Modeling Latent Neural Dynamics with Gaussian Process Switching Linear Dynamical Systems

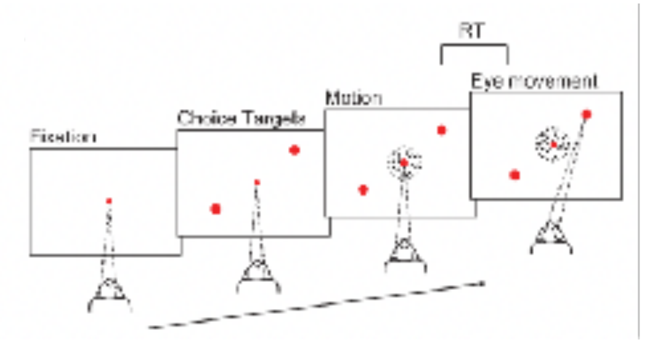
Amber Hu and Scott Linderman

Outline

- Scientific motivation
- Review of existing methods
- New modeling idea
- New inference algorithm
- Results & future work

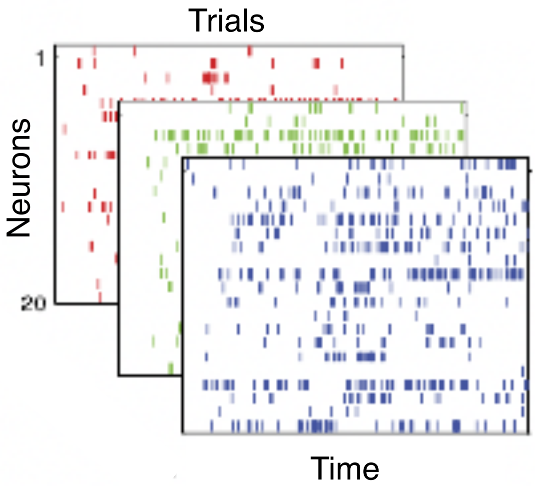
Analyzing neural data via latent dynamics

Experimental Data



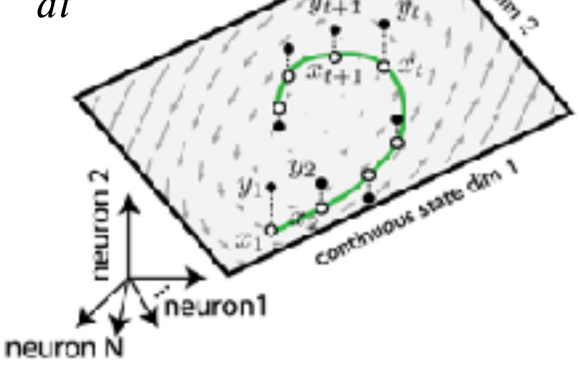
Experimental decision-making task

High-dimensional neural population activity (e.g. spike train recordings)



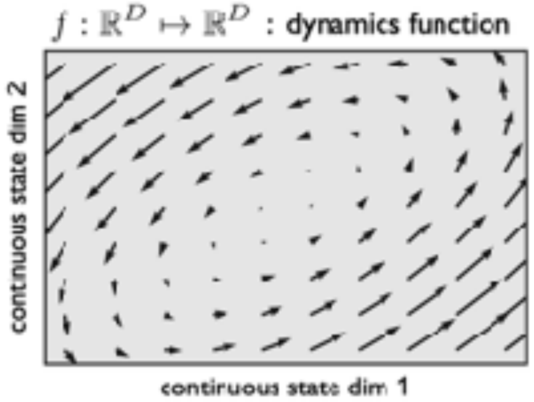
Dynamical Systems Analysis

$$\frac{dx}{dt} = f(x)$$



Noisy neural data are often adequately described by low-d latent variables

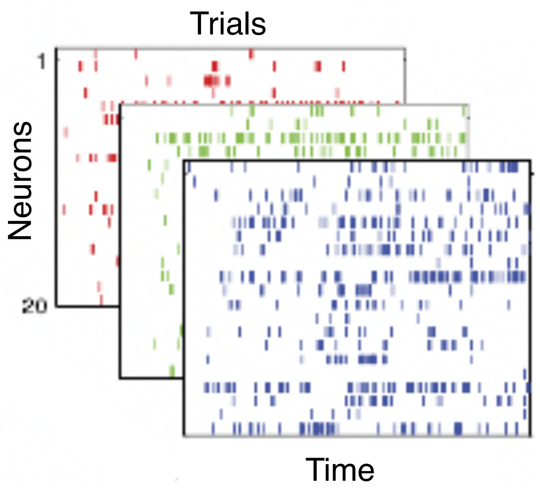
Rigorous analysis of neural activity via dynamical systems theory



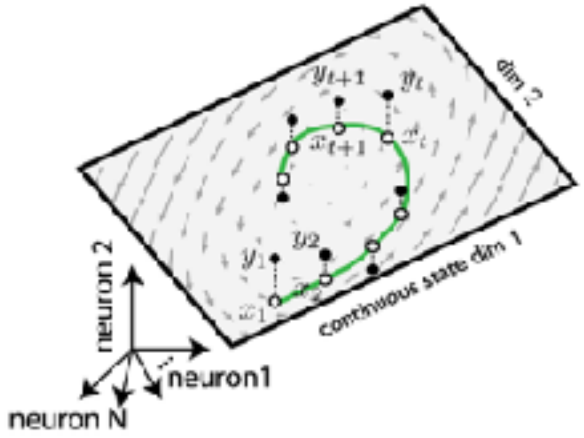
Cunningham and Yu (2014), Vyas et al. (2020), Stine et al. (2023)

Analyzing neural data via latent dynamics

Key question: How can we infer interpretable descriptions of the computations implemented by neural population activity?



?



We would like statistical methods which:

- Perform dimensionality reduction while modeling temporal structure
- Can incorporate prior knowledge about the data-generating process
- Produce latent representations which are interpretable for downstream analysis

Outline

- Scientific motivation

  Inferring interpretable descriptions of latent neural dynamics
- Review of existing methods
- New modeling idea
- New inference algorithm
- Results
- Future work

Methods for inferring latent dynamics

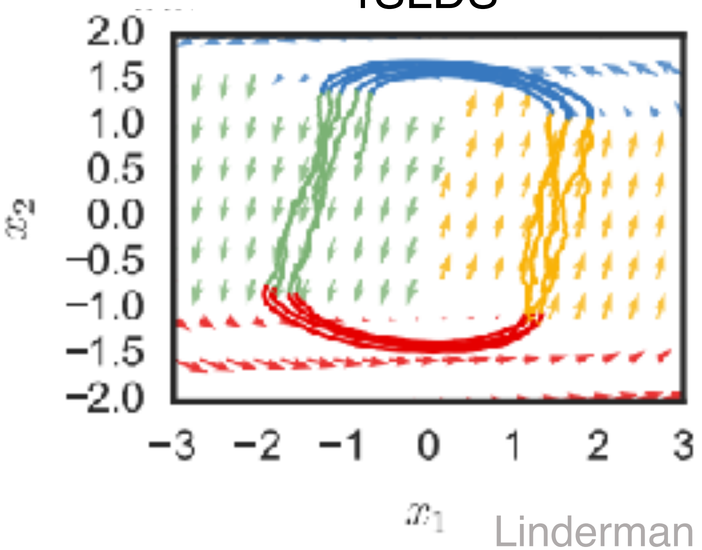
Probabilistic state-space models:

- Linear dynamical systems (LDS)
- Switching linear dynamical systems (SLDS)
- Nonlinear, non-Gaussian SSMs

Deep learning methods:

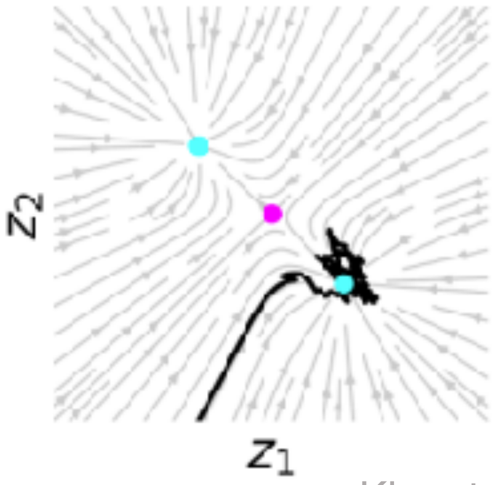
- Sequential VAEs (e.g. LFADS)
- Neural ODEs
- Deep state-space layers

rSLDS



Linderman et al. (2016)

Neural ODE

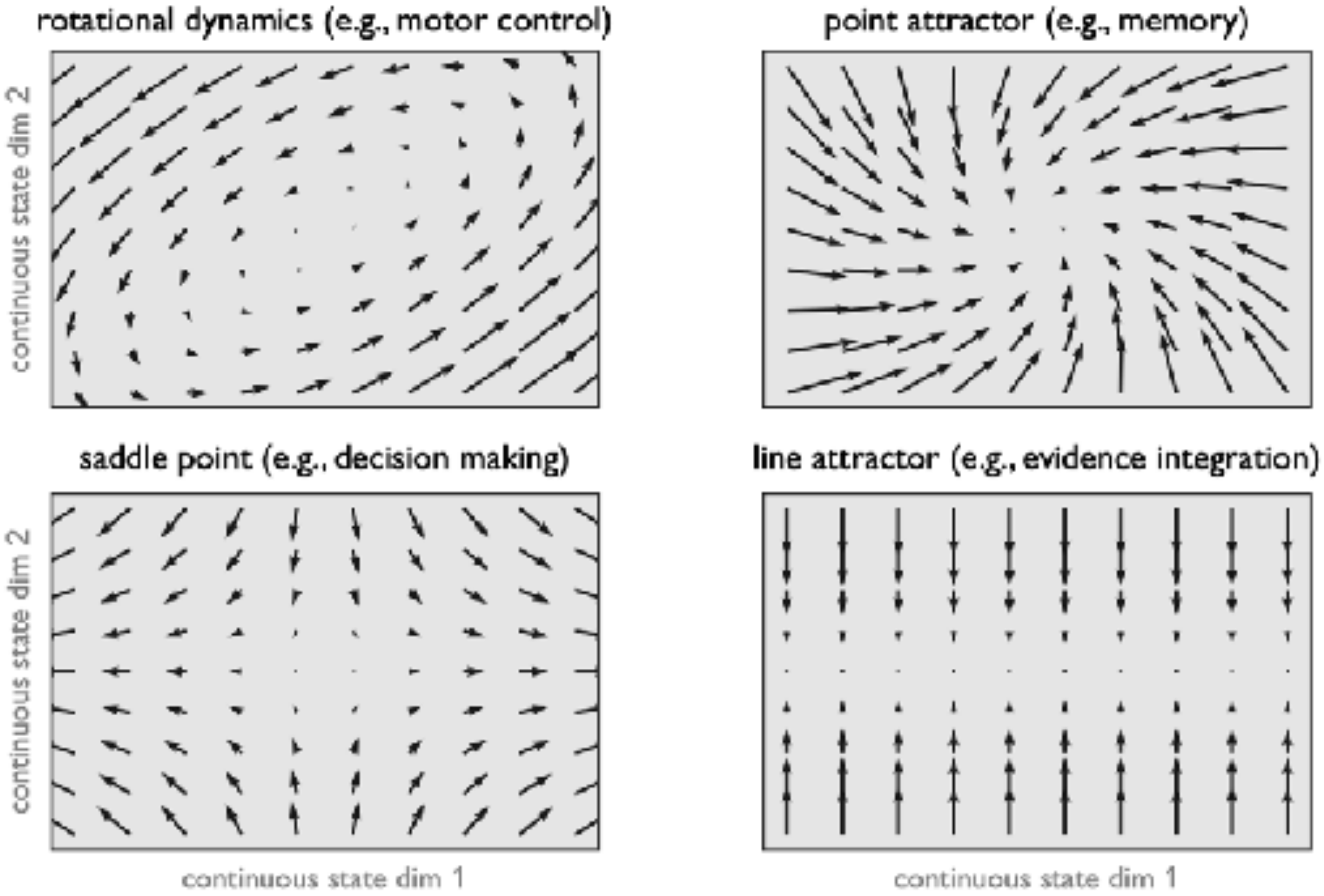


Kim et al. (2021)

Why linear dynamics?

