Jumps

We compute the gradient $\partial\mathcal{L}/\partial\Theta$ by using the reparametrization trick (Kingma & Welling, 2014), storing $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ instantiated in the forward pass, and using the same noise instances during the backward pass. Similar approaches have also been used in fitting circuit models in neuroscience (Duan et al., 2021) and in calibrating market models in finance (Giles & Glasserman, 2006).

$$\boldsymbol{z}\left(t^+_{u,j}\right) = \boldsymbol{z}\left(t_{u,j}\right) + \boldsymbol{g}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right) + \epsilon \odot \sqrt{\boldsymbol{\Sigma}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}, \qquad \epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

Due to Jia & Benson 2019 and Corner et al. 2018, the adjoint is

$$\boldsymbol{a}\left(t_{u,j}\right) = \frac{\partial\mathcal{L}}{\partial\boldsymbol{z}\left(t_{u,j}\right)} = \frac{\partial\mathcal{L}}{\partial\boldsymbol{z}\left(t^+_{u,j}\right)}\frac{\partial\boldsymbol{z}\left(t^+_{u,j}\right)}{\partial\boldsymbol{z}\left(t_{u,j}\right)} = \boldsymbol{a}\left(t^+_{u,j}\right)\left(\frac{\partial\boldsymbol{z}\left(t^+_{u,j}\right)}{\partial\boldsymbol{z}\left(t_{u,j}\right)}\right)$$

Then,

$$\begin{aligned}
\boldsymbol{a}\left(t_{u,j}\right) &= \boldsymbol{a}\left(t^+_{u,j}\right)\left(\boldsymbol{I} + \frac{\partial\boldsymbol{g}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}{\partial\boldsymbol{z}\left(t_{u,j}\right)} + \epsilon\odot\frac{\partial\sqrt{\boldsymbol{\Sigma}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}}{\partial\boldsymbol{z}\left(t_{u,j}\right)}\right) \\
&= \boldsymbol{a}\left(t^+_{u,j}\right) + \boldsymbol{a}\left(t^+_{u,j}\right)\frac{\partial\left[\boldsymbol{g}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)\right]}{\partial\boldsymbol{z}\left(t_{u,j}\right)} + \epsilon\odot\boldsymbol{a}\left(t^+_{u,j}\right)\frac{\partial\left[\sqrt{\boldsymbol{\Sigma}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}\right]}{\partial\boldsymbol{z}\left(t_{u,j}\right)}.
\end{aligned}$$

To compute $\boldsymbol{a}_\Theta\left(t_{u,j}\right)$, we let

$$\boldsymbol{z}_{\text{aug}}\left(t\right) = \begin{bmatrix}\boldsymbol{z}\\ \Theta\\ t\end{bmatrix}(t), \qquad \frac{d\boldsymbol{z}_{\text{aug}}\left(t\right)}{dt} = \begin{bmatrix} f_\theta(\boldsymbol{z}, t) + \delta_{t_u}(\boldsymbol{g}^u_\phi(\boldsymbol{z}, t) + \boldsymbol{\eta}_u)\\ \boldsymbol{0}\\ 1\end{bmatrix}$$

$$\boldsymbol{a}_{\text{aug}}\left(t\right) = \begin{bmatrix}\boldsymbol{a} & \boldsymbol{a}_\Theta & \boldsymbol{a}_t\end{bmatrix}(t)$$

$$\boldsymbol{g}_{\text{aug}}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right) = \begin{bmatrix}\boldsymbol{g}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)\\ 0\\ 0\end{bmatrix}, \quad \sqrt{\boldsymbol{\Sigma}_{\text{aug}}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)} = \begin{bmatrix}\sqrt{\boldsymbol{\Sigma}^u_\phi\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}\\ 0\\ 0\end{bmatrix}, \quad \epsilon_{\text{aug}} = \begin{bmatrix}\epsilon\\ 0\\ 0\end{bmatrix}$$

