Sample-efficient neural likelihood-free Bayesian inference of implicit HMMs

Sanmitra Ghosh[1,2] **Paul J. Birrell**[3,2] **Daniela De Angelis**[2,3]
{sanmitra.ghosh,paul.birrell,daniela.deangelis}@mrc-bsu.cam.ac.uk
[1]PhysicsX, London, [2]MRC Biostatistics Unit, University of Cambridge, [3]UK Health Security Agency

Abstract

Likelihood-free inference methods based on neural conditional density estimation were shown to drastically reduce the simulation burden in comparison to classical methods such as ABC. When applied in the context of any latent variable model, such as a Hidden Markov model (HMM), these methods are designed to only estimate the parameters, rather than the joint distribution of the parameters and the hidden states. Naive application of these methods to a HMM, ignoring the inference of this joint posterior distribution, will thus produce an inaccurate estimate of the posterior predictive distribution, in turn hampering the assessment of goodness-of-fit. To rectify this problem, we propose a novel, sample-efficient likelihood-free method for estimating the high-dimensional hidden states of an implicit HMM. Our approach relies on learning directly the intractable posterior distribution of the hidden states, using an autoregressive-flow, by exploiting the Markov property. Upon evaluating our approach on some implicit HMMs, we found that the quality of the estimates retrieved using our method is comparable to what can be achieved using a much more computationally expensive SMC algorithm.

1 INTRODUCTION

We consider the task of carrying out Bayesian inference of an *implicit* HMM, i.e. a HMM whose likelihood density function is analytically intractable. Such a model is only available as a simulator from which one can generate data that realistically, faithfully mimics the observed time course of some complex bio-physical system.

Due to the analytical intractability of the likelihood density function standard Bayesian inference methods cannot be applied to such an implicit HMM. Inference of such a model is typically carried out using approximate Bayesian computation (ABC) (Sisson et al., 2018), which only requires forward simulations from the model, see for example Martin et al. (2019); Picchini (2014); Toni et al. (2009).

Recently, a new class of likelihood-free inference methods, see Cranmer et al. (2020) for a review, were developed that use a neural network based emulator of the posterior density, the likelihood density and the likelihood ratio. Such methods were empirically shown to be much more sample-efficient (require fewer model simulations) (Lueckmann et al., 2021) than ABC. These methods were found to perform equally well across different models without problem specific tailoring of the neural network's architecture. Naturally, these methods appear as more preferable algorithmic choices for carrying out inference of an implicit HMM, in comparison to ABC.

These *neural likelihood-free* inference (NLFI) approaches, in the specific context of a latent variable problem such as a HMM, have so far been applied to carry out Bayesian inference partially by estimating only the marginal posterior of the parameters rather than the joint posterior of the hidden states and parameters. This is since a naive implementation of a neural network based density (or density-ratio) estimator may perform unreliably (with drastically reduced sample-efficiency) in estimating the joint posterior of the parameters and the high-dimensional hidden states, potentially for a lack of inductive biases. Estimation of the hidden states may or may not be of interest within a particular application domain. However, without estimating the joint posterior of the parameters and the hidden states the goodness-of-fit cannot be correctly as-

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

sessed. This is a severe limitation. Note that although ABC theoretically targets the joint distribution (see Appendix A.1 for details) it fails to estimate the hidden states adequately within a reasonable simulation budget.

Note that the task of inferring the posterior of the hidden states of an implicit HMM is in itself extremely challenging. This problem can only be solved using the Bootstrap sequential Monte Carlo (SMC) algorithm (Gordon et al., 1995). However, reliable performance of the Bootstrap SMC algorithm often requires a large number of model simulations, thus defeating the purpose of sample-efficient inference which motivates the use of NLFI. Thus, extending NLFI approaches to solve the joint estimation problem is highly non-trivial as this requires the development of a fundamentally new approach of estimating the hidden-states, that is much more sample-efficient in comparison to SMC.

In this paper we present such a novel technique which is based on learning an approximation of the posterior distribution of the hidden states using neural density estimation. After learning this posterior approximation, neural density estimators can be used to draw the full path of the hidden states recursively. Following are our salient contributions:

- We develop a sample-efficient method to obtain an approximation of the posterior distribution of the sample path of a HMM, without accessing the transition or the observation densities.
- Our approach, when combined with any off-the-shelf NLFI method, can be used, as a sample-efficient alternative to ABC, for carrying out full Bayesian inference of an implicit HMM.

2 BACKGROUND

We begin by first introducing the implicit HMM and then we will discuss the challenges of carrying out Bayesian inference. We can describe a HMM, for a latent Markov process $\boldsymbol{X}_t \in \mathbb{R}^K$ with a K-dimensional continuous state-space, as follows:

$$\boldsymbol{X}_t \sim f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{\theta}_f), \quad \boldsymbol{y}_t \sim g(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{\theta}_g) \tag{1}$$

where $\boldsymbol{\theta}_f, \boldsymbol{\theta}_g$ parameterise the transition $f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{\theta}_f)$ and the observation $g(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{\theta}_g)$ densities respectively. We consider the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_f, \boldsymbol{\theta}_g, \boldsymbol{X}_0)$ to include the initial state $\boldsymbol{X}_0$. Given a set of noisy observations of L out of the K states $\boldsymbol{y} \in \mathbb{R}^{M \times L}$ at M experimental time points, of the latent process, our goal is to infer the joint posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$, where $\boldsymbol{x} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{M-1})$ is the unobserved sample path of the process – the hidden states. The expression for the unnormalised posterior is given by

$$p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y}) \propto \left(\prod_{t=0}^{M-1} g(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{\theta}_g)\right)\left(\prod_{t=1}^{M-1} f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{\theta}_f)\right) \times p(\boldsymbol{\theta}), \tag{2}$$

where $p(\boldsymbol{\theta})$ is the prior distribution over the parameters and the initial values. Additionally, we are also interested in checking the goodness-of-fit, which, within the Bayesian context, is carried out by inspection of the posterior predictive distribution $p(\boldsymbol{y}^r|\boldsymbol{y})$ of generating *replicated data* (Gelman et al., 1996) $\boldsymbol{y}^r$. This distribution is given by

$$p(\boldsymbol{y}^r|\boldsymbol{y}) = \int p(\boldsymbol{y}^r|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{\theta}. \tag{3}$$

We assume that one can draw samples from $f(\cdot)$ and $g(\cdot)$, but cannot evaluate either or both of these densities. This assumption leads to an intractable likelihood density rendering the model implicit. This is the constrained setting for our work. Inference of $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$, in our implicit modelling context, can be carried out using the ABC algorithm which replaces the evaluation of the right hand side of Eq (2), upto a normalising constant, by using a distance function between simulated and real data. Although ABC jointly samples (Appendix A.1) the hidden states and the parameters, drawing the high-dimensional hidden states just using *rejection sampling* is highly inefficient. Although a more efficient variant of the basic ABC algorithm may employ a sophisticated technique to propose values of $\boldsymbol{\theta}$, the states are still updated using the prior of the Markov process as the proposal, thus falling back to rejection sampling, resulting in an exorbitant computational expense. Due to this computational burden ABC algorithms are rarely practically useful for inference of an implicit HMM where $f(\cdot)$ and $g(\cdot)$ are computationally expensive simulators.

