SIMPL: SCALABLE AND HASSLE-FREE OPTIMISATION OF NEURAL REPRESENTATIONS FROM BEHAVIOUR

Anonymous authors
Paper under double-blind review

ABSTRACT

High-dimensional neural activity in the brain is known to encode low-dimensional, time-evolving, behaviour-related variables. A fundamental goal of neural data analysis consists of identifying such variables and their mapping to neural activity. The canonical approach is to assume the latent variables *are* behaviour and visualize the subsequent tuning curves. However, significant mismatches between behaviour and the encoded variables may still exist — the agent may be thinking of another location, or be uncertain of its own — distorting the tuning curves and decreasing their interpretability. To address this issue a variety of methods have been proposed to learn this latent variable in an unsupervised manner; these techniques are typically expensive to train, come with many hyperparameters or scale poorly to large datasets complicating their adoption in practice. To solve these issues we propose SIMPL (Scalable Iterative Maximization of Population-coded Latents), an EM-style algorithm which iteratively optimizes latent variables and tuning curves. SIMPL is fast, scalable and exploits behaviour as an initial condition to further improve convergence and identifiability. We show SIMPL accurately recovers latent variables in biologically-inspired spatial and non-spatial tasks. When applied to a large rodent hippocampal dataset SIMPL efficiently finds a modified latent space with smaller, more numerous, and more uniformly-sized place fields than those based on behaviour, suggesting the brain may encode space with greater resolution than previously thought.

1 INTRODUCTION

Large neural populations in the brain are known to encode low-dimensional, time-evolving latent variables which are, oftentimes, closely related to behaviour (Afshar et al., 2011; Harvey et al., 2012; Mante et al., 2013; Carnevale et al., 2015). Coupled with the advent of modern neural recording techniques (Jun et al., 2017; Wilt et al., 2009) focus has shifted from single-cell studies to the joint analysis of hundreds of neurons across ever increasing time windows, where the goal is to extract these variables using a variety of statistical (Yu et al., 2008a; Cunningham & Yu, 2014; Kobak et al., 2016; Zhao & Park, 2017; Williams et al., 2020) and computational (Van der Maaten & Hinton, 2008; Pandarinath et al., 2018; Mackevicius et al., 2019) methods.

This paradigm shift is particularly pertinent in the context of the mammalian spatial memory and motor systems where celebrated discoveries have identified cells whose neural activity depends on behavioural variables such as position (O'Keefe & Dostrovsky, 1971; Hafting et al., 2005), heading direction (Taube et al., 1990), speed (McNaughton et al., 1983), distance to environmental boundaries/objects (Lever et al., 2009; Høydal et al., 2019) and limb movement direction(Georgopoulos et al., 1986) through complex and non-linear tuning curves. Characterising neural activity in terms of behaviour has been, and remains, a cornerstone practice in these fields, particularly navigation; however, the core assumption supporting it — that the latent variable encoded by neural activity *is and only is* the behavioural variable — is increasingly being called into question (Sanders et al., 2015; Whittington et al., 2020; George et al., 2024b).

The brain is not a passive observer of the world. Active internal processing like planning a future route (Spiers & Maguire, 2006) or recalling past positions (Squire et al., 2010) as well as observed phenomena such as replay (Carr et al., 2011), theta sweeps (Maurer et al., 2006), and predictive coding (Muller & Kubie, 1989; Mehta et al., 1997; Stachenfeld et al., 2017) will cause encoded

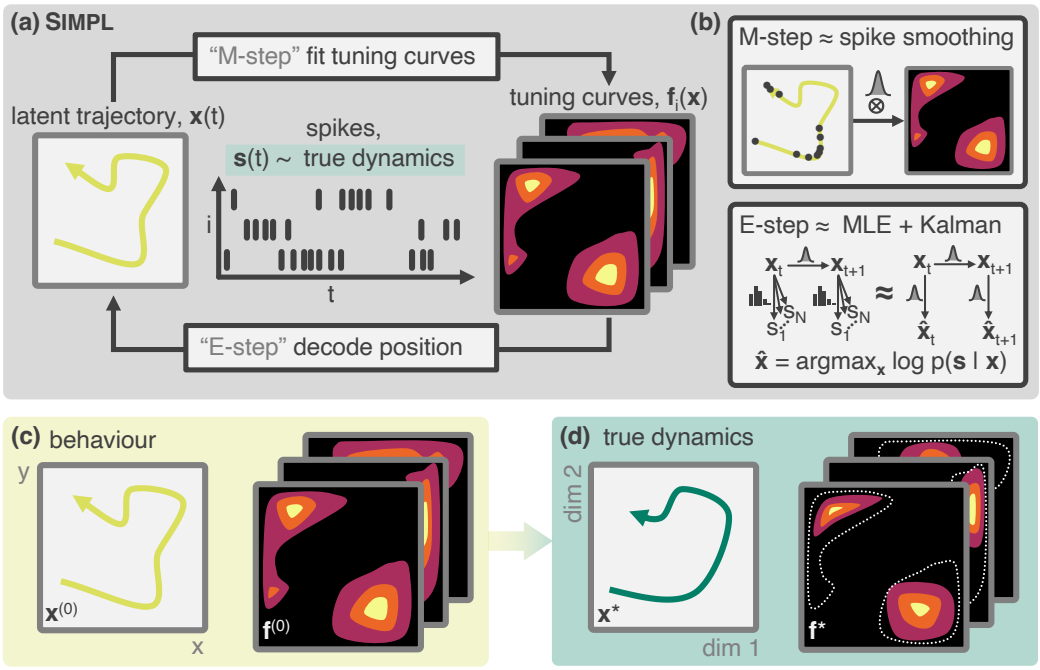

Figure 1: Schematic of the SIMPL algorithm. **(a)** A latent variable model for spiking data $(\mathbf{f}_i(\mathbf{x}), \mathbf{x}(t))$ is optimised by iterating a two-step procedure closely related to the expectation-maximization (EM, Dempster et al. 1977) algorithm: First, tuning curves are fitted to an initial estimate of the latent variable (the "M-step"), which are then used to *re*decode the latent variable (the "E-step"). **(b)** SIMPL fits tuning curves using kernel density estimation (KDE) with a Gaussian kernel (top) and decodes the latent variables by Kalman-smoothing maximum likelihood estimates. Measured behaviour **(c)** is used to initialise the algorithm as it is often closely related to the true generative latent variable of interest **(d)**.

variables to deviate from behaviour. Additionally, the brain is not a perfect observer; irreducible uncertainty due to limited, noisy or ambiguous sensory data can lead to similar encoding discrepancies. Experimental inaccuracies in measurement can contribute further. These hypotheses are supported by analyses which show that it is rarely, if ever, possible to perfectly decode "behaviour" from neural data (Glaser et al., 2020) as well as the observation that neurons show high variability under identical behavioural conditions (Fenton & Muller, 1998; Low et al., 2018).

All combined, these facts hint at a richer and more complex internal neural code. When this complexity is not accounted for, as is often the case, tuning curves will be blurred or distorted relative to their true form, weakening the validity of the conclusions drawn from them. As an explicit example, consider an animal situated at position X 'imagining' or 'anticipating' another position, Y, for which a cell is tuned (e.g. the cell has a place field at Y). This might trigger the cell to fire leading to the mistaken conclusion that the cell has a place field at location X.

Nonetheless, the observation that in such settings, behaviour is still a close-but-imperfect proxy for the true latent variable motivates the search for techniques tailored to *exploit* this link. Many modern techniques for neural data analysis were developed in a more general context and thus don't exploit behaviour(Gao et al., 2016; Gondur et al., 2023) at the cost of complexity and interpretability. Others do not model temporal dynamics(Zhou & Wei, 2020; Schneider et al., 2023; Lawrence, 2003), don't scale to large neural datasets (Wu et al., 2017; Nam, 2015), cannot model complex non-linear tuning curves(Pandarinath et al., 2018; Hurwitz et al., 2021; Duncker et al., 2019; Linderman et al., 2016; Gondur et al., 2023), or are not inately designed for spiking datasets(Lawrence, 2003; Krishnan et al., 2015). Moreover, many of these methods are conceptually complex, lack readily usable code implementations, or necessitate GPUs for efficient training restricting their accessibility and applicability as tools in experimental neuroscience.

Contributions Here we introduce SIMPL (Scalable Iterative Maximization of Population-coded Latents), a straightforward yet effective enhancement to the current paradigm. Our approach fits tuning curves to observed behaviour and refines these by iterating a two-step process: first we *de-*

code the latent variable from the previously estimated tuning curves; then, we *refit* the curves based on these decoded latents. SIMPL imposes minimal constraints on the structure of the tuning curves, scales well to large neural datasets and does not rely on neural network function approximators which can be hard to interpret and expensive to train. We theoretically analyse SIMPL and establish formal connections to expectation-maximisation (EM, Dempster et al. 1977) for a simple but flexible class of generative models. By exploiting behaviour as an initialization, SIMPL converges fast and alleviates issues to do with local minima and identifiability (Hyvärinen & Pajunen, 1999; Locatello et al., 2019). This allows it to reliably return refined tuning curves and latent variables which remain close to, but improve upon, their behavioural analogues readily admitting direct comparison. All in all, SIMPL is able to identify temporally smooth latents and complex tuning curves related to behaviour, while remaining computationally cheap and natively supporting spiking data, a distinguishing set of features in the field of latent variable models for neural data analysis.

TG Add a sentence about EM being a benefit because (i) it inherently permits initialisation in a way gradient-based methods dont and (ii) is very fast We first validate SIMPL on a dataset of synthetically generated 2D grid cells. We then apply SIMPL to rodent electrophysiological hippocampal data (Tanni et al., 2022) and show it modifies the latent space in an incremental but significant way: The optimised tuning curves explain the data better than their behavioural counterparts and contain sharper, more numerous place fields allowing for a reinterpretation of previous experimental results. Finally, we apply SIMPL to somatosensory dataset for a monkey performing a centre-out reaching task (Chowdhury et al., 2020) SIMPL, with a four-dimensional latent space, provides a good account of the data with the latent variables initialised to (and remaining correlated with) the monkeys hand-position and hand-velocity. SIMPL has only two hyperparameters and can be run on quickly on large neural datasets [1] without requiring a GPU. It outperforms popular alternative technique based on neural networks (Schneider et al., 2023; Zhao & Park, 2017) or Gaussian processes(Lawrence, 2003) and is over 30× faster. This make it a practical alternative to existing tools particularly of interest to navigational or motor-control communities where data is abundant and available behavioural variables are close to the true latent. We provide an open-source JAX-optimised (Bradbury et al., 2018) implementation of our code[2].

2 RELATED WORK

Probabilistic inference in neural data modulated by latent variables has been a major topic of study for decades — see, e.g. Tipping & Bishop (1999); Yu et al. (2006; 2008b;a); Macke et al. (2011); Mangion et al. (2011); Park et al. (2015); Gao et al. (2016); Hernandez et al. (2018); Dong et al. (2020); Zhou & Wei (2020); Gondur et al. (2023) — however not all methods were designed for the kind of data considered in this work. For instance, many methods contain a complex model of latent space dynamics, but combine these with simplistic tuning curves which restrict firing rates to linear, or exponential-linear, functions of the latent variable (Smith & Brown, 2003; Yu et al., 2008a; Macke et al., 2011; Duncker et al., 2019; Linderman et al., 2016; Pandarinath et al., 2018; Zoltowski et al., 2020; Sani et al., 2021; Hurwitz et al., 2021; Kim et al., 2021; Gondur et al., 2023). Such models cannot interpretably account account for the types of data we consider here, such as grid cells and place cells. Others methods do not, or cannot, make use of behaviour to support discovery of latent variables (Gao et al., 2016; Nam, 2015; Hernandez et al., 2018; Gondur et al., 2023). These methods take a fully "unsupervised" approach which is more general (for example to spike data without an obvious behavioural correlate) but which often comes at the expense of complexity and identifiability.

Algorithms that both don't restrict to simplistic intensity functions and exploit behaviour form a small but diverse set of relevant alternatives to SIMPL. Such behaviour-informed latent discovery and joint neural-behavioural analysis tools have become popular in recent years due to the explosion of large and readily available neural datasets taken from behaving animals, as well as the observation that behaviour can explain substantial variance in the neural dynamics.

Gaussian process latent variable models (GPLVMs, Lawrence (2003) form a family of methods that learn smooth, non-linear tuning curves by placing Gaussian process priors on them, and performing approximate marginal log-likelihood optimization on the latent variable. Popular implementations

[1] One-hour recordings of 200 neurons (10^6 spikes) takes 1 minute to run on a CPU laptop.

[2] Code and a demo can be found at: `https://anonymous.4open.science/r/simpl/`

leave the initial condition of this optimization user-defined, making such methods compatible with the behaviour-informed initialization also used by SIMPL. However, most such models were introduced outside of the neuroscience literature; because of this, popular variants use Gaussian (instead of Poisson) emission models (Lawrence, 2003; Wang et al., 2005; Jensen et al., 2020), or do not make smoothness assumptions on the latent trajectory (Jensen et al., 2020; Lawrence, 2003). P-GPLVM, which employs Poisson emissions and a Gaussian process prior on the latent trajectory, is an exception, but its cubic scaling with time points makes it impractical for hour-long datasets. In contrast, available GPLVM implementations(Bingham et al., 2018) use inducing point approximations to achieve linear time complexity.