$$
\begin{aligned}
\boldsymbol{z}_{\text{aug}}\left(t_{u,j}^{+}\right) &= \begin{bmatrix} \boldsymbol{z}\left(t_{u,j}\right) \\ \Theta \\ t_{u,j} \end{bmatrix} + \begin{bmatrix} \boldsymbol{g}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right) \\ 0 \\ 0 \end{bmatrix} + \epsilon_{\text{aug}} \odot \begin{bmatrix} \sqrt{\boldsymbol{\Sigma}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)} \\ 0 \\ 0 \end{bmatrix} \\
&= \boldsymbol{z}_{\text{aug}}\left(t_{u,j}\right) + \boldsymbol{g}_{\text{aug}}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right) + \epsilon_{\text{aug}} \odot \sqrt{\boldsymbol{\Sigma}_{\text{aug}}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{a}_{\text{aug}}\left(t_{u,j}\right) &= \boldsymbol{a}_{\text{aug}}\left(t_{u,j}^{+}\right)\left(\frac{\partial \boldsymbol{z}_{\text{aug}}\left(t_{u,j}^{+}\right)}{\partial \boldsymbol{z}_{\text{aug}}\left(t_{u,j}\right)}\right) \\
&= \begin{bmatrix} \boldsymbol{a} & \boldsymbol{a}_{\Theta} & \boldsymbol{a}_{t} \end{bmatrix}\left(t_{u,j}^{+}\right) \begin{bmatrix} \boldsymbol{I} + \frac{\partial \boldsymbol{g}_{\phi}^{u}}{\partial \boldsymbol{z}(t_{u,j})} + \epsilon \odot \frac{\partial \sqrt{\boldsymbol{\Sigma}_{\phi}^{u}}}{\partial \boldsymbol{z}(t_{u,j})} & \frac{\partial \boldsymbol{g}_{\phi}^{u}}{\partial \phi} + \epsilon \odot \frac{\partial \sqrt{\boldsymbol{\Sigma}_{\phi}^{u}}}{\partial \phi} & \frac{\partial \boldsymbol{g}_{\phi}^{u}}{\partial t_{u,j}} + \epsilon \odot \frac{\partial \sqrt{\boldsymbol{\Sigma}_{\phi}^{u}}}{\partial t_{u,j}} \\ 0 & \boldsymbol{I} & \boldsymbol{0} \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Therefore, the adjoints are

$$
\boldsymbol{a}\left(t_{u,j}\right) = \boldsymbol{a}\left(t_{u,j}^{+}\right) + \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\boldsymbol{g}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)\right]}{\partial \boldsymbol{z}\left(t_{u,j}\right)} + \epsilon \odot \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\sqrt{\boldsymbol{\Sigma}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}\right]}{\partial \boldsymbol{z}\left(t_{u,j}\right)}
$$

$$
\boldsymbol{a}_{\Theta}\left(t_{u,j}\right) = \boldsymbol{a}_{\Theta}\left(t_{u,j}^{+}\right) + \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\boldsymbol{g}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)\right]}{\partial \Theta} + \epsilon \odot \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\sqrt{\boldsymbol{\Sigma}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}\right]}{\partial \Theta}
$$

$$
\boldsymbol{a}_{t}\left(t_{u,j}\right) = \boldsymbol{a}_{t}\left(t_{u,j}^{+}\right) + \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\boldsymbol{g}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)\right]}{\partial t} + \epsilon \odot \boldsymbol{a}\left(t_{u,j}^{+}\right)\frac{\partial\left[\sqrt{\boldsymbol{\Sigma}_{\phi}^{u}\left(\boldsymbol{z}\left(t_{u,j}\right), t_{u,j}\right)}\right]}{\partial t}
$$

where ϵ is the same $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ instantiated during the forward pass. Computing the adjoints take $\mathcal{O}(M)$ memory and $\mathcal{O}(\tilde{M})$ time, similar to continuous adjoint sensitivity methods that utilize checkpointing (Serban & Hindmarsh, 2005; Rackauckas et al., 2020). M is the number of jump events and $\tilde{M}$ is the number of function evaluations.

B. Details of Section 4

B.1. Model Architectures

PLNDE is implemented with libraries in Julia's (Bezanson et al., 2017) SciML ecosystem—DifferentialEquations.jl and DiffEqFlux.jl (Rackauckas & Nie, 2017; Rackauckas et al., 2020). The FNN $\boldsymbol{f}_\psi$ of our model had the same architecture for all analyses done in this paper. $\boldsymbol{f}_\psi$ had three hidden layers with swish activations, with the first layer having 17 units, the second layer 23 units, and the third layer 17 units. The ODE solver used was Tsitouras 5/4 Runge-Kutta method, and automatically switched to a second order A-B-L-S-stable one-step ESDIRK method (TR-BDF2) whenever stiffness was detected (Rackauckas & Nie, 2017).

For Sections 4.1 and 4.2 in the main text, parameters $\Theta = \{\theta, \boldsymbol{\mu}_\phi^0, \boldsymbol{\Sigma}_\phi^0\}$ were jointly optimized with ADAM, where for the variational parameters ϕ, the constants $\boldsymbol{\mu}_\phi^0$ and $\boldsymbol{\Sigma}_\phi^0$ were fit separately for each trial, and $\boldsymbol{\Sigma}_\phi^0$ were constrained to be diagonal, with the diagonal elements less than 1. We let $\boldsymbol{\mu}_{\text{prior}}^0 = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_{\text{prior}}^0 = \boldsymbol{I}$. After training was finished, we evaluated on the test trials by freezing the generative parameters θ, and optimizing only the variational parameters ϕ on the test dataset via ADAM.

