SIχO: Smoothing Inference with Twisted Objectives

Dieterich Lawson,* **Allan Raventós,*** **Andrew Warrington,*** **Scott Linderman**

{jdlawson, aravento, awarring, scott.linderman}@stanford.edu

Stanford University

Abstract

Sequential Monte Carlo (SMC) is an inference algorithm for state space models that approximates the posterior by sampling from a sequence of target distributions. The target distributions are often chosen to be the *filtering* distributions, but these ignore information from future observations, leading to practical and theoretical limitations in inference and model learning. We introduce *SIXO*, a method that instead learns target distributions that approximate the *smoothing* distributions, incorporating information from all observations. The key idea is to use density ratio estimation to fit functions that warp the filtering distributions into the smoothing distributions. We then use SMC with these learned targets to define a variational objective for model and proposal learning. SIXO yields provably tighter log marginal lower bounds and offers more accurate posterior inferences and parameter estimates in a variety of domains.

1 Introduction

In this work we consider model learning and approximate posterior inference in probabilistic state space models. Sequential Monte Carlo (SMC) is a general-purpose method for these problems [1–3] that produces an unbiased estimate of the marginal likelihood as well as latent state trajectories (i.e. *particles*) that can be used to approximate posterior expectations. SMC can facilitate model learning via expectation-maximization or direct maximization of the marginal likelihood estimate [4, 5]. It can also be cast in a variational framework [6, 7] as a rich family of approximate posterior distributions that can be fit using stochastic gradient ascent and modern automatic differentiation methods [8–12].

The quality of SMC's marginal likelihood and posterior estimates is driven by two design decisions: the choice of *proposal distributions* and *target distributions*. The proposal distributions specify how particles propagate from one time step to the next, while the target distributions specify how those particles are weighted and which ones survive to future time steps. The most common SMC variant, filtering SMC, sets the targets to the *filtering distributions*, the conditional distributions over latent states $\mathbf{x}_{1:t} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$ given observations $\mathbf{y}_{1:t} = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$. The central issue is that the filtering distributions do not incorporate information from future observations $\mathbf{y}_{t+1:T}$.

Figure 1 illustrates why setting the target distributions to the filtering distributions can be problematic. In this example, the latent states follow a simple Gaussian random walk, but only the last step is observed. Thus, the filtering distributions reduce to the prior, a series of mean zero Gaussians, shown in Figure 1a. If the observation is far from the prior, the filtering distribution suddenly jumps at time $T = 10$. This is a recipe for disaster in SMC: the particles at time $T - 1$ will be distributed according to a mean zero Gaussian and very few will survive to the next time step, causing the variance of the SMC estimator to explode. Even if the proposals incorporate smoothing information, using filtering targets can cause particle degeneracy by resampling away high-quality particles, as seen in Figure 1c.

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

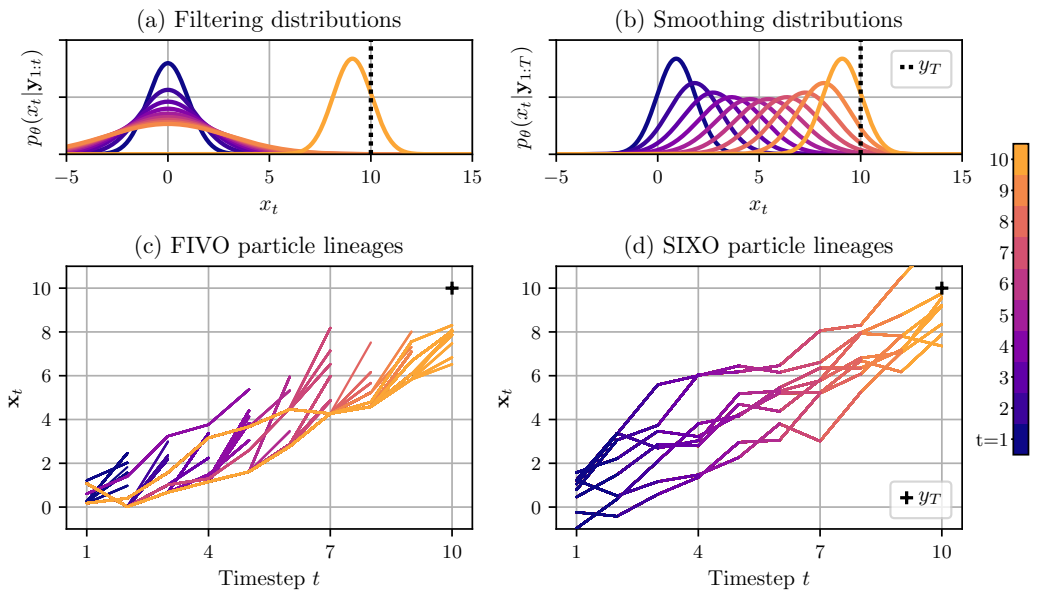


Figure 1: Theoretical and empirical target distributions for a Gaussian random walk with a single observation $y_T = 10$ and $\mathbf{y}_{1:T-1} = \varnothing$. **(a)** For $t = 1, \ldots, 9$ the filtering distributions reduce to a series of mean-zero Gaussians. At $t = T = 10$, the filtering distribution incorporates the observation y_T, resulting in a sudden shift and particle death. **(b)** In contrast, the smoothing distributions steadily shift towards the observation, matching the posterior perfectly. **(c)** The proposal learned by a previous method, FIVO [8–10], exploits smoothing information to propose particles upwards towards the observed value. However, FIVO is based on filtering SMC which "resists" this by resampling particles back towards the prior, resulting in particle degeneracy. **(d)** SIXO's proposal also leverages smoothing information, but proposed particles are preserved by the learned target distributions.

Suppose instead that the target distributions were the *smoothing distributions*—the conditional distributions over latents $\mathbf{x}_{1:t}$ given *all* observations $\mathbf{y}_{1:T}$. Figure 1b shows the smoothing distributions for the simple Gaussian random walk. Unlike the filtering distributions, the smoothing distributions shift steadily toward the observation over time. These slow, smooth changes are ideal for SMC: Figure 1d shows many particles surviving from one step to the next, resulting in a low-variance SMC estimator.

In practice we do not have access to the smoothing distributions—if we did, there would be no need for SMC! Here, we introduce a new method called SIXO: Smoothing Inference with Twisted Objectives. SIXO provides a unified approach for learning model and proposal parameters, as well as a set of *twisting functions* that warp the filtering distributions into targets that better approximate the smoothing distributions [13]. Like its predecessor FIVO [8–10], SIXO uses a variational approach, deriving a lower bound to the marginal likelihood. Unlike its predecessor, we prove that the SIXO bound can become tight, even with finitely many particles.

The key challenge with SIXO is learning the twisting functions. We find that a simple density ratio estimation approach works best, and we propose an algorithm that interleaves twist updates with updates to the model and proposal. Thus, SIXO offers a means of jointly learning model parameters, SMC proposals, and targets for accurate posterior inference.

Finally, we give empirical evidence to support our theoretical claims. Across a range of experiments with a Gaussian diffusion, a stochastic volatility model of currency exchange rates, and a Hodgkin-Huxley model of membrane potential in a neuron, SIXO consistently outperforms FIVO and related methods. We dissect these results to illustrate how learning better targets enables more effective posterior inference and model learning.

2 Background

Consider modeling sequential data $\mathbf{y}_{1:T} \in \mathcal{Y}^T$ using latent variables $\mathbf{x}_{1:T} \in \mathcal{X}^T$ with Markovian structure, and let the joint distribution factorize as

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_1)p_{\boldsymbol{\theta}}(\mathbf{y}_1 \mid \mathbf{x}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t) \tag{1}$$

with global parameters $\boldsymbol{\theta} \in \Theta$. We further assume that the conditional distributions $p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and $p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t)$ may depend nonlinearly on $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ respectively.

The marginal likelihood and posterior for this model class are not readily available from the joint distribution due to the intractable integral over the latents $\mathbf{x}_{1:T}$, i.e.

$$p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}) = \int_{\mathcal{X}^T} p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) \, \mathrm{d}\mathbf{x}_{1:T},$$

cannot easily be computed due to the form of the conditional distributions.

2.1 Sequential Monte Carlo

Sequential Monte Carlo is an algorithm for inference in state-space models that approximates the posterior $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ with a set of K weighted particles $\mathbf{x}_{1:T}^{1:K}$. These particles are constructed by approximately sampling from a sequence of target distributions $\{\pi_t(\mathbf{x}_{1:t})\}_{t=1}^T$, with the intuition that sampling from a series of distributions that gradually approach the posterior is easier than attempting to sample from it directly. The targets are often only available up to an unknown normalizing constant Z_t, so SMC uses the unnormalized targets $\{\gamma_t(\mathbf{x}_{1:t})\}_{t=1}^T$, which correspond to the normalized targets via $\pi_t(\mathbf{x}_{1:t}) = \gamma_t(\mathbf{x}_{1:t})/Z_t$.

SMC repeats three steps: First, a set of latents are sampled from a proposal distribution $q_{\boldsymbol{\theta}}(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^k, \mathbf{y}_{1:T})$ conditional on the current particles $\mathbf{x}_{1:t-1}^{1:K}$. Then, each particle is weighted using the unnormalized target $\gamma_t(\mathbf{x}_{1:t})$ to form an empirical approximation of the normalized target distribution. Finally, new particle trajectories $\mathbf{x}_{1:t}^{1:K}$ are drawn from this approximation to the normalized target.

Ideally the target distributions smoothly approach the posterior so that sampling from the target at time $t+1$ is easy given samples from the target at time t. As long as mild technical conditions are met and $\gamma_T(\mathbf{x}_{1:T}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$, SMC returns a consistent and unbiased estimate of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T})$ and a set of weighted particles approximating the posterior $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ [1–3]. For more details see Appendix B.1, and for a thorough treatment of SMC see [1–3].

2.2 Filtering SMC and Model Learning

The most commonly-used SMC algorithm is filtering SMC (FSMC), which sets the normalized targets to the *filtering* distributions, i.e. $\pi_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ and $\gamma_t(\mathbf{x}_{1:t}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$. Let $\widehat{Z}_{\mathrm{FSMC}}(\boldsymbol{\theta}, \mathbf{y}_{1:T})$ be the marginal likelihood estimator returned from running filtering SMC with proposal distributions $\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})\}_{t=1}^T$ which may share parameters with $p_{\boldsymbol{\theta}}$.

Previous work used filtering SMC to fit model parameters by ascending a lower bound on the log marginal likelihood called a *filtering variational objective* (FIVO) [8–10]. The FIVO bound is derived using Jensen's inequality and the unbiasedness of $\widehat{Z}_{\mathrm{FSMC}}$,

$$\mathcal{L}_{\mathrm{FIVO}}(\boldsymbol{\theta}, \mathbf{y}_{1:T}) \triangleq \mathbb{E}[\log \widehat{Z}_{\mathrm{FSMC}}(\boldsymbol{\theta}, \mathbf{y}_{1:T})] \leq \log \mathbb{E}[\widehat{Z}_{\mathrm{FSMC}}(\boldsymbol{\theta}, \mathbf{y}_{1:T})] = \log p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}), \tag{2}$$

and is optimized using stochastic gradient ascent in $\boldsymbol{\theta}$ [8–10, 14].

2.3 Smoothing SMC via Twisting Functions

The main disadvantage of filtering SMC is that the filtering distributions only condition on observations up to the current timestep t, ignoring future observations $\mathbf{y}_{t+1:T}$. This creates situations where future observations are highly unlikely given the current latent trajectories, which in turn causes particle death, high variance normalizing constant estimates, and poor inference and model learning [8, 13, 15]. Performing smoothing SMC would resolve this issue by choosing the *smoothing*

distributions as targets, i.e. $\pi_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:T})$ and $\gamma_t(\mathbf{x}_{1:t}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:T})$. Unfortunately, $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:T})$ is not readily available from the model and computing it is roughly as hard as the original inference problem.

However, $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:T})$ factors into the product of the filtering distributions, $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$, and the *lookahead* distributions, $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ (Appendix B.2). If the lookahead distributions can be well-approximated by a series of "twisting" functions [13], $\{r(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^T$, then running SMC with targets $\gamma_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) r(\mathbf{y}_{t+1:T}, \mathbf{x}_t)$ would approximate smoothing SMC. In this sense, the lookahead distributions are optimal twisting functions [13, 16].

Different twisting functions yield different SMC methods such as auxiliary particle filters and twisted particle filters [13, 17]. However, as long as the final unnormalized target $\gamma_T(\mathbf{x}_{1:T})$ is proportional to $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$ and regularity conditions are met, SMC will produce an unbiased estimate of the marginal likelihood, regardless of the choice of twisting functions [1, 3, 18]. Instead, the quality of the twisting functions affects the variance of SMC's marginal likelihood estimate.

3 SIXO: Smoothing Inference with Twisted Objectives

Our goal is to fit models by optimizing a lower bound on their log marginal likelihood constructed using smoothing SMC. To construct the lower bound, fix $r_{\boldsymbol{\psi}}(\mathbf{x}_T) = 1$ and let $\widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ be the marginal likelihood estimator returned from running SMC with unnormalized targets $\{p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^T$ and proposal distributions $\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T})\}_{t=1}^T$. Because the Tth unnormalized target is $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$, $\widehat{Z}_{\text{SIXO}}$ will be an unbiased estimator of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T})$ [1, 3]. This implies via Jensen's inequality that

$$\begin{aligned} \mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T}) &\triangleq \mathbb{E}\left[\log \widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})\right] \\ &\leq \log \mathbb{E}\left[\widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})\right] = \log p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}) \end{aligned} \tag{3}$$

i.e. $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ is a lower bound on the log marginal likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T})$ [14].

3.1 The Functional Form of the Twists

The structure of the lookahead distributions $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ suggests a functional form for $r_{\boldsymbol{\psi}}$ that accepts a single latent $\mathbf{x}_t$ and produces distributions over all future observations $\mathbf{y}_{t+1:T}$. Because the twists will be evaluated once per particle and timestep in an SMC sweep, this functional form would lead to an algorithm with $O(T^2)$ complexity. To reduce the complexity, we consider two methods: fixed-lag twisting and backwards twisting.