In particular, CEBRA (Schneider et al., 2023) learns a deterministic neural network–based mapping from spikes to latents using behaviour- or time-guided contrastive learning. Unlike most other methods, CEBRA does not natively learn a generative model nor tuning curves, which are of primary interest in our setting, instead focusing on latent visualisation and decoding, CEBRA also treats each data point independently, instead of modelling whole-trajectories; this prevents CEBRA from taking advantage of the temporal smoothness inherent in many underlying latent codes.

pi-VAE (Zhou & Wei, 2020) uses a variational autoencoder (Kingma & Welling, 2014) to infer the latent trajectories and learn tuning curves and using neural network function approximators. pi-VAE places a learnable prior, conditioned on behaviour, to the latent variable in order to obtain a model with provable identifiability properties. However, pi-VAE suffers from the same limitation as CEBRA in that it treats each data point as an i.i.d observation instead of a part of a whole trajectory.

Structured variational autoencoders (SVAEs) which extend the traditional VAE framework by incorporating additional structure into the latent space. PfLDS(Gao et al., 2016) imposes temporal structure equivalent to a linear dynamical system (LDS) by imposing block-tridiagonal struximate posterior. VIND(Hernandez et al., 2018) extends this to non-linear and spatially dependent Markovian dynamics. These methods are in spirit similar to SIMPL but

As previously discussed, the properties of large scale neural datasets suggest five desiderata on the algorithms used to analyse them. These are (1) the absence of restrictive tuning curve assumptions, (2) imposing smooth latent dynamics, (3) the presence of a spiking component (e.g. Poisson emissions), (4) the ability to exploit behaviour (including as an initial condition) and (5) scalability to large datasets. Surprisingly, none of the methods described in our literature review satisfy all five desiderata (see Table 3 in the Appendix). Here we propose SIMPL, a new method which fills this gap, while encouraging the improvement of existing methods to better handle the needs of modern neural data analysis.

3 METHOD

Here we provide a high-level description of the SIMPL algorithm. Comprehensive details, as well as a theoretical analysis linking SIMPL formally to expectation-maximization of a class of generative models, is provided in the Appendix B.

3.1 THE MODEL

SIMPL models *spike trains* of the form $\mathbf{s} \coloneqq (s_{ti})_{t=1,\ldots T}^{i=1,\ldots N}$, where s_{ti} represents the number of spikes emitted by neuron i between time $(t-1)\cdot\Delta t$ and $t\cdot\Delta t$, for some time discretization interval $\mathrm{d}t$. We denote $\mathbf{s}_t \coloneqq (s_{t1},\ldots,s_{tN})$ the vector of spike counts emitted by all neurons in the t-th time bin. SIMPL posits that such spike trains $\mathbf{s}$ are modulated by a *latent, continuously-valued, low-dimensional, time-evolving* variable $\mathbf{x} \coloneqq (\mathbf{x}_t)_{t=1,\ldots,T} \in \mathbb{R}^D$ through the following random process:

$$\begin{aligned} s_{ti} \mid \mathbf{x}_t &\sim \mathrm{Poisson}(f_i(\mathbf{x}_t)) \\ \mathbf{x}_{t+1} \mid \mathbf{x}_t &\sim \mathcal{N}(\mathbf{x}_t, \sigma_v^2\mathbf{I}), \end{aligned}$$

where $\sigma_v \coloneqq \mathrm{v}\cdot\mathrm{d}t$ and $\mathbf{x}_0 \sim \mathcal{N}(0,\sigma_0^2\mathbf{I})$. Here, v is some constant expected velocity hyperparameter. The resulting prior distribution $p(\mathbf{x}) = p(\mathbf{x}_0)\prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1})$ enforces a tunable (through v) amount of temporal smoothness in the trajectories. At each time step the latent variable $\mathbf{x}_t$ determines the instantaneous firing rate of each neuron via its intensity function f_i (hereon called its *tuning curve*, collectively denoted $\mathbf{f}$), which is unknown a priori, and which SIMPL will estimate.

Moreover, we make the common assumption that all neurons are *conditionally independent* given $\mathbf{x}_t$, i.e. $p(\mathbf{s}_t|\mathbf{x}_t) = \prod_{i=1}^{N} p(\mathbf{s}_{ti}|\mathbf{x}_t)$. Finally, we assume the latent variable $\mathbf{x}$ evolves only according to its previous state (it is Markovian), a common assumption in the neuroscience literature (see, e.g. George et al. 2021). This model has been previously studied in the literature (Smith & Brown, 2003; Macke et al., 2011), albeit using highly restrictive intensity function models, something which SIMPL avoids as discussed below.

3.2 THE SIMPL ALGORITHM

Outline We now seek an estimate of the true, unknown latent trajectory $\mathbf{x}^\star$ and tuning curves $\mathbf{f}^\star$ that led to some observed spike train, $\mathbf{s}$. SIMPL does so by iterating a two-step procedure closely related to the expectation-maximisation (EM) algorithm: first, tuning curves are fitted to an initial estimate of the latent variable (the "M-step"), which are then used to decode the latent variable (the "E-step"). This procedure is then repeated using the new latent trajectory, and so on until convergence.

The M-step In the M-step (or "fitting" step) of the e-th iteration SIMPL fits intensity functions to the current latent trajectory estimate $\mathbf{x}^{(e)}$ using kernel density estimation (KDE):

$$f_i^{(e)}(\mathbf{x}) := \frac{\sum_{t=1}^{T} s_{ti}\, k(\mathbf{x}, \mathbf{x}_t^{(e)})}{\sum_{t=1}^{T} k(\mathbf{x}, \mathbf{x}_t^{(e)})} \approx \frac{\text{\# spikes at x}}{\text{\# visits to x}} \tag{1}$$

The use of a smooth kernel allows extrapolation of the intuitive estimate on the right of Equation 1 to locations not present in the trajectory $\mathbf{x}^{(e)}$. In practice, we use a Gaussian kernel with bandwidth σ. Being a non-parameteric KDE estimator, such a tuning curve model is conceptually simple and free from the optimisation, misspecification or interpretability issues of most parametric models. It constitutes a notable departure from alternatives which use a neural networks(Zhou & Wei, 2020; Schneider et al., 2023) to model tuning curves and is particularly well suited to low-dimensional latent spaces.

The E-step In the E-step (or "decoding" step), SIMPL produces a new estimate $\mathbf{x}^{(e+1)}$ of the latent trajectory by smoothing — using the prior $p(\mathbf{x})$ — across time the (non-smooth) *maximum likelihood estimate* (MLE) $\widehat{\mathbf{x}}$ of $\mathbf{x}$, given $\mathbf{s}$ and $\mathbf{f}^{(e)}$. To do so, SIMPL computes a linear-Gaussian approximation of the conditional distribution $p(\widehat{\mathbf{x}}_t|\mathbf{x}_t) \approx \mathcal{N}(\mathbf{x}_t; \mathbf{\Sigma}_t)$. With this approximation, the variables $(\mathbf{x}, \widehat{\mathbf{x}})$ form a Linear Gaussian State Space Model, fully characterised by $\sigma_v^2 I$ (the transition noise covariance) and $\mathbf{\Sigma}_t$ (the observation noise covariance). This allows SIMPL to employ *Kalman Smoothing*, an efficient inference procedure for such models, to approximate $\mathbb{E}_{p(\mathbf{x}|\widehat{\mathbf{x}})}[\mathbf{x}]$.

Crucially, the linear-Gaussian approximation is *not* made on the spiking emissions $p(\mathbf{s}|\mathbf{x})$, which is non-Gaussian by design, but on $p(\widehat{\mathbf{x}}|\mathbf{x})$, a quantity which is provably asymptotically Gaussian in the many-neurons regime (full theoretical argument and an explicit formula for $\mathbf{\Sigma}_t$ in B.1). At a high level, SIMPL's E-step can thus be summarized as (see Fig. 1b, lower panel, for a graphical summary):

$$\begin{aligned} \widehat{\mathbf{x}}^{(e+1)} &:= \arg\max_{\mathbf{x}} \log p(\mathbf{s}|\mathbf{x}, \mathbf{f}^{(e)}) \\ \mathbf{x}^{(e+1)} &:= \mathbb{E}_{p(\mathbf{x}|\widehat{\mathbf{x}}_t^{(e+1)})}[\,\mathbf{x}\,] \approx \mathrm{KalmanSmooth}(\widehat{\mathbf{x}}^{(e+1)}; \sigma_v^2\mathbf{I}, \mathbf{\Sigma}_t) \end{aligned} \tag{2}$$

Behavioural initialization Spike trains often come alongside behavioural recordings which are thought to be closely related to the latent variable $\mathbf{x}$. SIMPL leverages this by setting $\mathbf{x}^{(0)}$, the initial decoded latent trajectory, to measured behaviour. We posit that such a *behavioural initialization* will place the first iterate of SIMPL in the vicinity of the true trajectory and tuning curves. This, in turn, faciliates the search for a good model which favours the true latent and tuning curves $(\mathbf{x}^\star, \mathbf{f}^\star)$ over alternative pairs $(\phi(\mathbf{x}^\star), \mathbf{f}^\star \circ \phi^{-1})$ whose latent space is *warped* by some invertible map ϕ, and which would explain the data equally well (i.e. solution pairs which are *isomorphic* to the ground truth). Through ablation studies, we confirm the beneficial effects of this behavioural initialization in the experiments section (see Fig. 2 and 3).

All in all, SIMPL is interpretable and closely matches common practice in neuroscience; moreover, it can be formally related to a generalized version of the EM-algorithm, for which theoretical guarantees may be obtained under suitable assumptions. We describe in detail the theoretical arguments justifying the validity of SIMPL as well as its connection to EM in the appendix.

4 RESULTS

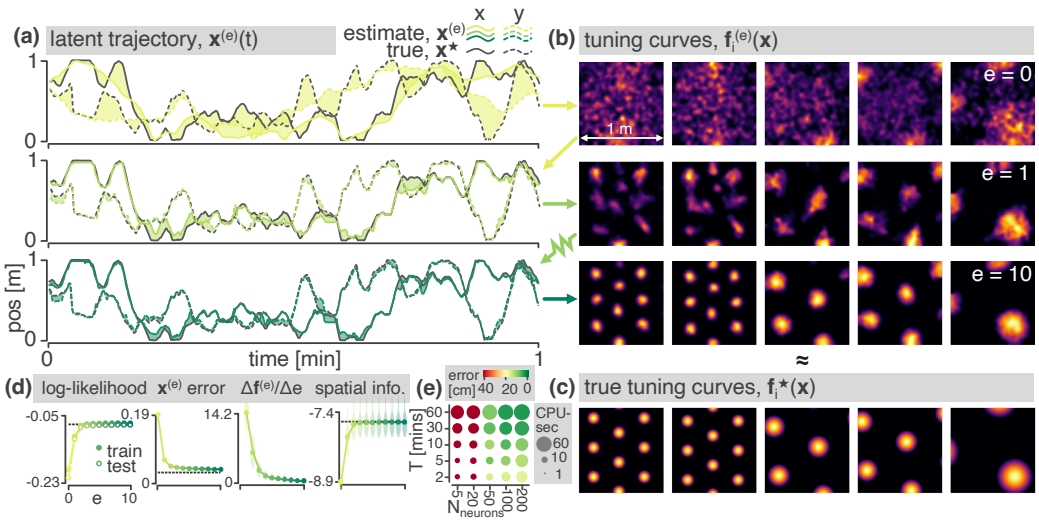

Figure 2: Results on a synthetic 2D grid cell dataset. An artificial agent locomotes a 1 m square environment for 1 hour (Δt = 0.1 s). Spikes are generated from N=225 artificial grid cells. **(a)** Estimated latent trajectories shown for epochs 0, 1 and 10. x and y positions are denoted by dotted and dashed lines respectively. Initial conditions are generated from the true latent (black) by the addition of smooth continuous Gaussian noise. **(b)** Tuning curve estimates for 5 examplar grid cells at epochs 0, 1 and 10. **(c)** Ground truth tuning curves. **(d)** Performance metrics: *Left:* log-likelihood of the train and test spikes (averaged per time step, dotted line shows ceiling performance on a model initialised with the true latent). *Middle-left:* Euclidean distance between the true and estimated latent trajectories (averaged per time step). *Middle-right:* Epoch-to-epoch change in the tuning curves. *Right:* Cell spatial information. Violin plots, where shown, display distributions across all 225 neurons. **(e)** A sweep over the number of cells and the duration of the trajectory.