For Section 4.3 in the main text, we let $\boldsymbol{u}$ be either R for right side and L for left side, and set the constant vectors $\boldsymbol{g}_\phi^R = -\boldsymbol{g}_\phi^L$. While $\boldsymbol{\mu}_\phi^0$, $\boldsymbol{\Sigma}_\phi^0$, $\boldsymbol{g}_\phi^R$, $\boldsymbol{\Sigma}_\phi^R$,$\boldsymbol{g}_\phi^L$, $\boldsymbol{\Sigma}_\phi^L$ can be more general and be some functions of $\boldsymbol{z}$ or $\boldsymbol{x}$, we let them be constants for our experiments. We fit constants $\boldsymbol{g}_\phi^R$, $\boldsymbol{g}_\phi^L$, $\boldsymbol{\Sigma}_\phi^R$, $\boldsymbol{\Sigma}_\phi^L$ separately for each pulse. $\boldsymbol{\Sigma}_\phi^R$ and $\boldsymbol{\Sigma}_\phi^L$ were constrained to be $\boldsymbol{\Sigma}_\phi^R = \boldsymbol{\Sigma}_\phi^L$ and diagonal, with the diagonal elements less than 0.01 (chosen to be a reasonable value relative to the mean jumps in Equation (28) of the main text). $\boldsymbol{g}_\phi^R$ and $\boldsymbol{g}_\phi^L$ were constrained to be between -1.2 and 1.2 with the $\tanh$ function and $\boldsymbol{g}_\phi^R = -\boldsymbol{g}_\phi^L$. We let the prior reflect the true parameters used to generate mutual inhibition dynamics, i.e., $\boldsymbol{\mu}_{\text{prior}}^R = \boldsymbol{C}^R$, $\boldsymbol{\mu}_{\text{prior}}^L = \boldsymbol{C}^L$ and $\boldsymbol{\Sigma}_{\text{prior}}^R = \boldsymbol{\Sigma}_{\text{prior}}^L = \kappa \boldsymbol{I}$ where $\kappa = 0.001$. $\boldsymbol{\mu}_\phi^0$ and $\boldsymbol{\Sigma}_\phi^0$ were constrained and fit in the same way as Sections 4.1 and 4.2. Therefore, parameters $\Theta = \{\theta, \boldsymbol{\mu}_\phi^0, \boldsymbol{\Sigma}_\phi^0, \boldsymbol{g}_\phi^R, \boldsymbol{g}_\phi^L, \boldsymbol{\Sigma}_\phi^R, \boldsymbol{\Sigma}_\phi^L\}$ were optimized.

For Section 4.4 in the main text, we used the same configuration as in Section 4.3.

If the dynamics we want to infer are complex, PLNDE may fall into a local minimum. To avoid this, for Sections 4.1–4.3, we first trained PLNDE on the first few ms of each trial, and gradually increased the interval that we use to train PLNDE, until finally we trained on the entire duration of each trial. While not necessary, doing this generally increased performance of PLNDE on the synthetic datasets. This method of "iteratively growing the fit" is introduced in an example in the documentation of DiffEqFlux.jl (Rackauckas et al., 2020). For Section 4.4, we did not iteratively grow the fit, as the median duration of the trials was only 0.5s.

As noted in Section A, although it is possible to directly compute Equation (14) via an ODE solver for each neuron n, if the number of neurons N becomes large, computing (14) may become less tractable. Thus, we binned our spikes with the maximal number of bins possible for spike times rounded to the thousandths—1001 bins. We used the same number of bins for PLDS, and used 100 bins for LFADS. We used 100 bins as GRU may not retain memory for longer time bins and potentially suffer more from vanishing and exploding gradients (Bengio et al., 1994; Hochreiter & Schmidhuber, 1997).

For LFADS and PLDS, we used the default initializations, hyperparameters and termination criteria given in their original repositories.

For PLDS, EM iterations ended whenever the specified tolerance was reached or the maximum number of iterations (set to be 150) was reached. PLDS was initialized with nuclear-norm minimization (Pfau et al., 2013), except for the nonlinear spiral dynamics having low mean population firing rates, where we initialized with exponential family PCA. This was because nuclear-norm minimization ended with only a couple of iterations and did not produce a meaningful result. For testing PLDS, after training, we performed the E-step on the test dataset.

We initialized the parameters of PLNDE with Glorot normal and trained for ≈ 5000 iterations. When there are instabilities in the solver, this often happens at the beginning of the training routine, and may typically be solved by either reducing the learning rate or with a different initialization. Different initializations of the model parameters, whenever there were no instabilities in the solver, resulted in phase portraits and performances that are similar.