- Linear dynamics express dynamical motifs which are hypothesized to underlie various kinds of neural computations

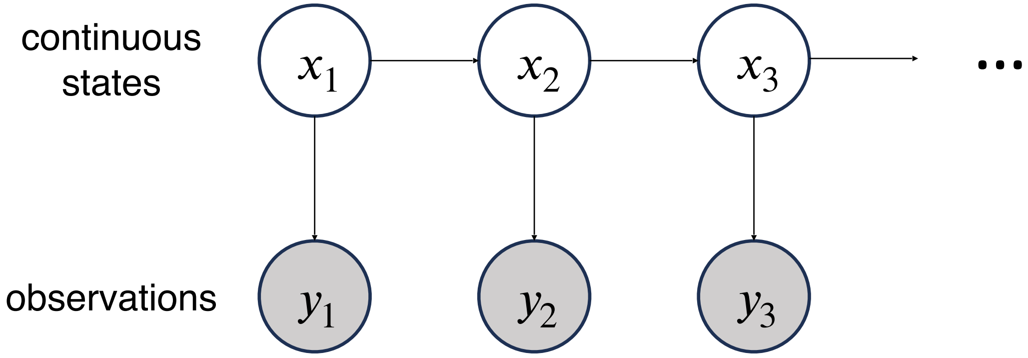
$$\frac{dx}{dt} = Ax + b$$



Vyas et al. (2020)

(Switching) linear dynamical systems

Linear dynamical system

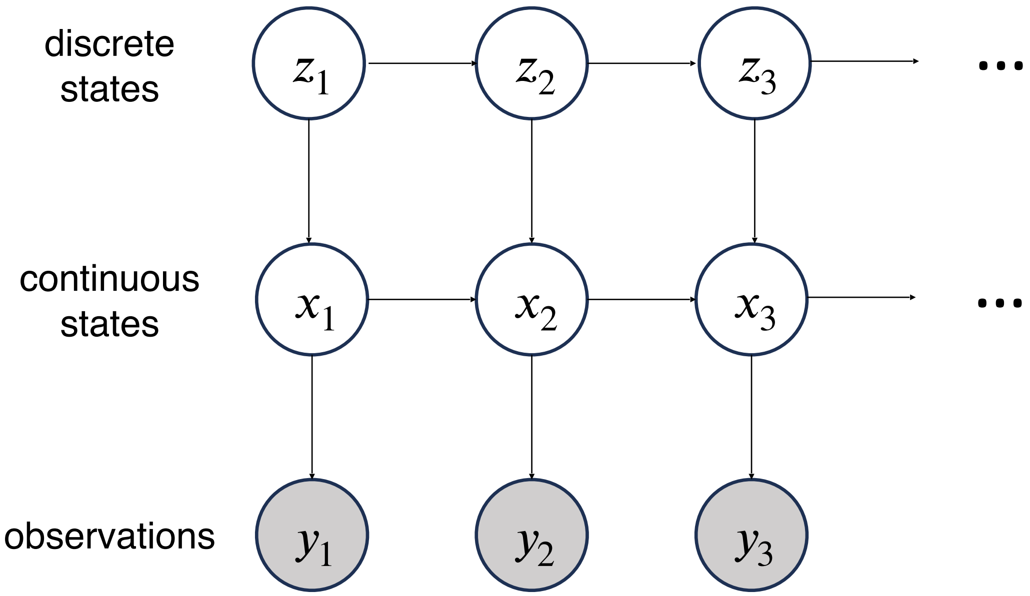


$$x_1 \sim \mathcal{N}\left(\mu_1,\ Q_1\right)$$

$$x_k \sim \mathcal{N}\left(Ax_{k-1} + b,\ Q\right),\quad k = 2\ \ldots,\ T$$

$$y_k \sim \mathcal{N}\left(Cx_k + d,\ R\right),\quad k = 1,\ \ldots,\ T$$

Switching linear dynamical system



$$z_1 \sim Cat\left(p_1\right),\quad z_k \sim Cat\left(\pi_{z_{k-1}}\right),\quad k = 2,\ \ldots,\ T$$

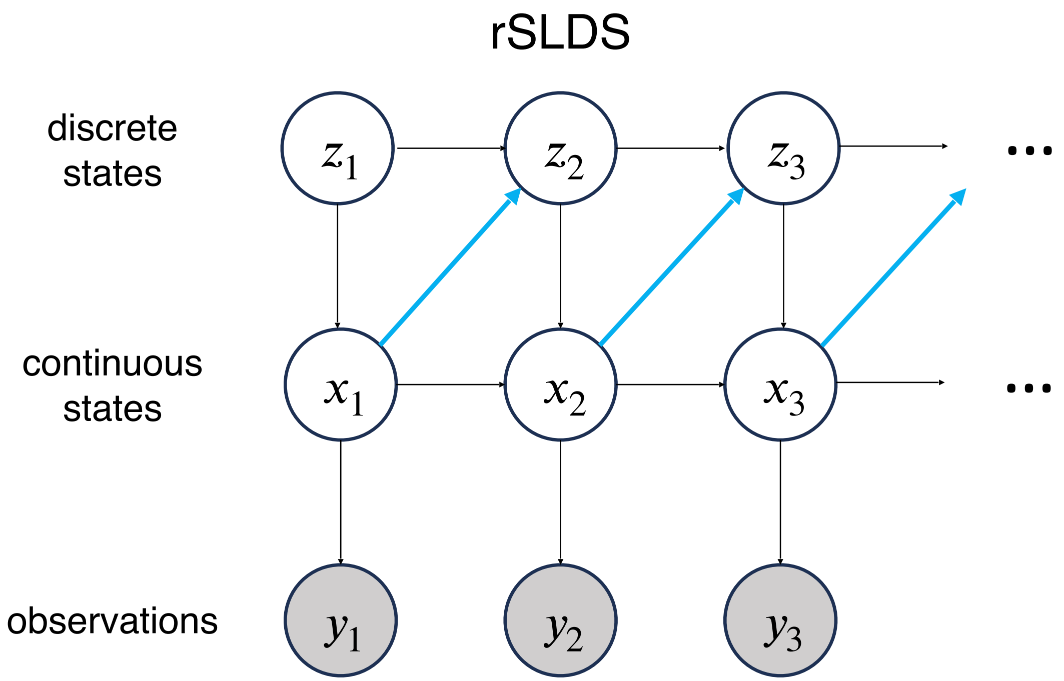
$$x_1 \sim \mathcal{N}\left(\mu_1, Q_1\right)$$

$$x_k \sim \mathcal{N}\left(A_{z_k} x_{k-1} + b_{z_k},\ Q_{z_k}\right),\quad k = 2\ \ldots,\ T$$

$$y_k \sim \mathcal{N}\left(Cx_k + d,\ R\right),\quad k = 1,\ \ldots,\ T$$

Recurrent SLDS

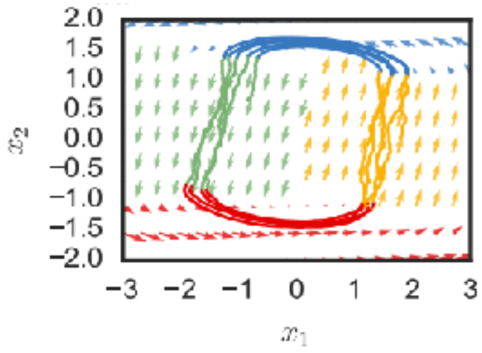
- Key idea: The linear system you are in should depend on your current location in continuous state space



Transition to next discrete state is modeled as a multiclass logistic regression:

$$p\left(z_k \,\middle|\, z_{k-1},\ x_{k-1}\right) \propto \exp(w^T x_{k-1} + r_{z_{k-1}})$$

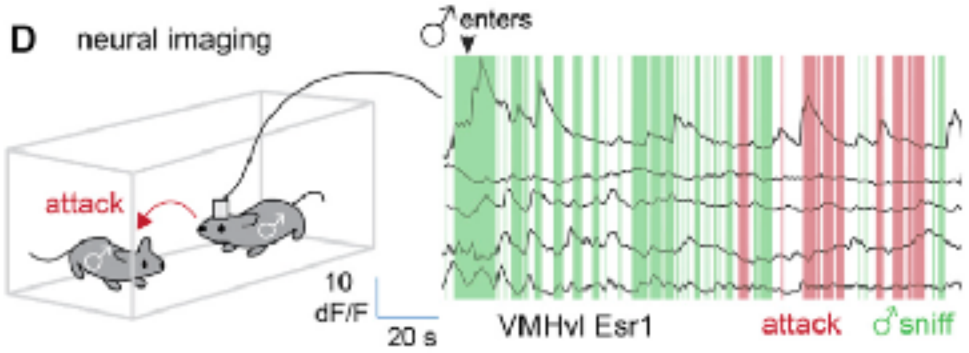
This leads to linear decision boundaries between discrete states.

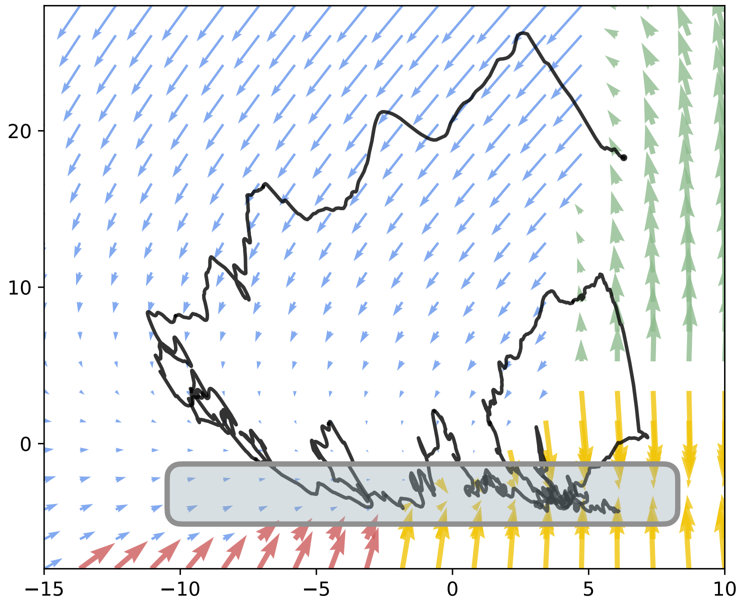


Linderman et al. (2016), Zoltowski et al. (2020)