Note that we can decompose the joint density using the product rule as follows:

$$p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y}) p(\boldsymbol{\theta}|\boldsymbol{y}). \tag{4}$$

With the above decomposition we can break down the task of inferring the joint distribution of $\boldsymbol{x}, \boldsymbol{\theta}$ into two sub-tasks of inferring separately the distributions $p(\boldsymbol{\theta}|\boldsymbol{y})$ and $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})$. Samples of $\boldsymbol{x}$ can then be drawn given samples of $\boldsymbol{\theta}$. Note that the task of inferring $p(\boldsymbol{\theta}|\boldsymbol{y})$ can be carried out, sample-efficiently, using any NLFI method. Inference of $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})$ can then be carried out using a Bootstrap SMC, or its ABC (and more inefficient) variant (Drovandi et al., 2016) when $g(\cdot)$ is unavailable. The posterior predictive distribution can

then be evaluated as follows:

$$p(\boldsymbol{y}^r|\boldsymbol{y}) \approx \int p(\boldsymbol{y}^r|\boldsymbol{x},\boldsymbol{\theta})p_{smc}(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{\theta}, \quad (5)$$

where $p_{smc}(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta})$ is a Bootstrap SMC estimate of the hidden states. To evaluate the above integral numerically, one has to run a particle filter for each $\boldsymbol{\theta}$ sample, which in turn will require as many simulations as the number of particles, resulting in a computation cost that in most cases will be higher than that of running NLFI for inferring $\boldsymbol{\theta}$ alone. Clearly, this makes SMC unusable as long as we need to evaluate the posterior predictive distribution. The particle Markov chain Monte Carlo (MCMC) algorithm (Andrieu et al., 2010), when $g(\cdot)$ is known, can produce samples from the true joint posterior distribution in Eq. (2). However, this algorithm also requires running SMC for each iteration of MCMC, making it computationally expensive to apply to complex models. Recently, neural network based methods have been proposed to infer a high-dimensional hidden states of a HMM (Schumacher et al., 2023; Ryder et al., 2021). However, it is unclear, given lack of comparison with exact algorithms, whether such methods can indeed recover the true posterior hidden states.

Note that standard sample-efficient alternatives to SMC such as EM family algorithms and more generally variational inference algorithms for state-space models are non-applicable due the implicitness of our model. Next, we briefly describe existing NLFI methods and highlight their limitations in estimating the joint posterior of $\boldsymbol{x},\boldsymbol{\theta}$, before explaining the proposed method.

2.1 Related work: Neural likelihood-free inference (NLFI)

If instead of the joint $p(\boldsymbol{\theta},\boldsymbol{x}|\boldsymbol{y})$ we only wish to estimate the marginal $p(\boldsymbol{\theta}|\boldsymbol{y})$, then a number of strategies based on conditional density estimation can be employed. For example, we can simulate pairs of $\boldsymbol{\theta},\boldsymbol{y}$ from their joint distribution and then subsequently create a training dataset, of N samples $\{\boldsymbol{\theta}^n,\boldsymbol{y}^n\}_{n=1}^N$, which can be utilised to train a conditional density estimator, constructed using a flexible function approximator such as a neural network, that can approximate the marginal posterior (Papamakarios and Murray, 2016) $p(\boldsymbol{\theta}|\boldsymbol{y}) \approx q_{\boldsymbol{\psi}}(\boldsymbol{\theta}|\boldsymbol{y})$ or the likelihood Papamakarios et al. (2019) $p(\boldsymbol{y}|\boldsymbol{\theta}) \approx q_{\boldsymbol{\psi}}(\boldsymbol{y}|\boldsymbol{\theta})$. Here $\boldsymbol{\psi}$ denotes the parameters of the function approximator used to build the density estimator. In the former case once we have trained an approximation to the posterior, using a density estimator, we can directly draw samples $\boldsymbol{\theta} \sim q_{\boldsymbol{\psi}}(\boldsymbol{\theta}|\boldsymbol{y}_o)$ by conditioning on a particular dataset $\boldsymbol{y}_o$. In the latter case we can use the trained density estimator of the likelihood to approximate the posterior $p(\boldsymbol{\theta}|\boldsymbol{y}) \propto q_{\boldsymbol{\phi}}(\boldsymbol{y}_o|\boldsymbol{\theta})p(\boldsymbol{\theta})$ and then draw samples from it using MCMC.

A neural network is used in this context either as a nonlinear transformation of the conditioning variables, within a mixture-of-Gaussian density as was proposed in Bishop (1994), or as a *normalizing-flow* (Rezende and Mohamed, 2015; Papamakarios et al., 2021) that builds a transport map (Parno, 2015) between a simple distribution (such as a standard Gaussian) and a complex one such as the likelihood/posterior density. Following the seminal work of Tabak and Turner (2013) a large amount of research is undertaken to build such transport maps using samples from the respective measures.

An alternative formulation of NLFI utilises the duality (Cranmer et al., 2015) between the optimal decision function of a probabilistic classifier and the likelihood ratio, $r(\boldsymbol{\theta}^a,\boldsymbol{\theta}^b) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}^a)}{p(\boldsymbol{y}|\boldsymbol{\theta}^b)}$ evaluated using two samples $\boldsymbol{\theta}^a$ and $\boldsymbol{\theta}^b$, to approximate the likelihood-ratio through training a binary classifier using samples from $p(\boldsymbol{y},\boldsymbol{\theta})$. This likelihood ratio can then be used as proxy within a MCMC accept/reject step as follows:

$$\min\left\{1, r(\boldsymbol{\theta}^*,\boldsymbol{\theta})\frac{k_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)}{k_{\boldsymbol{\theta}}(\boldsymbol{\theta}^*|\boldsymbol{\theta})p(\boldsymbol{\theta})}\right\}, \quad (6)$$

where $k_{\boldsymbol{\theta}}(\cdot)$ is a proposal density. Note that these NLFI methods carry out *amortised inference* that is there is no need to re-learn the density/density-ratio estimator for every new instances of the observations. However, we like to point out that the MCMC algorithms, associated with likelihood or likelihood ratio estimation based approaches, has to be re-run for each new dataset, which can be more time consuming than training the associated neural networks.

To increase sample-efficiency of these methods one can use them in a sequential manner (Durkan et al., 2018). After an initial round of NLFI, we are left with samples of $\boldsymbol{\theta}$ from its posterior distribution. We can subsequently use these samples to generate further simulated data concentrated around the given observations $\boldsymbol{y}_o$. This constitute a new training set on which a second round of NLFI can be applied to further refine the approximations. This process can be repeated for a number of rounds. Note that when a sequential process is used in conjunction with a density estimator for the posterior then the parameter samples from the second round are no longer drawn from the prior. Thus, different adjustments had been proposed, leading to different algorithms (Greenberg et al., 2019; Lueckmann et al., 2017), to overcome this issue.

We like to further point out that NLFI is applied using some summary statistic $s(\boldsymbol{y})$ of the data $\boldsymbol{y}$, a practise

carried over from the usage of ABC methods. There have been recent work (Chen et al., 2021) on using a neural network to generate the summary statistics.

Also see Appendix H for a brief review of other classical approaches for inference of implicit HMMs.