Fixed-lag twisting approximates the full lookahead distribution $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ using a fixed window of L observations, i.e. it models $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:t+L} \mid \mathbf{x}_t)$ [17, 19, 20]. We define the fixed-lag twisting functions $\{r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:t+L}, \mathbf{x}_t)\}_{t=1}^{T-1}$ as a sequence of functions which accept $\mathbf{x}_t \in \mathcal{X}$ and produce a distribution over $\mathbf{y}_{t+1:t+L} \in \mathcal{Y}^L$. This reduces the computational complexity to $O(TL)$, at the expense of only looking at L observations.

In our experiments we use an $L = 1$ twist that scores the next observation by approximating the one-step lookahead

$$p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1} \mid \mathbf{x}_t) = \int p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})\, p_{\boldsymbol{\theta}}(\mathbf{x}_{t+1} \mid \mathbf{x}_t)\, \mathrm{d}\mathbf{x}_{t+1} \tag{4}$$

with Gauss-Hermite quadrature [21]. We refer to this as the "quadrature twist". This approach is similar to the APF [17, 22], but uses numerical quadrature in place of sample-based integration.

Backwards twisting is motivated by rewriting the lookahead distributions using Bayes' rule,

$$p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})\, p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T})}{p_{\boldsymbol{\theta}}(\mathbf{x}_t)} \propto \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})}{p_{\boldsymbol{\theta}}(\mathbf{x}_t)}, \tag{5}$$

where we drop terms independent of $\mathbf{x}_t$ because the twisting functions will be used to score particles in SMC. Thus, we need only approximate $p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T}) / p_{\boldsymbol{\theta}}(\mathbf{x}_t)$. The numerator $p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})$

Algorithm 1 SIXO-DRE

procedure SIXO-DRE($\mathbf{y}_{1:T}$, $\boldsymbol{\theta}_0$, $\boldsymbol{\psi}_0$, S, N, K)
  for $s = 1, \ldots, S$ **do**
    $\boldsymbol{\psi}_s$ =TWIST-UPDATE($\boldsymbol{\theta}_{s-1}, \boldsymbol{\psi}_{s-1}, N$)
    $\boldsymbol{\theta}_s$ =MODEL-UPDATE($\mathbf{y}_{1:T}, \boldsymbol{\theta}_{s-1}, \boldsymbol{\psi}_s, N, K$)
  return $\boldsymbol{\theta}_S, \boldsymbol{\psi}_S$

procedure TWIST-UPDATE($\boldsymbol{\theta}, \boldsymbol{\psi}_0, N$)
  for $i = 1, \ldots, N$ **do**
    $\tilde{\mathbf{x}}_{1:T} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T})$
    $\mathbf{x}_{1:T}, \mathbf{y}_{1:T} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$
    $\mathcal{L}_{\mathrm{DRE}}(\boldsymbol{\psi}) = \frac{1}{T-1} \sum_{t=1}^{T-1} \log \sigma(\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)) + \log(1 - \sigma(\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \tilde{\mathbf{x}}_t)))$
    Compute $\boldsymbol{\psi}_i$ using the gradients of $\mathcal{L}_{\mathrm{DRE}}$ evaluated at $\boldsymbol{\psi}_{i-1}$
  return $\boldsymbol{\psi}_N$

procedure MODEL-UPDATE($\mathbf{y}_{1:T}, \boldsymbol{\theta}_0, \boldsymbol{\psi}, N, K$)
  for $i = 1, \ldots, N$ **do**
    $\widehat{Z}_{\mathrm{SIXO}}(\boldsymbol{\theta})$ = SMC($\{p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^{T}$, $\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T})\}_{t=1}^{T}$, K)
    Compute $\boldsymbol{\theta}_i$ using the biased gradients of $\widehat{Z}_{\mathrm{SIXO}}$ evaluated at $\boldsymbol{\theta}_{i-1}$
  return $\boldsymbol{\theta}_N$

procedure SMC($\{\gamma_t(\mathbf{x}_{1:t})\}_{t=1}^{T}$, $\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T})\}_{t=1}^{T}$, K)
  See Algorithm 2 in Appendix B.1.

is the reverse of the lookahead distributions—it is a distribution over a single latent conditioned on future observations. This makes it possible to parameterize the twists using a recurrent function approximator (e.g. a recurrent neural network or RNN) run backwards across the observations $\mathbf{y}_{1:T}$ to produce twist values for each timestep as a function of $\mathbf{x}_t^k$.

We define the backwards twists $\{r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^{T-1}$ as a sequence of positive, integrable, real-valued functions $\mathcal{Y}^{T-t} \times \mathcal{X} \to \mathbb{R}_+$ with parameters $\boldsymbol{\psi} \in \boldsymbol{\Psi}$. Parameterizing backward twists with a recurrent function approximator results in $O(T)$ complexity (see Appendix C.3) and allows the twist to condition on all future observations, making backwards twisting preferable to fixed-lag twisting.

3.2 Learning Twists

Ascending the Unified Objective One way to fit the twists, proposal, and model is to ascend $\mathcal{L}_{\mathrm{SIXO}}$ in the parameters of $p_{\boldsymbol{\theta}}$, $q_{\boldsymbol{\theta}}$, and $r_{\boldsymbol{\psi}}$, similar to FIVO [8–10]. The gradients of this objective include score-function terms that arise from the discrete resampling steps in SMC. We refer to ascending $\mathcal{L}_{\mathrm{SIXO}}$ with these unbiased gradients as SIXO-u. Because the resampling gradient terms have high variance, SIXO-u is impractical for complex settings [8, 11]. Lower-variance methods for estimating these gradients were explored by Lawson et al. [11] but found to be ineffective. We therefore seek an alternative method for training the twists. For a detailed discussion and derivation of the gradient, see Appendix C.1.

Density Ratio Estimation Note that the optimal backwards twist is proportional to the ratio of a "backwards message" $p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})$ and the latent marginal $p_{\boldsymbol{\theta}}(\mathbf{x}_t)$ (Equation 5). Thus, we can learn the backwards twist using density ratio estimation (DRE) [23, 24].

DRE via classification estimates the ratio of two densities $a(x)/b(x)$ by training a classifier to distinguish between samples from a and b. If such a classifier is trained using the logit link function, then its raw output will approximate $\log a(x) - \log b(x)$ up to a constant [24]. Using this approach, we interpret $\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)$ as the logit of a Bernoulli classifier, which is trained to distinguish between samples from $p_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{y}_{t+1:T})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}_t) p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T})$, which are available from the model. When trained in this way, $\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)$ will approximate $\log p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}_t)$ up to a constant, which can be ignored. For details see Appendix C.2 and Sugiyama et al. [24].

We use the DRE-learned twisting functions in an alternating scheme that first holds $p_{\boldsymbol{\theta}}, q_{\boldsymbol{\theta}}$ fixed and updates $r_{\boldsymbol{\psi}}$ using density ratio estimation, and then holds $r_{\boldsymbol{\psi}}$ fixed and updates $p_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{\theta}}$ by ascending a biased gradient estimator (no resampling terms) of $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi})$ in $\boldsymbol{\theta}$. We call the full alternating procedure for learning $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ SIXO-DRE, see Algorithm 1.

3.3 The SIXO Bound Can Become Tight

Maddison et al. [8] show that the FIVO bound can only become tight in models with uncommon dependency structures. We show that the SIXO bound can become tight for any model in the class defined in Section 2.

Proposition 1. Sharpness of the SIXO bound. *Let $p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$ be a latent variable model with Markovian structure as defined in Section 2, let $\mathcal{Q}$ be the set of possible sequences of proposal distributions indexed by parameters $\boldsymbol{\theta} \in \Theta$, and let $\mathcal{R}$ be the set of possible sequences of positive, integrable twist functions indexed by parameters $\boldsymbol{\psi} \in \Psi$. Assume that $\{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T})\}_{t=1}^{T} \in \mathcal{Q}$ and $\{p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)\}_{t=1}^{T-1} \in \mathcal{R}$. Finally, assume $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ has the unique optimizer $\boldsymbol{\theta}^*, \boldsymbol{\psi}^* = \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\psi}\in\Psi} \mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$.*

Then the following holds:

1. $q_{\boldsymbol{\theta}^*}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T})$ *for* $t = 1, \ldots, T$,
2. $r_{\boldsymbol{\psi}^*}(\mathbf{y}_{t+1:T}, \mathbf{x}_t) \propto p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ *up to a constant independent of* $\mathbf{x}_t$ *for* $t = 1, \ldots, T-1$,
3. $\mathcal{L}^K_{\text{SIXO}}(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*, \mathbf{y}_{1:T}) = \log p(\mathbf{y}_{1:T})$ *for any number of particles* $K \geq 1$.

Proof. See Appendix C.5. □

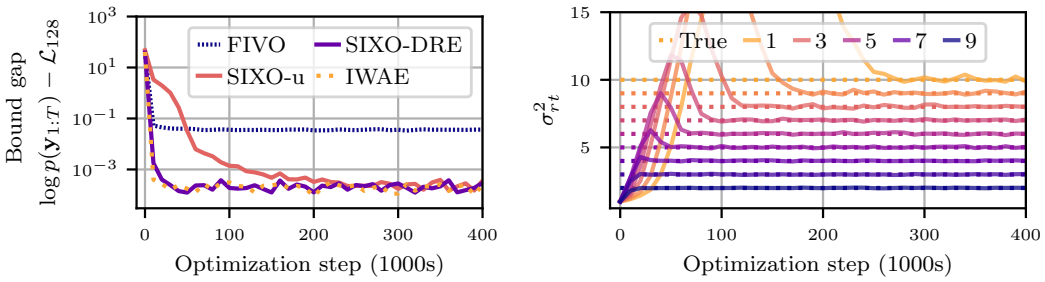
This is an important advantage of our work—the SIXO objective is the first to recover the true marginal likelihood with a finite number of particles while also being tailored to sequential tasks.

4 Related Work

Standard references for SMC include Doucet and Johansen [1], Naesseth et al. [2], and Del Moral [3] which provides a theoretical treatment of a generalization of SMC called Feynman-Kac formulae. Pitt and Shephard [17] introduced the auxiliary particle filter, an early smoothing SMC method which constructs an estimate of the one-step backwards message $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1} \mid \mathbf{x}_t)$ using simulations from the model. Smoothing SMC in general is discussed thoroughly in Briers et al. [15] and Del Moral et al. [25]. Later, Whiteley and Lee [13] introduced twisted particle filters, which perform SMC on a twisted model that approximates the smoothing distributions by multiplying the model's filtering distributions with "twisting" functions. Our work extends the theoretical framework in Whiteley and Lee [13] by proposing practical and effective methods for learning parametric twisting functions.

To make smoothing SMC computationally tractable, fixed-lag techniques use information from only a fixed window of future observations, as introduced in Clapp and Godsill [19] and surveyed in Lin et al. [20]. For example, Park and Ionides [26] use simulations from the model to estimate fixed-lag twisting functions and Doucet et al. [27] sample blocks of latents conditional on their observations via various Monte Carlo methods. These methods suffer from computational complexity that grows with the window size, and fail to take advantage of all future observations.

Other methods use twisting functions which depend on all observations. Most similar to our approach are Guarniero et al. [16], Heng et al. [28], and Zimmermann et al. [29] which learn parametric twists using a Bellman-like decomposition of the lookahead distributions $p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ in terms of the same distributions one step into the future. Our method instead uses independent DRE objectives for the twist at each timestep. Del Moral and Murray [30] use Gaussian processes to approximate the twists, Lindsten et al. [18] use traditional graphical model techniques such as loopy belief propagation and expectation propagation, and Ruiz and Kappen [31] use optimal control techniques. None of these approaches consider model learning and their twist parameterizations and learning techniques are highly specialized to specific problem settings.



(a) Bound gap for different methods.

(b) Convergence of twist parameters with SIXO-u.

Figure 2: Results for the Gaussian drift diffusion experiment presented in Section 5.1. The median across ten random seeds are shown. Further figures and discussion are included in Appendix D.1.

Fitting model parameters via stochastic gradient ascent on an evidence lower bound (ELBO) was introduced in Ranganath et al. [32], Hoffman et al. [33], Kingma and Welling [34], and later generalized to the Monte Carlo objectives (MCO) framework by Mnih and Rezende [14]. Since then, works have considered optimizing lower bounds defined by the normalizing constant estimators from multiple importance sampling [35], nested importance sampling, [36, 29], rejection sampling and Hamiltonian Monte Carlo [12], filtering SMC [8–10], and smoothing SMC [11, 37, 38]. The prior work on smoothing SMC used an objective defined by forward filtering backwards smoothing [15] which suffers from the same particle degeneracy issues as filtering SMC and cannot become tight. Kim et al. [39] optimize the importance weighted autoencoder (IWAE) bound [35] using a *baseline* derived from future likelihood estimates, but do not use SMC or resampling in their bound.

5 Experiments

We experimentally explore our claims that:

1. The SIXO bound can become tight while FIVO cannot.
2. DRE-learned twists enable better posterior inference than filtering SMC.
3. Model learning with SIXO provides better parameter estimates than FIVO.

5.1 Gaussian Drift Diffusion

We first consider a one-dimensional Gaussian drift diffusion process with joint distribution

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{1:T}, y_T\right) = \mathcal{N}\left(y_T \mid x_T + \alpha, \sigma_y^2\right) \mathcal{N}\left(x_1; \alpha, \sigma_x^2\right) \prod_{t=2}^{T} \mathcal{N}\left(x_t \mid x_{t-1} + \alpha, \sigma_x^2\right). \tag{6}$$

The single free model parameter is the drift $\alpha \in \mathbb{R}$, the state is $x_t \in \mathbb{R}$, and the observation is $y_T \in \mathbb{R}$. Figures 1a and 1b show that for $\alpha = 0$ the filtering and smoothing distributions in this model quickly diverge, which can lead to poor inference for filtering methods.