4.1 CONTINUOUS SYNTHETIC DATA: 2D GRID CELLS

Next we tested SIMPL on a realistic navigational task by generating a large artificial dataset of spikes from a population of $N = 225$ 2D grid cells — a type of neuron commonly found in the medial entorhinal cortex which activate on the vertices of a regular hexagonal grid (Hafting et al., 2005) — in a 1 m square environment. Grid cell tuning curves, $\mathbf{f}^\star$, were modelled as the thresholded sum of three planar waves at 0°, 60° and 120° to some offset direction (a commonly used model within the computational neuroscience literature (George et al., 2024a)) and, as observed in the brain, cells were arranged into three discrete modules, 75 cells per modules, of increasing grid scale from 0.3–0.8 m (Fig. 2c). Each cell had a maximum firing rate of 10 Hz. A latent trajectory, $\mathbf{x}^\star$, was then generated by simulating an agent moving around the environment for 1 hour under a smooth continuous random motion model replicating rodent foraging behaviour. Data was sampled at a rate of 10 Hz giving a total of $T = 36,000$ time bins ($\sim$ 800,000 spikes). All data was generated using the `RatInABox` package (George et al., 2024a).

$$\mathbf{x}^\star \sim \text{Smooth-continuous-random-walk} \quad \text{and} \quad s_{ti}|\mathbf{x}_t^\star \sim \text{Poi}(f_i^{\text{GC}}(\mathbf{x}_t^\star)) \tag{3}$$

The initial latent trajectory, $\mathbf{x}^{(0)}$, was generated by adding smooth Gaussian noise to the latent $\mathbf{x}$ such that, on average, the true latent and initial condition differed by 20 cm (Fig. 2a, top panel). This discrepancy, modelling the agent's own uncertainty in their position and/or a measurement error, was sufficient to obscure almost all structure from the initial grid cell tuning curves $\mathbf{f}^{(0)}(\mathbf{x})$ (Fig. 2b, top).

To assess performance we track to the log-likelihood of train and test spike (see Appendix D for details of how we partition the dataset) . We also calculate the Euclidean distance between the true and latent trajectory (Fig. 2d, middle-left), $T^{-1}\sum_t \|\mathbf{x}^{(e)}(t) - \mathbf{x}_t\|_2$, the epoch-to-epoch change in the tuning curves (Fig. 2d, middle-right) and the entropy (hereon called "spatial info", Fig. 2d, right) of the normalized tuning curves as a measure of how spatially informative they are.

SIMPL was then run for 10 epochs (total compute time 39.8 CPU-secs on a consumer grade laptop). The true latent trajectory and receptive fields were recovered almost perfectly and the log-likelihood of both train and test spikes rapidly approached the ceiling performance with negligible overfitting. As expected SIMPL performs better on larger datasets, Fig. 2e however, our testing shows performance is still good even with substantially small datasets (e.g. 50 cells for a duration of 5 minutes). Performance drops off sharply for datasets with less than 20 cells.

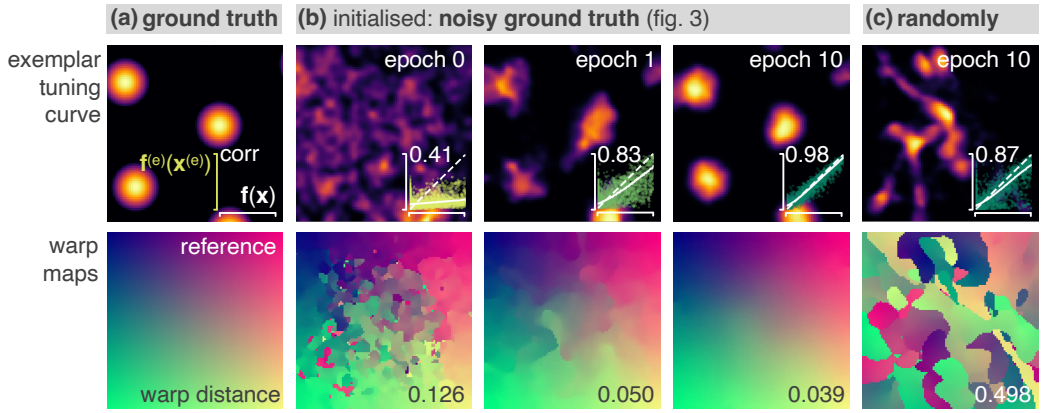

Figure 3: Latent manifold analysis: **(Top)** Examplar tuning curve in the ground truth latent space **(a)**, the latent space discovered by behaviourally-initialised-SIMPL after 0, 1 and 10 epochs **(b)** and the latent space discovered by SIMPL initialised with a random latent trajectory **(c)**. Inset scatter plots show the true and predicted firing rates of all neurons across all times as well as their correlation values ("accurate" models have higher correlations). **(Bottom)** Visualizations of the warp functions mapping each latent space to the "closest" location in ground truth as measured by the distance between the tuning curves population vectors.

Influence of behavioural initializations on performance Latent variable models trained with EM can experience two issues that usually complicate the scientific interpretability of their results. The first concerns the *quality* of the solution; does the algorithm converge on a good model of the data which predicts the spikes well? The second issue concerns *identifiability*; even if the recovered latent trajectory and tuning curves $(\mathbf{f}^{(e)}, \mathbf{x}^{(e)})$ are of high quality, they may differ from the true ones $(\mathbf{f}^\star, \mathbf{x}^\star)$ by some invertible "warp" ϕ in a way that does not affect the overall goodness of fit of the model. While SIMPL is a latent variable model, we show that behavioural initialization drastically minimizes the severity of both of these issues.

To do so, we first assess the absolute goodness–of–fit of SIMPL by computing the correlation between the estimated instantaneous firing rates $f^{(e)}(\mathbf{x}_t^{(e)})$ (a quantity invariant to warping) and the true ones. Our analysis shows that SIMPL converges to a highly accurate model (r=0.98) under behavioural initialization, but to a less accurate (though still quite accurate) one ($r = 0.87$) when initialised with a random latent trajectory which is uncorrelated with behaviour. Second, we estimate, quantify and visualize the warp map ϕ between SIMPL's estimates $(\mathbf{f}^{(e)}, \mathbf{x}^{(e)})$ and the ground truth $(\mathbf{f}^\star, \mathbf{x}^\star)$. We obtain this estimate by finding a mapping from the discovered latent space to the true latent space which minimizes the L2 difference between the tuning curves ($\phi(\mathbf{x}) = \arg\min_{\mathbf{y}} \|\mathbf{f}^\star(\mathbf{y}) - \mathbf{f}^{(e)}(\mathbf{x})\|_2$). We then quantify the "warpness" of this mapping by calculating the average distance between $\mathbf{x}$ and $\phi(\mathbf{x})$ across the environment, normalized by its characteristic length scale (1 m). This warp distance should be 0 for total un-warped models and $\mathcal{O}(1)$ for

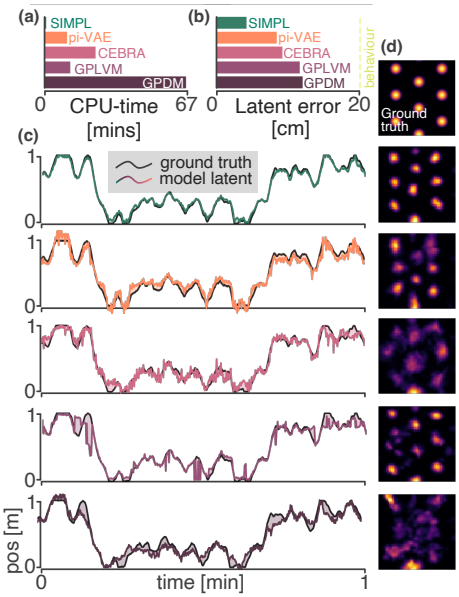

Figure 4: Comparison between SIMPL and CEBRA, GPLVM and pi-VAE.

heavy warps. We find that in addition to perfectly fitting the data, the solution found by SIMPL under behavioural initialization is minimally warped (warp dist = 0.050). In contrast, the good (but imperfect) solution found by SIMPL under random initialization is heavily warped (warp dist. = 0.498) in a fragmented manner. These results are shown in Fig. 3 and strongly motivate the use of behavioural initializations in latent variable models as an effective mean to encourage convergence towards latent spaces which are both accurate and un-warped with respect to the ground truth.

Benchmarking SIMPL against existing techniques We compared SIMPL to four popular methods for latent variable extraction: pi-VAE(Zhou & Wei, 2020), CEBRA(Schneider et al., 2023), GPLVM(Lawrence, 2003) and GPDM(Wang et al., 2005). Crucially, none of these methods make restrictive tuning curve assumptions and all can use behaviour to guide latent discovery. To this end, we initialise the latent variable estimates of GPLVM and GPDM to behaviour as is done for SIMPL (pi-VAE and CEBRA handle behaviour natively). All models were trained for their default number of training iterations. After training we aligned the latents to behaviour and visualise the them on top of the ground truth in Fig. 4. SIMPL recovered the true latent better than all other models, achieving a final error of 4.2 cm, half that of the next best performing model (pi-VAE, 8.4 cm, Fig. 4b). We posit that pi-VAE, CEBRA and GPLVM may suffer from the lack of a dynamical systems component in their generative models while GPDM may suffer from the data-subsampling required to cap the training time to less than two-hours. SIMPL converged in 40 seconds, over 15 times quicker than the next fastest (pi-VAE, 10.4 minutes, Fig. 4a). Except for GPDM, which required a GPU, all techniques were run and timed on a CPU. Only SIMPL produced sharp well-defined grid fields (Fig. 4d) close to the ground truth.

4.2 HIPPOCAMPAL PLACE CELL DATA

Next, we test SIMPL on a neural dataset from $N = 226$ hippocampal neurons recorded from a rat as it foraged in a large 3.5 m by 2.5 m environment for 2 hours (full details can be found in Tanni et al. 2022). The data was binned at 5 Hz (d$t = 0.2s$ giving $T = 36,000$ data samples, total $\sim$ 700,000 spikes). Place cells are a type of neuron commonly found in the hippocampus which activate when an animal is in a specific location in space (its "place field") and, like grid cells, are thought to be a key component of the brain's navigational system (O'Keefe, 1978). In large environments place cells are known to exhibit tuning curves with multiple place fields (Park et al., 2011).

We initialised SIMPL using the measured position of the animal and optimised for 10 epochs. The log-likelihood of test and train spikes increased, Fig. 5b, converging after approximately 4 epochs (compute time 41.2 CPU-secs). We then analysed statistics of the tuning curves before and after optimisation.: Tuning curves were visibly sharper after optimization, Fig. 5a; diffuse place fields — regions of elevated activity in a neurons tuning curve identified using a automatic procee, see Appendix E.5 — shrunk (e.g. see the third exemplar tuning curve, Fig. 5a) or split into multiple, smaller fields (second exemplar). Occasionally, new place fields appeared (fourth exemplar) or multiple place fields merged into a single larger field (fifth exemplar). Statistically, tuning curves had significantly more individual place fields (+19%, mean 1.14$\rightarrow$1.41 per cell, $p = 0.0035$ Mann Whitney U tests), substantially higher maximum firing rates (+45%, median 4.2$\rightarrow$6.1 Hz, $p = 9.8 \times 10^{-7}$) and were more spatially informative ($p = 0.038$). Individual place fields were substantially smaller (-25%, median 0.59$\rightarrow$0.44 m^2) and rounder (+8%, median 0.63$\rightarrow$0.68, $p = 0.0037$).

To ensure that these changes were not an artefact of the SIMPL algorithm we generated a control dataset by resampling spikes from the behaviour-fitted tuning curves, $\mathbf{s}_{\text{control}} \sim p(\cdot|\mathbf{x}^{(0)}, \mathbf{f}^{(0)})$. Control spikes thus had very similar temporal statistics and identical tuning curves to those in the original dataset but, crucially, were generated from a known ground truth model exactly equal to the initialization. Thus, any changes to the control spike tuning curves under SIMPL optimization can be considered artefactual and not fundamental to the underlying neural data. Notably, a measured effects were statistically insignificant (except for a slight *increase* in field area) and pointed in the *opposite* direction to those observed in the real data (except for roundness) (Fig. 5c). This control provides strong evidence that the changes observed in the real data are genuine and reflect the true nature of neural tuning curves in the brain.

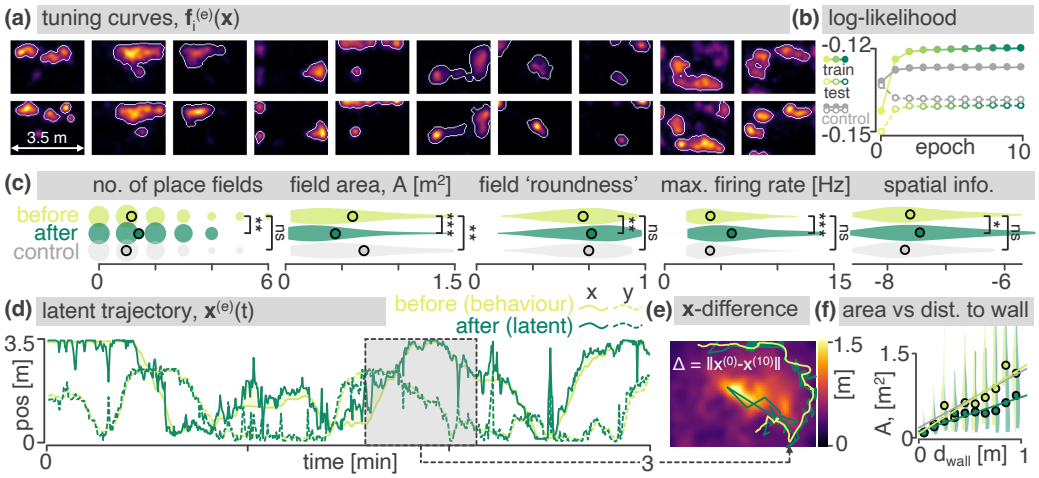

Figure 5: Results on a hippocampal place cell dataset collected by Tanni et al. (2022). **(a)** Exemplar tuning curves before and after optimization. Automatically identified place field boundaries shown in white. **(b)** Log-likelihood of test and train spikes. Equivalent results for a control model — fitted with spikes resampled from the behavioural place fields, $\mathbf{s}_{\text{control}} \sim p(\cdot|\mathbf{x}^{(0)}, \mathbf{f}^{(0)})$ — shown in grey. **(c)** Place field (before, after, control-after) analysis. Violin plots show the distributions over all place fields / place cells. **(d)** The final latent trajectory estimated from SIMPL (green) overlaid on top of the behaviour (used as initial conditions) (yellow). x and y coordinates shown with dotted and dashed lines respectively. **(e)** Behavioural discrepancy map: the average discrepancy $\|\mathbf{x}_t^{(0)} - \mathbf{x}_t^{(10)}\|_2$ as a function of the optimised latent $\mathbf{x}^{(10)}$. Overlaid is a snippet of the behavioural vs optimised true latent trajectory. **(f)** Median place field sizes, and distributions, as a function of the distance to the nearest.