3 LIMITATIONS OF NLFI METHODS

Can we ignore the joint distribution? The various NLFI techniques discussed so far are designed to solve the marginal problem of estimating $p(\boldsymbol{\theta}|\boldsymbol{y})$. The necessity of the estimation of the hidden states $\boldsymbol{x}$ is problem specific. In some applications estimation of the hidden states is of paramount importance whereas in others one may wish to ignore the hidden states. Irrespective of whether the interest is in estimating $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$ or $p(\boldsymbol{\theta}|\boldsymbol{y})$, in the process of modelling a physical phenomenon it is necessary to assess the goodness-of-fit. But when instead of the joint distribution we only have access to the marginal distribution (that is access to only samples of $\boldsymbol{\theta}$, the output of any NLFI method) then the posterior predictive distribution, instead of Eq. (3), can only be obtained as follows

$$p(\boldsymbol{y}^r|\boldsymbol{y}) \approx \hat{p}(\boldsymbol{y}^r|\boldsymbol{y}) = \int p(\boldsymbol{y}^r|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{\theta}, \tag{7}$$

where the joint posterior of $\boldsymbol{x}, \boldsymbol{\theta}$ is approximated as $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y}) \approx p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y})$, which is akin to drawing $\boldsymbol{x}$ from the prior $p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{t=1}^{M-1} f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{\theta})$, of the latent Markov process. As a result the credible intervals of $\hat{p}(\boldsymbol{y}^r|\boldsymbol{y})$ would be erroneously inflated, since in this case the latent sample path $\boldsymbol{x}$ is not correctly constrained by the data, leading to an incorrect assessment of the goodness-of-fit. This is a severe problem that needs to be addressed even in the case where we wish to estimate just the marginal $p(\boldsymbol{\theta}|\boldsymbol{y})$.

limitations of NFLI for inferring the joint: let us now consider the task of estimating the joint posterior $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$ using a NLFI method. If we want to approximate the posterior then we have to extend any chosen density estimator to target a high-dimensional vector $(\boldsymbol{\theta}, \mathrm{vec}(\boldsymbol{x}))$, where $\mathrm{vec} : \mathbb{R}^{K \times M} \to \mathbb{R}^{KM}$, which would invariably require a larger training set, and thus more simulations, in comparison to estimating only $\boldsymbol{\theta}$ (see Appendix F for an example). Alternatively, if we choose to approximate the likelihood density, then note that the accept/reject step of a MCMC scheme, targeting $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$, will be of the following form:

$$\min\left\{1, \frac{q_{\boldsymbol{\psi}}(\boldsymbol{y}_o|\boldsymbol{x}^*, \boldsymbol{\theta}^*) p(\boldsymbol{x}^*|\boldsymbol{\theta}^*) p(\boldsymbol{\theta}^*) k_{\boldsymbol{x}}(\boldsymbol{x}|\boldsymbol{x}^*) k_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q_{\boldsymbol{\psi}}(\boldsymbol{y}_o|\boldsymbol{x}_\cdot \boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) k_{\boldsymbol{x}}(\boldsymbol{x}^*|\boldsymbol{x}) k_{\boldsymbol{\theta}}(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}, \tag{8}$$

where $k_{\boldsymbol{x}}(\cdot)$, $k_{\boldsymbol{\theta}}(\cdot)$ are the proposal densities. Due to the intractability of $p(\boldsymbol{x}, \boldsymbol{\theta}) = p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta})$ our only option as a proposal, $k_{\boldsymbol{x}}(\cdot)$, is the prior (so that the proposal and prior of $\boldsymbol{x}$ cancel out in the above ratio) that is the transition density in equation 1. This will jeopardise the mixing of the MCMC sampler which in turn would require excessive simulation from the model. We would face the same limitation if we had chosen to emulate the likelihood ratio.

4 METHODS

4.1 Inferring hidden states

We can decompose the posterior of $\boldsymbol{x}$, by applying the product rule and then utilising the Markov property (see proof in Appendix A.2), as follows:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y}) = {} & p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2:1}, \boldsymbol{\theta}, \boldsymbol{y}) \\ & \times \prod_{t=1}^{M-2} p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}). \end{aligned} \tag{9}$$

Note that the above decomposition produces homogeneous factors $p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$. By dropping the dependency of the future sample point $\boldsymbol{X}_{t+1}$ and thus approximating each factor,

$$p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}) \approx p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}), \tag{10}$$

we can approximately decompose the posterior as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y}) \approx \prod_{t=1}^{M-1} p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}). \tag{11}$$

Although we are losing information, this approximation will still be a reasonable one as long as the information in $\boldsymbol{X}_{t+1}$ is largely contained in the pair $(\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$. Importantly, this approximation lets us easily draw the hidden states using ancestral sampling from the approximate factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$. Additionally, we can improve this approximation by employing importance sampling. That is we can obtain a sample from the correct factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ by drawing a weighted sample from the approximate one $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$, with weights given by

$$w_t(\boldsymbol{X}_t) = \frac{p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})}{p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})}. \tag{12}$$

Having introduced a technique for drawing the hidden states we must point out that except for linear-Gaussian models, these factors are never available in closed form. Thus, the decomposition in Eq. (9), in our knowledge, has never been explored in the context of classical filtering/smoothing methodologies.

Since these factors are homogeneous thus we can now emulate the approximate and true:

$$p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta}) \approx q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})$$
$$p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta}) \approx q_{\boldsymbol{\phi}_2}(\boldsymbol{X}_t|\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta}), \tag{13}$$

factors between any two consecutive time points $t, t-1$, using neural density estimators $q_{\boldsymbol{\phi}_1}(\cdot), q_{\boldsymbol{\phi}_2}(\cdot)$, where $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$ are the parameters of the respective neural networks, parameterising the density estimators in turn. With access to these neural density estimates of the correct and approximate factors, an approximation to the entire sample path can be generated again using importance sampling where weighted samples $\boldsymbol{X}_t$ can be recursively drawn at every time point, with weights given by

$$w_t(\boldsymbol{X}_t) = \frac{q_{\boldsymbol{\phi}_2}(\boldsymbol{X}_t|\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})}{q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})}. \tag{14}$$

In practise the above importance sampling can be carried out in two steps. First, we draw a cloud of P particles, at each time point, from the importance factor: $\hat{\boldsymbol{X}}_t^p \sim q_{\boldsymbol{\phi}_1}(\cdot|\hat{\boldsymbol{X}}_{t-1}^p, \boldsymbol{y}_t, \boldsymbol{\theta})$, and assign weight $w_t^p := w(\hat{\boldsymbol{X}}_t^p)$ to each particle $p = 1, \ldots, P$. We then resample a single $\boldsymbol{X}_t$, at each t from the particle cloud to construct the desired sample path $\boldsymbol{x}$.

Since we are using a density estimator, rather than the HMM itself, as above, we can use a large number of importance samples (we chose $P = 10^4$, throughout), unlike traditional SMC algorithms, without bothering about computational cost.