Example: Dynamics of aggression

Experimental setup



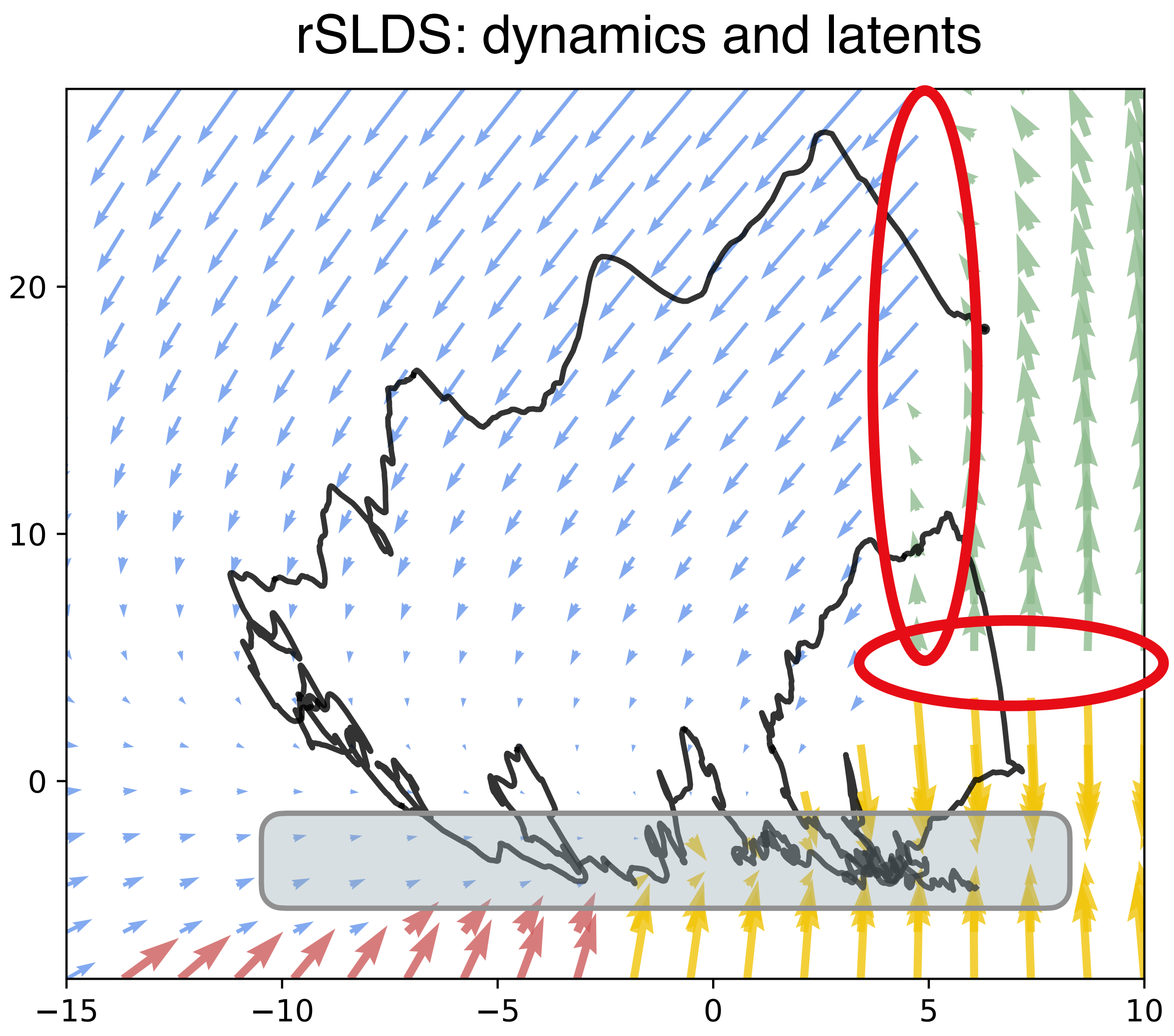


- In neuroscience, there is considerable interest in understanding latent dynamics underlying animal behavior
- Dataset: Calcium imaging of ventromedial hypothalamus neurons during aggressive behavior in mice

- Nair et al. fit a rSLDS with 4 linear regimes to the data
- rSLDS learns dynamics that form an "approximate line attractor" corresponding to an aggressive behavioral state
- Progression along the line attractor was correlated with the escalation of aggressive behavior, suggesting that it may encode an aggressive internal state

Nair et al. (2023)

A closer look at rSLDS



A few limitations of rSLDS:

- Often produces dynamics which transition sharply at regime boundaries
- Models dynamics as a stochastic mixture of linear systems at every time step
- Does not explicitly provide estimates of posterior uncertainty over inferred dynamics
    - How confident is the model in finding an approximate line attractor?

Outline

- Scientific motivation

  Inferring interpretable descriptions of latent neural dynamics

- Review of existing methods

  SLDS models decompose nonlinear dynamics into simpler components, but with some shortcomings

- New modeling idea
- New inference algorithm
- Results
- Future work

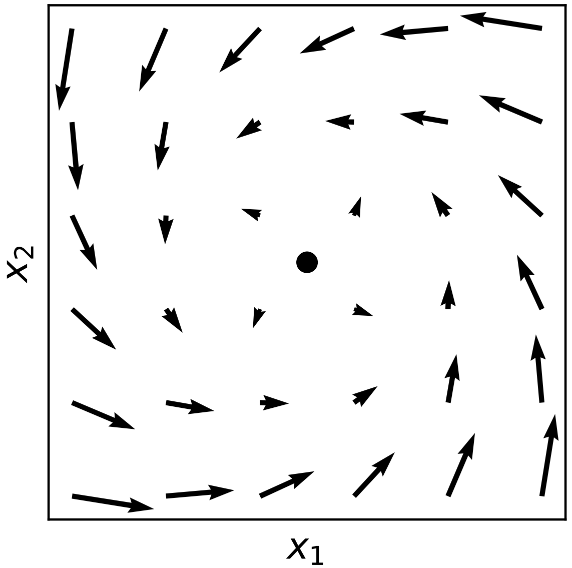
A new modeling idea

We propose a new model, the Gaussian process switching linear dynamical system (gpSLDS), which maintains the interpretability of rSLDS while addressing some of its drawbacks.

- ✓ Decomposes nonlinear dynamics into interpretable piecewise linear components
- ✓ Prior distribution on dynamics allows for posterior uncertainty estimates
- ✓ Smoothly transitioning dynamics at linear regime boundaries
- ✓ Produces a single set of dynamics instead of relying on discrete switching variables

GP-SDE framework

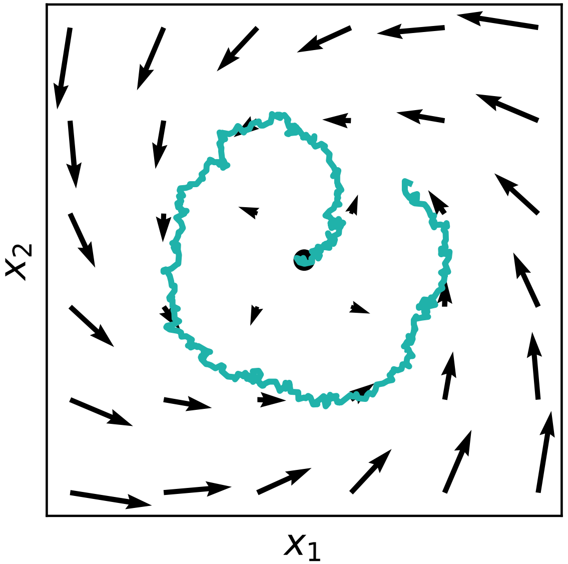
GP dynamics flow field



$$f_1(x), f_2(x) \sim \mathrm{GP}(0, K(\cdot\,,\cdot\,))$$

We propose a novel GP kernel to encode interpretable (“smooth piecewise linear”) structure into the dynamics prior

latent SDE trajectory

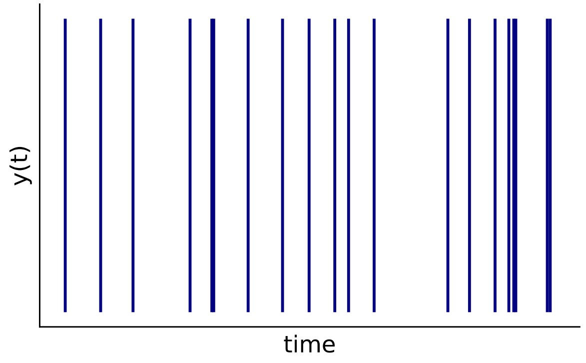


$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})dt + \boldsymbol{\Sigma}^{1/2}\mathbf{dw}$$

affine mapping to high-d space

irregularly sampled Gaussian observations

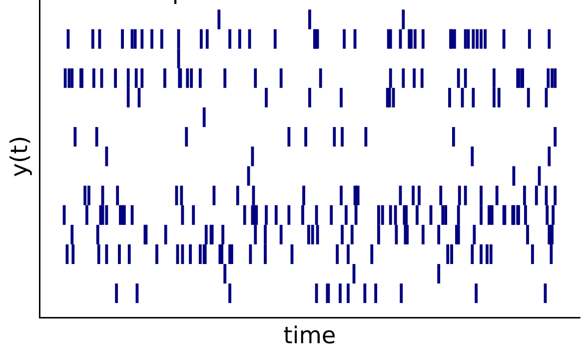
$$y_n(t_i) \sim \mathcal{N}(c_n^T x(t_i) + d_n,\ r)$$



Poisson process observations

$$y_n(t) \sim \mathcal{PP}(g(c_n^T x(t) + d_n))$$

$$g: \mathbb{R} \to \mathbb{R}_+$$

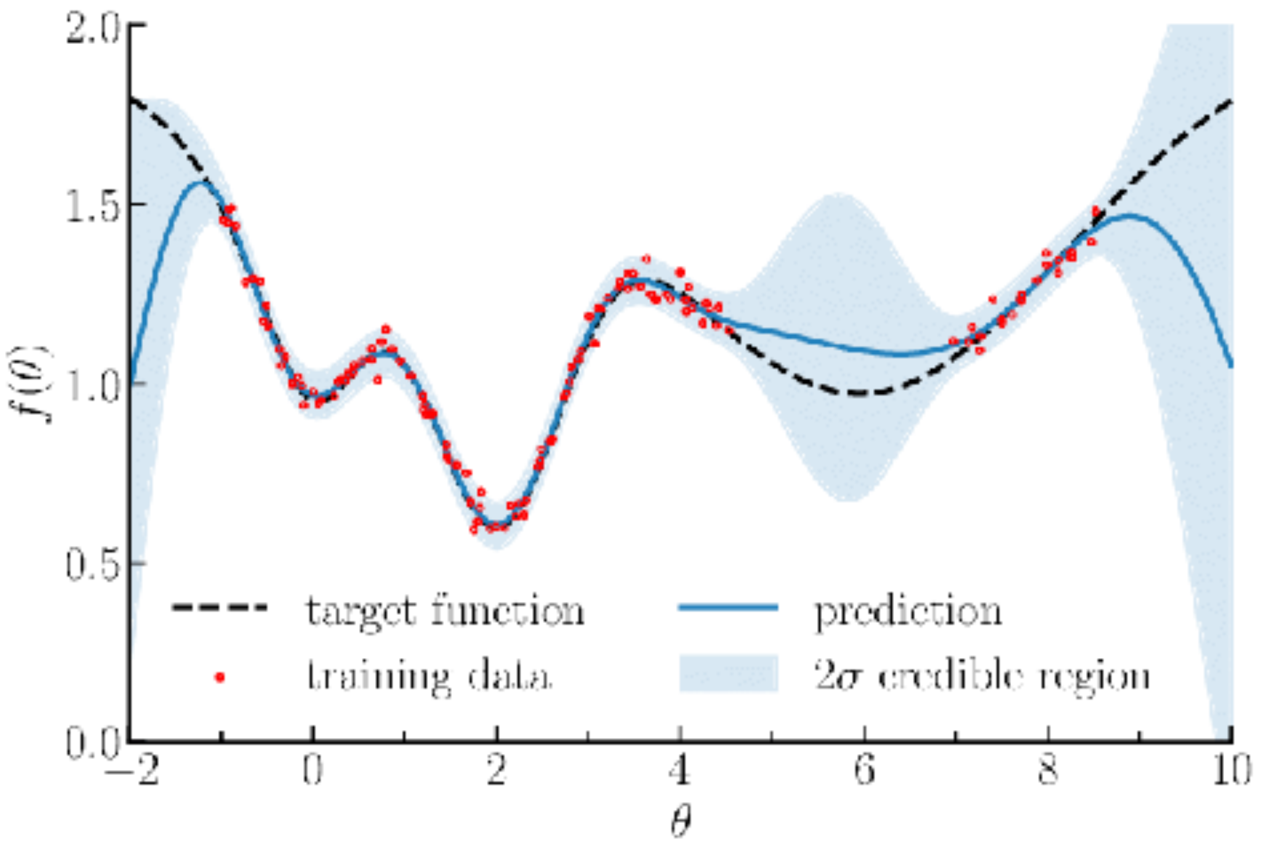


Gaussian processes

- Gaussian processes are distributions on functions $f : \mathbb{R}^D \mapsto \mathbb{R}$. (We can generalize to other domains as well.)