We compare joint model, proposal and twist learning using two variants of SIXO to variational inference with the IWAE bound [35] and FIVO with unbiased gradients [8–10]. All methods use an independent proposal at each time step parameterized as $q_t(x_t \mid x_{t-1}, y_T) = \mathcal{N}(x_t; f_t(x_{t-1}, y_T), \sigma_{qt}^2)$ where f_t is an affine function, a family which contains the optimal proposal. SIXO-u uses twists parameterized as $r_t(y_T, x_t) = \mathcal{N}(y_T; g_t(x_t), \sigma_{rt}^2)$ where g_t is an affine function, a family which contains the true lookahead distributions. The SIXO-DRE twist functions, $\log r_t(y_T, x_t)$, are parameterized as quadratic functions of x_t, where the parameters of the quadratic function are generated by a neural network with inputs (y_T, t). The true log density ratio will be quadratic in x_t, so if the neural network is sufficiently flexible, the true log density ratio function can be obtained.

Figure 2a shows the convergence of the variational bound for each method. As expected IWAE recovers a tight variational bound, whereas FIVO does not. While SIXO-u does recover a tight variational bound, the high variance of the unbiased gradient estimator makes it slower to converge,

Table 1: Performance of FIVO and SIXO on the SVM.

Method	Train $\mathcal{L}^4_{\text{Method}}$ (as in [9])	Train $\mathcal{L}^{2048}_{\text{BPF}}$	Test $\mathcal{L}^{2048}_{\text{BPF}}$
IWAE	$\mathbf{6940.18 \pm 1.17}$	$\mathbf{7019.38 \pm 2.99}$	3351.21 ± 2.35
FIVO	6921.29 ± 1.33	$\mathbf{7020.14 \pm 2.86}$	3352.51 ± 1.30
SIXO-q	6928.90 ± 1.24	$\mathbf{7019.65 \pm 2.97}$	3353.10 ± 1.58
SIXO-DRE	6931.51 ± 2.08	$\mathbf{7019.42 \pm 3.01}$	$\mathbf{3354.08 \pm 1.60}$

and impractical for non-toy problems. Conversely, SIXO-DRE achieves a tight bound while using the lower variance biased gradients. This motivates its use in more complex, non-linear settings where the unbiased FIVO gradients are not practical. Figure 2b shows that SIXO-u recovers the correct twist parameters. More figures illustrating the convergence of $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are included in Appendix D.1.

In Figures 1c and 1d we compare particle trajectories under FIVO and SIXO-u. We see that FIVO consistently proposes particles with high likelihood under the posterior distributions (identical to the smoothing distributions in this case) which are discarded by the resampling steps in filtering SMC. In contrast, SIXO both proposes particles with high posterior likelihood and retains them through the resampling steps by properly scoring particles under the twisted target distributions. These results empirically verify the theoretical claims made in Section 3.3.

5.2 Stochastic Volatility Model

We now apply SIXO to a stochastic volatility model (SVM) of monthly foreign exchange rates for $N = 22$ currencies, over the period 9/2007 to 8/2017 [40]. The SVM generative model is defined as

$$\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \mathbf{x}_t = \boldsymbol{\mu} + \boldsymbol{\phi} \odot (\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \boldsymbol{\nu}_t, \quad \mathbf{y}_t = \boldsymbol{\beta} \odot \exp\left(\frac{\mathbf{x}_t}{2}\right) \odot \varepsilon_t, \tag{7}$$

with transition noise $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, observation noise $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, I_{N\times N})$, states $\mathbf{x}_{1:T} \in \mathbb{R}^{T\times N}$, and observations $\mathbf{y}_{1:T} \in \mathbb{R}^{T\times N}$. All multiplications are performed element-wise, denoted by $\odot$. The proposal, $q_{\boldsymbol{\theta}}$, is structured as $q_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}) \propto \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ with means $\boldsymbol{\mu}_t \in \mathbb{R}^N$ and diagonal covariance matrices $\boldsymbol{\Sigma}_t \in \text{diag}(\mathbb{R}^N_+)$. We compare four approaches: IWAE, FIVO, SIXO with quadrature twist (SIXO-q), and SIXO with density ratio twist (SIXO-DRE). Note that the observations for this model are dense in time, so we would expect filtering-based approaches to perform well. See Appendix D.2 for more details.

Train Performance We first compare our methods in terms of their 4-particle log marginal likelihood lower bounds as in Naesseth et al. [9], e.g. for IWAE we report the IWAE bound and for FIVO we report the FIVO bound. Even though SIXO-q only scores a single future observation, it still obtains a 7-nat improvement over FIVO. SIXO-DRE, meanwhile, conditions on all future observations and obtains a 10-nat improvement over FIVO. All SMC-based methods, however, significantly underperform IWAE which seems to indicate that IWAE's unbiased gradients and use of smoothing information allow it to learn a proposal which is efficient at low numbers of particles. This performance does not carry over to model learning or other high-dimensional problems, however, and is at odds with previous work on this model and dataset [8–10].

We also estimate the learned model's training set marginal likelihood by computing a bootstrap particle filter's (BPF) log marginal lower bound with 2,048 particles, denoted $\mathcal{L}^{2048}_{\text{BPF}}$ [41]. Interestingly, a one-way ANOVA [42] does not reject the null hypothesis that the training set $\mathcal{L}^{2048}_{\text{BPF}}$ means are all equal ($p = 0.26$), suggesting that the log marginal likelihoods on training data are indistinguishable and training performance has saturated. It is clear, however, that among the SMC methods SIXO performs the best inference and makes the most efficient use of particles as the $\mathcal{L}^4_{\text{SIXO}}$ bounds are significantly higher than $\mathcal{L}^4_{\text{FIVO}}$ for similar models.

Test Performance We also compare methods on a held-out test set to evaluate model learning. We construct this test set using the same data source as the training set, but use the period of time since Naesseth et al. [9] was published, an extra 55 months. Again, we report BPF log marginal lower bounds with 2,048 particles and find that SIXO-DRE outperforms IWAE, SIXO-q and FIVO. All differences are statistically significant.

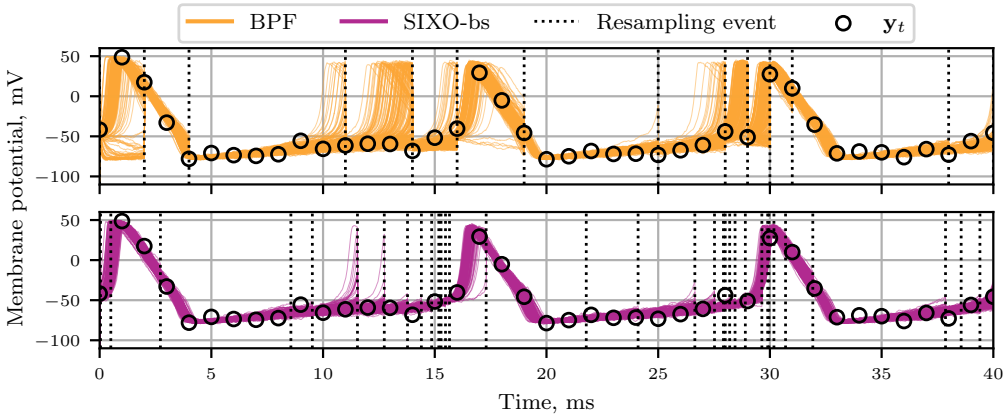


Figure 3: Comparison of trajectories generated by a BPF (top) and a SIXO-bs sweep (bottom) on synthetic data from the Hodgkin-Huxley model. Both sweeps use the true model parameters and a bootstrap proposal while SIXO-bs uses a learned twist. SIXO-bs resamples more frequently because the twist changes the SMC weights at each model step, while the BPF only changes weights at each observation. This allows SIXO-bs to resample away erroneous spikes right as they begin.

Table 2: Hodgkin-Huxley inference performance for different observation intervals.

Observation Interval	$\mathcal{L}^{32}_{\text{Method}}$ / number of observations			
	SIXO-sm	SIXO-bs	FIVO-fi	FIVO-bs
2ms	-1.66 ± 0.054	-1.75 ± 0.100	-22.80 ± 0.457	-2.57 ± 0.297
1ms	-1.18 ± 0.004	-1.36 ± 0.206	-11.86 ± 0.304	-2.36 ± 0.356
0.5ms	-1.06 ± 0.013	-1.17 ± 0.254	-6.09 ± 0.302	-2.17 ± 0.305
0.2ms	-0.92 ± 0.003	-1.00 ± 0.209	-2.49 ± 0.106	-1.93 ± 0.243

5.3 Hodgkin-Huxley Model

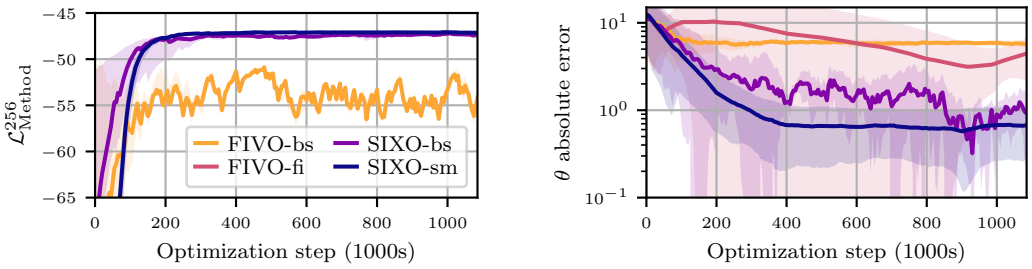
We conclude by comparing FIVO and SIXO on the Hodgkin-Huxley (HH) model of neural action potentials [43, 44]. A single neuron is represented with a four-dimensional state-space: the instantaneous membrane potential and the relative conductivity of three ion gates. A noise-corrupted and subsampled membrane potential can be obtained using electrodes [45] or voltage imaging [46]. The state of the gates, however, is not observable, and must be inferred from the noisy potential recordings. The physiological parameters governing the time-evolution of the system are also of interest, such as the base conductance of each of the ion channels.

We implement the HH model as a four-dimensional nonlinear state space model with Gaussian transition noise [47]. The observation is a single Gaussian-distributed value with mean equal to the instantaneous potential. Unless otherwise stated, we subsample observations by a factor of 50 to simulate an acquisition frequency of 1kHz (interval of 1ms). For more details, see Appendix D.3.

In this model action potentials, or spikes, are rare events that happen quickly and invoke a rapid change in the state. Therefore, filtering-based inference is particularly disadvantageous as noisy observations may trigger erroneous spikes or "miss" true spikes.

We compare four methods: SIXO with a DRE twist and a learned smoothing proposal (SIXO-sm), SIXO with a bootstrap proposal (SIXO-bs), FIVO with a learned filtering proposal (SIXO-fi), and FIVO with a bootstrap proposal (FIVO-bs).

Inference In Figure 3 we compare HH trajectories generated by a BPF and SIXO-bs. The BPF yields spurious spikes and misses the initiation of other spikes, illustrating the issue with filtering SMC for HH inference. SIXO-bs allows fewer spurious spikes to develop and fewer particles miss the onset of spikes. SIXO-bs also achieves a log marginal lower bound of -47.81 nats, higher than the -49.24 nats achieved by the BPF, showing that it performs more effective inference. In Table 2 we



(a) Validation set $\mathcal{L}^{256}_{\text{Method}}$ over training.

(b) Relative parameter error over training.

Figure 4: Hodgkin-Huxley model learning performance over training. Both SIXO-bs and SIXO-sm obtain a better parameter estimate and more stable bound than FIVO. FIVO-fi is not stable and achieves a low bound, not visible on Figure 4a.

Table 3: Hodgkin-Huxley model learning performance.

Bound	Proposal	Test $\mathcal{L}^{256}_{\text{Method}}$	Test $\mathcal{L}^{256}_{\text{BPF}}$	θ Relative Error
(True model)	(Bootstrap)	(N/A)	(-49.12)	(0.0)
FIVO	Bootstrap	-51.32 ± 0.73	-51.32 ± 0.73	0.45 ± 0.03
FIVO	Filtering	-660.62 ± 283.50	-115.74 ± 80.12	0.78 ± 0.20
SIXO	Bootstrap	-48.98 ± 0.20	-49.40 ± 0.79	0.10 ± 0.06
SIXO	**Smoothing**	$\mathbf{-48.73 \pm 0.14}$	$\mathbf{-49.30 \pm 0.71}$	$\mathbf{0.05 \pm 0.02}$

study this effect more extensively, comparing performance across different observation frequencies. We see, as predicted, that the performance of SIXO is more consistent across observation frequencies than FIVO, supporting the claim that twists assist in inference when observations are sparse. For more results, see Table 4 in Appendix D.3.

Model Learning We conclude by comparing FIVO and SIXO for parameter recovery in Figure 4 and Table 3. We see that FIVO-bs converges to a poor parameter estimate, and recovers a poor variational bound. FIVO-fi does not converge (details and full figure in Appendix D.3). The SIXO methods recover much better parameter estimates, and achieve the highest bound values, with SIXO-sm outperforming all other methods in terms of final performance and training stability.

6 Conclusions, Limitations, and Future Work

In this work we proposed a method of learning twisting functions for smoothing SMC via density ratio estimation. Our approach ascends a lower bound on the log marginal likelihood that can theoretically become tight, a first for SMC objectives. We verified our theoretical claims by experimentally demonstrating improvements over existing techniques in inference and model learning.

There are, however, important limitations to our approach. Training and evaluating the twist requires additional computational effort compared to FIVO, and requires further hyperparameter tuning. Although we consider SIXO to be mainly used for offline settings, extending SIXO to online settings could yield practical benefit. Finally, there is a known pathology in DRE methods where the ratio may be poorly estimated if the difference between the densities in the ratio is very large [48]. Thus, new methods for learning the twist are important topics for future work.

Acknowledgements and Disclosure of Funding We thank Matt MacKay for helpful discussions and edits, and the anonymous reviewers for their feedback during the review process. This work was supported by grants from the Simons Collaboration on the Global Brain (SCGB 697092), the NIH BRAIN Initiative (U19NS113201 and R01NS113119). Some of the computation for this work was made possible by Microsoft Education Azure cloud credits. Dieterich Lawson was supported in part by a Stanford Data Science Fellowship.

References

[1] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In Dan Crisan and Boris Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.

[2] Christian A. Naesseth, Fredrik Lindsten, Thomas B. Schön, et al. Elements of sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

[3] Pierre Del Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*, volume 88. Springer, 2004.

[4] Christophe Andrieu, Arnaud Doucet, Sumeetpal S. Singh, and Vladislav B. Tadic. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92 (3):423–438, 2004.