B.2. Spiral Dynamics

The set of equations:

$$\begin{aligned}\dot{z}_1 &= -4z_1^3 - 4z_1 - 80z_2^3 - 80z_2 \\ \dot{z}_2 &= 80z_1^3 + 80z_1 - 4z_2^3 - 4z_2 \\ \dot{z}_3 &= -12z_3^3 - 12z_3\end{aligned}$$

generates nonlinear spiral dynamics similar to the spiral example in (Brunton et al., 2016). We used spike trains from 2^3, 3^3, 4^3, 5^3, 6^3, 7^3, $2 \cdot 7^3$, and $3 \cdot 7^3$ trials for training. The initial values of each trial were chosen from $2 \times 2 \times 2$, $3 \times 3 \times 3$, $4 \times 4 \times 4$, $5 \times 5 \times 5$, $6 \times 6 \times 6$, and $7 \times 7 \times 7$ grids in $[-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$ when training with 2^3, 3^3, 4^3, 5^3, 6^3, 7^3 trials, respectively. When we train with $2 \cdot 7^3$ trials, we use the $7 \times 7 \times 7$ grid, where two trials have the same initial value, and when we train with $3 \cdot 7^3$ trials, three trials have the same initial value. Each trial was 1 second long. For this dataset, we tested two different cases, 1) the case where the mean firing rate across the neural population is high (6.62 spikes/s for 7^3 training trials, see Figure 2B in the main text, left column, example spike trains) and 2) low (1.12 spikes/s for 7^3 training trials, see Figure 2C in the main text, left column, example spike trains). The mapping from the latent trajectories to spike times followed Equations (2), where each element of $\boldsymbol{C}$ was chosen arbitrarily by drawing from $U[8, 9]$ and randomly multiplying by either 1 or -1, for Case 1). For Case 2), each element was drawn from $U[2, 3]$ and was randomly multiplied by either 1 or -1. $\boldsymbol{d} = \mathbf{0}$ for both cases. The spike times were rounded to the nearest thousandths. We evaluated the models on 7^3 test trials. Initial values of the test trials were not chosen from a grid. Each trial had a different initial value coming from a uniform distribution with range $[-0.25, 0.25] \times [-0.25, 0.25] \times [-0.25, 0.25]$. Therefore, the initial values in the test dataset were never seen in training.

We found that PLNDE can accurately infer the phase portrait of these spiral dynamics and reconstruct the test latent trajectories more accurately than LFADS and PLDS (see the main text for details). In these experiments, we assumed we know the latent dimensionality ($= 3$), not just for PLNDE but also for LFADS and PLDS. However, we often do not know the true latent dimensionality of the system we observe. In such a case, the optimal dimensionality may be recovered by comparing the test loss (ELBO) and the test log-likelihood (LL) of each neuron across models that assume different latent dimensions. Recovering the optimal dimensionality with this approach was possible for Case 1 (Figure 5).

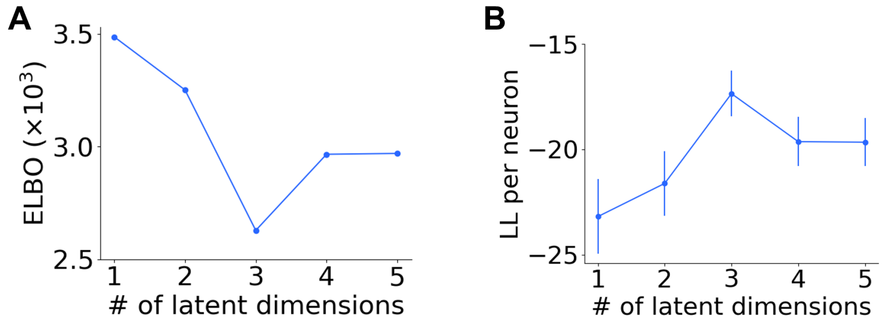

Figure 5. Recovering the dimensionality of the three-dimensional nonlinear spiral dynamical system with PLNDE. (A) Test evidence lower bounds (ELBO) of PLNDEs that assume different latent dimensions. (B) The average test log-likelihoods per neuron for PLNDEs that assume different latent dimensions. The error bars indicate ± 1 standard deviations. The average test log-likelihood for dimension 3 was significantly higher than that of either dimension 2 or 4 ($p < 0.0001$; two-sided Mann-Whitney U tests).

B.3. FitzHugh-Nagumo

Spike times of 50 neurons were generated from a system governed by the FitzHugh-Nagumo oscillator:

$$\dot{z}_1 = \rho\tau\left(z_1 - \frac{1}{3}z_1^3 - z_2\right) + I_{\text{input}}$$
$$\dot{z}_2 = \frac{\tau}{\rho}\left(z_1 + a - bz_2\right)$$

where $a = b = 0$, $\rho = 2$, $\tau = 15$ and $I_{\text{input}} = 0$. We used spike trains from different numbers of trials (100, 200, 300, 500, 1000, 2000) to train the models. The initial values of each trial were chosen from a 10×10 grid in $[-2, 2] \times [-2, 2]$. The mapping from the latent trajectories to spike times followed Equations (2), where each element of $\boldsymbol{C}$ was chosen arbitrarily by drawing from $U[1, 2]$ and randomly multiplying by either 1 or -1. $\boldsymbol{d} = \boldsymbol{0}$. The spike times were rounded to the nearest thousandths. When we train using 100 trials, each trial thus has a different initial value. When we train using 200 trials, two trials have the same initial value, and when we train using 2000 trials, twenty trials have the same initial value. Each trial was 1 second long. The mean firing rate across the neural population in the 100 training trials was 11.94 spikes/s. We evaluated the models on 100 test trials, with the initial value for each trial randomly generated from $[-3.5, 3.5] \times [-3.5, 3.5]$. Therefore, the initial values in the test dataset were never seen in training.