After 10 iterations of optimization the latent trajectory $\mathbf{x}^{(10)}$ remained highly correlated with the behaviour ($R^2 = 0.86$, fig. 5d) occasionally diverging for short period as the latent "jumped" to and from a new location, as if the animal was mentally teleporting itself (one such "jump" is visualized in Fig. 5e). The close correspondence between the optimised latent and the behaviour allows us to directly compare when, and where, they diverge. We calculated the discrepancy between the optimised latent and the behaviour at each time point, $\|\mathbf{x}_t^{(0)} - \mathbf{x}_t^{(10)}\|_2$, and visualized this as a heat map overlaid onto the latent space (Fig. 5e). Discrepancy was minimal around the edges of the environment and peaked near the centre, consistent with the hypothesis that sensory input is less reliable in the centre of the environment (where there are fewer visual and tactile cues) to guide self-localisation resulting in a larger average discrepancy between the optimised latent and the behaviour.

Tanni et al. (2022) found that place field size increased with distance from the nearest wall in the environment. Our observation — that latent-behaviour discrepancy is highest in the centre of the environment — suggests a possible explanation: place fields in the centre of the environment are not larger but *appear* larger because they are distorted and blurred by the discrepancy which is largest near the centre of the environment. To test this we binned place fields according to their distance to the nearest wall (measured with respect to the place fields centre of mass) and plotted the median field size against distance (Fig. 5f). Optimized place fields, much like behavioural place fields, were the smallest near the walls and grew with distance (replicating Tanni et al. (2022)), but this correspondence broke down around ~ 0.5 m after which the optimised size distribution flattened off (something not observed in the control). A majority of the shrink in place field size thus came from larger place fields near the centre of the environment not the smaller ones near the walls. This result suggests that a substantial fraction of the increased size of place fields away from walls is not a fundamental feature of the neural tuning curves themselves but can be attributed to a behaviour-induced distortion in the tuning curves, an artefact which can be corrected for by optimising the latent with SIMPL.

5 DISCUSSION

We introduced SIMPL, a tool for optimizing tuning curves and latent trajectories using a technique which refines estimates obtained from behaviour. It hinges on two well-established sub-routines — fitting and decoding — which are widely used by both experimentalists and theorists for analysing neural data. By presenting SIMPL as an iterative application of these techniques, we aim to make latent variable modelling more accessible to the neuroscience community.

Furthermore, we see SIMPL as a specific instance of a broader class of latent optimization algorithms. In principle *any* arbitrary curve fitting procedure and *any* arbitrary decoder could be coupled into a candidate algorithm for optimizing latents from neural data. Our specific design choices, while attractive due to their conceptual simplicity, will also come with limitations. For example, we predict KDE won't scale well to very high dimensional latent spaces (Györfi et al., 2006). In these instances users could consider substituting this component with a parametric model, e.g. a neural network, which are known to perform better in high dimensions (Bach, 2017), potentially at the cost of compute time.

Our synthetic analysis focussed on settings where behaviour and the true latent differed only in an unbiased manner. It would be interesting to determine if SIMPL's strong performance extends to more complex perturbations. In the brain, fast, non-local and asymmetric perturbations are common; for instance "replay"(Carr et al., 2011) where the latent jumps to another location in the environment. Likewise, during theta sweeps (Maurer et al., 2006), the encoded latent moves away from the agent. This forward-biased discrepancy could theoretically induce a backward-biased skew in behavioural place fields, even if the true tuning curves remain unskewed. If this is the case, latent dynamics —— and tools like SIMPL for extracting them — could help reinterpret the predictive nature of place field tuning curves (Stachenfeld et al., 2017; Fang et al., 2023; Bono et al., 2023; George et al., 2023), similar to how latent optimization reduced the asymmetry in place field sizes further from walls (Fig. 5f).

REFERENCES

Afsheen Afshar, Gopal Santhanam, M Yu Byron, Stephen I Ryu, Maneesh Sahani, and Krishna V Shenoy. Single-trial neural correlates of arm movement preparation. *Neuron*, 2011.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, (19):1–53, 2017.

Patrick Billingsley. Statistical methods in markov chains. *The annals of mathematical statistics*, 1961.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Rafal Bogacz, Eric Brown, Jeff Moehlis, Philip Holmes, and Jonathan D Cohen. The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological review*, 2006.

Jacopo Bono, Sara Zannone, Victor Pedrosa, and Claudia Clopath. Learning predictive cognitive maps with spiking neurons during behavior and replays. *eLife*, pp. e80671, 2023.

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

Ralph A Bradley and John J Gart. The asymptotic properties of ml estimators when sampling from associated populations. *Biometrika*, 1962.

Federico Carnevale, Victor de Lafuente, Ranulfo Romo, Omri Barak, and Néstor Parga. Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty. *Neuron*, 2015.

Margaret F Carr, Shantanu P Jadhav, and Loren M Frank. Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature neuroscience*, 2011.

Raeed H Chowdhury, Joshua I Glaser, and Lee E Miller. Area 2 of primary somatosensory cortex encodes kinematics of the whole arm. *eLife*, pp. e48198, 2020.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 2014.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 1977.

Eric L Denovellis, Anna K Gillespie, Michael E Coulter, Marielena Sosa, Jason E Chung, Uri T Eden, and Loren M Frank. Hippocampal replay of experience at real-world speeds. *eLife*, pp. e64505, 2021.

Zhe Dong, Bryan Seybold, Kevin Murphy, and Hung Bui. Collapsed amortized variational inference for switching nonlinear dynamical systems. In *International Conference on Machine Learning*, pp. 2638–2647. PMLR, 2020.

Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *International conference on machine learning*. PMLR, 2019.

Ching Fang, Dmitriy Aronov, LF Abbott, and Emily L Mackevicius. Neural learning rules for generating flexible predictions and computing the successor representation. *eLife*, pp. e80680, 2023.

André A Fenton and Robert U Muller. Place cell discharge is extremely variable during individual passes of the rat through the firing field. *Proceedings of the National Academy of Sciences*, 1998.

Ronald Aylmer Fisher. Theory of statistical estimation. In *Mathematical proceedings of the Cambridge philosophical society*. Cambridge University Press, 1925.

Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. *Advances in Neural Information Processing Systems*, 2016.

Dileep George, Rajeev V Rikhye, Nishad Gothoskar, J Swaroop Guntupalli, Antoine Dedieu, and Miguel Lázaro-Gredilla. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature communications*, 2021.

Tom M George, William de Cothi, Kimberly L Stachenfeld, and Caswell Barry. Rapid learning of predictive maps with stdp and theta phase precession. *eLife*, pp. e80663, 2023.

Tom M George, Mehul Rastogi, William de Cothi, Claudia Clopath, Kimberly Stachenfeld, and Caswell Barry. Ratinabox, a toolkit for modelling locomotion and neuronal activity in continuous environments. *eLife*, 2024a.

Tom M George, Kimberly L Stachenfeld, Caswell Barry, Claudia Clopath, and Tomoki Fukai. A generative model of the hippocampal formation trained with theta driven local learning rules. *Advances in Neural Information Processing Systems*, 2024b.

Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

Joshua I Glaser, Ari S Benjamin, Raeed H Chowdhury, Matthew G Perich, Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *eNeuro*, (4), 2020.

Rabia Gondur, Usama Bin Sikandar, Evan Schaffer, Mikio Christian Aoi, and Stephen L Keeley. Multi-modal gaussian process variational autoencoders for neural and behavioral data. *arXiv preprint arXiv:2310.03111*, 2023.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 2005.

Christopher D Harvey, Philip Coen, and David W Tank. Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, 2012.

Daniel Hernandez, Antonio Khalil Moretti, Ziqiang Wei, Shreya Saxena, John Cunningham, and Liam Paninski. Nonlinear evolution via spatially-dependent linear dynamics for electrophysiology and calcium data. *arXiv preprint arXiv:1811.02459*, 2018.

Pierre Hodara, Nathalie Krell, and Eva Löcherbach. Non-parametric estimation of the spiking rate in systems of interacting neurons. *Statistical Inference for Stochastic Processes*, 2018.

Øyvind Arne Høydal, Emilie Ranheim Skytøen, Sebastian Ola Andersson, May-Britt Moser, and Edvard I Moser. Object-vector coding in the medial entorhinal cortex. *Nature*, 2019.

Cole Hurwitz, Akash Srivastava, Kai Xu, Justin Jude, Matthew Perich, Lee Miller, and Matthias Hennig. Targeted neural dynamical modeling. *Advances in Neural Information Processing Systems*, 34:29379–29392, 2021.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 1999.

Kristopher Jensen, Ta-Chu Kao, Marco Tripodi, and Guillaume Hennequin. Manifold gplvms for discovering non-euclidean latent structure in neural data. *Advances in Neural Information Processing Systems*, 2020.

James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Cağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 2017.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Timothy D Kim, Thomas Z Luo, Jonathan W Pillow, and Carlos D Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In *International Conference on Machine Learning*, pp. 5551–5561. PMLR, 2021.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

Dmitry Kobak, Wieland Brendel, Christos Constantinidis, Claudia E Feierstein, Adam Kepecs, Zachary F Mainen, Xue-Lian Qi, Ranulfo Romo, Naoshige Uchida, and Christian K Machens. Demixed principal component analysis of neural population data. *eLife*, 2016.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Neil Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16, 2003.

Colin Lever, Stephen Burton, Ali Jeewajee, John O'Keefe, and Neil Burgess. Boundary vector cells in the subiculum of the hippocampal formation. *Journal of Neuroscience*, 2009.

Itay Lieder, Vincent Adam, Or Frenkel, Sagi Jaffe-Dax, Maneesh Sahani, and Merav Ahissar. Perceptual bias reveals slow-updating in autism and fast-forgetting in dyslexia. *Nature neuroscience*, 2019.

Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, 2019.

Ryan J Low, Sam Lewallen, Dmitriy Aronov, Rhino Nevers, and David W Tank. Probing variability in a cognitive map using manifold inference from neural dynamics. *BioRxiv*, 2018.

Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in Neural Information Processing Systems*, 2011.

Emily L Mackevicius, Andrew H Bahle, Alex H Williams, Shijie Gu, Natalia I Denisenko, Mark S Goldman, and Michale S Fee. Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience. *eLife*, 2019.

Andrew Zammit Mangion, Ke Yuan, Visakan Kadirkamanathan, Mahesan Niranjan, and Guido Sanguinetti. Online variational inference for state-space models with point-process observations. *Neural Computation*, 2011.

Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 2013.

Andrew P Maurer, Stephen L Cowen, Sara N Burke, Carol A Barnes, and Bruce L McNaughton. Organization of hippocampal cell assemblies based on theta phase precession. *Hippocampus*, 2006.

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

Bruce L McNaughton, Carol A Barnes, and JJEBR O'Keefe. The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely-moving rats. *Experimental brain research*, 1983.

Mayank R Mehta, Carol A Barnes, and Bruce L McNaughton. Experience-dependent, asymmetric expansion of hippocampal place fields. *Proceedings of the National Academy of Sciences*, 1997.

Robert U Muller and John L Kubie. The firing of hippocampal place cells predicts the future position of freely moving rats. *Journal of Neuroscience*, 1989.

Hooram Nam. Poisson extension of gaussian process factor analysis for modeling spiking neural populations. *Master's thesis, Department of Neural Computation and Behaviour, Max Planck Institute for Biological Cybernetics, Tübingen.[Google Scholar]*, 2015.

J O'Keefe. The hippocampus as a cognitive map, 1978.

John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

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

EunHye Park, Dino Dvorak, and André A Fenton. Ensemble place codes in hippocampus: Ca1, ca3, and dentate gyrus place cells have multiple place fields in large environments. *PloS one*, 2011.

Mijung Park, Gergo Bohner, and Jakob H Macke. Unlocking neural population non-stationarities using hierarchical dynamics models. *Advances in Neural Information Processing Systems*, 2015.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

Michael L Platt and Paul W Glimcher. Neural correlates of decision variables in parietal cortex. *Nature*, 1999.

Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 1965.

Honi Sanders, César Rennó-Costa, Marco Idiart, and John Lisman. Grid cells and place cells: an integrated view of their navigational and memory function. *Trends in neurosciences*, 2015.

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

Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 2023.