We denote the above strategy (see Algorithm 1 for the pseudocode) of drawing the hidden states $\boldsymbol{x}$ using neural density estimates of the true and approximate factors (both being essentially an incremental posterior), collectively as an *incremental density estimator* (IDE). Using the IDE we can approximate the posterior predictive in (3) now as follows:

$$p(\boldsymbol{y}^r|\boldsymbol{y}) \approx \int p(\boldsymbol{y}^r|\boldsymbol{x},\boldsymbol{\theta}) q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2},\boldsymbol{\theta},\boldsymbol{y}) \prod_{t=1}^{M-2} w_t(\boldsymbol{X}_t) q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{\theta}. \tag{15}$$

In summary our strategy, see Figure 1 for an overview, for drawing samples $\boldsymbol{x}, \boldsymbol{\theta}$ from an approximation of their joint posterior is as follows. We first infer the parameter marginal $p(\boldsymbol{\theta}|\boldsymbol{y})$ using any chosen off-the-shelf NLFI method (see section 2.1) and draw samples of $\boldsymbol{\theta}$. In parallel we train an IDE using a subset of simulations used in inferring $p(\boldsymbol{\theta}|\boldsymbol{y})$. Then for each sample $\boldsymbol{\theta}$, and an observed dataset $\boldsymbol{y}_o$, we can use the trained IDE recursively to obtain the sample path $\boldsymbol{x}$ conditioned on the sample $\boldsymbol{\theta}$ and the dataset $\boldsymbol{y}_o$.

Algorithm 1 Hidden states prediction using IDE

Input: Posterior parameter samples $\{\boldsymbol{\theta}^l\}_{l=1}^L$, observed time series $\boldsymbol{y}_o = (\boldsymbol{y}_{o_1}, \ldots, \boldsymbol{y}_{o_M})$ of length M, number of particles P, neural density estimators $q_{\boldsymbol{\phi}_1}(\cdot), q_{\boldsymbol{\phi}_2}(\cdot)$ introduced in Eq (13).
1. Generate importance samples
for $l = 1$ **to** L **do**
  for $t = 1$ **to** $M-1$ **do**
    for $p = 1$ **to** P **do**
      Draw importance samples of the hidden states $\hat{\boldsymbol{X}}_t^{l,p} \sim q_{\boldsymbol{\phi}_1}(\cdot|\hat{\boldsymbol{X}}_{t-1}^{l,p}, \boldsymbol{y}_{o_t}, \boldsymbol{\theta}^l)$.
      Obtain importance weights $w_t^{l,p}(\hat{\boldsymbol{X}}_t^l) = \frac{q_{\boldsymbol{\phi}_2}(\hat{\boldsymbol{X}}_t^{l,p}|\hat{\boldsymbol{X}}_{t+1}^{l,p},\hat{\boldsymbol{X}}_{t-1}^{l,p},\boldsymbol{y}_{o_t},\boldsymbol{\theta}^l)}{q_{\boldsymbol{\phi}_1}(\hat{\boldsymbol{X}}_t^{l,p}|\hat{\boldsymbol{X}}_{t-1}^{l,p},\boldsymbol{y}_{o_t},\boldsymbol{\theta}^l)}$.
    end for
    Normalise the weights, set $w_t^{l,p} = \frac{w_t^{l,p}}{\sum_{p=1}^P w_t^{l,p}}$
  end for
end for
2. Generate weighted samples
for $l = 1$ **to** L **do**
  for $t = 1$ **to** $M-1$ **do**
    Resample an index r from the set $\{1, \ldots, P\}$, with respective weights $\{w_t^{l,1}, \ldots, w_t^{l,P}\}$.
    Set $\boldsymbol{X}_t^l = \hat{\boldsymbol{X}}_t^{l,r}$.
  end for
end for
Output: Hidden states $\boldsymbol{X} \in \mathbb{R}^{M \times L}$.

Limitations: There are two fundamental assumptions behind our approach. Firstly, we assume that samples of $\boldsymbol{\theta}$ are drawn from a good approximation to the true unknown posterior. However, this may not be true when inference of $\boldsymbol{\theta}$ is done on a very limited simulation budget. Secondly, we assume that there is no model miss-specification and the actual observations come from the joint distribution $p(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})$ used for learning the IDE. This can be a strong assumption when modelling a new phenomenon.

4.2 Training the incremental density estimator

For simplicity we will provide details about the neural density estimation for the approximate factor $q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$, from which we can draw the sample path recursively. The corresponding neural density estimator for the true factor can be constructed and trained analogously.

We chose a *masked autoregressive flow* (MAF) as the incremental neural density estimator. MAF is built upon the idea of chaining together a series of autoregressive functions, and can be interpreted as a normalizing-flow (Papamakarios et al., 2017). That is we can rep-

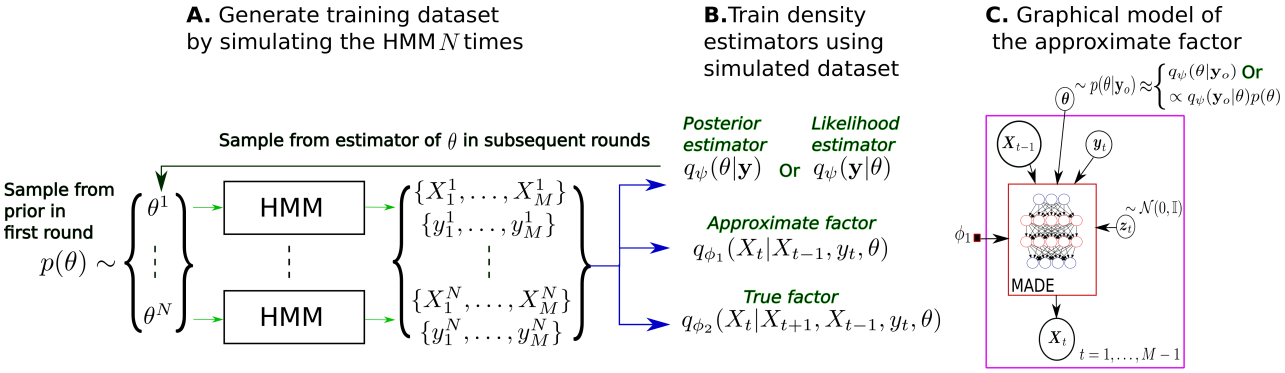


Figure 1: The process of using neural density estimators to approximate the joint posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$ of a HMM. First, the HMM (simulator) is used to generate a training dataset $\{\boldsymbol{\theta}^n, \boldsymbol{x}^n, \boldsymbol{y}^n\}_{n=1}^N$ (**A**), which is then used to train three neural density estimators (**B**). Training of the estimators of $\boldsymbol{\theta}$ happens sequentially through multiple rounds, generating more simulated data in the process. Once trained, given an observed time series $\boldsymbol{y}_o$, for each posterior sample of $\boldsymbol{\theta}$ drawn using its estimator, the approximate factor recursively generates (**C**) *importance samples* of the latent path. The hidden states are resampled from these importance samples using weights that are the ratio of the true and approximate factors (see Algorithm 1 for the pseudocode).

resent $q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ as a transformation of a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbb{I})$ (or another simple distribution) through a series of J autoregressive functions $h^1_{\boldsymbol{\phi}_1}, \ldots, h^J_{\boldsymbol{\phi}_1}$, parameterised by $\boldsymbol{\phi}_1$, each of which is dependent on the triplet $(\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$:

$$\boldsymbol{X}_t = \boldsymbol{z}_J, \quad \text{where} \quad \begin{aligned} \boldsymbol{z}_0 &= \mathcal{N}(\mathbf{0}, \mathbb{I}) \\ \boldsymbol{z}_j &= h^j_{\boldsymbol{\phi}_1}(\boldsymbol{z}_{j-1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}) \end{aligned}. \tag{16}$$

Each h_j is a bijection with a lower-triangular Jacobian matrix, implemented by a *Masked Autoencoder for Distribution Estimation* (MADE) (Germain et al., 2015), and is conditioned on $(\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$. Using the formula for change of variable the density is given by

$$q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbb{I}) \prod_{j=1}^{J} \left| \det \left(\frac{\partial h^j_{\boldsymbol{\phi}_1}}{\partial \boldsymbol{z}_{j-1}} \right) \right|^{-1}. \tag{17}$$

We can learn the parameters $\boldsymbol{\phi}_1$ by maximising the likelihood. To do this we create a training dataset consisting of N examples. We first sample N values of the parameter $\{\boldsymbol{\theta}^n\}_{n=1}^N$ from its prior and for each $\boldsymbol{\theta}^n$ we simulate the sample path of the states and the observations using (1). Each training example is then created by collecting the random variables: $(\boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n)$, and $\boldsymbol{X}_j^n$, as the input-target pair, where $(i, j) = (0, 1), (2, 3), \ldots, (M-2, M-1)$. Clearly, even with a small number of model simulations (a few thousands) we can create a large training dataset to learn an expressive neural density estimator.