We found that PLNDE can accurately infer the phase portrait of FitzHugh-Nagumo, capturing the limit cycle and the unstable fixed point (Figure 6B–C). We trained PLNDE, PLDS and LFADS on different numbers of training trials and found that PLNDE significantly outperformed LFADS and PLDS in reconstructing test latent trajectories in all numbers of trials we considered ($p < 0.0001$; Mann-Whitney U tests; Figure 6F) except for when there were 2000 training trials ($p = 0.0364$ between LFADS and PLNDE; Mann-Whitney U test; the rightmost circles in Figure 6F). We also found that PLNDE significantly outperformed LFADS and PLDS in reconstructing test firing rates in all numbers of trials we considered ($p < 0.0001$; Mann-Whitney U tests; Figure 6G).

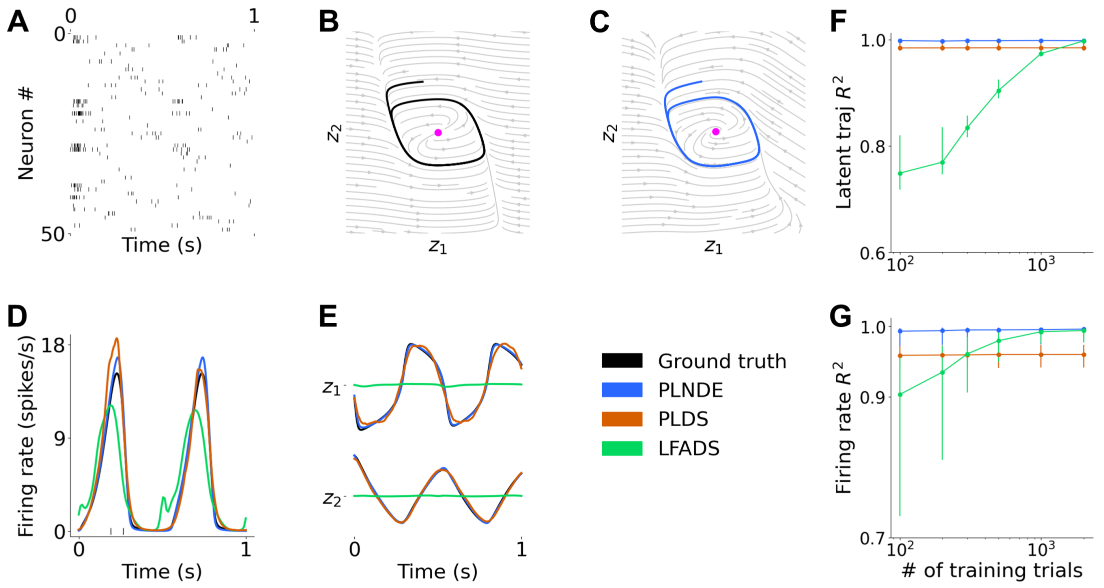

Figure 6. PLNDE accurately infers phase portrait and fixed point. PLNDE outperforms LFADS and PLDS in inferring single-trial latent trajectories and individual neural firing rates for FitzHugh-Nagumo. (A) Spike trains of the 50 neurons in this dataset in an example test trial. (B) True phase portrait. Black line indicates the example trajectory that generated the spike times in A. Magenta circle indicates the true unstable fixed point. (C) Inferred phase portrait affine-transformed to match the true phase portrait. Blue line indicates the inferred trajectory. Magenta circle indicates the inferred unstable fixed point. (D) An example test firing rate of an example neuron. Model fits are from training with 100 trials. (E) The example trajectory in B–C unrolled in time. Model fits are from training with 100 trials. (F) Each circle indicates the median R^2 of the true and inferred latent trajectories. The error bars indicate the first and third quartiles. (G) Same as F, but for firing rates.

We also considered the case when $a = b = 0$, $\rho = 2$, $\tau = 15$, but $I_{\text{input}} \neq 0$. We used spike trains of 50 neurons from 100 trials to fit PLNDE. The initial values of each trial were chosen from a 10×10 grid in $[-2, 2] \times [-2, 2]$. The mapping from the latent trajectories to spike times followed Equations (2), where each element of $\boldsymbol{C}$ was chosen arbitrarily by drawing from $U[1, 2]$ and randomly multiplying by either 1 or -1. $\boldsymbol{d} = \boldsymbol{0}$. The spike times were rounded to the nearest thousandths. Each trial was 1 second long. There were 10 different conditions where the currents input to the system were different for each condition. Each condition had 10 trials. The input currents were damped oscillations that followed the equations below.