[5] Shixiang Shane Gu, Zoubin Ghahramani, and Richard E. Turner. Neural adaptive sequential Monte Carlo. *Advances in neural information processing systems*, 28, 2015.

[6] David Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[7] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[8] Chris J. Maddison, Dieterich Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Teh. Filtering variational objectives. *Advances in Neural Information Processing Systems*, 30, 2017.

[9] Christian A. Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

[10] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential Monte Carlo. In *6th International Conference on Learning Representations*, 2018.

[11] Dieterich Lawson, George Tucker, Christian A. Naesseth, Chris Maddison, Ryan P. Adams, and Yee Whye Teh. Twisted variational sequential Monte Carlo. In *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[12] Dieterich Lawson, George Tucker, Bo Dai, and Rajesh Ranganath. Energy-inspired models: Learning with sampler-induced distributions. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1): 115–141, 2014.

[14] Andriy Mnih and Danilo Rezende. Variational inference for Monte Carlo objectives. In *International Conference on Machine Learning*, pages 2188–2196. PMLR, 2016.

[15] Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

[16] Pieralberto Guarniero, Adam M. Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017.

[17] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[18] Fredrik Lindsten, Jouni Helske, and Matti Vihola. Graphical model inference: Sequential Monte Carlo meets deterministic approximations. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Tim C. Clapp and Simon J. Godsill. Fixed-lag smoothing using sequential importance sampling. In José-Miguel Bernardo, James Berger, Philip Dawid, and Adrian Smith, editors, *Bayesian Statistics 6*, pages 743–751. Citeseer, 1999.

[20] Ming Lin, Rong Chen, and Jun S. Liu. Lookahead strategies for sequential Monte Carlo. *Statistical Science*, 28(1):69–94, 2013.

[21] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1964.

[22] Adam M Johansen and Arnaud Doucet. A note on auxiliary particle filters. *Statistics & Probability Letters*, 78(12):1498–1504, 2008.

[23] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[24] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[25] Pierre Del Moral, Arnaud Doucet, and Sumeetpal Singh. Forward smoothing using sequential Monte Carlo. Technical Report CUED/F-INFENG/TR 638, Cambridge University, 2010.

[26] Joonha Park and Edward L. Ionides. Inference on high-dimensional implicit dynamic models using a guided intermediate resampling filter. *Statistics and Computing*, 30(5):1497–1522, 2020.

[27] Arnaud Doucet, Mark Briers, and Stéphane Sénécal. Efficient block sampling strategies for sequential Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 15(3): 693–711, 2006.

[28] Jeremy Heng, Adrian N. Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904–2929, Oct 2020. ISSN 0090-5364. doi: 10.1214/19-aos1914.

[29] Heiko Zimmermann, Hao Wu, Babak Esmaeili, and Jan-Willem van de Meent. Nested variational inference. *Advances in Neural Information Processing Systems*, 34:20423–20435, 2021.

[30] Pierre Del Moral and Lawrence M. Murray. Sequential Monte Carlo with highly informative observations. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):969–997, 2015.

[31] Hans-Christian Ruiz and Hilbert J. Kappen. Particle smoothing for hidden diffusion processes: Adaptive path integral smoother. *IEEE Transactions on Signal Processing*, 65(12):3191–3203, 2017.

[32] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, pages 814–822. PMLR, 2014.

[33] Matthew D. Hoffman, David Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[34] Diederik Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

[35] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations*, 2016.

[36] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. Nested sequential monte carlo methods. In *International Conference on Machine Learning*, pages 1292–1301. PMLR, 2015.

[37] Antonio Moretti, Zizhao Wang, Luhuan Wu, Iddo Drori, and Itsik Pe'er. Variational objectives for Markovian dynamics with backward simulation. In *ECAI 2020*, pages 1371–1378. IOS Press, 2020.

[38] Antonio Moretti, Zizhao Wang, Luhuan Wu, and Itsik Pe'er. Smoothing nonlinear variational objectives with sequential Monte Carlo. *ICLR Workshop: Deep Generative Models for Highly Structured Data*, 2019.

[39] Geon-Hyeong Kim, Youngsoo Jang, Hongseok Yang, and Kee-Eung Kim. Variational inference for sequential data with future likelihood estimates. In *International Conference on Machine Learning*, pages 5296–5305. PMLR, 2020.

[40] Siddhartha Chib, Yasuhiro Omori, and Manabu Asai. Multivariate stochastic volatility. In *Handbook of financial time series*, pages 365–400. Springer, 2009.

[41] Neil Gordon, David Salmond, and Adrian Smith. Novel approach to nonlinear/non-gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, April 1993.

[42] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992.

[43] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500, 1952.

[44] Peter Dayan and Laurence F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[45] Justin M. Kita and R. Mark Wightman. Microelectrodes for studying neurobiology. *Current Opinion in Chemical Biology*, 12(5):491–496, 2008.

[46] Darcy S. Peterka, Hiroto Takahashi, and Rafael Yuste. Imaging voltage in neurons. *Neuron*, 69 (1):9–21, 2011.

[47] Quentin J.M. Huys and Liam Paninski. Smoothing of, and parameter estimation from, noisy biophysical recordings. *PLoS computational biology*, 5(5):e1000379, 2009.

[48] Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

[49] Robert Price. A useful theorem for nonlinear devices having Gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

[50] Georges Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. In *Annales des Télécommunications*, volume 19-9, pages 203–220. Springer, 1964.

[51] Tim Salimans and David A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

[52] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[53] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[54] 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/google/jax.

[55] Diederik Kingma, Jimmy Ba, Yoshua Bengio, and Yann LeCun. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

[56] Charles W. Dunnett. Pairwise multiple comparisons in the unequal variance case. *Journal of the American Statistical Association*, 75(372):796–800, 1980.

[57] Derek C. Sauder and Christine E. DeMars. An updated recommendation for multiple comparisons. *Advances in Methods and Practices in Psychological Science*, 2(1):26–44, 2019.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[59] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. *Advances in neural information processing systems*, 29, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We present a method build on FIVO [8] to include smoothing information.
   (b) Did you describe the limitations of your work? [Yes] We especially highlight the persistent problem of biased gradient estimators in non-trivial models, and the dependency on the variational family used for each set of function approximators.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We foresee no negative social impacts arising from this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Sketch proofs are included in the main paper, and expanded proofs and commentaries are included in the supplement.
   (b) Did you include complete proofs of all theoretical results? [Yes] See above.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is available publicly at the URL listed.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Extensive written configurations are included in the supplement. Weights and Biases was used for experimental logging. Code released contains hyperparameters for reproduction of results.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We list the types of nodes and runtimes for the algorithms presented.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendices for SI$\mathcal{X}$O: Smoothing Inference with Twisted Objectives

A Table of Notation

Name	Symbol	Notes
Sequence length	T	$T \in \mathbb{N}$
Timestep	t	$t \in \{1, \dots, T\}$
Latent state	$\mathbf{x}_t$	$\mathbf{x}_t \in \mathcal{X}$
Observation	$\mathbf{y}_t$	$\mathbf{y}_t \in \mathcal{Y}$
Observation sequence	$\mathbf{y}_{1:T}$	$\mathbf{y}_{1:T} \in \mathcal{Y}^T$
Number of SMC particles	K	$K \in \mathbb{N}$
k^{th} particle latent trajectory	$\mathbf{x}_{1:t}^k$	$\mathbf{x}_{1:t}^k \in \mathcal{X}^t$
Model and proposal parameters	$\boldsymbol{\theta}$	$\boldsymbol{\theta} \in \boldsymbol{\Theta}$
Twist parameters	$\boldsymbol{\psi}$	$\boldsymbol{\psi} \in \boldsymbol{\Psi}$
Joint distribution	$p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$	Distribution on $\mathcal{X}^T \times \mathcal{Y}^T$
t^{th} transition distribution	$p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$	Conditional distribution on $\mathcal{X}$
t^{th} observation distribution	$p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t)$	Conditional distribution on $\mathcal{Y}$
Proposal distributions	$\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T})\}_{t=1}^T$	T conditional distributions on $\mathcal{X}$
Twist functions	$\{r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^{T-1}$	$T-1$ positive integrable functions in $\mathcal{X} \times \mathcal{Y}^{T-t-1} \times \boldsymbol{\Psi} \to \mathbb{R}_{\geq 0}$
Filtering distributions	$\{p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})\}_{t=1}^T$	T conditional distributions on $\mathcal{X}^t$
Smoothing distribution	$\{p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:T})\}_{t=1}^T$	T conditional distributions on $\mathcal{X}^t$
Unnormalized target distributions	$\{\gamma_t(\mathbf{x}_{1:t})\}_{t=1}^T$	T positive integrable functions in $\mathcal{X}^t \to \mathbb{R}_{\geq 0}$
Normalized target distributions	$\{\pi_t(\mathbf{x}_{1:t})\}_{t=1}^T$	T distributions on $\mathcal{X}^t$
t^{th} normalizing constant	Z_t	Positive real, $\pi_t(\mathbf{x}_{1:t}) = \gamma_t(\mathbf{x}_{1:t})/Z_t$
Bootstrap particle filter	BPF	SMC using $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ as the proposal and no twist.
Filtering sequential Monte Carlo	FSMC	SMC with filtering distributions as targets.
SIXO-unified	SIXO-u	SIXO objective optimized w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ through direct and unbiased gradient ascent of the SIXO bound.
SIXO-quadrature	SIXO-q	SIXO objective with r_t parameterized using the quadrature twist, and optimized using ascent of the unified SIXO bound.
SIXO-density ratio estimation	SIXO-DRE	SIXO with r_t parameterized as the density ratio estimate twist, and learned using the alternating method.
SIXO with DRE and bootstrap proposal	SIXO-bs	Used in HH experiments.
SIXO with DRE and learned smoothing proposal	SIXO-sm	Used in HH experiments.
FIVO with bootstrap proposal	FIVO-bs	Used in HH experiments. Same forward sweep as a BPF.
FIVO with learned filtering proposal	FIVO-fi	Used in HH experiments.
BPF bound	$\mathcal{L}_{\text{BPF}}^K$	Log marginal likelihood bound from a K-particle BPF.
FIVO bound	$\mathcal{L}_{\text{FIVO}}^K$	FIVO bound (2) with K particles.
SIXO bound	$\mathcal{L}_{\text{SIXO}}^K$	SIXO bound (3) with K particles.
DRE loss	$\mathcal{L}_{\text{DRE}}$	Loss used to learn twist with DRE.

B Background

B.1 Sequential Monte Carlo

Sequential Monte Carlo (SMC) is a popular method for sampling from posterior distributions with sequential structure. For a thorough introduction we refer the reader to Doucet and Johansen [1] and Naesseth et al. [2]. We reproduce the general SMC algorithm in Algorithm 2.

Algorithm 2 Sequential Monte Carlo

procedure SMC($\{\gamma_t(\mathbf{x}_{1:t})\}_{t=1}^T$, $\{q_t(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})\}_{t=1}^T$, K)
  $w_0^{1:K} = 1, \quad \widehat{Z}_0 = 1$
  for $t = 1, \ldots, T$ **do**
    for $k = 1, \ldots, K$ **do**
      $\mathbf{x}_t^k \sim q_t(\mathbf{x}_t | \mathbf{x}_{1:t-1}^k)$
      $\mathbf{x}_{1:t}^k = (\mathbf{x}_{1:t-1}^k, \mathbf{x}_t^k)$
      $\alpha_t^k = \dfrac{\gamma_t(\mathbf{x}_{1:t}^k)}{\gamma_t(\mathbf{x}_{1:t-1}^k) q_t(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}^k)}$
      $w_t^k = w_{t-1}^k \alpha_t^k$
    $\widehat{Z_t/Z_{t-1}} = \dfrac{\sum_{k=1}^K w_t^k}{\sum_{k=1}^K w_{t-1}^k}$
    $\widehat{Z}_t = \widehat{Z}_{t-1}(\widehat{Z_t/Z_{t-1}})$
    if should resample **then**
      for $k = 1, \ldots, K$ **do**
        $a_t^k \sim \text{Categorical}(w_t^{1:K})$
        $\tilde{\mathbf{x}}_{1:t}^k = \mathbf{x}_{1:t}^{a_t^k}$
      $\mathbf{x}_{1:t}^{1:K} = \tilde{\mathbf{x}}_{1:t}^{1:K}$
      $w_t^{1:K} = 1$
  return $\widehat{Z}_T$, $\mathbf{x}_{1:T}^{1:K}$

B.2 Factoring the Smoothing Distributions

Here we show that the smoothing distributions can be factored into the filtering distributions and the lookahead distributions.

Let $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$ be a model as defined in (1). Then

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{y}_{1:t}, \mathbf{x}_{1:t}) \tag{8}$$

$$= p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t), \tag{9}$$

where $\mathbf{y}_{t+1:T}$ is conditionally independent of $(\mathbf{y}_{1:t}, \mathbf{x}_{1:t-1})$ given $\mathbf{x}_t$ because of the Markov structure of $p_{\boldsymbol{\theta}}$.

C Methods

C.1 The Gradients of SMC

The original FIVO papers [8–10] used biased gradients to optimize $\mathcal{L}_{\text{FSMC}}$ which ignore score-function gradient terms arising from the discrete resampling operations. We use the same biased gradient estimator in SIXO-DRE when optimizing $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ in terms of $\boldsymbol{\theta}$. We also examine SIXO-u, where we use the full unbiased gradient estimator (see (17)). We derive both estimators here.

Assume resampling occurs at each timestep, let $\mathbf{A}_t = a_t^{1:K}$ and $\mathbf{X}_t = \mathbf{x}_t^{1:K}$ be the ancestor indices and latents for all particles at time t, and let $\mathbf{A} = (\mathbf{A}_1, \ldots, \mathbf{A}_{T-1})$ and $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$ be the

full sequence of ancestor indices and latents. We can then write the probability distribution over $\mathbf{X}, \mathbf{A}$ that defines SMC as

$$p_{\text{SMC}}(\mathbf{X}, \mathbf{A}) = \left(\prod_{k=1}^{K} q(\mathbf{x}_1^k) \right) \prod_{t=2}^{T} \prod_{k=1}^{K} q(\mathbf{x}_t^k | \mathbf{x}_{1:t-1}^{a_{t-1}^k}) \overline{\alpha}_{t-1}^{a_{t-1}^k} \tag{10}$$

where $\overline{\alpha}_t^i = \alpha_t^i / (\sum_{k=1}^{K} \alpha_t^k)$ is the normalized incremental weight. When the proposals and targets of SMC are parametric functions of $\boldsymbol{\theta}$, both q and $\overline{\alpha}$ will depend on $\boldsymbol{\theta}$.