In Figure 1 we outline the process of creating this training dataset. Given these training examples $\boldsymbol{\phi}$ can be learnt, using gradient ascent, through maximising the total likelihood:

$$\mathcal{L}(\boldsymbol{\phi}_1) = \sum_{n=1}^{N} \sum_{i=0,j=1}^{M-2,M-1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_j^n|\boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n), \tag{18}$$

which is equivalent to minimising the forward Kullback–Leibler divergence $\mathrm{KL}\left(p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})||q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})\right)$ (Papamakarios et al., 2019) between the approximate factor and its neural density estimate.

Pseudocode of sampling and training of the IDE is provided in Appendix A.3.

5 EVALUATIONS

We evaluated the proposed approach in two stages. First, we used a nonlinear Gaussian state-space model which has a tractable approximate factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$. This tractability lets us compare the IDE with a conditionally optimal SMC algorithm (more accurate than bootstrap SMC). In the next stage, we used two implicit biological HMMs models to evaluate the IDE's usefulness in accurately estimating the hidden states and the posterior predictive distribution.

5.1 State-space model with a tractable approximate factor

We considered a state-space model, that has a tractable approximate factor, to evaluate the quality of approximation of the hidden states $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta})$ in a classical *filtering context* (Särkkä, 2013). Specifically, we used

the following model:

$$\begin{aligned} \boldsymbol{X}_t &\sim \mathcal{N}(\boldsymbol{A}\gamma(\boldsymbol{X}_{t-1}), \sigma_x^2 \mathbb{I}) \quad t \geq 1 \\ \boldsymbol{y}_t &\sim \mathcal{N}(\boldsymbol{B}\boldsymbol{X}_t, \sigma_y^2 \mathbb{I}), \end{aligned} \tag{19}$$

where $\gamma(\boldsymbol{X}) = \sin(\exp(\boldsymbol{X}_{t-1}))$, applied elementwise, $\boldsymbol{A} = \mathbb{I}_{K\times K}$, $B = 2\boldsymbol{A}$ and $\boldsymbol{X}_0 = \boldsymbol{0}$.

We considered a moderately high-dimensional state-space, $K = L = 10$, with a long enough time series, $M = 500$, to challenge SMC algorithms. Naturally, a model setup that is challenging for a SMC algorithm will suffice as a good test-bed for the IDE. We considered the parameters σ_x, σ_y to be known and fixed, thus conditioning the IDE only on $(\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$.To estimate the sample path $\boldsymbol{x}$ we applied the IDE (see Appendix B.1 for details of the neural networks used), the bootstrap SMC, and the conditionally optimal SMC (Doucet et al., 2000), also known as the guided SMC, that uses $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$, a Gaussian distribution, see Appendix B.1, as the importance proposal. Note that the guided SMC is not applicable in case of an implicit model. However, due to its superior performance, when we have a tractable $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$, we have used this algorithm as the *gold standard* for this model.

The goal of this experiment was to find out how the quality of estimation, by methods that require simulation, vary with the number of simulations. Essentially comparing the sample-efficiency. For the IDE this is determined by the training set size, and for SMC the number of particles. We chose the (i) *mean squared error* (MSE) between the true hidden states and it's posterior mean, to quantify the bias, and (ii) the 90% *empirical coverage* (EC), of the ground truth, to quantify the quality of uncertainty estimation.

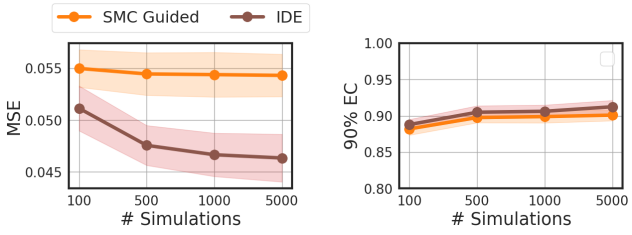


Figure 2: Estimation of the **hidden states** of a **nonlinear state-space** model. The quality of approximations was quantified using the MSE and 90% EC, summarised using the mean (solid line) and 95% confidence intervals (shaded area), across 10 datasets.

In Figure 2 we plot these metrics, summarised across 10 simulated datasets. The Bootstrap SMC performed poorly and its particle system completely degenerated within a few time steps resulting in a comparatively high MSE (≥ 0.4) for all the datasets. Thus, we avoid plotting the Bootstrap SMC results in Figure 2. We found similar degeneration for other experimental settings (see Appendix B.2). The IDE outperformed the guided SMC algorithm, in terms of both the metrics. It produced a noticeably smaller MSE than what was produced by the guided SMC using the largest particle population, 5000. Thus, it is clearly evident that although the guided SMC's importance proposal is better than a vanilla Bootstrap SMC, a large number of particles (much larger than the maximum, 5000, used in this experiment) is required to make its performance comparable to the IDE. Especially, for a problem with a large value of $K \times M$. This experiment also shows that for complex high-dimensional implicit models, where guided SMC is inapplicable, IDE, using significantly fewer simulations, may perform at par or better than the bootstrap SMC.

5.2 Implicit biological HMMs

To evaluate the usefulness of the IDE for accurate estimation of the posterior predictive distribution $p(\boldsymbol{y}^r|\boldsymbol{y})$, we considered two biological HMMs: (i) a stochastic Lotka-Volterra (LV) (Wilkinson, 2018), and (ii) a prokaryotic autoregulator (PKY) (Golightly and Wilkinson, 2011) model. Further details of the dynamics, data generation and priors can be found in Appendix C. The hidden states for these models evolve as a pure Markov jump process (MJP) and thus the density $f(\cdot)$ is unavailable. Throughout we used simulated data so that we are cognisant of the ground truth. We used a tractable observation density $g(\cdot)$ to facilitate the Bootstrap SMC algorithm, run with 100 particles following Golightly and Wilkinson (2011). SMC estimates were considered as the baseline.

We chose the following competing approaches. First, the ABC-SMC algorithm (Toni et al., 2009), which produces samples from the joint distribution $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$, that can be used to evaluate the posterior predictive in (3). Except ABC-SMC all other approaches rely on the availability of parameters samples from the marginal $p(\boldsymbol{\theta}|\boldsymbol{y})$, which can be obtained using any off-the-shelf NLFI method such as the ones discussed in section 2.1. Once samples of $\boldsymbol{\theta}$ become available then samples of $\boldsymbol{x}$ can be drawn from its posterior using the IDE, the SMC (since $g(\cdot)$ is available), or simply from the prior transition $p(\boldsymbol{x}|\boldsymbol{\theta})$ (which we denote as PrDyn). Samples from $p(\boldsymbol{y}^r|\boldsymbol{y})$ can then be drawn using (15) for IDE, (5) for SMC and (7) for PrDyn.