Condition 1: $I_{\text{input}}(t) = 40 \sin(8\pi t) \exp(-t)$

Condition 2: $I_{\text{input}}(t) = 20 \sin(4\pi t) \exp(-t)$

Condition 3: $I_{\text{input}}(t) = 10 \sin(5\pi t) \exp(-t)$

Condition 4: $I_{\text{input}}(t) = 50 \cos(\pi t) \exp(-t)$

Condition 5: $I_{\text{input}}(t) = 0.1 \cos(\pi t) \exp(-t)$

Condition 6: $I_{\text{input}}(t) = 40 \cos(8\pi t) \exp(-t)$

Condition 7: $I_{\text{input}}(t) = 20 \cos(4\pi t) \exp(-t)$

Condition 8: $I_{\text{input}}(t) = 10 \cos(5\pi t) \exp(-t)$

Condition 9: $I_{\text{input}}(t) = 10 \sin(\pi t) \exp(-t)$

Condition 10: $I_{\text{input}}(t) = 0$ (No current)

We evaluated the models on 100 test trials with each trial having different initial values. The initial values were randomly generated from $[-3.5, 3.5] \times [-3.5, 3.5]$. We used one of the input currents in the 10 conditions used in training for each test trial's input current, so that for the 100 test trials, there are 10 different conditions with each condition having 10 test trials.

We found that PLNDE can infer the phase portrait of FitzHugh-Nagumo even when there are continuous input currents to the system, capturing the limit cycle and the unstable fixed point (Figure 7A). Also, the reconstructed test latent trajectories and test firing rates matched the ground truth well (Figure 7B–C) when the latents were perturbed with different types of damped oscillation inputs (Figure 7D). When we quantified the reconstruction accuracy with R^2 for each test trial, the median was 0.922 (Figure 7E).

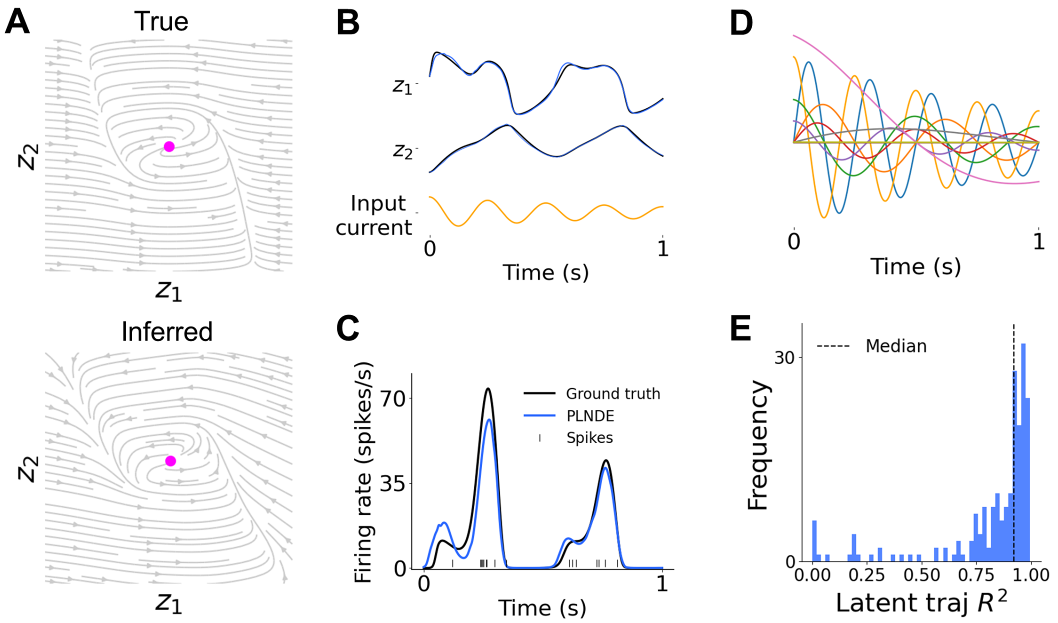

Figure 7. PLNDE can infer phase portrait and fixed point in the presence of continuous input streams in FitzHugh-Nagumo. (A) Same as Figure 2A, but for FitzHugh-Nagumo when $I_{\text{input}} = 10$. Magenta circles indicate the true and inferred unstable fixed points. (B–C) Black lines show a true example latent trajectory and firing rate while blue lines show the posterior means. (D) Currents input to the system colored differently for each condition. Bright orange is the input current used in B. (E) Histogram of R^2 between the true test trajectories and the posterior means (median=0.922).