To emphasize its dependence on $\boldsymbol{\theta}$ when run with parametric twists and proposals we will write $p_{\text{SMC}}(\mathbf{X}, \mathbf{A})$ as $p(\mathbf{X}, \mathbf{A}; \boldsymbol{\theta})$. Then the gradient of $\mathcal{L}_{\text{SIXO}}$ is defined as

$$\nabla_{\boldsymbol{\theta}}(\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta})) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}, \mathbf{A} \sim p(\mathbf{X}, \mathbf{A}; \boldsymbol{\theta})} [\log \widehat{Z}(\mathbf{X}, \mathbf{A}, \boldsymbol{\theta})] \tag{11}$$

where we have rewritten $\widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ as $\widehat{Z}(\mathbf{X}, \mathbf{A}, \boldsymbol{\theta})$ to emphasize its dependence on the random variables $\mathbf{X}$ and $\mathbf{A}$ and suppress its dependence on $\boldsymbol{\psi}$ and $\mathbf{y}_{1:T}$.

The first step is to reparameterize the expectation in terms of continuous noise instead of $\mathbf{X}$ [49, 50, 34, 51, 52]. Assume $\mathbf{X}$ is from a reparameterizable distribution and let $\phi_{\mathbf{X}}(\boldsymbol{\theta}, \epsilon)$ be a function that deterministically combines continuous noise ϵ and the parameters $\boldsymbol{\theta}$ to produce a sample $\mathbf{X}$. Then we have

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}, \mathbf{A}} [\log \widehat{Z}(\mathbf{X}, \mathbf{A}, \boldsymbol{\theta})] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\epsilon, \mathbf{A}} [\log \widehat{Z}(\phi_{\mathbf{X}}(\boldsymbol{\theta}, \epsilon), \mathbf{A}, \boldsymbol{\theta})]. \tag{12}$$

We will further abuse notation by writing $\widehat{Z}(\phi_{\mathbf{X}}(\boldsymbol{\theta}, \epsilon), \mathbf{A}, \boldsymbol{\theta})$ as $\widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta})$. Rewriting the expectation (12) as an integral gives

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\epsilon, \mathbf{A}} [\log \hat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta})] = \nabla_{\boldsymbol{\theta}} \int \log \hat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) p(\epsilon, \mathbf{A}; \boldsymbol{\theta}) d\epsilon d\mathbf{A}. \tag{13}$$

Assuming that we can differentiate under the integral allows us to break the integrand apart using the product rule as

$$\int \nabla_{\boldsymbol{\theta}} (\log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta})) p(\epsilon, \mathbf{A}; \boldsymbol{\theta}) + \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} (p(\epsilon, \mathbf{A}; \boldsymbol{\theta})) d\epsilon d\mathbf{A}. \tag{14}$$

The left hand term in the integrand of (14) equals $\mathbb{E}_{\epsilon, \mathbf{A}} [\nabla_{\boldsymbol{\theta}} \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta})]$, the expectation of a gradient that can be estimated using simple Monte Carlo. The right hand term is a "score-function gradient" [53] which can be rewritten using the fact that $\nabla_{\boldsymbol{\theta}}(\log f(\boldsymbol{\theta})) = \nabla_{\boldsymbol{\theta}}(f(\boldsymbol{\theta}))/f(\boldsymbol{\theta})$ as

$$\int \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} (\log p(\epsilon, \mathbf{A}; \boldsymbol{\theta})) p(\epsilon, \mathbf{A}; \boldsymbol{\theta}) d\epsilon d\mathbf{A} \tag{15}$$

which in turn equals the expectation

$$\mathbb{E}_{\epsilon, \mathbf{A}} \left[\log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\epsilon, \mathbf{A}; \boldsymbol{\theta}) \right]. \tag{16}$$

Writing both terms together gives the full unbiased gradient that is amenable to estimation with simple Monte Carlo,

$$\mathbb{E}_{\epsilon, \mathbf{A}} \left[\nabla_{\boldsymbol{\theta}} \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) + \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\epsilon, \mathbf{A}; \boldsymbol{\theta}) \right]. \tag{17}$$

Similar to prior work [8–11] we find that the term on the right hand side of (17) has prohibitively high variance which inhibits learning. Dropping it gives the biased SMC gradient estimator used in SIXO-DRE,

$$\mathbb{E}_{\epsilon, \mathbf{A}} [\nabla_{\boldsymbol{\theta}} \log \widehat{Z}(\epsilon, \mathbf{A}, \boldsymbol{\theta})], \tag{18}$$

which can be estimated using open-source autodiff software [54].

The derivation above is adaptable for any resampling schedule that does not depend on the parameters (and by extension, the weights), but many common resampling schemes such as effective sample size resampling do not meet this requirement. If the resampling scheme depends on the parameters of the model, it introduces additional gradient terms which are not described here. Thus for all methods in experiments where compare to SIXO-u we use a fixed resampling schedule.

C.2 Density Ratio Estimation

Density ratio estimation (DRE) considers estimating ratios of densities, e.g. $a(x)/b(x)$ with $a(x)$ and $b(x)$ defined on the same probability space and $b(x) > 0$ for all x. Instead of estimating $a(x)$ and $b(x)$ individually and then forming the ratio, an alternative approach is to directly estimate the *odds* that a given sample of x was drawn from a.

Let $p(x, z) = p(z)p(x \mid z)$ be an expanded generative model for x defined as

$$z \sim \text{Bernoulli}(\alpha), \tag{19}$$

$$x \sim a(x) \quad \text{if} \quad z = 1, \tag{20}$$

$$x \sim b(x) \quad \text{if} \quad z = 0 \tag{21}$$

with $\alpha \in (0, 1)$. We can now write the density ratio in terms of conditionals in this generative model,

$$a(x)/b(x) = p(x \mid z = 1) \,/\, p(x \mid z = 0) \tag{22}$$

$$= \left(\frac{p(x)p(z = 1 \mid x)}{p(z = 1)}\right) / \left(\frac{p(x)p(z = 0 \mid x)}{p(z = 0)}\right), \tag{23}$$

$$= \left(\frac{p(z = 0)}{p(z = 1)}\right) / \left(\frac{p(z = 0 \mid x)}{p(z = 1 \mid x)}\right), \tag{24}$$

$$= \left(\frac{1 - \alpha}{\alpha}\right) \left(\frac{p(z = 1 \mid x)}{p(z = 0 \mid x)}\right). \tag{25}$$

Thus, the density ratio can be rewritten as proportional to the odds that x was drawn from $a(x)$ instead of $b(x)$.

Density ratio estimation via classification suggests training a binary classifier with supervised learning to predict z given x [24, 23]. Let $\sigma(x) = 1/(1 + e^{-x})$ be the sigmoid function and let $g_{\boldsymbol{\psi}}(x)$ be a classifier trained with Bernoulli loss to maximize the log probability of a dataset $z_{1:N}, x_{1:N}$ sampled IID from $p(x, z)$. Specifically, $\boldsymbol{\psi}$ is fit by minimizing $\mathcal{L}_{\text{DRE}}(\boldsymbol{\psi})$, defined as

$$\mathcal{L}_{\text{DRE}}(\boldsymbol{\psi}) \triangleq \mathbb{E}_{z_{1:N}, x_{1:N} \sim p(x,z)} \left[\log \prod_{i=1}^{N} \text{Bernoulli}(z; \sigma(g_{\boldsymbol{\psi}}(x))\right], \tag{26}$$

$$= \mathbb{E}_{z_{1:N}, x_{1:N} \sim p(x,z)} \left[\sum_{i=1}^{N} z_i \log(\sigma(g_{\boldsymbol{\psi}}(x_i))) + (1 - z_i) \log(\sigma(g_{\boldsymbol{\psi}}(x_i)))\right]. \tag{27}$$

If trained in this way, the raw output of $g_{\boldsymbol{\psi}}(x)$ will approximate the log-odds that x came from $a(x)$ instead of $b(x)$, i.e.

$$g_{\boldsymbol{\psi}}(x) \approx \log\left(\frac{p(z = 1 \mid x)}{1 - p(z = 1 \mid x)}\right) = \log\left(\frac{p(z = 1 \mid x)}{p(z = 0 \mid x)}\right). \tag{28}$$

The log of the density ratio can then be expressed as

$$\log a(x)/b(x) = \log(1 - \alpha) - \log(\alpha) + \log\left(\frac{p(z = 1 \mid x)}{p(z = 0 \mid x)}\right), \tag{29}$$

$$\approx \log(1 - \alpha) - \log(\alpha) + g_{\boldsymbol{\psi}}(x). \tag{30}$$

Assuming a fixed α parameter, the log-ratio of densities is then proportional to the logit produced by $g_{\boldsymbol{\psi}}$ up to an additive constant. Thus as long as we can sample training pairs (x, z) from the expanded generative model above, we can estimate ratios of densities by training a binary classifier.

C.3 Backwards Density Ratio Twist Architecture

We now provide further implementation details for the the backwards DRE twist we propose. The twist is defined through two neural networks, as illustrated in Figure 5. First, a recurrent neural network (RNN) is passed *backwards* over the data to create a sequence of encodings, denoted $\mathbf{e}_{2:T}$. These backward encodings are computed upfront, before the SMC sweep, with a cost complexity that

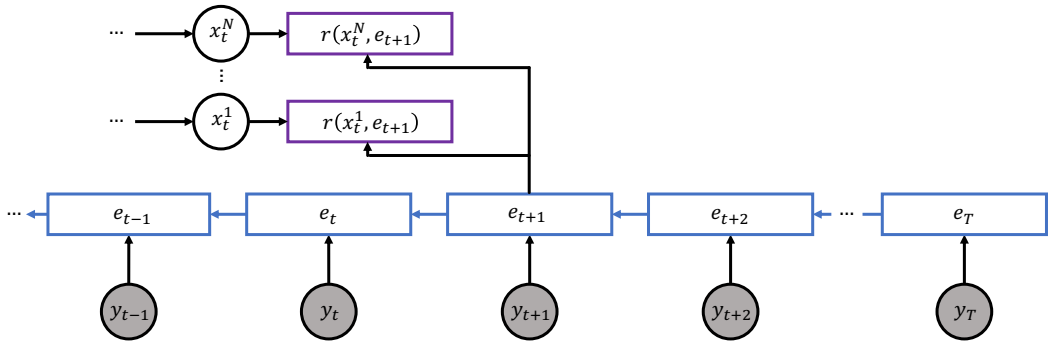


Figure 5: The backwards DRE twist architecture. Shown in blue is the recurrent backwards encoding, denoted here as $\mathbf{e}_t$, computed before the sweep. This computation will be shared across all particles. At time t, the twist value for each particle is computed by taking the backwards encoding at the next timestep, $\mathbf{e}_{t+1}$, concatenating it with the particle, and passing it through an MLP, shown in purple. As such, the twist complexity is linear in both time and the number of particles.

is linear in the length of the sequence. To evaluate the twist value of the nth particle at time t, denoted r_t^n, a second "head" MLP is used. The head MLP accepts as inputs the backwards encoding at the *next* timestep, $\mathbf{e}_{t+1}$, and the particle value, $\mathbf{x}_t^n$, and produces a single real-valued scalar, representing the logarithm of the twist value, $\log r_t^n$. The total cost of application of the head MLP is linear in time and the number of particles. As the head MLP reuses the backwards encoding, the cost of applying the backwards RNN is amortized across the number of particles used in the forward sweep. Note that no twist is computed at $t = T$, i.e. $r_T^k \triangleq 1$.

C.4 Alternating Density Ratio Twist Training

To train the density ratio twist functions we define a supervised maximum likelihood update that is applied offline from any update to the model and proposal parameters. This update is shown in Algorithm 1, labeled as DRE.

To define this update, we use the approach in Section C.2 and set $a(x) = p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})$ and $b(x) = p_{\boldsymbol{\theta}}(\mathbf{x}_t)$. To generate positive and negative examples for DRE, we first sample a set of M latent state and observation trajectories from the generative model, $\mathbf{x}_{1:T}^{1:M}, \mathbf{y}_{1:T}^{1:M} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$. Because these are samples from joint distribution they are also samples from the conditionals $\{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})\}_{t=1}^{T-1}$. We will refer to these samples as *positive samples*. We can then draw a second set of samples, but discard the observed data, $\tilde{\mathbf{x}}_{1:T}^{1:M} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T})$. We will refer to these as *negative samples*. The sets of positive and negative examples form the data on which we will train the twist classifier. Generating examples sequentially in this manner is cheap and parallelizable, allowing us to use relatively large values of M.

To train the DRE twist, we first pass the RNN backwards over sampled synthetic observed data $\mathbf{y}_{2:T}^m$ to generate the sequence of encodings $\mathbf{e}_{2:T}^m$ (noting that we flip the resulting encodings so they are "forward" in time). To evaluate the probability of a positive classification we concatenate the encoding $\mathbf{e}_{t+1}^m$ with $\mathbf{x}_t^m$ at each timestep, feed the concatenation into the head MLP, and take the output as a positive example Bernoulli logit. To evaluate the probability of a negative classification we take the same sequence of encodings, $\mathbf{e}_{2:T}^m$, concatenate them at each time step with $\tilde{\mathbf{x}}_{1:T-1}^m$, feed the result at each timestep into the head MLP, and take the result as a negative example Bernoulli logit. These outputs are used to compute the cross-entropy loss as written in Algorithm 1 at each timestep, which we average across time and across positive and negative examples to create the final loss. This approach allows us to estimate an entire sequence of ratios, $\{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})/p_{\boldsymbol{\theta}}(\mathbf{x}_t)\}_{t=1}^{T-1}$, for positive and negative examples using a single RNN backwards pass.