- Equivalently, a GP is a continuous set of random variables $\{f(x) : x \in \mathbb{R}^D\}$; i.e., a *stochastic process*.

- The defining property of GPs is that the function values at any finite collection of points are *jointly Gaussian*.



Gaussian processes

- We say $f \sim \mathrm{GP}(\mu(\cdot), K(\cdot, \cdot))$ if

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_N) \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) \cdots K(x_1, x_N) \\ \vdots \quad\quad \vdots \\ K(x_N, x_1) \cdots K(x_N, x_N) \end{bmatrix} \right)$$

  for all finite subsets of points $\{x_1, \ldots, x_N\} \subset \mathbb{R}^D$.

- Here, $\mu : \mathbb{R}^D \to \mathbb{R}$ is the **mean function** and $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the **covariance function**, or **kernel**.

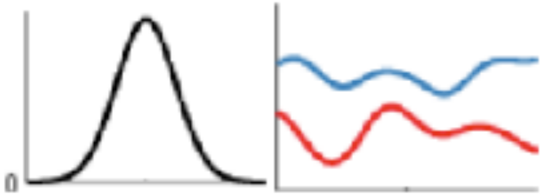
- The covariance matrix obtained by applying the covariance function to each pair of data points above is called the **Gram matrix**.

- The covariance function must be positive definite; i.e. the Gram matrix must be positive definite for any subset of points.

Kernel functions

- The choice of kernel allows for a wide range of prior distributions on functions.

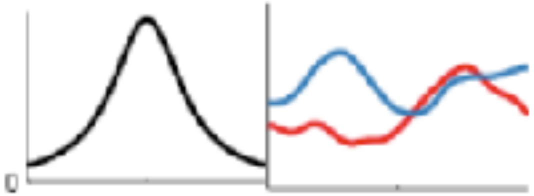
Squared Exponential Kernel



A.K.A. the Radial Basis Function kernel,

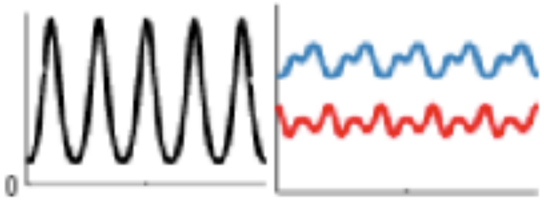
$k_{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$

Rational Quadratic Kernel



$k_{RQ}(x, x') = \sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha}$

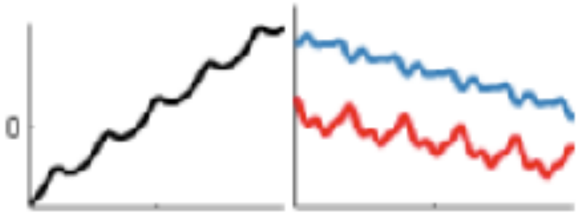
Periodic Kernel



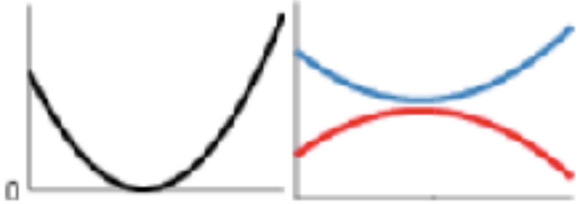
$k_{Per}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2}\right)$

- You can even add or multiply kernels to make new ones.

Linear plus Periodic



Linear times Linear



See https://www.cs.toronto.edu/~duvenaud/cookbook/

GP prior on linear functions

- $f(\cdot) \sim \mathrm{GP}(0, K(\cdot,\cdot))$
- When $K(\cdot,\cdot)$ is a linear kernel, we get a distribution over linear functions

$$K_{\text{lin}}(x, x') = (x - c)^\top M (x' - c) + \sigma_0^2$$

Intercept hyperparameter

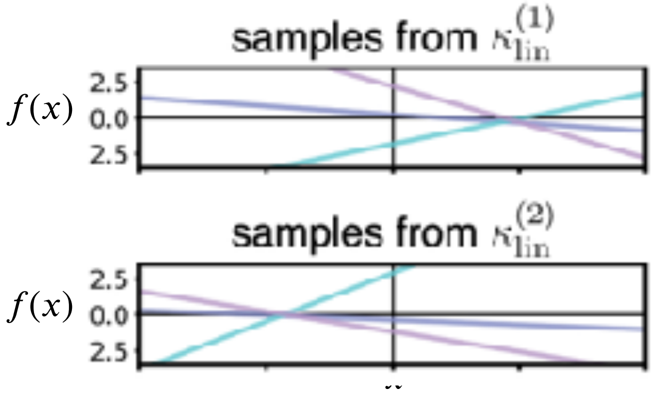
Slope variance hyperparameter

Global variance hyperparameter

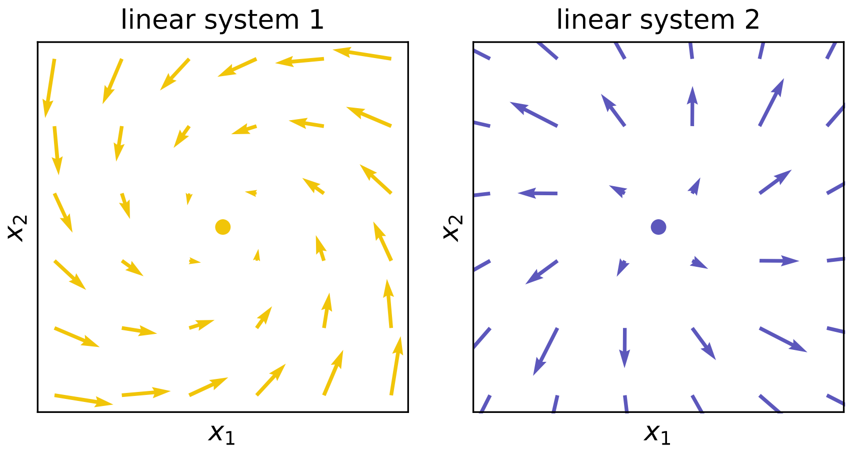
- This is equivalent to the Bayesian linear regression model,

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0$$
$$(\boldsymbol{\beta}, \beta_0)^T \sim \mathcal{N}\left(\boldsymbol{0}, \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{M}\boldsymbol{c} \\ -\boldsymbol{c}^T\boldsymbol{M} & \boldsymbol{c}^T\boldsymbol{c} + \sigma_0^2 \end{pmatrix}\right)$$

Random linear 1D functions



Random linear 2D functions



$$f : \mathbb{R}^D \mapsto \mathbb{R}^D$$
$$f_d \sim \mathrm{GP}(0, K_{\text{lin}}(\cdot,\cdot)), \quad d = 1,\ldots,D$$

GP prior on piecewise constant functions

- Let $\left(\mathscr{A}_1, \ldots, \mathscr{A}_J\right)$ be a partition of $\mathbb{R}^K$.

- Define $\pi(\boldsymbol{x})$ as the one-hot feature vector:

$$\pi(\boldsymbol{x}) = \left(\mathbb{I}\left[\boldsymbol{x} \in \mathscr{A}_1\right], \ \ldots, \ \mathbb{I}\left[\boldsymbol{x} \in \mathscr{A}_J\right]\right)^T$$

- The product between entries of this one-hot vector defines a “partition kernel”:

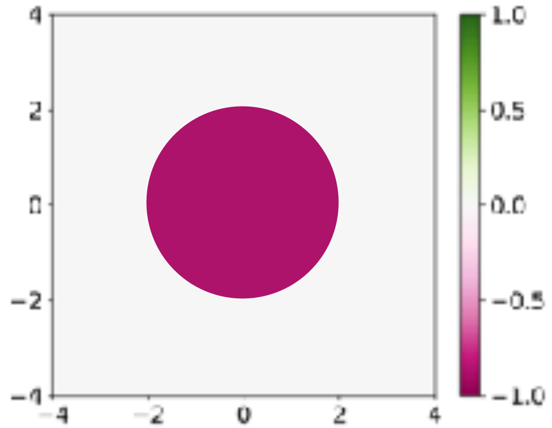
$$k_{part}^{(j)}(\boldsymbol{x}, \boldsymbol{x}') = \pi_j(\boldsymbol{x})\pi_j(\boldsymbol{x}')$$

which yields piecewise constant functions of the form,

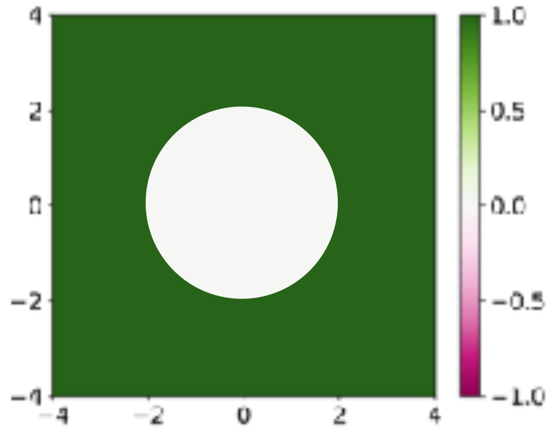
$$f(x) = c_j \quad \text{where } x \in \mathscr{A}_j$$

Sample from GPs with partition kernels

$k_{part}^{(1)}(\boldsymbol{x}, \boldsymbol{x}')$



$k_{part}^{(2)}(\boldsymbol{x}, \boldsymbol{x}')$



$\mathscr{A}_1 \colon \{\boldsymbol{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 4\}$

GP prior on piecewise constant functions

- Let $(\mathscr{A}_1, \ldots, \mathscr{A}_J)$ be a partition of $\mathbb{R}^K$.