B.4. Mutual Inhibition with External Input Stimuli

Spike trains from 150 neurons were generated from a system that has the following attractor dynamics:

$$
\begin{aligned}
\dot{z}_1 &= \tau \left(-z_1 + \frac{1}{1 + \exp\left(k\left(z_2 - \gamma\right)\right)} \right) \\
&\quad + \left(\delta_{t,t_R} \left(C_1^R + \eta_{R,1} \right) + \delta_{t,t_L} \left(C_1^L + \eta_{L,1} \right) \right) \\
\dot{z}_2 &= \tau \left(-z_2 + \frac{1}{1 + \exp\left(k\left(z_1 - \gamma\right)\right)} \right) \\
&\quad + \left(\delta_{t,t_R} \left(C_2^R + \eta_{R,2} \right) + \delta_{t,t_L} \left(C_2^L + \eta_{L,2} \right) \right)
\end{aligned}
$$

where $\tau = 10$, $k = 16$, and $\gamma = 0.5$. A similar model was studied in Wong & Wang 2006 and Piet et al. 2017 to describe the dynamics of two mutually inhibiting neural populations accumulating evidence for decision-making. For these 150 neurons, we assumed that the animal is doing the auditory decision-making task in Figure 1B of the main text, where there are two input trains of pulses of 30 Hz, one for the left and one for the right side of the chamber. A pulse on the right side induces a mean jump of $\boldsymbol{C}^R$, and a pulse on the left side induces a mean jump of $\boldsymbol{C}^L$. We let

$$
\boldsymbol{C}^R = \begin{bmatrix} 0.05 \\ -0.05 \end{bmatrix}, \quad \boldsymbol{C}^L = \begin{bmatrix} -0.05 \\ 0.05 \end{bmatrix}.
$$

We used 100, 200, 300, 500, 1000, 2000 trials to train PLNDE, PLDS, and LFADS. The initial values of each trial were chosen from a 10×10 grid in $[-1, 2] \times [-1, 2]$. The mapping from the latent trajectories to spike times followed Equations (2), where each element of $\boldsymbol{C}$ was chosen arbitrarily by drawing from $U[3, 4]$ and randomly multiplying by either 1 or -1. $\boldsymbol{d} = \boldsymbol{0}$. The spike times were rounded to the nearest thousandths. When we train using 100 trials, each trial thus has a different initial value. When we train using 200 trials, two trials have the same initial value, and when we train using 2000 trials, twenty trials have the same initial value. Each trial was 1 second long. The mean firing rate across the neural population in the 100 training trials was 21.50 spikes/s. For this dataset, we tested two different cases, 1) the case where $\eta_L = \eta_R = \boldsymbol{0}$ and 2) the case where $\eta_L, \eta_R \sim \mathcal{N}(\boldsymbol{0}, \kappa \boldsymbol{I})$ with $\kappa = 0.001$. For Case 1), we evaluated the models on 100 test trials, with each trial having a different initial value coming from the 10×10 grid in $[-1, 2] \times [-1, 2]$ (Figure 8). The left and right click times used in testing were never seen at training. We found that PLNDE trained on 100 trials significantly outperformed PLDS and LFADS trained on 1000 trials or less in reconstructing both the latent trajectories and the firing rates ($p < 0.0001$; two-sided Mann-Whitney U tests; Figure 8F). LFADS trained on 2000 trials outperformed PLDS in reconstructing the latent trajectories and the firing rates ($p < 0.0001$; two-sided Mann-Whitney U tests; Figure 8F), but still did not perform better than PLNDE trained on 100 training trials in reconstructing the latent trajectories and the firing rates ($p < 0.0001$; two-sided Mann-Whitney U tests; Figure 8F). Figure 3 in the main text shows the results of Case 2).