To estimate $p(\boldsymbol{\theta}|\boldsymbol{y})$, required for IDE, SMC and PrDyn approaches, we used two sequential NLFI methods. One based on learning the likelihood density (SNLE) and the other one based on learning the likelihood-ratio (SRE). It was recently shown in (Durkan et al., 2020)

Table 1: Estimates of the **hidden states** of the **Lotka-Volterra** and **Prokaryotic autoregulator** models. Except ABC-SMC, we denoted all other methods using the following convention: method to estimate $\boldsymbol{x}$ (NLFI algorithm for estimating $\boldsymbol{\theta}$). Metrics were summarised by mean ± standard deviation across 10 simulated datasets. The baseline is SMC.

The **Lotka-Volterra** model				The **Prokaryotic autoregulator** model			
Methods	MSE	90% EC	CV	Methods	MSE	90% EC	CV
ABC-SMC	3654.41 ± 2239.14	0.99 ± 0.0091	0.79 ± 0.16	ABC-SMC	17.28 ± 4.76	0.99 ± 0.01	1.03 ± 0.11
PrDyn (SRE)	3585.03 ± 1938.88	0.98 ± 0.02	0.75 ± 0.16	PrDyn (SRE)	19.71 ± 6.68	0.99 ± 0.01	0.80 ± 0.02
PrDyn (SNLE)	4165.14 ± 1869.35	0.97 ± 0.02	0.81 ± 0.16	PrDyn (SNLE)	20.14 ± 7.51	0.99 ± 0.01	0.72 ± 0.04
SMC (SRE)	57.19 ± 7.97	0.93 ± 0.03	0.07 ± 0.01	SMC (SRE)	3.05 ± 0.61	0.93 ± 0.03	0.73 ± 0.02
SMC (SNLE)	56.86 ± 7.79	0.92 ± 0.03	0.07 ± 0.01	SMC (SNLE)	2.98 ± 0.59	0.94 ± 0.02	0.65 ± 0.04
IDE (SRE)	58.13 ± 6.11	0.95 ± 0.01	0.08 ± 0.01	IDE (SRE)	2.83 ± 0.54	0.97 ± 0.01	0.72 ± 0.02
IDE (SNLE)	57.85 ± 6.89	0.95 ± 0.01	0.08 ± 0.01	IDE (SNLE)	2.78 ± 0.46	0.98 ± 0.01	0.66 ± 0.04

that SRE (Hermans et al., 2020) is equivalent to a certain form of sequential learning of the posterior density (SNPE-C) (Greenberg et al., 2019) and both can be unified under a common framework on contrastive learning (Gutmann and Hyvärinen, 2010). Thus, by using SNLE and SRE we can cover the general ambit of sequential NLFI approaches. Further details of the neural network architecture, optimisation and other relevant details for IDE/SNLE/SRE are given in Appendix D. We used a fixed budget of simulations respectively for ABC-SMC (which jointly estimates $\boldsymbol{\theta}, \boldsymbol{x}$) and SNLE/SRE (used for estimating $\boldsymbol{\theta}$). This was done to rule out major differences in the estimates of $\boldsymbol{\theta}$ among ABC-SMC and NLFI methods so that the differences in estimates of $\boldsymbol{x}$, and subsequently $\boldsymbol{y}^r$, cannot be attributed to differences in parameter estimates. These simulation budgets were informed by previous studies such as Lueckmann et al. (2021) that compared the sample-efficiency between ABC-SMC and NLFI methods for estimating $p(\boldsymbol{\theta}|\boldsymbol{y})$. Thus, for both models, while using SNLE/SNRE, we used 30 rounds and the posterior samples from the final round were collected. For both models we generated 5000 training examples in the first round and in the subsequent rounds we generated 1000 examples. The simulations generated in the first round were used to learn the parameters $\boldsymbol{\phi}$ of the IDE. We limited the ABC-SMC to use no more than 10^7 simulations from the model. ABC-SMC is far less sample-efficient in comparison to SNLE/SRE (Lueckmann et al., 2021). Hence, we used considerably more simulations, in case of ABC-SMC, to ensure that the parameter estimates are as close as possible to SNLE/SRE. Further details of ABC-SMC implementation are given in Appendix D.

For inferring the parameters using ABC, SNLE/SRE we used summary statistics chosen to preserve the dynamical properties (e.g limit cycles). Note that IDE and SMC require full data, and PrDyn does not use data. Additionally, these methods use the same $\boldsymbol{\theta}$ values. Thus, these methods' relative performances are not influenced by the choice and use of summary statistics.

As before we considered the MSE (between the ground truth and the posterior mean) and 90% EC as metrics to quantify the quality of estimates of the replicated data $\boldsymbol{y}^r$ and the hidden states $\boldsymbol{x}$. However, methods such as the PrDyn are bound to overestimate the uncertainty (see section 3). Thus, we have also quantified the dispersion of $p(\boldsymbol{y}^r|\boldsymbol{y})$ and $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})$ using *coefficient of variation* (CV): the ratio of the posterior standard deviation and posterior mean, averaged across the time points.

The NLFI approaches were implemented using the `sbi`[1] package (Tejero-Cantero et al., 2020). We implemented the stochastic simulation algorithm (Gillespie, 1977) in `C++` to simulate the LV and PKY models. All the experiments were carried out on a high-performance computing cluster. Our code is available at `https://github.com/sg5g10/HMM`.

Additional experiments: In Appendix E we have also carried out evaluations on the PKY model without using summary statistics, using the SRE (which can learn summaries on the fly) and ABC-SMC, where we see no major differences in performance. Additionally, we have also run an experiment with the LV model to show the perils of trying to estimate $\boldsymbol{x}, \boldsymbol{\theta}$ jointly using a neural density estimator. See Appendix F for details.

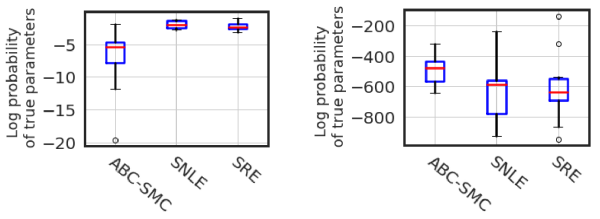
Results: We quantified the quality of estimation of the posterior distribution of the hidden states in Table 1 and subsequently the posterior predictive distribution in Table 2. In both these Tables, except ABC-SMC, we denoted all other methods using the following convention: **method to estimate $\boldsymbol{x}$ (NLFI algorithm for estimating $\boldsymbol{\theta}$)**. We summarised the chosen metrics across 10 simulated datasets. Clearly, using the IDE we were able to produce an estimate of the hidden states and subsequently the posterior predictive

[1] https://www.mackelab.org/sbi/

Table 2: Estimates of the **posterior predictive distribution** of the **Lotka-Volterra** and **Prokaryotic autoregulator** models, summarised across 10 simulated datasets. The baseline is SMC.