- Define $\pi(\boldsymbol{x})$ as the one-hot feature vector:

$$\pi(\boldsymbol{x}) = \left(\mathbb{I}\left[\boldsymbol{x} \in \mathscr{A}_1\right], \ \ldots, \ \mathbb{I}\left[\boldsymbol{x} \in \mathscr{A}_J\right]\right)^T$$

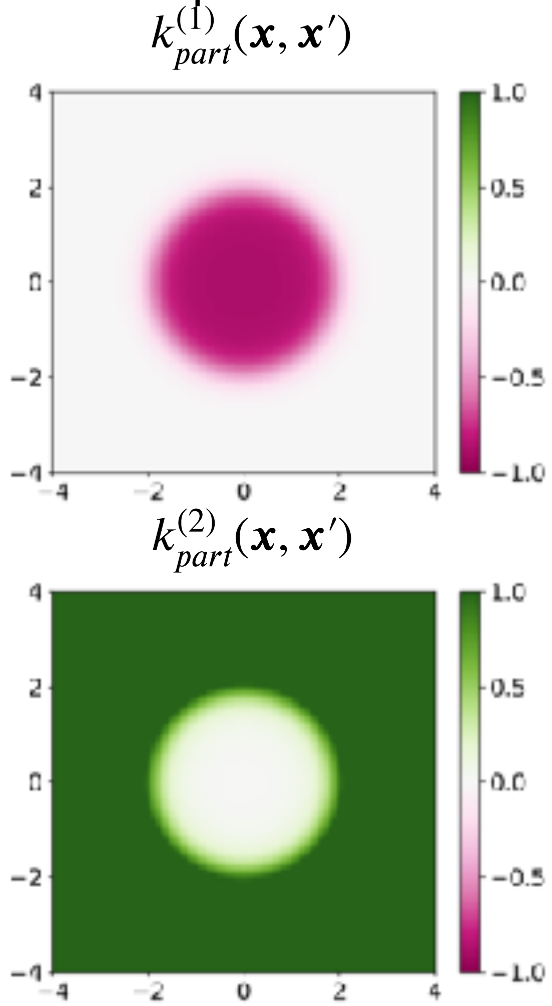
- The product between entries of this one-hot vector defines a “partition kernel”:

$$k_{part}^{(j)}(\boldsymbol{x}, \boldsymbol{x}') = \pi_j(\boldsymbol{x})\pi_j(\boldsymbol{x}')$$

  which yields piecewise constant functions.

- Now suppose $\pi(\boldsymbol{x}) \in \Delta_J$ varies smoothly. Then samples from a GP with this kernel are smoothly-interpolating piecewise constant functions.

Sample from GPs with smooth partition kernels



$\mathscr{A}_1$: $\{\boldsymbol{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 4\}$

gpSLDS: Smooth, piecewise linear dynamics

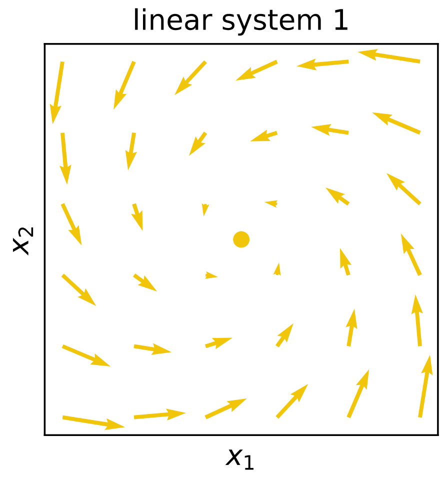
Our "smoothly switching linear" kernel is a weighted sum of linear kernels, with weights determined by the partition kernel.

$$k_{ssl}(\boldsymbol{x},\ \boldsymbol{x}') = \sum_{j=1}^{J} \kappa_{lin}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')\kappa_{part}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')$$

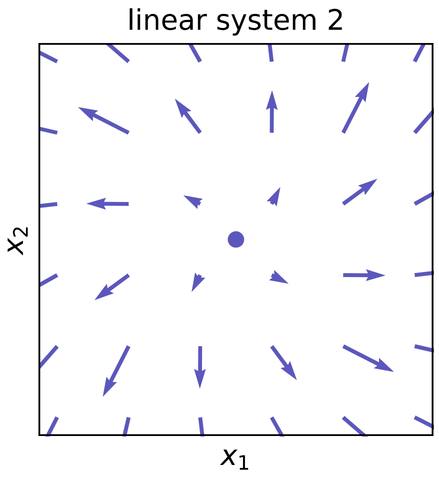
We parametrize the partition weights via a multiclass logistic regression:

$$\pi(\boldsymbol{x}) = \left(\pi_1(\boldsymbol{x})\ldots\pi_J(\boldsymbol{x})\right)^T = \mathrm{softmax}(\boldsymbol{W\phi}(\boldsymbol{x})/\tau)$$

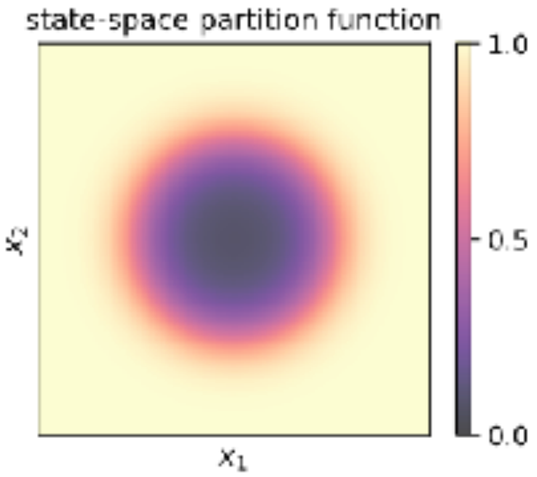
Balancing interpretability and flexibility:
- ✓ Interpretable composition of GP kernels
- ✓ Piecewise linear dynamics
- ✓ Smooth dynamics at boundaries



$k_{lin}^{(1)}(\cdot,\ \cdot)$



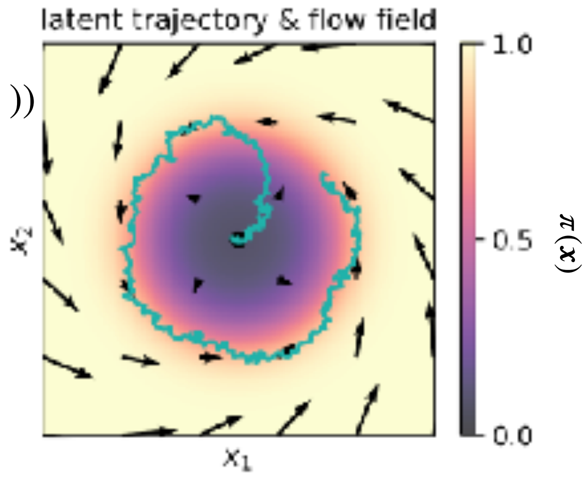
$k_{lin}^{(2)}(\cdot,\ \cdot)$



$\pi_1(\boldsymbol{x})$

$f_d(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{ssl}(\cdot,\ \cdot))$

$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})dt + \boldsymbol{\Sigma}^{1/2}\mathrm{d}\mathbf{w}$



Outline

- Scientific motivation

  Inferring interpretable descriptions of latent neural dynamics

- Review of existing methods

  SLDS models decompose nonlinear dynamics into simpler components, but with some shortcomings

- New modeling idea

  gpSLDS uses a novel kernel for a prior on smoothly switching piecewise-linear dynamics

- New inference algorithm

- Results

- Future work

y: observations
x: latent trajectories
f: dynamics
u: inducing points
θ: kernel params (partition & smoothness)

Inference and learning for GP-SDE

- Prior on latent states $\boldsymbol{x}(t)$ is nonlinear and non-Gaussian due to $f(\cdot)$
- We build off a variational EM framework first proposed by Archambeau et al. (2007), Titsias (2009), and Duncker et al. (2019) to infer $\boldsymbol{x}$, $\boldsymbol{f}$ and learn kernel hyperparameters θ

$$q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{x}) \prod_k p(f_k \mid u_k, \theta) q(u_k)$$

Latent SDE trajectory:
Gaussian Markov process

$$q(\boldsymbol{x}):\ d\boldsymbol{x} = \left(-\boldsymbol{A}(t)\boldsymbol{x} + \boldsymbol{b}(t)\right)dt + \boldsymbol{\Sigma}^{1/2} d\boldsymbol{w}$$
$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$$

Gaussian process dynamics:
Sparse approx. with inducing variables

$$q(\boldsymbol{f}) = \int p(\boldsymbol{f} \mid \boldsymbol{u}, \boldsymbol{\theta}) q(\boldsymbol{u}) d\boldsymbol{u}$$

$$N(\boldsymbol{u}_k \mid \boldsymbol{m}_{\boldsymbol{u}}^k, \boldsymbol{S}_{\boldsymbol{u}}^k)$$

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ : kernel params (partition & smoothness)

Variational Inference and Learning

- The evidence lower bound (ELBO) of the model is

$$\mathscr{L}[q,\theta] = \mathbb{E}_{q(x)}[\log p(y \mid x)] - \mathbb{E}_{q(f)}\left[\mathrm{KL}(q(x) \,\|\, p(x \mid f))\right] - \sum_{d=1}^{D} \mathrm{KL}(q(u_d) \,\|\, p(u_d \mid \theta))$$

- We use a variational family of **Gaussian Markov processes**,

$$q(x) : dx = (-Ax + b)dt + \Sigma^{1/2} dW$$

where $W(t)$ is a standard D-dimensional Brownian motion.

- On any discrete grid of points, $q(x_{0:T})$ is jointly Gaussian,

$$q(x_{0:T} \mid \eta) \propto \exp\left\{ -\frac{1}{2} \sum_{t=0}^{T} x_t^\top J_t x_t - \sum_{t=0}^{T-1} x_{t+1}^\top L_t x_t + \sum_{t=0}^{T} x_t^\top h_t \right\}$$

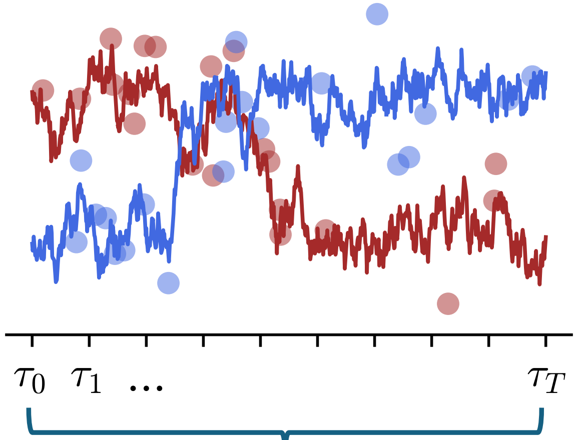
where $\eta = \{J_t, L_t, h_t\}$ are the natural parameters.

- **Goal:** find η and θ to maximize the ELBO.

Variational Inference and Learning

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ: kernel params (partition & smoothness)

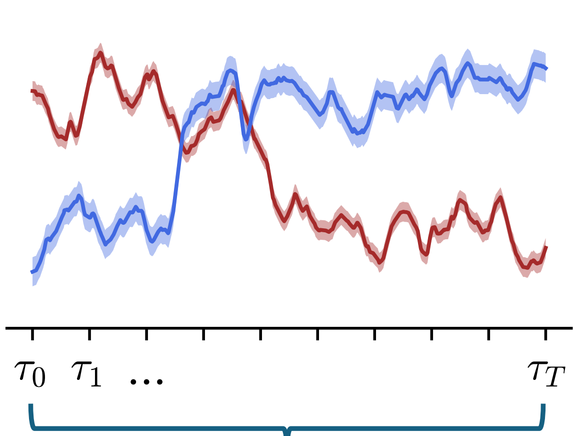
Prior process and observations



marginalize & discretize

$\tilde{p}(\boldsymbol{x}_{0:T})$

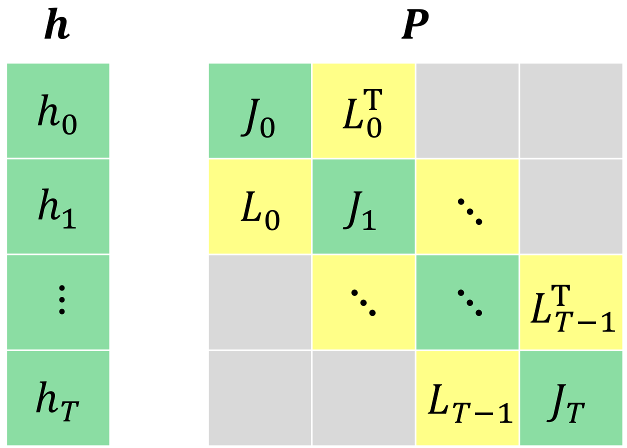
Inferred latent state posterior



marginalize

$q(\boldsymbol{x}_{0:T}) = N_I(\boldsymbol{x}_{0:T}|\boldsymbol{h},\boldsymbol{P})$

Variational natural parameters



Natural Gradient Ascent

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ: kernel params (partition & smoothness)

- Rather than simply performing (stochastic) gradient ascent on the ELBO, we can obtain faster rates of convergence with **natural gradient ascent**.