Anne C Smith and Emery N Brown. Estimating a state-space model from point process observations. *Neural Computation*, 2003.

Hugo J Spiers and Eleanor A Maguire. Thoughts, behaviour, and brain dynamics during navigation in the real world. *Neuroimage*, 2006.

Larry R Squire, Anna S van der Horst, Susan GR McDuff, Jennifer C Frascino, Ramona O Hopkins, and Kristin N Mauldin. Role of the hippocampus in remembering the past and imagining the future. *Proceedings of the National Academy of Sciences*, 2010.

Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 2017.

Sander Tanni, William De Cothi, and Caswell Barry. State transitions in the statistically stable place cell population correspond to rate of perceptual change. *Current Biology*, 2022.

Jeffrey S Taube, Robert U Muller, and James B Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *Journal of Neuroscience*, 1990.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

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

James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 2020.

Alex Williams, Anthony Degleris, Yixin Wang, and Scott Linderman. Point process models for sequence detection in high-dimensional neural spike trains. *Advances in Neural Information Processing Systems*, 2020.

Brian A Wilt, Laurie D Burns, Eric Tatt Wei Ho, Kunal K Ghosh, Eran A Mukamel, and Mark J Schnitzer. Advances in light microscopy for neuroscience. *Annual review of neuroscience*, 2009.

Anqi Wu, Nicholas A Roy, Stephen Keeley, and Jonathan W Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. *Advances in Neural Information Processing Systems*, 2017.

Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Expectation propagation for inference in nonlinear dynamical models with poisson observations. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. IEEE, 2006.

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

Byron M Yu, John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. Neural decoding of movements: From linear to nonlinear trajectory models. In *Neural Information Processing: 14th International Conference, ICONIP 2007, Kitakyushu, Japan, November 13-16, 2007, Revised Selected Papers, Part I 14*. Springer, 2008b.

Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Computation*, 2017.

Ding Zhou and Xue-Xin Wei. Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-vae. *Advances in Neural Information Processing Systems*, 2020.

Petr Znamenskiy and Anthony M Zador. Corticostriatal neurons in auditory cortex drive decisions during auditory discrimination. *Nature*, 2013.

David Zoltowski, Jonathan Pillow, and Scott Linderman. A general recurrent state space framework for modeling neural dynamics during decision-making. In *International Conference on Machine Learning*, pp. 11680–11691. PMLR, 2020.

Supplementary Material for "SIMPL: Scalable and hassle-free optimisation of neural representations from behaviour"

A BACKGROUND

A.1 EXPECTATION MAXIMIZATION

Expectation Maximization (EM, Dempster et al. 1977) is a widely used paradigm to perform statistical estimation in latent variable models. The goal of EM is to maximise the *Free Energy*, a lower bound on the log-likelihood $\log p(\mathbf{s}; \mathbf{f})$ of the data, given by (following the notations of Section 3.1):

$$\mathcal{F}(\mathbf{f}, q) := \mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})] - \mathbb{E}_{q(\mathbf{x})}[\log q(\mathbf{x})] \leq \log p(\mathbf{s}; \mathbf{f}),$$

where q is some probability distribution on the latent variable $\mathbf{x}$. Importantly, for a given set of intensity functions $\mathbf{f}$, $\mathcal{F}$ is maximised, and the lower bound becomes "tight", at $q^\star := p(\mathbf{x}|\mathbf{s}\,; \mathbf{f})$, i.e. the posterior distribution of the latent variable given the $\mathbf{s}$ and $\mathbf{f}$. Moreover, for a fixed q, the only $\mathbf{f}$-dependent term in $\mathcal{F}$ is $\mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})]$. To maximise $\mathcal{F}(\mathbf{f}, q)$ — and thus also increase the log-likelihood — EM produces a sequence $(\mathbf{f}^{[e]})_{e\geq 0}$ of parameters $\mathbf{f}^{[e]}$ by invoking, at each step e and given $\mathbf{f}^{[e-1]}$, two well known subroutines:

- **E-step**: Define $q^{[e]} := p(\mathbf{x}|\mathbf{s}\,; \mathbf{f}^{[e-1]})$; compute $\mathcal{F} \longmapsto \mathbb{E}_{q^{[e]}}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})]$
- **M-step**: Compute $\mathbf{f}^{[e]} := \arg\max_{\mathbf{f}} \mathcal{F}(\mathbf{f}, q^{[e]}) = \arg\max_{\mathbf{f}} \mathbb{E}_{q^{[e]}}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})]$

with the property that $\log p(\mathbf{s}; \mathbf{f}^{[e]}) \geq \log p(\mathbf{s}; \mathbf{f}^{[e-1]})$ for all e, grounding the use of EM to maximise the likelihood of the data. As the E-step computes specific posterior expectations, a tractable E-step often implies the ability to compute in particular posterior means and variances, the most valuable expectations in the context of decoding the latent variable from behaviour. Thus, in the context of neural data, EM offers a framework to both estimate intensity functions via maximum likelihood, and to 'decode' the variable encoded by the neurons, here by taking the mean of the posterior.

Finally, note that while the E-step writes an expectation under the full posterior $q^{[e]} := p(\mathbf{x}|\mathbf{s}, \mathbf{f}^{[e-1]})$, only specific marginals of this posterior may actually be needed depending on the structure of the joint distribution, as further discussed in Section A.2.

Impossibility of Exact EM for Gaussian-Modulated Poisson Processes The E-step of the EM algorithm requires computing a function *defined* as an expectation w.r.t $p(\mathbf{x}|\mathbf{s}\,; \mathbf{f}^{[e-1]})$. In the case of Hidden Markov Models, such expectations are intractable to compute in closed form, unless the latent variable $\mathbf{x}$ is discrete, or both the transition and the emission probabilities are Gaussian (with mean and variance depending linearly on $\mathbf{x}$, (Rauch et al., 1965)). In particular, exact inference in the model described in Section 3.1 is impossible because the emission probabilities are Poisson with mean given by a non-linear function of $\mathbf{x}$ via each neurons tuning curve.

In order to perform statistical inference for our spike train model, SIMPL runs an approximation of Exact EM, which we detail below. At a high level the goal is to convert the non-linear, non-Gaussian spiking observations, into a variable which is linear and Gaussian with resepct to the latent, thus can be solved using a Kalman smoother.

A.2 LINEAR GAUSSIAN STATE SPACE MODELS AND KALMAN SMOOTHING

Linear Gaussian State Space Models (LGSSM) are dynamical systems of the form:

$$\begin{aligned} \mathbf{z}_{t+1} &= F_t \mathbf{z}_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0_d, Q_t) \\ \mathbf{x}_t &= H_t \mathbf{z}_t + \delta_t, \quad \delta_t \sim \mathcal{N}(0_m, R_t). \end{aligned} \tag{4}$$

where $z \in \mathbb{R}^d$, $x \in \mathbb{R}^m$, $F_t, Q_t \in \mathbb{R}^{d\times d}$, $H_t \in \mathbb{R}^{p\times d}$ and $R_t \in \mathbb{R}^{m\times m}$. LGSSMs can be used as latent variable models given some observed data $\mathbf{x}$, where $\mathbf{z}$ is treated as a latent variable. While these models are limited in their expressiveness, their benefits are that inference (and in setting, "E-steps") can be done very efficiently: not only is the posterior $p(\mathbf{z}_1, \ldots, \mathbf{z}_T|\mathbf{x}_1, \ldots, \mathbf{x}_T)$

a Gaussian distribution (of dimension Td), but all of its marginals and pairwise marginals $p(\mathbf{z}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T), p(\mathbf{z}_t, \mathbf{z}_{t+1}|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ (crucially, the only distributions needed for learning the parameters of LGSSM via EM) can be computed jointly in $\mathcal{O}(T)$ time using a technique known as Kalman Smoothing (Kalman, 1960; Rauch et al., 1965). Such a scaling contrasts with naive binning-based alternatives for (approximate) inference in continuous, non-Gaussian State Space Models, which require maintaining an estimate of each bin — a vector of size n (no. bins) where n grows *exponentially* with the dimension of the latent space, as used in e.g. Denovellis et al. 2021. Instead, for LGSSMs, the Gaussianity means only the mean and covariance of the marginal posterior distributions — of size d and d^2 respectively — need to be stored. This is not memory intensive and, perhaps more importantly, the Kalman Filter proceeds to compute them in a combined $\mathcal{O}(T)$ time. In our experiments, we found the cost of the Kalman Filter to be negligible relative to the KDE evaluations which are the main computational bottleneck of SIMPL.

B SIMPL AS AN APPROXIMATE EM ALGORITHM

B.1 MLE-BACKED APPROXIMATE E-STEP

Instead of $q^{[e]} = p(\mathbf{x}|\mathbf{s}\,;\mathbf{f}^{[e-1]})$, SIMPL computes an approximation $\widehat{q}^{[e]}$ to $q^{[e]}$, allowing for both statistical estimation and uncertainty-aware trajectory decoding. As a first step towards obtaining $\widehat{q}^{[e]}$, SIMPL first performs Maximum Likelihood Estimation (MLE) on the latent trajectory $\mathbf{x}$. Instead of returning a posterior on $\mathbf{x}$, MLE returns a point estimate of the *true* trajectory that led to the observed spike train $\mathbf{s}$. In particular, MLE does not use the prior knowledge encoded by $p(\mathbf{x})$. The MLE $\widehat{\mathbf{x}}$ of $\mathbf{x}$ given $\mathbf{s}$ is given by:

$$\widehat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p(\mathbf{s}|\mathbf{x}\,;\mathbf{f}^{[e-1]}) = \arg\max_{\mathbf{x}} \sum_{t=1}^{T}\sum_{i=1}^{N} \log p(s_{ti}|\mathbf{x}_t\,;\mathbf{f}^{[e-1]})$$

$$\implies \widehat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t} \sum_{i=1}^{N} \log p(s_{ti}|\mathbf{x}_t\,;\mathbf{f}^{[e-1]}).$$

The second equality follows from the conditional independence structure of the HMM. This maximisation problem can be solved independently for each t, yielding the formula for $\widehat{\mathbf{x}}_t$ given by the third equality. As a function of $\mathbf{s}$, the MLE $\widehat{\mathbf{x}}$ is itself a random variable. In the many neurons limit, under certain regularity assumptions, the distribution of this random variable converges to a Gaussian, a fact known as *asymptotic normality*. We restate a formal statement of this result in the case of independent, but non identically distributed observations [3] originally established in Bradley & Gart (1962), and reformulated using the notations of the model at hand. For simplicity, we will consider the case where only P distinct intensity functions $\mathbf{f}_1, \ldots, \mathbf{f}_P$ exist, although versions of this result exist without this assumption.

Theorem B.1 (Asymptotic Normality of the MLE). *Let $\mathbf{x}_t^\star \in \mathbb{R}^d$. Let $\mathbf{s} = (s_{1t}, \ldots, s_{Nt})$ be independent random variables with probability densities $p(s_{ti}|\mathbf{x}_t^\star; \mathbf{f}_{t(i)})$, where $t(i) \in \{1, \ldots, P\}$ is the index of the intensity function $f_{t(i)}$ that generated the spike train s_{ti}. For $p \in 1, \ldots, P$, denote n_p the number of times the intensity function f_p appeared in the sequence $\mathbf{f}_{t(i)}$. Assume that the MLE $\widehat{\mathbf{x}}_t$ exists and it is unique. Then, under mild regularity conditions, we have:*

$$\sqrt{N}\left(\widehat{\mathbf{x}}_t - \mathbf{x}_t^\star\right) \xrightarrow[N\to\infty]{\mathrm{d}} \mathcal{N}(0, \mathcal{I}(\mathbf{x}_t^\star)^{-1})$$

where $\mathcal{I}(\mathbf{x}_t^\star) := \sum_{p=1}^{P} \mu_p \mathbb{E}_{p(\mathbf{s}_t;\mathbf{f}_p)} \mathrm{Hess}(\log p(\mathbf{s}_t|\mathbf{x}_t^\star; \mathbf{f}_p))$ is the Fisher Information matrix of the model at $\mathbf{x}_t^\star$, $\xrightarrow{\mathrm{d}}$ means convergence in distribution, and we defined $\mu_p := \lim_{N\to\infty} \frac{n_p}{N}$.

The asymptotic Gaussianity of the MLE in the many neurons limit suggests performing approximate inference in a surrogate Hidden Markov Model, with the same transition probabilities $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ as the original ones, but where the observations $\mathbf{s}$ are replaced by the previously computed MLE $\widehat{\mathbf{x}}$ of the latent variable. Leveraging Theorem B.1, SIMPL approximates the emission probabilities $p(\widehat{\mathbf{x}}_t|\mathbf{x}_t)$ by the Gaussian distribution $\mathcal{N}(\mathbf{x}_t, \mathbf{\Sigma}_t)$, where $\mathbf{\Sigma}_t := (N\mathcal{I}(\widehat{\mathbf{x}}_t))^{-1} \approx (N\mathcal{I}(\mathbf{x}_t))^{-1}$.