C.5 The SIXO Bound Can Become Tight

Proposition 1. (Reproduced from Section 3.3) Sharpness of the SIXO bound. *Let $p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$ be a latent variable model with Markovian structure as defined in Section 2, let $\mathcal{Q}$ be the set of*

possible sequences of proposal distributions indexed by parameters $\boldsymbol{\theta} \in \Theta$, and let $\mathcal{R}$ be the set of possible sequences of positive, integrable twist functions indexed by parameters $\boldsymbol{\psi} \in \Psi$. Assume that $\{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T})\}_{t=1}^{T} \in \mathcal{Q}$ and $\{p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)\}_{t=1}^{T-1} \in \mathcal{R}$. Finally, assume $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ has the unique optimizer $\boldsymbol{\theta}^*, \boldsymbol{\psi}^* = \arg\max_{\boldsymbol{\theta}\in\Theta, \boldsymbol{\psi}\in\Psi} \mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$.

Then the following holds:

1. $q_{\boldsymbol{\theta}^*}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T})$ *for* $t = 1, \ldots, T$,
2. $r_{\boldsymbol{\psi}^*}(\mathbf{y}_{t+1:T}, \mathbf{x}_t) \propto p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$ *up to a constant independent of* $\mathbf{x}_t$ *for* $t = 1, \ldots, T-1$,
3. $\mathcal{L}^K_{\text{SIXO}}(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*, \mathbf{y}_{1:T}) = \log p(\mathbf{y}_{1:T})$ *for any number of particles* $K \geq 1$.

Proof. First we reproduce the proof in the main text that $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T}) \leq p(\mathbf{y}_{1:T})$ for any setting of $\boldsymbol{\theta}, \boldsymbol{\psi}$. As previously, fix $r_{\boldsymbol{\psi}}(x_T) = 1$ and let $\widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ be the normalizing constant estimator returned by SMC run with targets $\{p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^{T}$ and proposals $\{q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T}))\}_{t=1}^{T}$. Then,

$$\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T}) \triangleq \mathbb{E}\left[\log \widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})\right] \tag{31}$$

$$\leq \log \mathbb{E}\left[\widehat{Z}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})\right] \tag{32}$$

$$= \log p(\mathbf{y}_{1:T}), \tag{33}$$

where (32) holds by Jensen's inequality and the concavity of $\log$, and (33) holds by the unbiasedness of SMC's marginal likelihood estimator.

Because $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T}) \leq p(\mathbf{y}_{1:T})$ and we assume $\mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$ has a unique optimizer, any setting of $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ that causes the bound to hold with equality must be $\boldsymbol{\theta}^*, \boldsymbol{\psi}^* = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\psi}} \mathcal{L}_{\text{SIXO}}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{y}_{1:T})$. Thus, we conclude the proof by showing that the twists $\{p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)\}_{t=1}^{T-1}$ and proposals $\{p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T})\}_{t=1}^{T}$ cause the bound to hold with equality.

We proceed by induction on t, the timestep in the SMC sweep. We will show that for $t = 1, \ldots, T$, $\widehat{Z}_t = p(\mathbf{y}_{1:T})$. In proving this we will also show that for each t, w_t^k either equals 1 or $p(\mathbf{y}_{1:T})$ for $k = 1, \ldots, K$, depending on whether resampling occurred.

For $t = 1$ note that

1. $\gamma_1(\mathbf{x}_1^k) = p(\mathbf{x}_1^k, \mathbf{y}_1) p(\mathbf{y}_{2:T} \mid \mathbf{x}_1^k)$,
2. $\gamma_0 \triangleq 1$,
3. and $q_1(\mathbf{x}_1^k) = p(\mathbf{x}_1^k \mid \mathbf{y}_{1:T})$.

Taken together this implies that the incremental weight α_1^k is

$$\alpha_1^k = \frac{p(\mathbf{x}_1^k, \mathbf{y}_1) p(\mathbf{y}_{2:T} \mid \mathbf{x}_1^k)}{p(\mathbf{x}_1^k \mid \mathbf{y}_{1:T})} = \frac{p(\mathbf{x}_1^k, \mathbf{y}_{1:T})}{p(\mathbf{x}_1^k \mid \mathbf{y}_{1:T})} = p(\mathbf{y}_{1:T}) \tag{34}$$

which does not depend on k. Because $w_0^k \triangleq 1$, we have that $w_1^k = w_0^k \alpha_1^k = p(\mathbf{y}_{1:T})$ for all k. This in turn implies

$$\widehat{Z_1/Z_0} = \frac{\sum_{k=1}^{K} w_1^k}{\sum_{k=1}^{K} w_0^k} = \frac{K p(\mathbf{y}_{1:T})}{K} = p(\mathbf{y}_{1:T}), \tag{35}$$

which when combined with the fact that $\widehat{Z}_0 \triangleq 1$ yields

$$\widehat{Z}_1 = \widehat{Z}_0 (\widehat{Z_1/Z_0}) = p(\mathbf{y}_{1:T}). \tag{36}$$

If resampling occurs at the end of step 1, all weights $w_1^{1:K}$ will be set to 1. Thus we have shown that $\widehat{Z}_1 = p(\mathbf{y}_{1:T})$ and $w_1^{1:K} = p(\mathbf{y}_{1:T})$ or 1.

Now assume that $\widehat{Z}_{t-1} = p(\mathbf{y}_{1:T})$ and $w_{t-1}^{1:K}$ equals 1 or $p(\mathbf{y}_{1:T})$. Again, we derive the incremental weights α_t^k by noting that

1. $\gamma_t(\mathbf{x}_{1:t}^k) = p(\mathbf{x}_{1:t}^k, \mathbf{y}_{1:t})p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t^k)$,
2. $\gamma_{t-1}(\mathbf{x}_{1:t-1}^k) = p(\mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:t-1})p(\mathbf{y}_{t:T} \mid \mathbf{x}_{t-1}^k)$,
3. and $q_t(\mathbf{x}_t^k) = p(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})$

which yields α_t^k as

$$\alpha_t^k = \frac{p(\mathbf{x}_{1:t}^k, \mathbf{y}_{1:t})p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t^k)}{p(\mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:t-1})p(\mathbf{y}_{t:T} \mid \mathbf{x}_{t-1}^k)p(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})} \tag{37}$$

$$= \frac{p(\mathbf{x}_{1:t}^k, \mathbf{y}_{1:T})}{p(\mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})p(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})} \tag{38}$$

$$= \frac{p(\mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})p(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:T})}{p(\mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})p(\mathbf{x}_t^k \mid \mathbf{x}_{1:t-1}^k, \mathbf{y}_{1:T})} \tag{39}$$

$$= 1 \tag{40}$$

for $k = 1, \ldots, K$.

Now there are two cases depending on the value of the weights at the previous timestep. If $w_{t-1}^{1:K} = 1$, then $w_t^k = w_{t-1}^k \alpha_t^k = 1$ for all k, implying that $\widehat{Z_t/Z_{t-1}} = 1$. Alternatively, if $w_{t-1}^{1:K} = p(\mathbf{y}_{1:T})$ then $w_t^k = p(\mathbf{y}_{1:T})$ for all k which also implies that $\widehat{Z_t/Z_{t-1}} = 1$. Given that $\widehat{Z_t/Z_{t-1}} = 1$ in both cases, and that $\widehat{Z}_{t-1} = p(\mathbf{y}_{1:T})$, we have that

$$\widehat{Z}_t = \widehat{Z}_{t-1}(\widehat{Z_t/Z_{t-1}}) = p(\mathbf{y}_{1:T}). \tag{41}$$

Finally, if resampling occurs then the weights $w_t^{1:K}$ will be set to 1. Thus we have shown that $\widehat{Z}_t = p(\mathbf{y}_{1:T})$ and $w_t^{1:K} = p(\mathbf{y}_{1:T})$ or 1 for each $t = 1, \ldots, T$.

We note that we have used $r_{\boldsymbol{\psi}^*}(\mathbf{y}_{t+1:T}, \mathbf{x}_t) = p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$. In the case where $r_{\boldsymbol{\psi}^*}(\mathbf{y}_{t+1:T}, \mathbf{x}_t) \propto p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$, we have that $\widehat{Z}_t \propto p(\mathbf{y}_{1:T})$ for $t < T$. All the proportionality constants for each $r_{\boldsymbol{\psi}^*}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)$ cancel out in (41), and hence we still obtain $\widehat{Z}_T = p(\mathbf{y}_{1:T})$ as required. □

Note that we have incidentally shown that all weights are equal at each step of SMC for the optimal proposals and twisting functions. This implies that the variance of the importance weights is minimized (i.e. is 0), and if effective sample size is used to trigger resampling, resampling will never occur.

D Experiments

Code for reproduction of all experiments is released here: https://github.com/lindermanlab/sixo.

D.1 Gaussian Drift Diffusion

D.1.1 Model Details

The one-dimensional Gaussian drift-diffusion process has joint distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, y_T) = p_{\boldsymbol{\theta}}(x_1)\left(\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(x_t \mid x_{t-1})\right) p_{\boldsymbol{\theta}}(y_T \mid x_T),$$

$$= \mathcal{N}(x_1\,; \alpha, 1)\left(\prod_{t=2}^{T} \mathcal{N}(x_t\,;\, x_{t-1} + \alpha, 1)\right)\mathcal{N}(y_T\,;\, x_T + \alpha, 1),$$

where the free parameters of the model are $\boldsymbol{\theta} = \{\alpha\} \in \boldsymbol{\Theta} = \mathbb{R}$, the state is $x_t \in \mathcal{X} = \mathbb{R}$, and the observed data are $\mathbf{y}_{1:T} = y_T \in \mathbb{R}$. Training data are sampled from this joint distribution with $\alpha = 1$. Note that the distributions we show in Figure 1 were generated with $\alpha = 0$ for clarity.

D.1.2 Analytic Forms

The tth marginal of the filtering distribution for $t < T$ is

$$p_{\boldsymbol{\theta}}(x_t) \;=\; \mathcal{N}(x_t \;;\; t\alpha, t). \tag{42}$$

The tth marginal of the smoothing distributions can be derived as follows:

$$\begin{aligned} p(x_t \mid y_T) &\propto p(x_t)p(y_T \mid x_t), && (43)\\ &= \mathcal{N}\left(x_t; t\alpha, t\right) \mathcal{N}\left(y_T; x_t + \alpha(T-t+1), T-t+1\right), \\ &= \mathcal{N}\left(x_t; t\alpha, t\right) \mathcal{N}\left(x_t; y_T - \alpha(T-t+1), T-t+1\right). && (44) \end{aligned}$$

Noting that the product of two Gaussian densities is also Gaussian:

$$\mathcal{N}\left(x; \mu_1, \sigma_1^2\right) \mathcal{N}\left(x; \mu_2, \sigma_2^2\right) \propto \mathcal{N}\left(x; \frac{\sigma_2^2\mu_1 + \sigma_1^2\mu_2}{\sigma_1^2 + \sigma_2^2}, \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\right),$$

allows us to combine the two Gaussian distributions in (44):

$$\begin{aligned} p(x_t \mid y_T) &\propto \mathcal{N}\left(x_t; \frac{(T-t+1)t\alpha + t(y_T - \alpha(T-t+1))}{t + (T-t+1)}t, \frac{t(T-t+1)}{t+(T-t+1)}\right) \\ &= \mathcal{N}\left(x_t; \frac{t}{T+1}y_T, \frac{t(T-t+1)}{T+1}\right). \end{aligned}$$

Hence the smoothing distribution is a Gaussian distribution with analytically computable mean and variance terms.

The filtering and smoothing distributions are equal at $t = T$. It is interesting to note that for $\alpha = \frac{y_T}{T+1}$, which is the maximum likelihood drift parameter for a *single* datapoint y_T, the sequence of filtering and smoothing distributions have the same means. However, the variances are different for all $t < T$; in particular, the smoothing distribution variance peaks in the middle of the timeseries, whereas the variance of the filtering distribution is monotonically increasing for $t < T$, and then drops at $t = T$.

According to Proposition 1, we expect the proposal recovered by SIXO, q_{θ_t}, to match the conditional of the smoothing distribution:

$$\begin{aligned} q_{\boldsymbol{\theta}_1}(x_1 \mid y_T) = p_{\boldsymbol{\theta}}(x_1 \mid y_T) \quad &= \mathcal{N}\left(x_1 \;;\; \frac{y_T}{T+1}, \frac{T}{T+1}\right) \quad \text{for } t = 1, \\ q_{\boldsymbol{\theta}_t}(x_t \mid x_{t-1}, y_T) = p_{\boldsymbol{\theta}}(x_t \mid x_{t-1}, y_T) \;&= \mathcal{N}\left(x_t \;;\; \frac{(T-t+1)x_{t-1} + y_T}{T-t+2}, \frac{T-t+1}{T-t+2}\right) \quad \text{otherwise}. \end{aligned}$$

Note that the mean is an affine function with bias equal to zero.

Furthermore, we also expect the optimal twist distribution to be equal to the true lookahead distribution:

$$r_{\boldsymbol{\psi}_t}(y_T \mid x_t) \;=\; p_{\boldsymbol{\theta}}(y_T \mid x_t) \;=\; \mathcal{N}\left(y_T \;;\; x_t + \alpha(T-t+1), T-t+1\right) \quad \forall t \in 1, \ldots, T-1. \tag{45}$$

D.1.3 SIXO Variants for the Gaussian Drift Diffusion

For all experiments we use a Gaussian proposal at each timestep, parameterized as $q_{\boldsymbol{\theta}_t}(x_t \mid x_{t-1}, y_T) = \mathcal{N}(x_t; f_t(x_{t-1}, y_T), \sigma_{qt}^2)$ where f_t is a general affine function of x_{t-1} and y_T. There are therefore $4T-1$ proposal parameters to learn (T biases, T y_T coefficients, $T-1$ x_{t-1} coefficients, and T variances σ_{qt}^2).

We test four variants of SIXO:

1. **SIXO-u** learns $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ by gradient ascent on the unified bound given in (3) using the unbiased gradients (reparameterization and resampling gradients). We parameterize the twists as $r_t(y_T, x_t) = \mathcal{N}(y_T; g_t(x_t), \sigma_{rt}^2)$ for $t < T$, where g_t is a learnable affine function and σ_{rt}^2 is also learned.