- Let $\mathscr{F}(\eta) = \mathbb{E}_{\bar{q}(z|\eta)}\left[\nabla_\eta \log \bar{q}(z\,|\,\eta)\,\nabla_\eta \log \bar{q}(z\,|\,\eta)^\top\right]$ denote the **Fisher information matrix**.

- The natural gradient ascent update is,

$$\eta^{(j+1)} = \arg\min_\eta - \eta^\top \nabla_\eta \mathscr{L}(\eta^{(j)}) + \frac{1}{\rho} \cdot \underbrace{\frac{1}{2}(\eta - \eta^{(j)})^\top \mathscr{F}(\eta^{(j)})(\eta - \eta^{(j)})}_{\approx \mathrm{KL}(\bar{q}(z|\eta^{(j)}) \,\|\, \bar{q}(z|\eta))}$$

$$\implies \eta^{(j+1)} = \eta^{(j)} + \rho[\mathscr{F}(\eta^{(j)})]^{-1} \nabla_\eta \mathscr{L}(\eta^{(j)})$$

- If we used the identity matrix instead of the Fisher information matrix, we would recover the standard gradient ascent step. The difference is that the natural gradient step accounts for the curvature of the ELBO.

Inferring posterior on dynamics

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ: kernel params (partition & smoothness)

- Duncker et al. (2019) extended the original algorithm to incorporate inducing points, which allows for tractable inference of the dynamics function $f(\cdot)$.
- They show that the dynamics variational distribution, $q(\boldsymbol{u}_k) = \mathcal{N}(\boldsymbol{u}_k \,|\, \boldsymbol{m}_{\boldsymbol{u}}^k,\ \boldsymbol{S}_{\boldsymbol{u}}^k)$, can be updated conveniently in closed form.
- Then, after fitting the model, we can easily recover the inferred posterior on dynamics at any new input location x^* using Gaussian conjugacy:

$$
\begin{aligned}
q\Big(f_k\big(\boldsymbol{x}^*\big)\Big) &= \int p\big(f_k(\boldsymbol{x}^*) \mid \boldsymbol{u}_k,\ \Theta\big)\ N\big(\boldsymbol{u}_k \mid \boldsymbol{m}_{\boldsymbol{u}}^k, , \boldsymbol{S}_{\boldsymbol{u}}^k\big)\, d\boldsymbol{u}_k \\
&= \int N\Big(f_k\big(\boldsymbol{x}^*\big) \mid \boldsymbol{k}_{x^*z}\boldsymbol{K}_{zz}^{-1}\boldsymbol{u}_k,\ k_{x^*x^*} - \boldsymbol{k}_{x^*z}\boldsymbol{K}_{zz}^{-1}\boldsymbol{k}_{zx^*}\Big) N\big(\boldsymbol{u}_k \mid \boldsymbol{m}_{\boldsymbol{u}}^k, \boldsymbol{S}_{\boldsymbol{u}}^k\big)\, d\boldsymbol{u}_k \\
&= N\Big(f_k\big(\boldsymbol{x}^*\big) \mid \boldsymbol{k}_{x^*z}\boldsymbol{K}_{zz}^{-1}\boldsymbol{m}_{\boldsymbol{u}}^k,\ k_{x^*x^*} - \boldsymbol{k}_{x^*z}\boldsymbol{K}_{zz}^{-1}\boldsymbol{k}_{zx^*} + \boldsymbol{k}_{x^*z}\boldsymbol{K}_{zz}^{-1}\boldsymbol{S}_{\boldsymbol{u}}^k\boldsymbol{K}_{zz}^{-1}\boldsymbol{k}_{zx^*}\Big)
\end{aligned}
$$

An improved parameter learning objective

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ : kernel params (partition & smoothness)

Learn hyperparameters $\boldsymbol{\Theta}$ by maximizing a partially optimized ELBO,

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \left\{ \max_{q(\boldsymbol{u})} L(q(\boldsymbol{x}), q(\boldsymbol{u}),\ \boldsymbol{\Theta}) \right\}$$

- The inner maximization can be performed in closed form.
- This approach can be thought of as jointly maximizing the ELBO with respect to both $q(\boldsymbol{u})$ and $\boldsymbol{\Theta}$, which helps circumvent local optima during variational EM.

$\boldsymbol{y}$: observations
$\boldsymbol{x}$: latent trajectories
$\boldsymbol{f}$: dynamics
$\boldsymbol{u}$: inducing points
θ: kernel params (partition & smoothness)

A new "collapsed" vEM algorithm

- Putting it all together, our collapsed vEM algorithm is:

1) Update posterior on latents $q^*(\boldsymbol{x})$ using natural gradient ascent.

2) Compute new learning objective, which results after performing closed-form maximization of the ELBO with respect to $q(\boldsymbol{u})$

3) Perform gradient descent on this partially optimized objective with respect to $\boldsymbol{\Theta}$ until convergence

4) Given the new $\boldsymbol{\Theta}$, compute the optimal $q^*\left(\boldsymbol{u}_d\right) = \mathcal{N}(\boldsymbol{u}_d \,|\, \boldsymbol{m}_d^{u*},\ \boldsymbol{S}_d^{u*})$.

Outline

- Scientific motivation

  Inferring interpretable descriptions of latent neural dynamics

- Review of existing methods

  SLDS models decompose nonlinear dynamics into simpler components, but with some shortcomings

- New modeling idea

  gpSLDS uses a novel kernel for a prior on piecewise-linear dynamics

- New inference algorithm

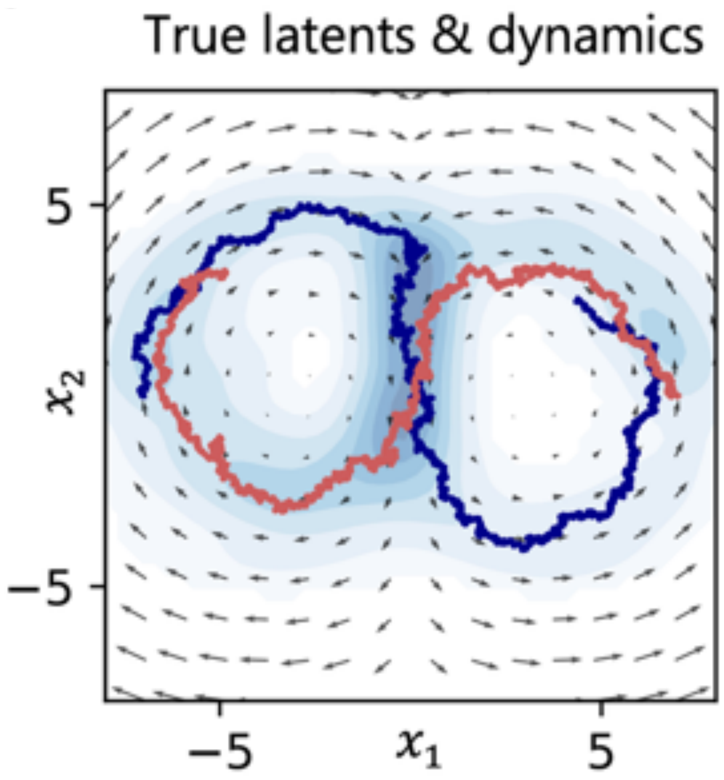
  We developed a natural gradient ascent algorithm for GP-SDEs called "SING"

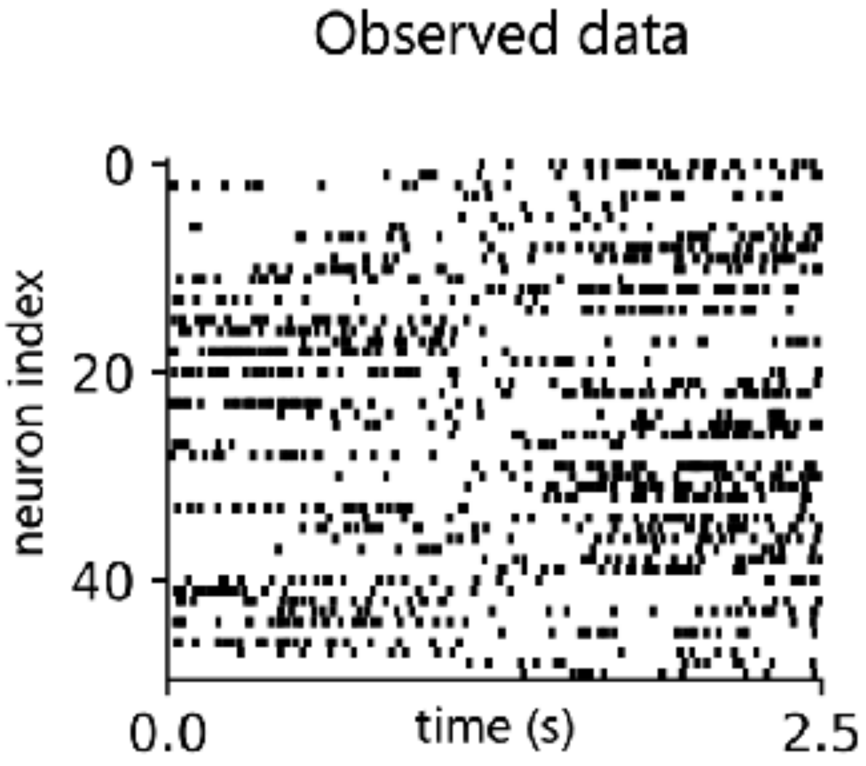
- Results

- Future work

Synthetic example: 2 rotation systems

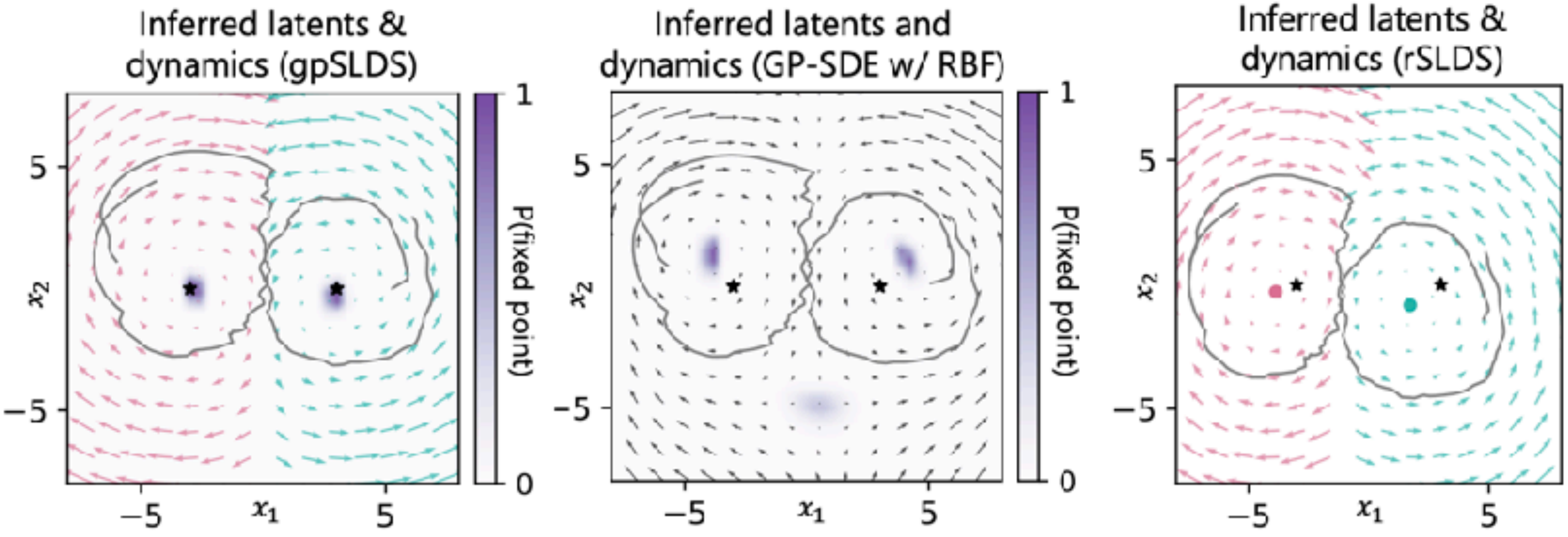
- 2 linear rotation systems that combine smoothly at $x_1 = 0$
- 30 trials of Poisson process observations from 50 output dimensions (“neurons”)