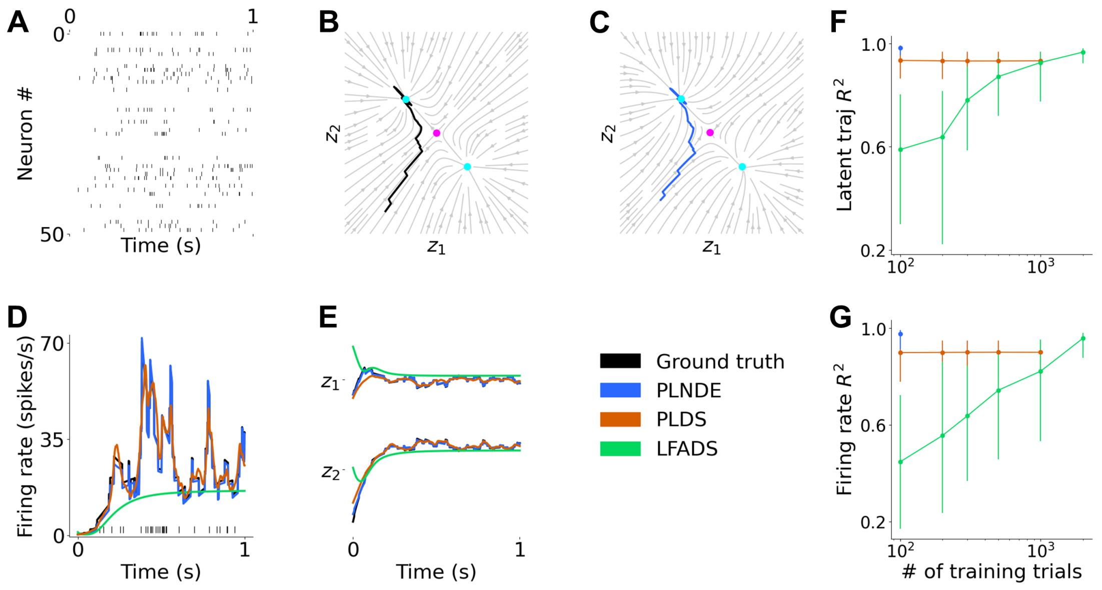

Figure 8. PLNDE accurately infers phase portrait and fixed points. PLNDE outperforms LFADS and PLDS in inferring single-trial latent trajectories and individual neural firing rates for the mutual inhibition dynamics. (A) An example test trial with spike trains of the first 50 neurons. (B) True phase portrait. Black line indicates the trajectory that generated the spike times in the example trial in A. The cyan circles indicate the true stable fixed points, while the magenta circle indicates the true unstable fixed point. (C) Inferred phase portrait affine-transformed to match the true phase portrait. Blue line indicates the inferred trajectory. The cyan circles indicate the inferred stable fixed points, while the magenta circle indicates the inferred unstable fixed point. (D) An example test firing rate of an example neuron. Model fits are from training with 100 trials. (E) The example trajectory in B–C unrolled in time. Model fits are from training with 100 trials. (F) Each circle indicates the median R^2 between the true and inferred latent trajectories. The error bars indicate the first and third quartiles. (G) Same as F, but for neural firing rates.

C. Number of Parameters

The table below shows the number of model parameters for each dataset (with 343 training trials for the spiral and with 100 training trials for FitzHugh-Nagumo and the mutual inhibition dynamics).

	PLNDE	LFADS	PLDS
SPIRAL	3,086	382,288	630
FITZHUGH-NAGUMO	1,459	304,838	164
MUTUAL INHIBITION	1,761	383,138	468

D. Computing Infrastructure and Runtime

We trained LFADS on Princeton's Tiger cluster. It is a Dell computer cluster with 320 NVIDIA P100 GPUs across 80 Broadwell nodes. LFADS took up one of these nodes, using one GPU that has 16GB of memory. LFADS took no more than 1 day to train on 1700–2000 trials, and usually took 3–5 hours to fit an average session ($\approx$ 300 trials).

We trained PLDS and PLNDE on the Spock cluster in the Princeton Neuroscience Institute. A node in this cluster contains two quad-core Xeon X5570 processors (Intel), operating at 2.7GHz, and 24GB of RAM. PLDS and PLNDE took no more than 4 days to train on 1700–2000 trials, and usually took less than 1 day to fit an average session ($\approx$ 300 trials). We did not utilize GPUs and parallelize training routines for PLDS and PLNDE, but we expect the runtimes will become shorter with parallelization.

References

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

Brunton, B. W., Botvinick, M. M., and Brody, C. D. Rats and humans can optimally accumulate evidence for decision-making. *Science*, 340(6128):95–98, 2013.

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

Corner, S., Sandu, C., and Sandu, A. Adjoint sensitivity analysis of hybrid multibody dynamical systems, 2018.

Duan, C. A., Pagan, M., Piet, A. T., Kopec, C. D., Akrami, A., Riordan, A. J., Erlich, J. C., and Brody, C. D. Collicular circuits for flexible sensorimotor routing. *Nature Neuroscience*, 2021.

Duncker, L., Bohner, G., Boussard, J., and Sahani, M. Learning interpretable continuous-time models of latent stochastic dynamical systems. *Proceedings of the 36th International Conference on Machine Learning*, 97:1726–1734, 2019.

Giles, M. and Glasserman, P. Smoking adjoints: Fast monte carlo greeks. *Risk*, 19(1):88–92, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Jia, J. and Benson, A. R. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2014.

Palm, C. Intensitätsschwankungen im fernsprechverker. *Ericsson Technics*, 1943.

Pfau, D., Pnevmatikakis, E. A., and Paninski, L. Robust learning of low-dimensional dynamics from large neural ensembles. *Advances in Neural Information Processing Systems*, 26:2391–2399, 2013.

Piet, A. T., Erlich, J. C., Kopec, C. D., and Brody, C. D. Rat prefrontal cortex inactivations during decision making are explained by bistable attractor dynamics. *Neural Computation*, 29:2861–2886, 2017.

Rackauckas, C. and Nie, Q. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., and Ramadhan, A. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. K. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. *Advances in Neural Information Processing Systems*, 32:5320–5330, 2019.

Serban, R. and Hindmarsh, A. C. CVODES: The sensitivity-enabled ode solver in sundials. In *Proceedings of the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 6: 5th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A, B, and C, pp. 257–269, 2005.

Wong, K. F. and Wang, X. J. A recurrent network mechanism of time integration in perceptual decisions. *Journal of Neuroscience*, 26(4):1314–1328, 2006.