[3] The i.i.d case was established in Fisher (1925)

By treating the covariance matrices $\mathbf{\Sigma}_t$ as deterministic instead of depending on $\mathbf{x}_t$, the variables $(\mathbf{x}_t, \widehat{\mathbf{x}}_t)$ form a Linear Gaussian State Space Model, with hidden variables $\mathbf{x}_t$ and observed variables $\widehat{\mathbf{x}}_t$ given by:

$$\begin{aligned} \widehat{\mathbf{x}}_t \mid \mathbf{x}_t &\sim \mathcal{N}(\mathbf{x}_t, \mathbf{\Sigma}_t) \\ \mathbf{x}_{t+1} \mid \mathbf{x}_t &\sim \mathcal{N}(\mathbf{x}_t, \sigma_v^2 \mathbf{I}), \end{aligned} \tag{5}$$

for $\sigma_v = \mathrm{v} \cdot dt$. This model is precisely an instance of Linear Gaussian State Space Models defined in Equation 4, with latent variable $z_t := \mathbf{x}_t$, observation $\widehat{\mathbf{x}}_t$, and the four matrices set to:

$$\begin{aligned} F_t &= \mathbf{I} && \text{(constant)} \\ H_t &= \mathbf{I} && \text{(constant)} \\ Q_t &= \sigma_v^2 \mathbf{I} && \text{(constant)} \\ R &= \mathbf{\Sigma}_t && \text{(time-varying).} \end{aligned}$$

This correspondence allows SIMPL to compute an approximation of the marginal posterior distributions $p(\mathbf{x}_t|\mathbf{s}) \approx p(\mathbf{x}_t|\widehat{\mathbf{x}})$ using Kalman Smoothing (Kalman, 1960; Rauch et al., 1965). This posterior is then used as the approximation $\widehat{q}^{[e]}$ to $q^{[e]}$ in SIMPL's E-step. Finally, $\mathcal{F}(f, \widehat{q}^{[e]})$ is approximated by sampling from $\widehat{q}^{[e]}$, and computing the empirical average of $\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})$. Importantly, obtaining the MLE estimates $\widehat{\mathbf{x}}_t$ can be obtained in parallel for all t; the only sequential procedure remaining being the Kalman Smoothing step.

B.2 SPIKE SMOOTHING AS AN APPROXIMATE M-STEP

In the M-step, one maximises $\mathbb{E}_{\widehat{q}^{[e]}}[\log p(\mathbf{x}, \mathbf{s}\,; \mathbf{f})]$ w.r.t to the intensity functions (tuning curves) $\mathbf{f} = (f_1, \cdots, f_N)$. This step is often done by specifying a parametric model for each f, and then optimizing the parameters. However parametric models come with diadvantages, for example if the true function cannot be accurately represented by the parameteric model, the final procedure will suffer from a bias that does not vanish in the large sample limit. While one could use a neural network (whose bias can be made arbitrarily small by increasing the number of neurons), neural networks can be hard to interpret and expensive to train. Instead, SIMPL uses a non-parametric approach that is both training-free and interpretable. To do so, SIMPL samples from its approximate posterior $\tilde{\mathbf{x}} \sim \widehat{q}^{[e]}$, and computes a non-parametric estimate (Hodara et al., 2018) of the intensity functions f_i given by:

$$\widehat{f}_i^{[e]}(\mathbf{x}) := \frac{\sum_{t=1}^{T} s_{ti}\, k(\mathbf{x}, \tilde{\mathbf{x}}_t)}{\sum_{t=1}^{T} k(\mathbf{x}, \tilde{\mathbf{x}}_t)}. \tag{6}$$

Here, $k : \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}_+$ is some kernel function.

We propose an explanation of the above formula as the generalization of an M-step: for a fixed $\widehat{q}^{[e]}$, $\mathbb{E}_{p(\mathbf{s})\widehat{q}^{[e]}(\mathbf{x})} \log p(\mathbf{s}, \mathbf{x}\,; \mathbf{f})$ equals (up to a constant) the negative KL divergence between the "data" distribution [4] $p(\mathbf{s})\widehat{q}^{[e]}(\mathbf{x}|\mathbf{s})$ and the model $p(\mathbf{s}, \mathbf{x}; \mathbf{f})$. Thus, an M-step can be understood as minimizing this KL divergence approximately, by replacing the expectation over $p(\mathbf{s})$ by an empirical average over the true data $\mathbf{s}$, an approximation which is asymptotically consistent in the large number of time-steps limit under suitable ergodicity conditions (Billingsley, 1961). SIMPL relaxes this approximation further, replacing the expectation over $\widehat{q}^{[e]}(\mathbf{x}|s)$ by a one-sample estimate of it through $\tilde{\mathbf{x}}$. Moreover, it does not use the KL as a loss function, but instead performs model fitting in a non-parametric manner. Under this procedure, the existing guarantees regarding the EM algorithm do not hold – on the other hand, SIMPL's M-step precisely matches spike smoothing, a fast and standard practice in neuroscience.

C IMPLEMENTATION DETAILS

Below we provide some implementation details that were important to maximise the computational efficiency of the method.

[4] We denote $q^k(x)$ by $q^k(x|s)$ to highlight the dependence between x and s.

C.1 MAXIMIZING SIMPL'S COMPUTATIONAL EFFICIENCY

C.1.1 COMPUTATIONAL BOTTLENECKS IN SIMPL

A single evaluation of the log-likelihood $\log p(\mathbf{s}_t|\mathbf{x}_t)$ requires evaluating the KDE-based rate map estimates given in Equation 6, which takes $\mathcal{O}(T)$ time as it involves a sum across timesteps. Moreover, this $\mathcal{O}(T)$-length calculation will be repeated T-times for each step of the Kalman smoother in order to (1) compute the MLEs $\widehat{\mathbf{x}}_t$ (which naively require gradient ascent on $\log p(\mathbf{s}_t|\mathbf{x}_t)$) and (2) evaluate the MLE variance $\mathbf{\Sigma}_t \coloneqq (N\mathcal{I}(\widehat{\mathbf{x}}_t))^{-1} = (N\mathbf{H}_x(\log p(\mathbf{s}|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t))^{-1}$. All in all, an exact implementation of SIMPL E-step thus has a quadratic $\mathcal{O}(T^2)$ time complexity, which is prohibitive for long datasets. Moreover, the second-order differentiation needed to compute $\mathcal{I}(\widehat{\mathbf{x}}_t)$ is also computationally expensive (formally, in introduces a large constant factor in front of the $O(T^2)$ term). In the next sections, we describe additional approximations which allow SIMPL to estimate the MLE and its variance in $\mathcal{O}(T)$ time and without differentiating the rate maps.

C.1.2 LINEAR-TIME, MLE ESTIMATION

Naive gradient-based solution The naive way to calculate the MLE $\widehat{\mathbf{x}}_t$ is to evaluate all N tuning curves (recall each evaluation costs $\mathcal{O}(T)$) for some location $\mathbf{x}$, use these to establish the log-likelihood $\log p(\mathbf{s}_t|\mathbf{x})$, calculate the gradient of this log-likelihood w.r.t. $\mathbf{x}$, and then take, for example, k gradient descent steps to find the MLE. This process is repeated for each timestep t, leads to a quadratic time complexity of $\mathcal{O}(kNT^2)$.

SIMPL's approach To compute the MLE in linear time SIMPL bypasses the need to recalculate the tuning curves at each time step by, instead, binning them onto a discretised grid of points once at the start of each iteration.

Formally SIMPL computes n evaluations the tuning curves $\tilde{\mathbf{f}} \coloneqq (\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_n) \coloneqq (\mathbf{f}(\mathbf{g}_1), \ldots, \mathbf{f}(\mathbf{g}_n))$ on a grid of n points $\mathcal{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_n)$. This has time complexity $\mathcal{O}(NnT)$. We use a uniform rectangular grid of points (the smallest rectangle containing the full observed behavioural variable) of spacing dx. For example, in a 1 m $\times$ 1 m environment with $dx = 0.02$ m, this would yield a grid of 50$\times$50 points ($n = 2500$).

Then, given $\tilde{\mathbf{f}}$, SIMPL then discretizes the log-likelihood functions $\log p(\mathbf{s}_t|\mathbf{x})$ over that same grid:

$$\begin{aligned} \tilde{l}_{it} \coloneqq \log p(\mathbf{s}_t|\mathbf{g}_i) &= \sum_{j=1}^{N} \log p(s_{tj}|\mathbf{g}_i) = \sum_{j=1}^{N} \log \frac{e^{-\tilde{f}_{ij}} \tilde{f}_{ij}^{s_{tj}}}{s_{tj}!} \\ &= -\sum_{j=1}^{N} \tilde{f}_{ij} + s_{tj} \log \tilde{f}_{ij} - \log s_{tj}! \end{aligned} \tag{7}$$

where we noted $\tilde{f}_{ij} \coloneqq (\tilde{\mathbf{f}}_i)_j$. Finally, given such evaluations, SIMPL set its approximation of the MLE to be

$$\widehat{\mathbf{x}}_t \coloneqq \arg\max_{\mathbf{g} \in \mathcal{G}} \log p(\mathbf{s}_t|\mathbf{g}) = \arg\max_i \tilde{l}_{it}$$

This way of calculating the MLE has linear time complexity yielding an improvement for $n < kT$.

C.1.3 LINEAR-TIME DERIVATIVE-FREE MLE VARIANCE ESTIMATION

A similar strategy could be employed to also compute $\mathcal{I}(\widehat{\mathbf{x}}_t) \coloneqq -\mathbf{H}_x(\log p(\mathbf{s}_t|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t)$, which appears in $\mathbf{\Sigma}_t$. Here $\mathbf{H}_x$ is the Hessian operator defined as $\mathbf{H}_x(f)(x) \coloneqq \nabla_x^2 f(x)$. To do so, one could compute the Hessian of the rate maps and their logarithm on that grid, from which any $\mathbf{H}_x(\log p(\mathbf{s}|\widehat{\mathbf{x}}_t))(\widehat{\mathbf{x}}_t)$ at the grid-point-based MLE obtained above can be evaluated as $\mathbf{H}_x(\log p(\mathbf{s}_t|\mathbf{g}_i))(\mathbf{g}_i) = -\sum_{j=1}^{N} \mathbf{H}_x(f_j)(\mathbf{g}_i) + s_{tj}\mathbf{H}_x(\log f_j)(\mathbf{g}_i)$. However, we found that differentiating $\mathbf{f}$ could be slow. To further improve computational efficiency, SIMPL produces an estimation of $\mathbf{\Sigma}_t$ by instead *estimating the variance of the posterior distribution* $p(\mathbf{x}_t|\mathbf{s}_t) \propto p(\mathbf{x}_t)p(\mathbf{s}_t|\mathbf{x}_t) = p(\mathbf{s}_t, \mathbf{x}_t)$. The posterior variance and the MLE variance are expected to closely match, as discussed in our theoretical justification above. Moreover, as this posterior is available analytically up to the normalizing constant $p(\mathbf{s}_t)$, its variance can be approximately computed by

binning $p(\mathbf{x}_t|\mathbf{s}_t)$ onto the same grid $\mathcal{G}$ introduced above, yielding the following fast estimator for $\mathbf{\Sigma}_t$.

$$\mathbf{\Sigma}_t \approx \operatorname{Cov} p(\mathbf{x}_t|\mathbf{s}_t) \approx \frac{\sum_i \tilde{p}_{it}(\mathbf{g}_i - \boldsymbol{\mu}_t)(\mathbf{g}_i - \boldsymbol{\mu}_t)^T}{\sum_i \tilde{p}_{it}}, \quad \boldsymbol{\mu}_t := \frac{\sum_i \mathbf{g}_i \tilde{p}_{it}}{\sum_i \tilde{p}_{it}} \tag{8}$$

where $\tilde{p}_{it} := \exp(\tilde{l}_{it}) = p(\mathbf{s}_t|\mathbf{g}_i)$. Intuitively, this is equivalent to fitting a multivariate Gaussian to the binned likelihood map. The covariance matrix of this Gaussian is then used as an approximation of the MLE variance. We provide a theoretical argument justifying the validity of this formula below.

Theoretical Justification Equation 8 is justified by the Bernstein Von Mises theorem, which states that the difference (in total variation) between the posterior distribution and the distribution of the MLE vanishes in the many neurons limit. We restate this theorem using the notations of our paper, assuming a unique rate map, and without stating some of the required regularity assumptions for simplicity. We refer the reader to (Van der Vaart, 2000, Theorem 10.1, p.141–144) for the full version.

Theorem C.1 (Bernstein-von Mises)**.** *Let $\mathbf{x}_t^\star \in \mathbb{R}^d$. Let $\mathbf{s}_t = (s_{1t}, \dots, s_{Nt})$ be i.i.d random variables with probability density $p(\mathbf{s}_t|\mathbf{x}_t^\star; \mathbf{f})$. Assume that the MLE $\widehat{\mathbf{x}}_t$ exists and it is unique. Then, under mild regularity conditions, for any prior p on $\mathbf{x}_t$, we have:*

$$\|p(\mathbf{x}_t|\mathbf{s}_t) - \mathcal{N}(\widehat{\mathbf{x}}_t, (N\mathcal{I}(\mathbf{x}_t^\star))^{-1})\|_{\mathrm{TV}} \underset{N\to\infty}{\overset{p(\mathbf{s}_t)}{\to}} 0$$

where $\overset{p(\mathbf{s})}{\to}$ denotes convergence in probability, and $\|\cdot\|_{\mathrm{TV}}$ denotes the Total Variation norm on bounded measures.