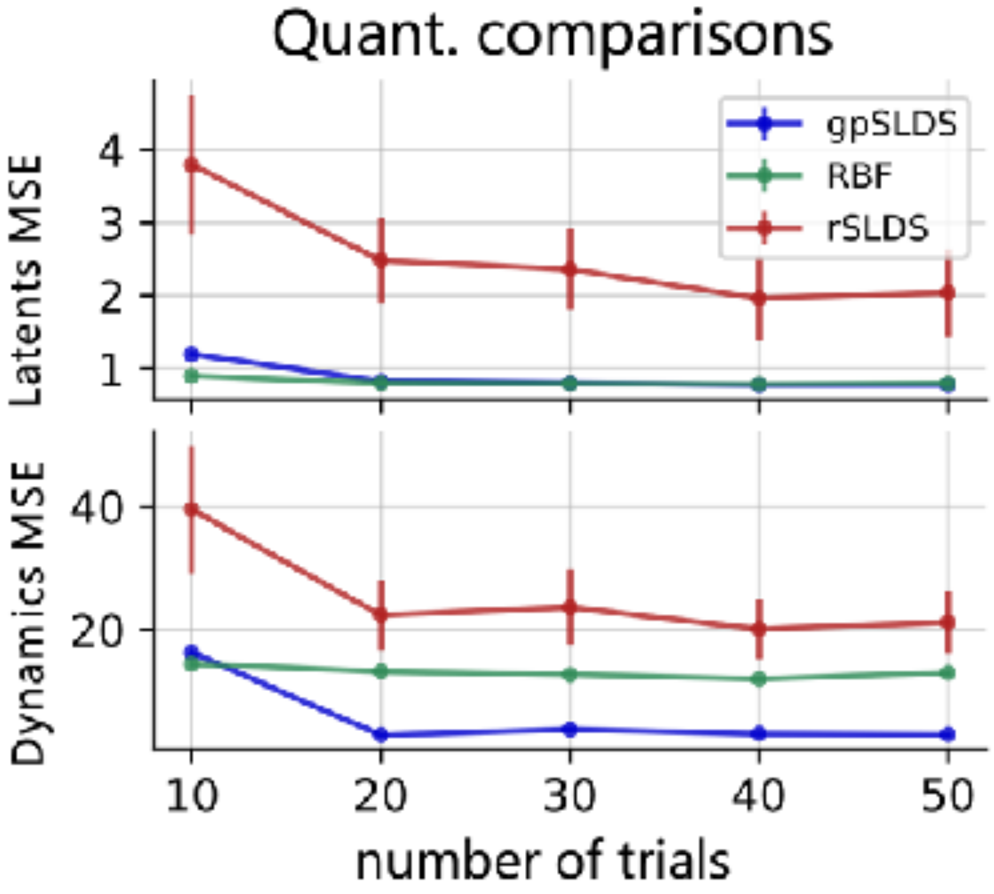
Synthetic example: 2 rotation systems

- The gpSLDS more accurately recovers the true latent trajectories, rotation dynamics, and decision boundaries compared to competing methods



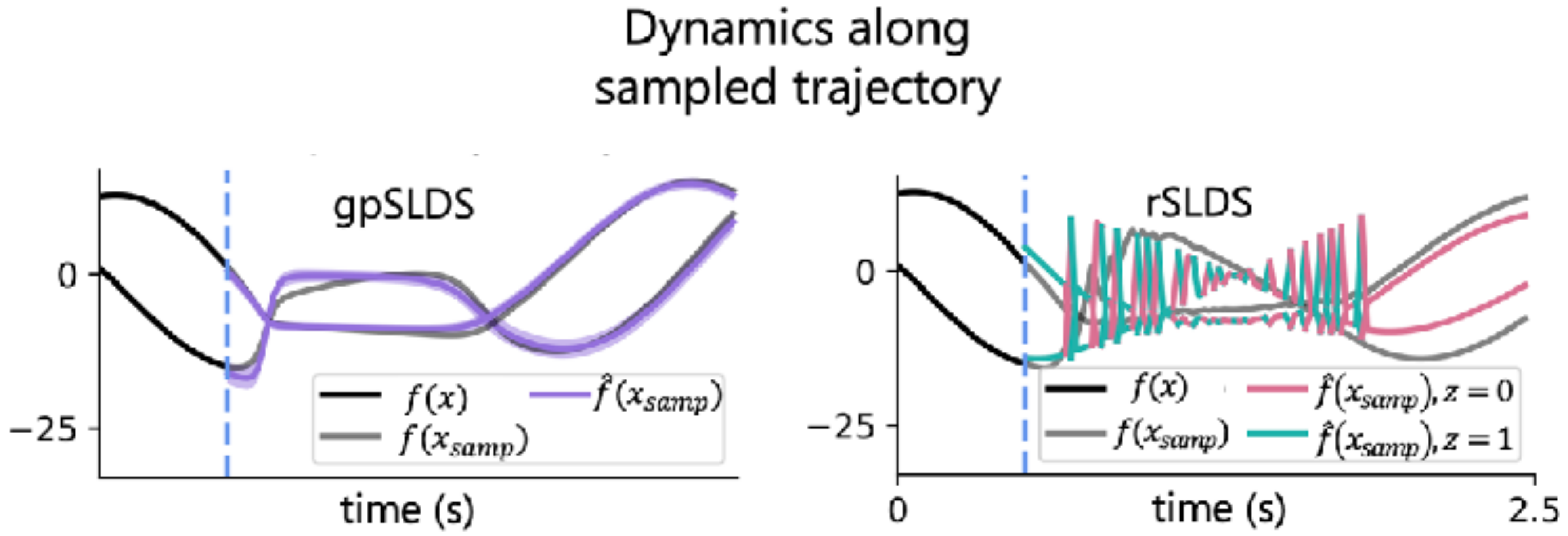
Synthetic example: 2 rotation systems

- The gpSLDS more accurately recovers the true latent trajectories, rotation dynamics, and decision boundaries compared to competing methods

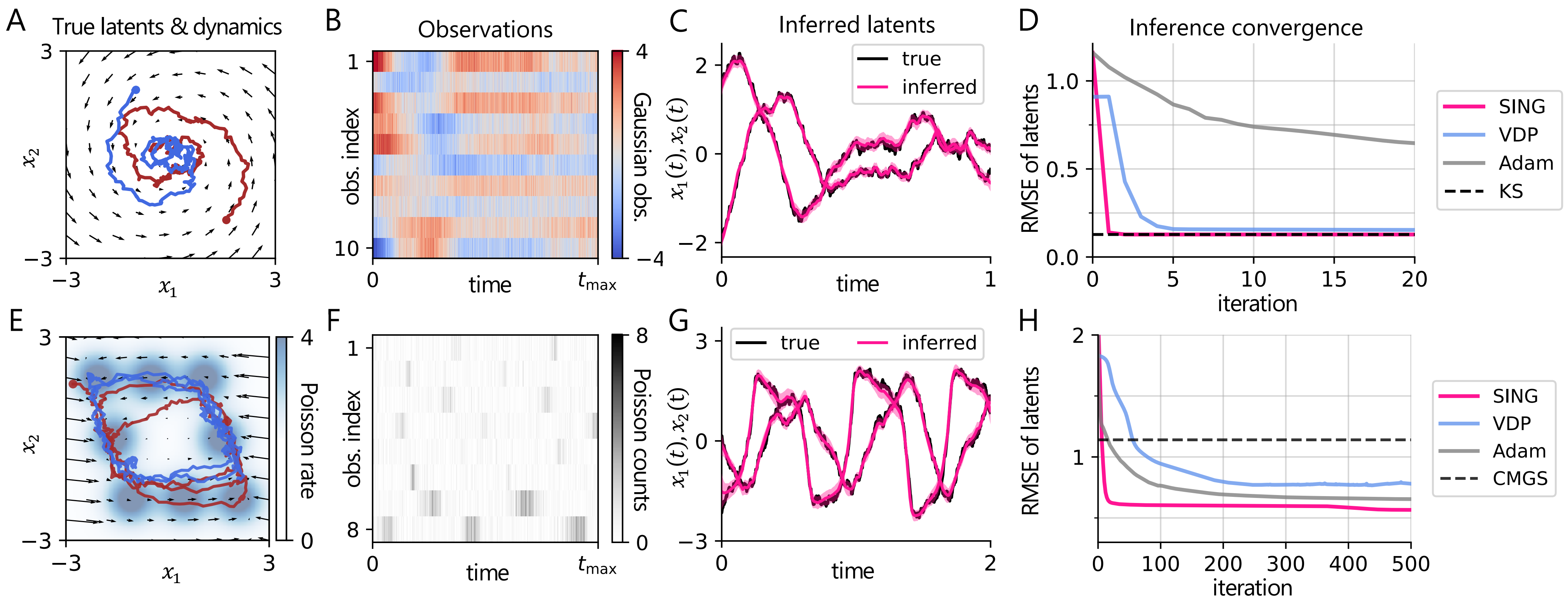


Synthetic example: 2 rotation systems

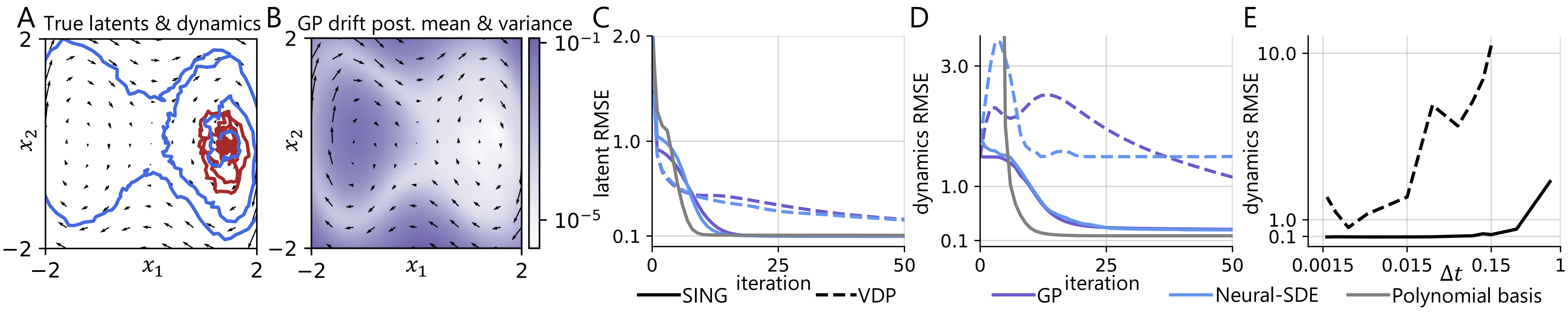
- The gpSLDS produces smooth simulated dynamics that match the true dynamics, and expresses uncertainty directly in function space
- By contrast, the rSLDS expresses uncertainty by oscillating between the two linear systems, producing uninterpretable dynamics