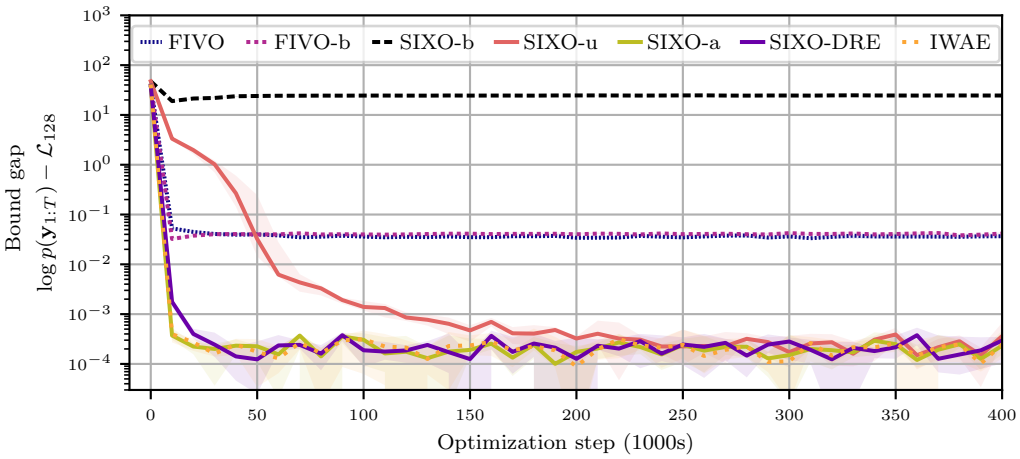


Figure 6: Convergence of the bound for all methods discussed in Section D.1. Of these lines, FIVO-b, SIXO-b and SIXO-a were omitted from Figure 2 the main text. Median and quartiles across ten random seeds are shown.

2. **SIXO-DRE** as defined in Algorithm 1 learns $\boldsymbol{\theta}$ by ascending the bound (3) *using the biased reparameterization gradients* as in (18). The twist parameters $\boldsymbol{\psi}$ are then fit using a density ratio update. The twist is parameterized as an MLP that produces the coefficients of a quadratic function over $\mathbf{x}_t$ as a function of y_T and t. This quadratic function is evaluated at $\mathbf{x}_t$ to compute the $\log r_\psi$ value.

3. **SIXO-a** (not included in Figure 2) uses the analytic form for the twist as a function of $\boldsymbol{\theta}$ and y_T (specified in (45)). There are no free parameters to learn for this twist, and $\boldsymbol{\theta}$ is learned by ascending the bound (3) using the biased reparameterization gradients in (18).

4. **SIXO-b** (not included in Figure 2) learns $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ by gradient ascent on the unified bound given in (3) using biased reparameterization gradients (18). We parameterize the twists as $r_t(y_T, x_t) = \mathcal{N}(y_T; g_t(x_t), \sigma^2_{rt})$ for $t < T$, where g_t is a learnable affine function and σ^2_{rt} is also learned.

Note that the true distributions lie within the variational families (assuming a sufficiently expressive MLP for SIXO-DRE, which is not unreasonable). In all of these models we initialize the parameter $\alpha = 0$.

D.1.4 Results

We compare the four different variants of SIXO to IWAE, FIVO with biased gradients, and FIVO with unbiased gradients. For clarity, we omitted some of these comparisons from Figure 2 in the main text, but include them here in Figure 6. Individual seeds for each experiment were run using two CPU cores, 8Gb of memory, and had a runtime of no longer than five hours. We sample 1,000 synthetic trajectories of length $T = 10$ from the model and perform minibatch stochastic gradient descent using the ADAM optimizer [55] with a learning rate of 1×10^{-3}, a minibatch size of 32, and 10 particles. In the case of SIXO-DRE we alternate between 1,000 updates to the model and proposal and 1,000 twist updates, where the tilt minibatch size is 64. We report and plot the means across ten random seeds. The variances for FIVO, FIVO-b, SIXO-b are too small to be seen.

The model drift α is initialized to zero, affine function weights are initialized to zero, affine function biases are initialized to zero, and σ^2_{qt} and σ^2_{rt} are initialized to one.

In Figure 6 we show the convergence of the bound across all seven methods we considered. FIVO with unbiased gradients (FIVO) performs comparably to FIVO with biased gradients (FIVO-b), however, converges slightly more slowly. We find that SIXO-b performed worse than all the other methods, similar to the results in Lawson et al. [11]. We therefore omitted it from the main paper for brevity. However, this result motivated us to find alternative methods.

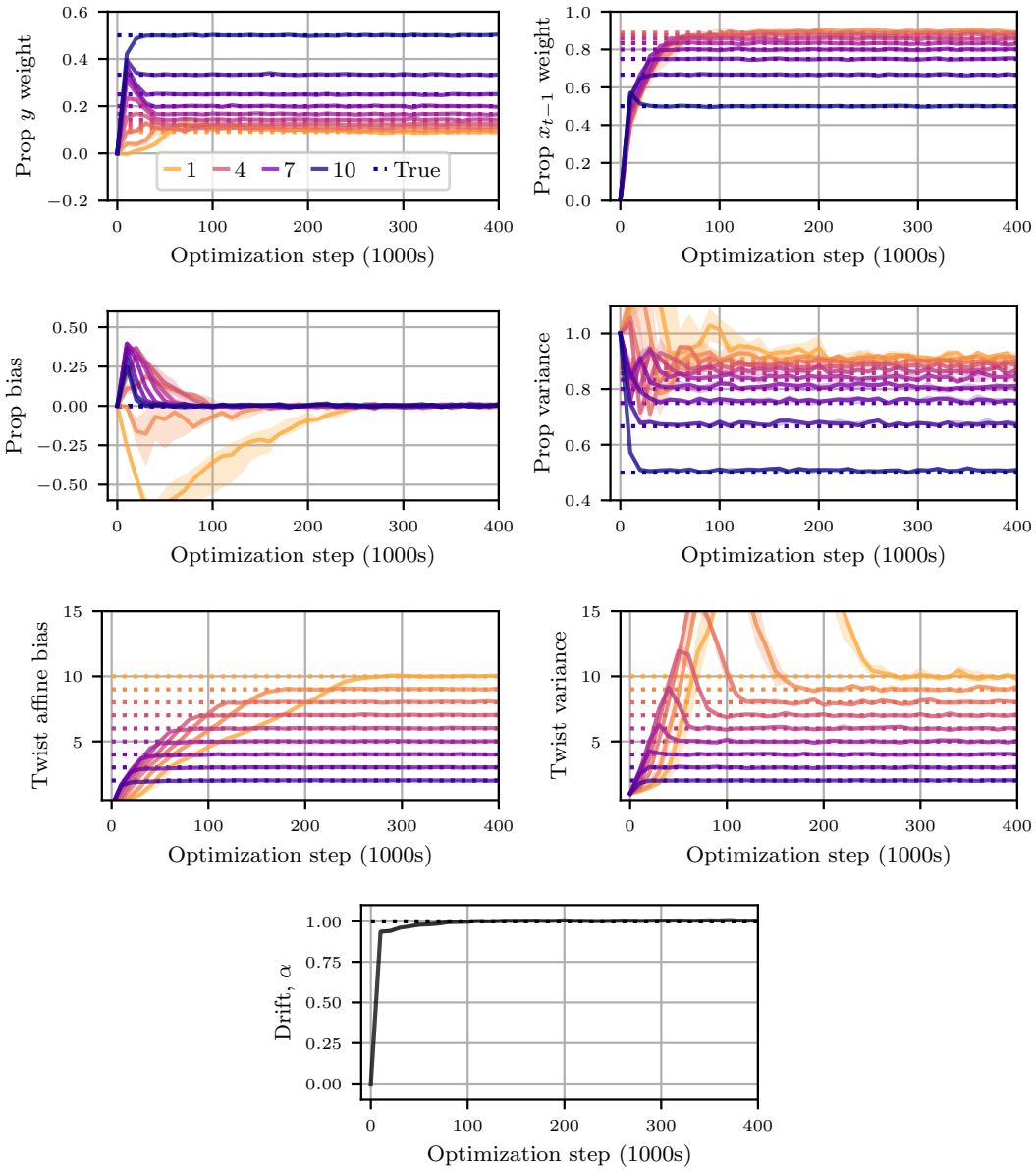


Figure 7: Convergence for all free parameters in the GDD experiment using SIXO-u. The drift parameter (α) is constant across time. Coloring of lines indicates the time $t \in 1, \ldots, 10$. Median and quartiles across ten random seeds are shown.

All of SIXO-u, SIXO-DRE and SIXO-a converge to the correct solution and achieve a tight variational bound (parameter convergence is shown in Figure 7 for SIXO-u). SIXO-u converges to a tight bound most slowly, but SIXO-DRE and SIXO-a converge quickly at a rate similar to IWAE. This result shows that SIXO is able to recover the optimal model, true posterior, and true twists, and that it can recover a tight variational bound. It also shows that SIXO-DRE converges in a rate commensurate with the best-case convergence of SIXO-a. Future work will investigate whether the use of a twist makes the reparameterization gradient less biased.

Figures 1c and 1d show the lineages for two SMC sweeps: one using a model and proposal learned using FIVO, and one using a model, proposal and twist learned using SIXO-u. We resample at every timestep using systematic resampling. In each color we show the unweighted filtering particles prior to resampling (i.e. the propagated particles) at time t, and then the smoothing particles after

performing a backwards pass from that timestep. As such, we are showing the family of smoothing and predictive distributions. We see that FIVO proposes particles towards the observation, but that these particles are often preferentially resampled back towards the prior distribution. This results in gross particle degeneracy. In contrast, the lineages for SIXO are nearly perfect, with every particle being resampled exactly once.

D.2 Stochastic Volatility Model (SVM)

D.2.1 Model Details

The SVM models an N-dimensional state-space is defined as follows:

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}\right) = p_{\boldsymbol{\theta}}\left(\mathbf{x}_1\right) p_{\boldsymbol{\theta}}\left(\mathbf{y}_1 \mid \mathbf{x}_1\right) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t), \tag{46}$$

$$\mathbf{x}_1 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right), \quad \mathbf{x}_t = \boldsymbol{\mu} + \boldsymbol{\phi}\left(\mathbf{x}_{t-1} - \boldsymbol{\mu}\right) + \boldsymbol{\nu}_t, \quad \mathbf{y}_t = \boldsymbol{\beta} \exp\left(\frac{\mathbf{x}_t}{2}\right) \boldsymbol{\varepsilon}_t, \tag{47}$$

where the states and observations are defined as:

$$\mathbf{x}_{1:T} = \mathbf{x}_{1:T} \in \mathcal{X}^T = \mathbb{R}^{T \times N}, \quad \mathbf{y}_{1:T} = \mathbf{y}_{1:T} \in \mathcal{Y}^T = \mathbb{R}^{T \times N}, \tag{48}$$

the transition and observation noise terms are defined as:

$$\boldsymbol{\nu}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right), \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbb{I}^{N \times N}\right). \tag{49}$$

and all multiplications are performed element-wise. The model has free parameters defined as:

$$\theta = \left\{\boldsymbol{\mu}, \boldsymbol{\phi}, \boldsymbol{\beta}, \mathbf{Q}\right\}, \quad \text{where} \quad \boldsymbol{\mu} \in \mathbb{R}^N, \quad \boldsymbol{\phi} \in [0,1]^N, \quad \boldsymbol{\beta} \in \mathbb{R}_+^N, \quad \mathbf{Q} \in \operatorname{diag}(\mathbb{R}_+^N), \tag{50}$$

such that there are $4N$ model parameters. The model learning objective is to recover the free parameters, $\boldsymbol{\theta}$, given observed data $\mathbf{y}$. The data we consider are the monthly returns from $N = 22$ currencies over the period from 9/2007 to 8/2017, transformed into the log domain. As a result, $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{119 \times 22}$.

D.2.2 Results

We use a fixed proposal per timestep parameterized as a Gaussian perturbation to $p_{\boldsymbol{\theta}}$ as in [9],

$$q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{51}$$

Thus the parameters specific to q are the means $\boldsymbol{\mu}_t \in \mathbb{R}^N$ and covariance matrices $\boldsymbol{\Sigma}_t \in \operatorname{diag}(\mathbb{R}_+^N)$ for $t = 1, \ldots, T$.

SIXO-q uses a parameterless quadrature twist, so we skip twist optimization and use biased reparameterization gradients of the SIXO bound to fit the model and proposal. For the quadrature twist we use Gauss-Hermite quadrature with degree five.

For SIXO-DRE, we model the twist using the backwards RNN method introduced in Section 3.2. The twist is parameterized with a one-layer LSTM with 128 hidden units and a one-layer MLP with 128 hidden units. The twist is learned using the alternating DRE method described in Section 3.2. We generate a batch of 32,000 synthetic trajectories from the current model, and perform minibatch stochastic gradient descent using the ADAM optimizer [55] with a learning rate of 3×10^{-3} and a minibatch size of 64. We apply 1,000 twist updates (corresponding to two epochs) and then apply 1,000 updates to the model and proposal. The model and proposal parameters are updated using the ADAM optimizer [55] with a learning rate of 1×10^{-4}. We use four particles per SMC sweep, and average across four datasets per model and proposal update.

Individual FIVO and SIXO-q seeds for each experiment were run using four CPU cores, 24Gb of memory, and had a wallclock time of no longer than twenty four hours. SIXO-DRE had a longer runtime of five days, however, competitive results were achieved within two days.

Train Bound Performance We compare four methods: IWAE (learning $\boldsymbol{\theta}$), FIVO (learning $\boldsymbol{\theta}$), SIXO-q (learning $\boldsymbol{\theta}$ with a quadrature twist) and SIXO-DRE (learning $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ using a DRE twist). All methods (other than IWAE) use the biased resampling gradients of (18). We show the median and

quartiles across five random seeds. Each μ_n is initialized from $\mathcal{N}(0, 0.3)$ (with $n \in 1, \ldots, 22$). ϕ_n is learned in the unconstrained space $\mathbb{R}$, and is transformed to $[0, 1]$ by passing the raw ϕ through a hyperbolic tangent function. The unconstrained ϕ_n's are initialized from $\mathcal{N}(\operatorname{arctanh}(0.1), 0.3)$, and $\log \beta_n$ is initialized from $\mathcal{N}(\log 1.0, 0.3)$. Finally, $\log Q_n$ is initialized from $\mathcal{N}(\log 1.0, 0.3)$. The train bound value we report is taken as the average train bound after 75% of training. We also report $\mathcal{L}_{\text{BPF}}^{2048}$ as the bound evaluated using a BPF with 2,048 particles. This tests the performance of the learned model opposed to inference performance.