From this theorem, we thus have that the (random) posterior distribution behaves (in total variation) as a Gaussian whose covariance matrix is precisely the asymptotic variance of the MLE. Note however that convergence in total variation does not a priori imply convergence of variances. Further work could examine under which assumptions such a convergence of variances may hold. In practice, we found that this approximation yielded a satisfying trade-off between performance and accuracy.

C.2 Iterative Linear Realignment of the trajectories

To improve the identifiability properties and the numerical stability of SIMPL, we also transform the decoded latent trajectory at each iteration using a linear mapping which maximally aligns it with behaviour defined as $\mathbf{x}_t^{(e)} \leftarrow \mathbf{M}\mathbf{x}_t^{(e)} + \mathbf{c}$ where $\mathbf{M}, \mathbf{c} = \arg\min \sum_t \|\mathbf{x}_t^{(0)} - (\mathbf{M}\mathbf{x}_t^{(e)} + \mathbf{c})\|$. This approach ensures the scale, orientation and centre of the optimised latent trajectory are tied to behaviour, preventing accumulation of linear shifts/rotations across iterations and allowing us to interpret the latent relative to, and in the same units as, behaviour. We suspect that performing this alignment on all iterates *after* the optimisation would yield similar results. Because the transformed latent necessarily has similar scale to the behaviour — which was used to set the size of the discretised environment — we can reuse the same discrete grid for the latent avoiding the need to rediscretize the environment at each iteration.

C.3 Hyperparameters settings

SIMPL has two model hyperparameters:

- v: the diffusion rate for Kalman smoothing, which sets a prior over expected velocity of the latent variable. Units are in ms^{-1}.
- σ: the bandwidth of the kernel used in the M-step to smooth spikes. Units are in m.

Additionally there are some implementation-specific parameters:

- $\mathrm{d}x$: the bin size for the variance estimation of the MLE. Units are in m.
- $\mathrm{d}t$: the time step of the discretization of the latent variable. Units are in s.

- E: the number of iterations of the EM algorithm.

Finally, in all simulations we used a test fraction of 10% and held out 'speckled' data segments of length 1 second to evaluate the performance of the model. We provide in Table 1 the value of these hyperparameters for the Artificial Grid Cell Dataset and the Real Hippocampal Dataset.

Table 1: Hyperparameters settings

Dataset	v	σ	$\mathrm{d}x$	$\mathrm{d}t$	E
Artificial Grid Cell Dataset (Fig. 2)	0.4 ms^{-1}	0.02 m	0.02 m	0.1 s	10
Real Hippocampal Dataset (Fig. 5)	1.0 ms^{-1}	0.1 m	0.04 m	0.2 s	10
Motor task dataset (2D) [5] (Fig. 7c&d)	1.0	0.1	0.02	0.05 s	10
Motor task dataset (4D) (Fig. 7e)	1.5	0.09	0.1	0.05 s	10

D TEST-TRAIN PARTITIONING

To assess performance we partition the spike data matrix, $\mathbf{s}$, into testing and training sets, $\mathcal{S}_{\text{test}}, \mathcal{S}_{\text{train}}$. Inference is performed solely on the training set and we then track the log-likelihood of data in both sets (Fig. 2d, left), e.g. $\ell^{(e)} = |\mathcal{S}_{\text{test}}|_{\text{test}}^{-1} \sum_{(i,t)\sim\mathcal{S}^{\text{test}}} \log p(s_{ti}|\mathbf{x}_t^{(e)}, \mathbf{f}_i^{(e)})$. This partitioning requires careful consideration: entire time intervals cannot be withheld for testing without impairing the model's ability to infer the latent over this period. Likewise, entire neurons cannot be withheld without impairing the model's capacity to estimate their tuning curves. Instead, we adopt a speckled train-test mask previously used in latent variable modelling set-ups (Williams et al., 2020) which withholds for testing extended chunks of time bins arranged in an irregular "speckled" pattern across the data matrix (totalling 10% of the data).

E ADDITIONAL RESULTS

E.1 TOY MODEL OF A DISCRETE LATENT VARIABLE TASK

Before testing SIMPL on a large temporally continuous dataset we constructed a smaller dataset akin to a discrete two-alternative forced choice task (2AFC) (Fig. 6) — a widely studied decision–making paradigm (Platt & Glimcher, 1999; Bogacz et al., 2006; Znamenskiy & Zador, 2013; Lieder et al., 2019). The true latent states $\mathbf{x}_t^\star \in \{0, 1\}$ are binary and have no temporal structure (here subscript t indexes *trials* not time), analagous to a series of random "left" or "right" choices (Fig. 6b). This latent state is stochastically encoded by a population of neurons with random tuning curves giving the Bernoulli emission probabilities under each latent state:

$$f_i^\star(\mathbf{x}) = \begin{cases} f_{i0} \sim \mathcal{U}(0,1) & \mathbf{x} = 0, \\ f_{i1} \sim \mathcal{U}(0,1) & \mathbf{x} = 1, \end{cases}$$

$$\mathbf{x}_t^\star \sim \text{Bernoulli}(0.5) \quad \text{and} \quad s_{ti}|\mathbf{x}_t \sim \text{Bernoulli}(f_i^\star(\mathbf{x}_t^\star)).$$

Data is then sampled for $T = 50$ trials and $N = 15$ neurons as shown in Fig. 6. Initial conditions, $\mathbf{x}_t^{(0)}$, are generated from the true latent by randomly resampling a fraction of trials $\rho = 0.5$ (Fig. 6b). This partial resample represents an initial discrepancy between the behavioural measurement and the true internal state of the agent.

We perform inference on this dataset using a reduced version of the model (SIMPL-R). In the M-step, tuning curves were fitted by calculating the average activity of a neuron across each latent condition (e.g. $f_i^{(e)}(\mathbf{x}) = \sum_t s_{ti}\delta(\mathbf{x}_t^{(e)}, \mathbf{x}) / \sum_t \delta(\mathbf{x}_t^{(e)}, \mathbf{x})$, conceptually similar to KDE). For the E-step, each latent was the decoded according to the maximum likelihood estimate under the observed spikes and tuning curve estimates from the previous epoch: $\mathbf{x}_t^{(e+1)} = \arg\max_{\mathbf{x}} \sum_i \log p(s_{ti}|\mathbf{x}, f_i^{(e)})$ (there is no time dependence between latents, thus no Kalman smoothing). This process was repeated for 5 epochs and, with high reliability, converged on the true latents after approximately two (Fig. 6c & d, distributions show repeat for 1000 randomly seeded datasets, dotted lines show ceiling performance on a model perfectly initialised with noiseless $\mathbf{x}^{(0)} = \mathbf{x}^\star$). We repeated this experiment for various

values of ρ: latent recovery was almost perfect when ρ was small (i.e. when the initial conditions were close to the true latent), dropping off as ρ approached 1. At $\rho = 1$ when the conditions were *completely* random, the model was biased to recover a latent space that is either perfectly correlated or perfectly anti-correlated ("left" $\leftrightarrow$ "right") with the true latent (Fig. 6c, right), a valid isomorphism discussed more in the upcoming sections.

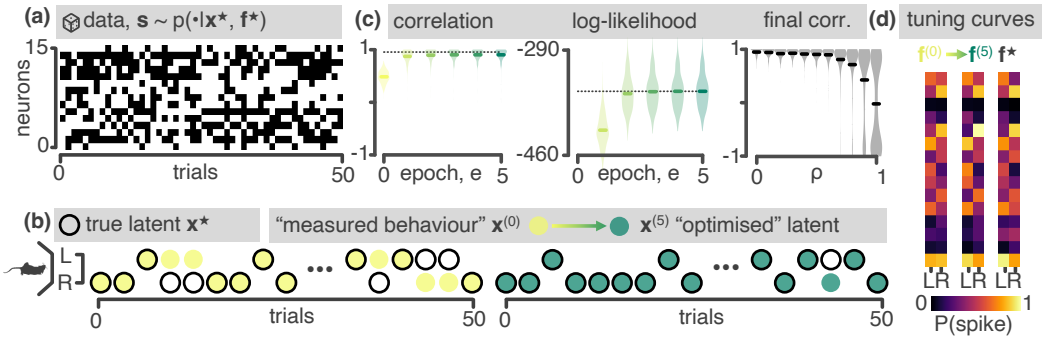

Figure 6: A two-alternative forced choice task (2AFC) toy-model. **(a)** Data generation: Spikes are sampled from a simple generative model. For each of T=50 independent trials a random binary latent — analogous to a "left" or "right" choice — is encoded by a population of N=15 neurons with randomly initialised tuning curves. **(b)** Model performance: Starting from a noisy estimate (yellow) of the true latent (black) where a fraction $\rho = 0.5$ of trials are resampled, SIMPL-R recovers the true latent variables (green) with high accuracy. **(c)** *Left:* Correlation between $\mathbf{x}^{(e)}$ and $\mathbf{x}^\star$. *Middle:* Log-likelihood, $\log p(\mathbf{s}|\mathbf{x}^{(e)}, \mathbf{f}^{(e)})$. *Right:* Final correlation between $\mathbf{x}^{(5)}$ and $\mathbf{x}^\star$ as a function of initialization noise ρ. Violin plots show distributions over 1000 randomly seeded datasets, dotted lines show ceiling performance of a perfectly initialised model ($\mathbf{x}^{(0)} = \mathbf{x}^\star$) **(d)** Tuning curves.

E.2 SOMATOSENSORY CORTEX DATA DURING A HAND-REACHING TASK

To assess the generality of SIMPL beyond navigational/hippocampal datasets we tested it on data from the somatosensory cortex of a macaque monkey performing a centre-out hand-reaching task Chowdhury et al. (2020). During this recording the monkey made a series of reaches to a target in one of 8 directions, 7. On about half of the trials the reach was "active" whereby the monkey moved the manipulandum towards the target by itself and, on the other half the reach was "passive", whereby the monkey's hand was bumped in the direction of one of the targets by a force applied to the manipulandum, forcing the monkey to correct and return the cursor to the centre. We binned the data ($N = 65$ neurons, $T = 37$ mins, 1.02×10^6 spikes) at 20 Hz and trained SIMPL models on the entire dataset (active and passive reaches as well as the inter-trial intervals) for 10 epochs.

First we trained SIMPL with a 2D latent initialised to the measured x- and y-hand position of the monkey (Fig. 7c). The log-likelihood of the test spikes reliably increased during training (Fig. 7c) following which we visualised the latent trajectory, averaged across trial type aligned to movement onset time (i.e. reach direction, Fig. 7c top-right). We found the latent trajectory had diverged from, but remained correlated with, hand-position (correlation = 0.59). Individual trial types had distinct but overlapping trajectory motifs in the optimised latent space. We then trained SIMPL but with hand velocity, rather than position, as the initial condition (Fig. 7d). This model performed comparably, converging to an almost identical log-likelihood as the position model. After optimisation, the latent correlated only weakly with hand-velocity (corr.= 0.41).

Finally, we trained SIMPL with a 4D latent. Two of the dimensions were initialised with x- and y-hand position whilst the other two were initialised with x- and y-hand velocity. This model performed better than either of the two 2D models, converging to a higher log-likelihood. The latent dimensions initialised to position remained highly correlated with position (corr. = 0.74) and the latent dimensions initiales with velocity remained correlated with velocity (corr. = 0.57). The latent trajectory was also more structured, with distinct non-overlapping motifs for each trial type. We visualised two-dimensional slices of the four-dimensional tuning curves for each neuron and found that they had sharp and well-defined receptive fields similar to place fields in the hippocampus.