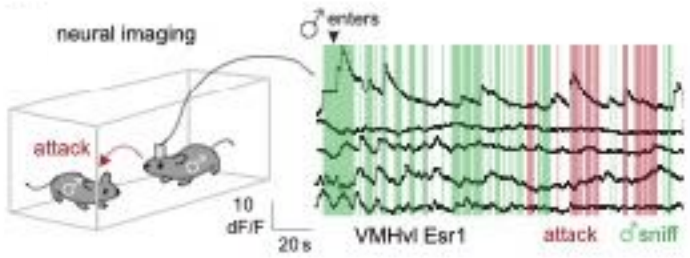
Synthetic examples: natural gradient ascent is much faster



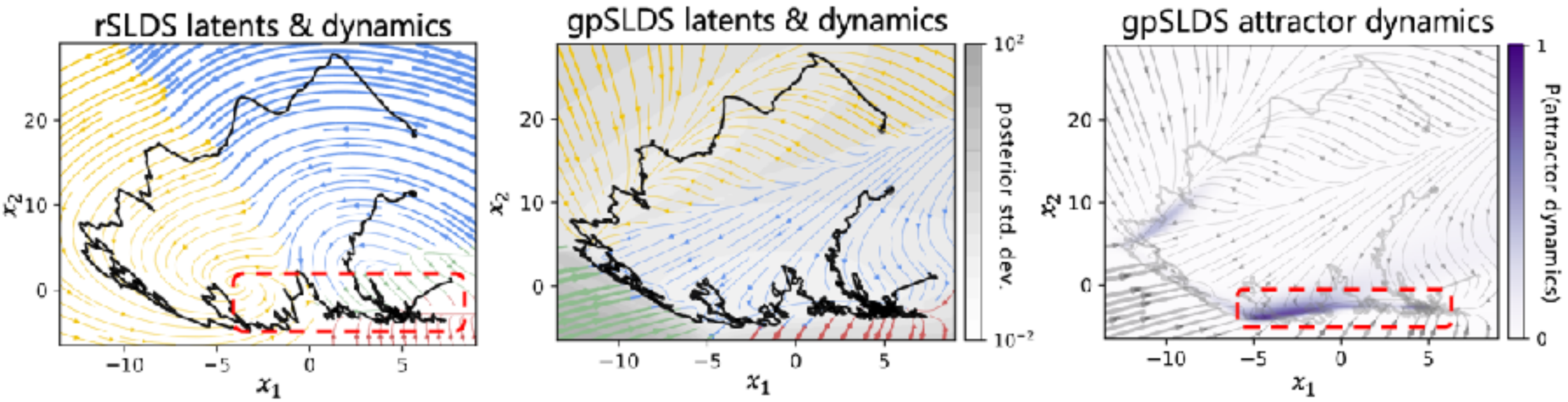
Synthetic examples: Gaussian process posterior captures uncertainty about the dynamics function



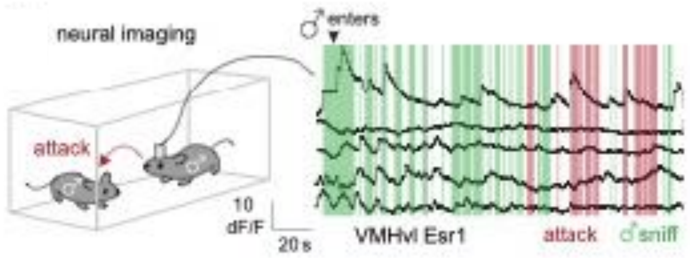
Revisiting dynamics of aggression



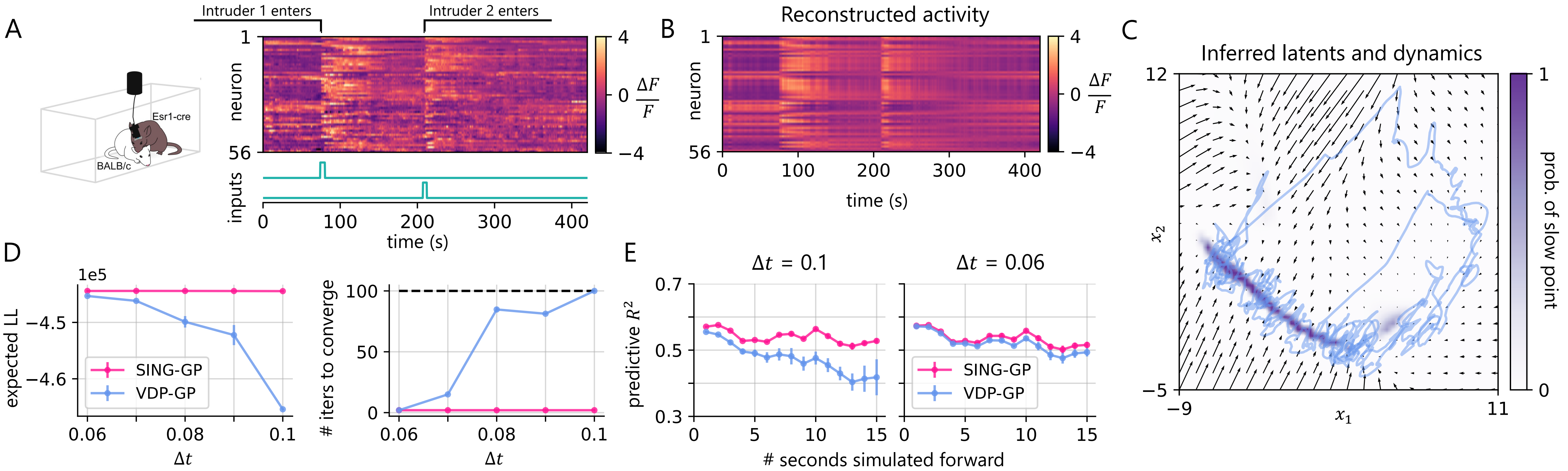
- We use the gpSLDS to revisit the analyses of Nair et al. (2023), which applied rSLDS models to calcium imaging recorded during aggression in mice
- Both methods infer similar latent trajectories and plausible flow fields
- Further, the gpSLDS can allow us to precisely identify the approximate line attractor, and more generally to estimate model confidence in dynamics



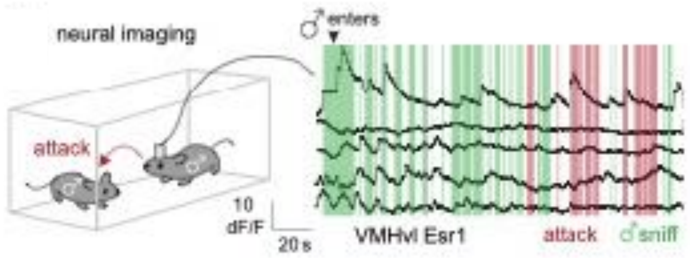
Revisiting dynamics of aggression



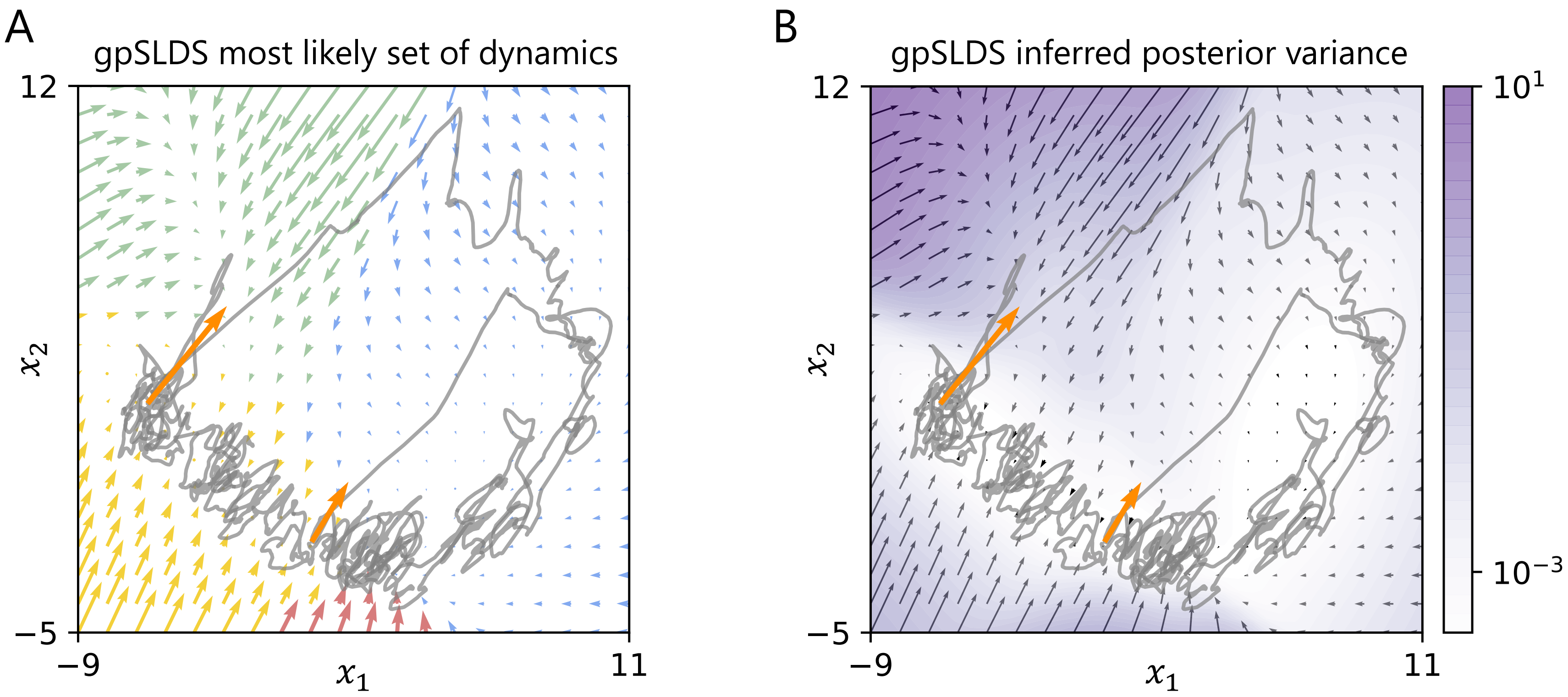
- We also revisited the data from Vinograd, Nair et al (2024).



Revisiting dynamics of aggression



- We also revisited the data from Vinograd, Nair et al (2024).



Outline

- Scientific motivation

  Inferring interpretable descriptions of latent neural dynamics

- Review of existing methods

  SLDS models decompose nonlinear dynamics into simpler components, but with some shortcomings

- New modeling idea

  gpSLDS uses a novel kernel for a prior on piecewise-linear dynamics

- New inference algorithm

  We developed a natural gradient ascent algorithm for GP-SDEs called “SING”

- Results

  gpSLDS recovers generative parameters on synthetic data and finds key dynamical structures in real data

- Conclusion

Future directions

A couple things we are working on now…

- Extending the gpSLDS model to time-varying dynamics

- Continuing collaborations with experimentalists to apply our method to new neuroscience datasets

Thanks!

The GP-SLDS was presented at NeurIPS ’24, and we recently submitted SING to NeurIPS ’25.

Find more at:

- Paper 📃: https://arxiv.org/abs/2408.03330
- Code 👩🏻‍💻: https://github.com/lindermanlab/gpslds