A one-way ANOVA [42] rejected the null hypothesis that the mean train bounds are equal ($p < 0.0001$), and a post-hoc Dunnett T3 test [56] found that all methods are statistically significantly different ($p < 0.01$ for all pairs). For the $\mathcal{L}_{\text{BPF}}^{2048}$ values, a one-way ANOVA failed to reject the null hypothesis that the train bounds are equal ($p = 0.26$), so all entries are bolded. This methodology was recommended in Sauder and DeMars [57].

Test Set Performance We also report the performance on a held-out dataset constructed from the new data since Naesseth et al. [9] was published. The test $\mathcal{L}_{\text{BPF}}^{2048}$ we report is the bound evaluated using a BPF with 2,048 particles, averaged across all checkpoints after 75% of training. A one-way ANOVA rejected the null hypothesis that the mean test bounds are equal ($p < 0.0001$), and a post-hoc Dunnett T3 test found the mean SIXO-DRE bound to be the highest ($p < 0.01$ for all pairs).

D.3 Hodgkin-Huxley Model

We provide a brief overview of the model here for completeness, but refer the reader to Chapter 5.6 of Dayan and Abbott [44] for more detailed information.

D.3.1 Model Details

The HH model is a physiologically grounded model of neural action potentials [43] defined through a set of four nonlinear differential equations. Each neuron is defined by four state variables: the instantaneous membrane potential $v(t) \in \mathbb{R}$, the potassium channel activation $n(t) \in [0, 1]$, sodium channel activation $m(t) \in [0, 1]$, and sodium channel inactivation $h(t) \in [0, 1]$. The channel states represent the aggregated probability that the given channel is active. The state evolves according to:

$$C_m \frac{\mathrm{d}v(t)}{\mathrm{d}t} = i_{\text{ext}}(t) - g_L(v(t) - E_L) - g_K n^4(v(t) - E_K) - g_{Na} m^3 h(v(t) - E_{Na}). \tag{52}$$

The membrane capacitance C_m is often defined to be 1.0. The first term, $i_{\text{ext}}(t)$ is the externally injected current. The second term represents the net current through cell membrane due to the potential difference between the intracellular and extracellular mediums, often referred to as the *leakage current*. This current is a function of the membrane capacitance, g_L, and the potential of the extracellular medium, E_L, where the potential difference across the membrane is then $v(t) - E_L$. The third term represents the net current through the membrane as a result of the potassium channels as a function of the channel state $n(t)$, the channel capacitance g_K, and the potassium reversal potential E_K. The final term represents the current through the membrane as a result of the sodium channel states, both $m(t)$ and $h(t)$, the sodium channel conductance g_{Na}, and the sodium reversal potential E_{Na}. The channel states evolve according to:

$$\frac{\mathrm{d}n(t)}{\mathrm{d}t} = \alpha_n(v(t))(1 - n) - \beta_n(v(t))n, \tag{53}$$

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \alpha_m(v(t))(1 - n) - \beta_m(v(t))m, \tag{54}$$

$$\frac{\mathrm{d}h(t)}{\mathrm{d}t} = \alpha_h(v(t))(1 - n) - \beta_h(v(t))h, \tag{55}$$

where $\alpha_n, \alpha_m, \alpha_h, \beta_n, \beta_m, \beta_h$ are all fixed scalar functions of the membrane potential. We discretize this continuous-time differential equation into a discrete-time latent variable model by integrating using Euler integration with an integration timestep of 0.02ms (similarly to Huys and Paninski [47]). This defines the deterministic time-evolution of the neural state.

We follow a similar approach as Huys and Paninski [47] and add Gaussian random noise to each of the four states at each timestep. The membrane potential is unconstrained, and so we can add

noise directly. The gate states, however, are constrained to the range $[0, 1]$. Huys and Paninski [47] use truncated Gaussian noise to avoid pushing the state outside the constrained range. We use a different approach and transform the constrained states into an unconstrained state by applying the inverse sigmoid function to the raw gate value. This has the effect of modifying the variance of the perturbation in constrained space as a function of the state (heteroscedastic noise in constrained space). However, this hetereoscedasticity carries with it a favorable intuition. The magnitude of the noise term is reduced (after being pushed through a sigmoid) close the limits. This means that the same noise kernel provides smaller perturbations close to the extremes, while still retaining a larger permissible perturbations in the mid-range. The model is integrated with a timestep of 0.02ms. We add zero-mean Gaussian noise to the potential with variance scaled by the integration timestep, $\sigma_v^2 = 9\mathrm{mVs}^{-1} \times 0.02\mathrm{ms} = 0.18\mu\mathrm{V}$. The unconstrained gate variables are perturbed by zero-mean Gaussian noise with variance also scaled by the integration timestep, $\sigma^2_{\{n,m,h\}} = 0.1\mathrm{s}^{-1} \times 0.02\mathrm{ms} = 0.000002$. Observations are sampled from a Gaussian emission distribution centered on the current membrane potential with variance 25mV. Observations are generated every 50 timesteps, corresponding to an acquisition rate of 1kHz.

We initialize the potential according to a Gaussian distribution with mean equal to -65mV and a standard deviation of 25mV. The unconstrained gate variables are initialized from a Gaussian distribution with mean defined by an estimate of the steady-state value $x = \mathrm{sigmoid}\left({}^{\alpha_x(-65)}/_{\alpha_x(-65)-\beta_x(-65)}\right)$, where x represents the n, m or h states.

To iterate the model, we first constrain the state by passing the gate variables through a sigmoid. The potential is already unconstrained and so requires no transform. We then iterate the model given the constrained state. The iterated state is then unconstrained by passing the gate states through the logit function (inverse of the sigmoid function). The noise term is then added to the iterated, unconstrained state. Observations are then generated by sampling from the emission distribution every 50 steps. We generate traces with 2,048 timesteps, corresponding to approximately 40ms.

D.3.2 Results

For the experiments presented in Section 5.3 we use either a bootstrap, filtering or smoothing proposal. The bootstrap proposal is defined as $q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. The filtering proposal runs an RNN forward over the observed data to create an encoding of all previous observations. This encoding is then concatenated with a Transformer-style embedding of the elapsed time since the last observation [58], and is input into an MLP to produce the residual density with is multiplied with the prior density from which particles are then proposed (a type of *resq* proposal [59]). The smoothing proposal runs two separate RNNs forwards and backwards over the data to encode all previous and future observations. Both RNN encodings and the time encoding are concatenated and used as input to an MLP. For SIXO, the twists are parameterized by an RNN running backwards over the observations. The hidden state of the RNN is concatenated with a time encoding and fed into an MLP to produce the twist value. All RNNs in these experiments have a hidden state of size 32, and all MLPs have two hidden layers of size 32. Individual seeds for each experiment were run using three CPU cores, 9Gb of memory, and had a runtime time of no longer than five days.

Inference In Figure 3 we compare the inference performance of a BPF (equivalent to FIVO-bs), to SIXO-bs. The true generative model was used in both cases. This highlights the advantage of using a learned twist under a fixed proposal.

In Table 4 we compare different inference methodologies in the true model across different observation intervals and number of particles. FIVO-fi performs poorly, and drops off dramatically with the number of particles and sampling interval. The performance of SIXO (both bootstrap and smoothing) outperforms FIVO-bs, and is more consistent across the number of particles.

Model Learning For the experiments presented in Figure 4 and Table 3 in Section 5.3 we learn the constant external current input, i_{ext} in (52). We generate 10,000 training sequences, and evaluate on 64 validation and test sequences, all generated with $\theta^{\mathrm{True}} = i_{\mathrm{ext}} = 13\mathrm{mV}$. We use five seeds, deterministically initialized i_{ext} to a value uniformly spaced between $1.3\mu\mathrm{A}$ and $37.7\mu\mathrm{A}$, corresponding to a relative error of between -0.9 and 1.9. We use $K = 4$ particles per SMC sweep during learning, when evaluating the model and proposal gradients (cf. Line 11 of Algorithm 1), and average across four sequences per parameter update. We use the same proposal and twist

Table 4: Comparison of inference performance for different observation intervals and numbers of particles. We normalize $\mathcal{L}^K_{\text{Method}}$ by the number of observations in the sequence for comparison. We compare SIXO-DRE with a smoothing proposal (SIXO-sm), SIXO-DRE with a bootstrap proposal (SIXO-bs), FIVO with a filtering proposal (FIVO-fi), and filtering with a bootstrap proposal (FIVO-bs, equivalent here to a BPF).

Number of Particles (K)	$\mathcal{L}^K_{\text{Method}}$ / number of observations			
	SIXO-sm	SIXO-bs	FIVO-fi	FIVO-bs
	Observation interval: 2ms			
4	-5.25 ± 1.706	-6.77 ± 1.678	-19.98 ± 0.206	-15.2 ± 2.446
8	-2.90 ± 0.118	-3.56 ± 0.539	-20.92 ± 0.449	-7.91 ± 0.528
16	-2.07 ± 0.097	-2.21 ± 0.238	-21.57 ± 0.558	-3.95 ± 0.520
32	-1.66 ± 0.054	-1.75 ± 0.100	-22.80 ± 0.457	-2.57 ± 0.297
64	-1.50 ± 0.028	-1.54 ± 0.050	-22.02 ± 0.248	-1.69 ± 0.072
128	-1.43 ± 0.013	-1.45 ± 0.010	-21.31 ± 0.364	-1.51 ± 0.049
256	-1.41 ± 0.008	-1.41 ± 0.005	-20.58 ± 0.341	-1.44 ± 0.014
	Observation interval: 1ms			
4	-2.50 ± 1.948	-4.56 ± 1.941	-16.14 ± 0.843	-16.1 ± 1.688
8	-1.30 ± 0.011	-2.24 ± 0.561	-14.93 ± 0.498	-8.28 ± 0.618
16	-1.21 ± 0.017	-1.41 ± 0.176	-13.29 ± 0.301	-4.00 ± 0.612
32	-1.18 ± 0.004	-1.36 ± 0.206	-11.86 ± 0.304	-2.36 ± 0.356
64	-1.17 ± 0.004	-1.21 ± 0.045	-10.11 ± 0.219	-1.43 ± 0.081
128	-1.17 ± 0.002	-1.18 ± 0.008	-8.58 ± 0.129	-1.27 ± 0.048
256	-1.16 ± 0.002	-1.17 ± 0.001	-7.00 ± 0.192	-1.20 ± 0.015
	Observation interval: 0.5ms			
4	-2.12 ± 1.182	-4.05 ± 1.522	-5.73 ± 0.149	-15.4 ± 1.997
8	-1.20 ± 0.024	-1.83 ± 0.393	-5.48 ± 0.107	-7.76 ± 0.837
16	-1.11 ± 0.035	-1.18 ± 0.096	-5.62 ± 0.168	-3.88 ± 0.620
32	-1.06 ± 0.013	-1.17 ± 0.254	-6.09 ± 0.302	-2.17 ± 0.305
64	-1.04 ± 0.014	-1.05 ± 0.017	-5.93 ± 0.236	-1.31 ± 0.066
128	-1.03 ± 0.006	-1.03 ± 0.005	-5.72 ± 0.298	-1.10 ± 0.043
256	-1.02 ± 0.002	-1.02 ± 0.001	-5.19 ± 0.233	-1.04 ± 0.015
	Observation interval: 0.2ms			
4	-1.28 ± 0.185	-3.57 ± 1.369	-2.36 ± 0.081	-13.71 ± 1.753
8	-1.00 ± 0.009	-1.57 ± 0.362	-2.43 ± 0.205	-6.69 ± 0.800
16	-0.95 ± 0.018	-1.02 ± 0.081	-2.43 ± 0.075	-3.37 ± 0.416
32	-0.92 ± 0.003	-1.00 ± 0.209	-2.49 ± 0.106	-1.93 ± 0.243
64	-0.91 ± 0.010	-0.91 ± 0.009	-2.31 ± 0.097	-1.17 ± 0.082
128	-0.90 ± 0.002	-0.90 ± 0.007	-2.02 ± 0.059	-0.97 ± 0.030
256	-0.90 ± 0.001	-0.89 ± 0.001	-1.79 ± 0.089	-0.92 ± 0.011

architectures described above. We take 400 steps in the model per 100 steps in the twist. We perform a hyperparameter search over the learning rate for the model, proposal, and twist, and a learning rate decay schedule that halves the learning rate at specific steps. The optimal parameters are shown in Table 5. We selected the best hyperparamters based on validation-set accuracy, and then evaluated the

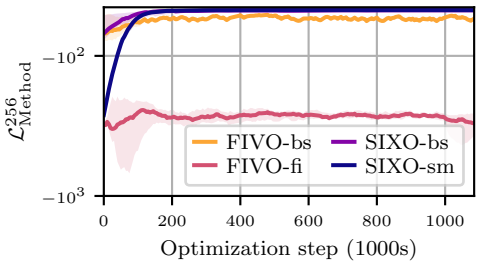


Figure 8: Full version of Figure 4a showing the poor performance of FIVO-fi.

final three checkpoints of each selected run on the test set 10 times to produce $\mathcal{L}^{256}_{\text{BPF}}$ and $\mathcal{L}^{256}_{\text{Method}}$ values shown.

Table 5: Learning rate hyperparameters for the model learning experiment presented in Section 5.3.

Method	Model LR	Proposal LR	Twist LR	Learning rate decay schedule (1000s)
FIVO-bs	0.01	N/A	N/A	[150, 300, 600]
FIVO-fi	0.0003	0.0002	N/A	[100, 200, 400]
SIXO-bs	0.01	N/A	0.0005	[800, 1600, 3200]
SIXO-sm	0.0003	0.0002	0.003	[80, 160, 320]