E.3 HYPERPARAMETER SWEEP

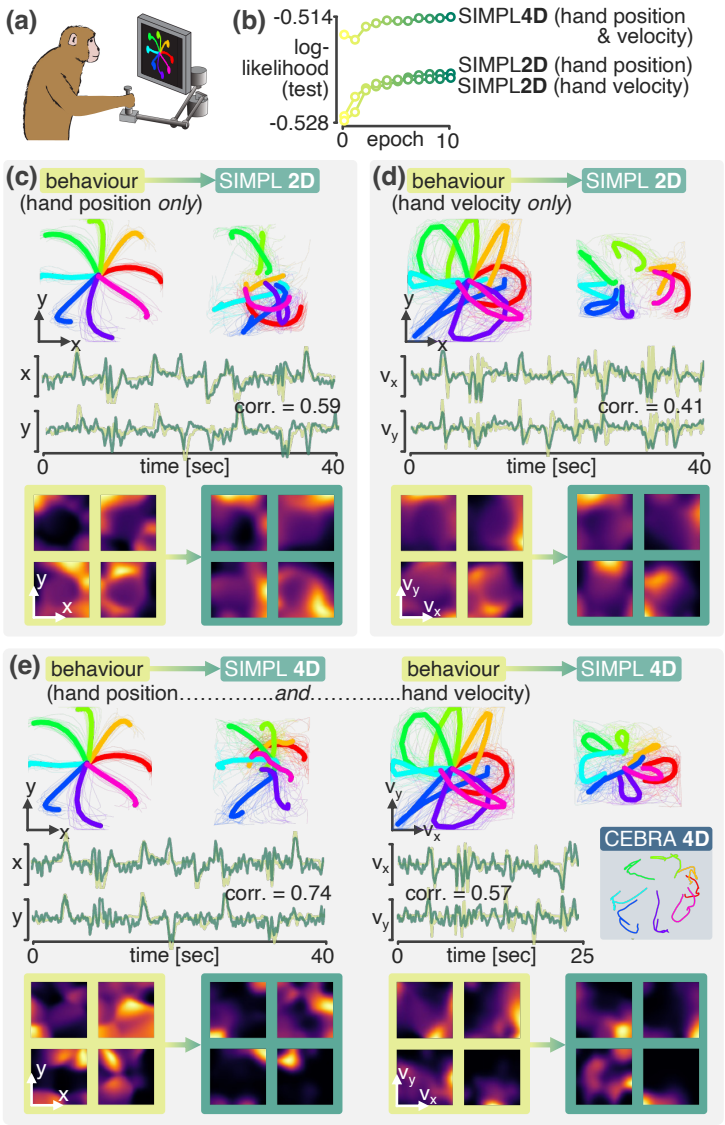

Figure 7: SIMPL applied to a somatosensory cortex dataset. **(a)** A macaque perform a serier of centre-out reaches and $N = 65$ neurons from the somatosensory cortex are recorded. **(b)** Log-likelihood curves for the three SIMPL models described in panels c-e. **(c)** SIMPL is trained with a 2D latent initialised to the $x-$ and $y-$position of the monkeys hand. Top-left show the raw behaviour averaged across all (active) trial aligned to movement onset time (-100ms – 500ms). Top-right shows SIMPL's latent after optimisation. Middle shows 40 seconds of behaviour (yellow) and latent (green) for 40 seconds. Bottom shows four example tuning curves at epochs 0 (behaviour) and 10 (optimised). **(d)** Same as c but with hand velocity as the initial condition. **(e)** Same as c but with a 4D latent where dimensions 1 and 2 are initialised with hand position and dimensions 3 and 4 are initialised with hand velocity. Inset panel shows an equivalent 2D visualisation of a 4D latent embedding generated using CEBRA trained with position as the behavioural label. Figure adapted from Schneider et al. (2023).

We swept over the two hyperparameters v (the velocity prior) and σ (the KDE bandwidth) to assess how sensitive SIMPL is to these hyperparameters, as shown in Figure 8. For this we used the same synthetic grid cell dataset used in Fig. 2. Notably, SIMPL's performance (measured in terms of the final error, see panel b) is relatively stable across a wide range of hyperparameters; kernel bandwidths between 0.1 cm and 5 cm and velocity priors between 0.2 m/s and 1 m/s all yield similar performance. When the tuning curves are confirmed that kernel bandwidth has a significant effect on their appearance. Broader kernels give smoother tuning curves eventually blurring the individual grid fields together whilst narrower kernels give sharper tuning curves eventual leading to overfitting where individual spikes are resolved.

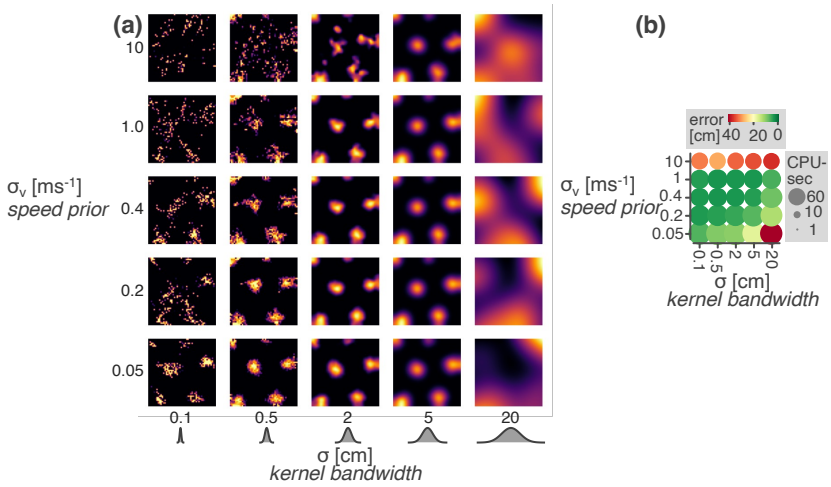

Figure 8: Performance of SIMPL on the synthetic grid cell dataset as a function of the hyperparameters v (speed prior) and σ (kernel bandwidth). **(a)** Tuning curves. **(b)** Final error between the latent and ground truth (colour) and total compute time (size).

E.4 NON-CONTINUOUS HIPPOCAMPAL REPLAY DATASET

Since SIMPL places an explicit prior on latent trajectories which are smooth and continuous we tested whether it could be used to model a dataset where the latent variable is non-continuous. For this we simulated a synthetic "replay" dataset from $N = 225$ small Gaussian place cells. In this dataset the latent variable and behaviour perfectly match except for regular, brief periods of "replay" where the latent variable jumps to a new location. Using the same hyperparameters as in the main text we found that SIMPL was able to recover the latent variable, capturing (or "decoding") the replay events with high accuracy (Fig. 9), despite its smoothness prior.

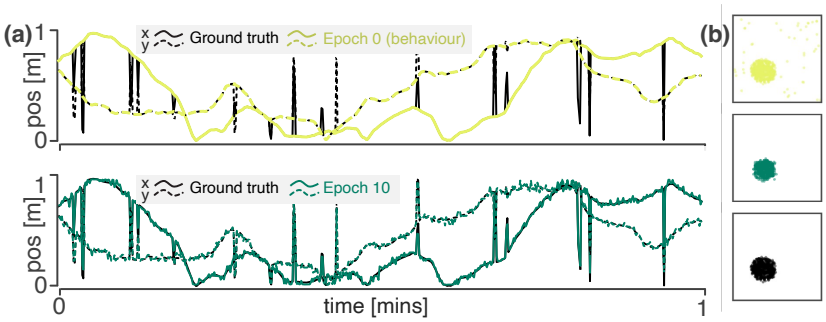

Figure 9: A synthetic hippocampal "replay" dataset. **(a)** One minute of trajectory, x-coordinate in solid line, y-coordinate in dashed. The behaviour (light-green, top panel) is smooth, actually matching the latent most of the time except when the latent takes regular, brief discontinuous jumps reminiscent of hippocampal replay events. After optimisation SIMPL is able to recover the latent (dark-green, bottom panel) and capture the replay events with high accuracy. **(b)** Spike raster plots; spikes plotted against the behaviour, optimised latent and ground truth latent.

E.5 PLACE CELL ANALYSIS

In Fig. 5, it was shown that the tuning curves of place cells in the hippocampus undergo statistically significant changes when optimised using SIMPL. For this analysis, individual place fields were automatically identified as isolated regions of elevated activity within a cells tuning curve. This was done by thresholding the activity of each neuron at 1 Hz and identifying contiguous regions of activity with a peak firing rate above 2 Hz and a total area less than half that of the full environment, similar to previous work (Tanni et al., 2022).

Further to the results shown in the main text we discovered that only *place cells* — here defined as a cell with one of more place *fields* — show statistically significant changes. Non-place cells — cells with no place fields — did not show statistically significant changes in their maximum firing rate (median -6.5%, p=0.96) nor spatial information (median +0.4%, p=0.74) (Fig. 10).

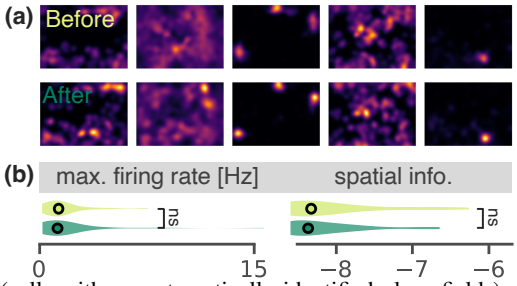

Figure 10: Non-place cells (cells with no automatically identified place fields) are not statistically affected by SIMPL. **(a)** Tuning curves before and after optimisation for five randomly selected non-place cells. **(b)** Distributions (median shown in black circle) of maximum firing rates and spatial information for non-place cell tuning curves before and after optimisation.

E.6 FURTHER EXPERIMENTS WITH GPLVM

Among the methods compared to SIMPL in the manuscript GPLVM and a GPDM were the only ones with a misspecified emission model (a Gaussian models for spike data). We investigated the impact of this misspecification by running GPLVM on a control dataset generated with the same tuning curves but with a Gaussian noise model, instead of a Poisson one. For the Gaussian data, as for the Poisson data, the tuning curves specified the means of the observations and we set the standard deviation to a fixed value of 0.1. The results are shown in figure 11. While GPLVM performed slightly better on the control dataset the improvement remained small compared to the difference between GPLVM and SIMPL (Fig. 4). Other differences between SIMPL and GPLVM (e.g. the inducing point approximation used for GPLVM, optimisation issues, or other forms of misspecification/model differences) must be more important.

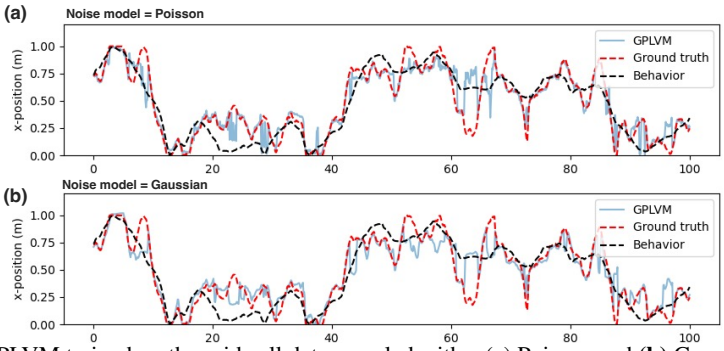

Figure 11: GPLVM trained on the grid cell data sampled with a (a) Poisson nad **(b)** Gaussian noise model.

F SUMMARY TABLE OF RELATED METHODS

Here we summarize some of the most relevant LVM and dimensionality reduction techniques in the context of our five key desiderata as described in the related work section. These are:

1. Complex tuning curves: Does the model learn/infer non-linear tuning curves as opposed to linear/exponential-linear/etc. tuning curves.

2. Smooth latent dynamics: Does the model impose smooth temporal dynamics on the latent space (e..g by assuming a linear dynamical system, Gaussian process or usign an RNN), as opposed to treating each time point independently.
3. Spike-friendly: Was the method built with spikes in mind, e.g. for probabilistic model this refers to whether the generative noise model is Poisson as opposed to, say, Gaussian.
4. Exploit behaviour: Does/can the model use behaviour (e.g. as an observation, contrastive loss-target, or initialisation) to guide latent discovery.
5. Scalable: Can the model scale to datasets of long duration. Specifically, in available open-source implementations of the method does training/inference have near-linear time complexity. Note this does *not* mean that compute time is necessarily fast in an absolute sense, just that scaling is linear.

Model	Complex tuning curves	Smooth latent dynamics	Spike-friendly	Exploit behaviour	Scalable
SIMPL (Our method)	Yes	Yes	Yes	Yes[§]	Yes
GPLVM (Lawrence, 2003)	Yes	N/S	No	Yes[§]	Yes[#]
P-GPLVM (Wu et al., 2017)	Yes	Yes	Yes	Yes[§]	No
M-GPLVM (Jensen et al., 2020)	Yes	No	No	No	No
VIND (Hernandez et al., 2018)	Yes	Yes	Yes	N/S[‡]	Yes
PfLDS (Gao et al., 2016)	Yes	Yes	Yes	No	Yes
pi-VAE (Zhou & Wei, 2020)	Yes	No	Yes	Yes	Yes
CEBRA (Schneider et al., 2023)	N/A	No	Yes	Yes	Yes
LFADS (Pandarinath et al., 2018)	No	Yes	Yes	Yes	Yes
TNDM (Hurwitz et al., 2021)	No	Yes	Yes	Yes	Yes
GP-SDEs (Duncker et al., 2019)	No	Yes	N/S	N/S	Yes
rSLDS (Linderman et al., 2016)	No	Yes	No	Yes[§]	Yes
GPDM (Wang et al., 2005)	Yes	Yes	No	Yes[§]	No
MM-GPVAE (Gondur et al., 2023)	No	Yes	Yes	Yes	Yes
MM-GPVAE (Gondur et al., 2023)	No	Yes	Yes	Yes	Yes
PSID (Sani et al., 2021)	No	Yes	No	Yes	Yes
GPFA (Yu et al., 2008a)	No	Yes	No	No	Yes[#]
SSMDM (Zoltowski et al., 2020)	No	Yes	Yes	Yes	Yes
PLNDE (Kim et al., 2021)	No	Yes	Yes	Yes	Yes
GLDS (Kalman, 1960)	No	Yes	No	No	Yes
DKF (Krishnan et al., 2015)	Yes	Yes	No	No	Yes
PLDS (Macke et al., 2011)	No	Yes	Yes	No	Yes
UMAP (McInnes et al., 2018)	N/A	No	No	No	No
TSNE (Van der Maaten & Hinton, 2008)	N/A	No	No	No	No
pPCA (Pearson, 1901; Tipping & Bishop, 1999)	No	No	No	No	Yes
dPCA (Kobak et al., 2016)	No	No	No	Yes	Yes

Table 3: A table of comparable models and their properties. N/A means the criterion is not applicable to the model. N/S means the criterion is not specified or may be dependent on implementation specifics. Techniques in **bold** are compared to on our benchmark dataset in Fig. 4 [#]: If using a induction point variant.
[†]: No current inducing variant which could improve scalability.
[§] [illegible]