The **Lotka-Volterra** model				The **Prokaryotic autoregulator** model			
Methods	MSE	90% EC	CV	Methods	MSE	90% EC	CV
ABC-SMC	3791.01 ± 2239.27	0.98 ± 0.01	0.80 ± 0.16	ABC-SMC	22.23 ± 5.45	0.98 ± 0.02	0.34 ± 0.07
PrDyn (SRE)	3671.44 ± 1876.06	0.97 ± 0.02	0.76 ± 0.16	PrDyn (SRE)	24.87 ± 7.46	0.98 ± 0.01	0.33 ± 0.03
PrDyn (SNLE)	4260.82 ± 1827.63	0.96 ± 0.02	0.82 ± 0.16	PrDyn (SNLE)	25.20 ± 8.48	0.98 ± 0.02	0.32 ± 0.03
SMC (SRE)	55.50 ± 13.08	0.98 ± 0.01	0.12 ± 0.02	SMC (SRE)	2.15 ± 0.46	0.99 ± 0.01	0.11 ± 0.01
SMC (SNLE)	54.61 ± 12.43	0.98 ± 0.01	0.12 ± 0.02	SMC (SNLE)	1.76 ± 0.54	0.99 ± 0.01	0.11 ± 0.01
IDE (SRE)	100.69 ± 28.02	0.96 ± 0.03	0.12 ± 0.02	IDE (SRE)	2.36 ± 0.43	0.99 ± 0.01	0.12 ± 0.02
IDE (SNLE)	99.21 ± 27.39	0.96 ± 0.03	0.12 ± 0.02	IDE (SNLE)	2.05 ± 0.51	0.99 ± 0.01	0.12 ± 0.01

that is closer or better (hidden states of PKY model) to what can be achieved using SMC (which is highly sample-inefficient). We noticed that all the methods were producing higher values of the empirical coverage. For methods such as PrDyn such higher values do not indicate good uncertainty quantification. Rather such high coverage, for these methods, indicate credible intervals that are wide enough to always contain the ground truth. This was verified upon inspecting the CV metric which indicated significantly higher dispersion, for PrDyn and ABC-SMC, methods which draws $\boldsymbol{x}$ using its prior ($f(\cdot)$), indicating overestimation of the uncertainty. Notice the overestimation of uncertainty in plots of the posterior sample paths (Appendix G.1). PrDyn and ABC-SMC, by relying on the prior of the Markov process for proposing the hidden states, end-up producing a highly biased and under-confident estimate of the posterior distribution of the hidden states and subsequently the posterior predictive distribution. In Figure 3 we have compared the accuracy of



(a) (b)

Figure 3: Accuracy of parameter estimates for the **Lotka-Volterra** (a) and **Prokaryotic autoregulator** (b) models, assessed using the log probability of the true generative parameter vector, summarised across the 10 datasets. The log probabilities were obtained by fitting a mixture of multivariate Gaussian densities to 500 samples drawn from an estimate of $p(\boldsymbol{\theta}|\boldsymbol{y})$ obtained using each method.

parameters estimates obtained using ABC-SMC and the NLFI methods. Accuracy of the estimates were evaluated as the log probability of the true parameter vector under a mixture of multivariate Gaussian densities fitted to 500 samples drawn from an estimate of $p(\boldsymbol{\theta}|\boldsymbol{y})$ obtained using each method. We did not notice any drastic difference in accuracy and thus the difference in the estimation of posterior predictive were largely influenced by the estimates of the hidden states. The marginal densities of the parameter posteriors, for one dataset, are shown in Appendix G.2.

6 CONCLUSION

Neural likelihood-free methods have been previously benchmarked using implicit HMMs and are proposed as a computationally cheaper alternative to classical methods such as ABC-SMC in surrounding literature. We have shown that both classical as well as neural likelihood-free methods, by ignoring accurate state estimation, can lead to a grossly incorrect assessment of the goodness-of-fit. We thus proposed a novel technique to approximately estimate the hidden states once samples from the posterior (of the parameters) have been obtained using any likelihood-free method. Our technique, based on learning the posterior Markov process, using an autoregressive flow, produced estimate of the hidden states closer to what can be obtained using SMC, albeit with much fewer simulations.

Acknowledgements

We like to thank the anonymous reviewers for their helpful comments and suggestions. We like to also thank the anonymous reviewers of NeurIPS 2022 and ICLR 2023 conferences, who had kindly provided constructive comments on an earlier version of this work. SG was supported by the Medical Research Council (Unit programme number MC UU 00002/11).

References

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018.

Gael M Martin, Brendan PM McCabe, David T Frazier, Worapree Maneesoonthorn, and Christian P Robert. Auxiliary likelihood-based approximate bayesian computation in state space models. *Journal of Computational and Graphical Statistics*, 28(3): 508–522, 2019.

Umberto Picchini. Inference for sde models via approximate bayesian computation. *Journal of Computational and Graphical Statistics*, 23(4):1080–1100, 2014.

T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P.H Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, February 2009.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48): 30055–30062, 2020.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 343–351. PMLR, 13–15 Apr 2021.

Neil Gordon, David Salmond, and Craig Ewing. Bayesian state estimation for tracking and guidance using the bootstrap filter. *Journal of Guidance, Control, and Dynamics*, 18(6):1434–1443, 1995.

Andrew Gelman, Xiao-Li Meng, and Hal Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760, 1996.

Christopher C Drovandi, Anthony N Pettitt, and Roy A McCutchan. Exact and approximate bayesian inference for low integer-valued time series models with intractable likelihoods. *Bayesian Analysis*, 11(2): 325–352, 2016.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Lukas Schumacher, Paul-Christian Bürkner, Andreas Voss, Ullrich Köthe, and Stefan T Radev. Neural superstatistics for bayesian estimation of dynamic cognitive models. *Scientific Reports*, 13(1):13778, 2023.

Thomas Ryder, Dennis Prangle, Andrew Golightly, and Isaac Matthews. The neural moving average model for scalable variational inference of state space models. In *Uncertainty in Artificial Intelligence*, pages 12–22. PMLR, 2021.

George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, 29, 2016.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019.

Christopher M Bishop. Mixture density networks. 1994.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Matthew David Parno. *Transport maps for accelerated Bayesian computation*. PhD thesis, Massachusetts Institute of Technology, 2015.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66 (2):145–164, 2013.

Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*, 2015.

Conor Durkan, George Papamakarios, and Iain Murray. Sequential neural methods for likelihood-free inference. *arXiv preprint arXiv:1811.08723*, 2018.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR, 2019.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in neural information processing systems*, 30, 2017.

Yanzhi Chen, Dinghuai Zhang, Michael U. Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In *Ninth International Conference on Learning Representations (ICLR 2021)*, May 2021. URL `https://iclr.cc/Conferences/2021/Dates`. Ninth International Conference on Learning Representations

2021, ICLR 2021 ; Conference date: 04-05-2021 Through 07-05-2021.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

Simo Särkkä. *Bayesian filtering and smoothing*. Number 3. Cambridge university press, 2013.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10 (3):197–208, 2000.

Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2018.

Andrew Golightly and Darren J Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle markov chain monte carlo. *Interface focus*, 1(6):807–820, 2011.

Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning*, pages 2771–2781. PMLR, 2020.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free mcmc with amortized approximate ratio estimators. In *International Conference on Machine Learning*, pages 4239–4248. PMLR, 2020.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J. Gonçalves, David S. Greenberg, and Jakob H. Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020. doi:10.21105/joss.02505. URL `https://doi.org/10.21105/joss.02505`.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Sample-efficient neural likelihood-free Bayesian inference of implicit HMMs: Supplementary Materials

A Derivations of ABC and incremental posteriors of HMM

A.1 Joint distribution for HMM using ABC

NLFI methods are designed to efficiently sample from the marginal distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$. In ABC although the desired outcome often is the marginal distribution, however it is easy to show that for a latent variable model, such as an implicit HMM, ABC does indeed target an approximation of the joint distribution $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$.

In ABC we rely upon simulation of a pseudo-data $\hat{\boldsymbol{y}}$, when the likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ is intractable. The operating principle of any standard ABC algorithm, based on rejection sampling (Pritchard et al., 1999), MCMC (Marjoram et al., 2003) or SMC (Toni et al., 2009; Del Moral et al., 2012), is to jointly sample the parameters $\boldsymbol{\theta}$ and the pseudo-data $\hat{\boldsymbol{y}}$ from their posterior density (Marin et al., 2012)

$$p_\epsilon(\boldsymbol{\theta}, \hat{\boldsymbol{y}}|\boldsymbol{y}) = \frac{\mathbb{1}_\epsilon \left\{d(s(\hat{\boldsymbol{y}}), s(\boldsymbol{y}) < \epsilon)\right\} p(\hat{\boldsymbol{y}}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int \mathbb{1}_\epsilon \left\{d(s(\hat{\boldsymbol{y}}), s(\boldsymbol{y}) < \epsilon)\right\} p(\hat{\boldsymbol{y}}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}}, \tag{1}$$

where $\mathbb{1}_\epsilon(\cdot)$ is the indicator function, $d(\cdot)$ is a chosen distance metric, $\epsilon > 0$ and we consider the summary $s(\cdot)$ to be sufficient. The desired marginal posterior then follows as

$$p_\epsilon(\boldsymbol{\theta}|\boldsymbol{y}) = \int p_\epsilon(\boldsymbol{\theta}, \hat{\boldsymbol{y}}|\boldsymbol{y}) d\hat{\boldsymbol{y}}. \tag{2}$$

Note that the pseudo-data distribution $p(\hat{\boldsymbol{y}}|\boldsymbol{\theta})$ appearing in (1) is not required analytically in any of the ABC algorithms. This distribution is essentially the generative model under consideration.

For the HMM such a pseudo data is sampled from the distribution

$$p(\hat{\boldsymbol{y}}, \boldsymbol{x}|\boldsymbol{\theta}) = \left(\prod_{t=0}^{M-1} g(\hat{\boldsymbol{y}}_t|\boldsymbol{X}_t, \boldsymbol{\theta}) \right) \left(\prod_{t=1}^{M-1} f(\boldsymbol{X}_t|\boldsymbol{X}_{t_1}, \boldsymbol{\theta}) \right), \tag{3}$$

where $f(\cdot)$, $g(\cdot)$ and thus $p(\hat{\boldsymbol{y}}, \boldsymbol{x}|\boldsymbol{\theta})$ need not be analytically tractable, just a sample $\hat{\boldsymbol{y}}$ of the pseudo-data from this distribution is required. Sampling from this distribution is essentially the process of forward sampling from the generative model of the HMM (see main text). Considering $\hat{\boldsymbol{y}}$ alone from the pair $(\hat{\boldsymbol{y}}, \boldsymbol{x})$ we have a sample of the pseudo-data drawn from its marginal $p(\hat{\boldsymbol{y}}|\boldsymbol{\theta})$. Thus, when ABC is applied to the HMM the joint density in (1) is replaced by a density over the triplet $(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}})$ given by

$$p_\epsilon(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}}|\boldsymbol{y}) = \frac{\mathbb{1}_\epsilon \left\{d(s(\hat{\boldsymbol{y}}), s(\boldsymbol{y}) < \epsilon)\right\} p(\hat{\boldsymbol{y}}, \boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int \mathbb{1}_\epsilon \left\{d(s(\hat{\boldsymbol{y}}), s(\boldsymbol{y}) < \epsilon)\right\} p(\hat{\boldsymbol{y}}, \boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}}, \tag{4}$$

from which samples of the pair $(\boldsymbol{\theta}, \boldsymbol{x})$ is distributed from $p_\epsilon(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$. And the corresponding ABC marginal posterior is given by

$$p_\epsilon(\boldsymbol{\theta}|\boldsymbol{y}) = \int p_\epsilon(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}}|\boldsymbol{y}) d\hat{\boldsymbol{y}} d\boldsymbol{x}. \tag{5}$$

From (4) it is evident that any ABC algorithm applied to the HMM will target the joint distribution $p_\epsilon(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$. However, this distribution will only be an approximation to the true posterior $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$, since $\epsilon \neq 0$ (considering $s(\cdot)$ to be sufficient). Note that since $\boldsymbol{x}$ is sampled from its prior thus if ϵ is set to zero (or a small value) then a practically infeasible amount of simulations is required to produce an ABC posterior $p(\boldsymbol{\theta}, \boldsymbol{x}|\boldsymbol{y})$ that can approximate closely the true posterior.

A.2 Deriving the *incremental posterior* decomposition

We can decompose the posterior of $\boldsymbol{x}$, using the product rule, as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{y}) = p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2:1},\boldsymbol{\theta},\boldsymbol{y})p(\boldsymbol{X}_{M-2:1}|\boldsymbol{\theta},\boldsymbol{y}). \tag{6}$$

Let us first consider the first factor from the above equation, $p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2:1},\boldsymbol{\theta},\boldsymbol{y})$. We can obtain from this the density of the last sample points $\boldsymbol{X}_{M-1}$, conditioned on all other random variables, by applying the Markov property and retaining only the terms that involve it, given by:

$$\begin{aligned} p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2},\ldots,\boldsymbol{X}_1,\boldsymbol{\theta},\boldsymbol{y}) &\propto p(\boldsymbol{y}|\boldsymbol{X}_{M-1},\boldsymbol{X}_{M-2},\ldots,\boldsymbol{X}_1,\boldsymbol{\theta})p(\boldsymbol{X}_{M-1},\boldsymbol{X}_{M-2},\ldots,\boldsymbol{X}_1|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &\propto p(\boldsymbol{\theta})\left(\prod_{t=0}^{M-1} g(\boldsymbol{y}_t|\boldsymbol{X}_t,\boldsymbol{\theta}_g)\right)\left(\prod_{t=1}^{M-1} f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{\theta}_f)\right) \\ &\propto g(\boldsymbol{y}_{M-1}|\boldsymbol{X}_{M-1},\boldsymbol{\theta}_g)f(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2},\boldsymbol{\theta}_f)p(\boldsymbol{\theta}), \end{aligned} \tag{7}$$

which is simply the density $p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2},\boldsymbol{y}_{M-1},\boldsymbol{\theta})$.

We can also write the conditional distribution of any intermediate sample point $\boldsymbol{X}_t$ among the remaining ones $\boldsymbol{X}_{M-2:1}$, by again applying the Markov property and retaining only the terms that involve it, given by:

$$\begin{aligned} p(\boldsymbol{X}_t|\boldsymbol{X}_{M-1},\ldots,\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\ldots,\boldsymbol{X}_1,\boldsymbol{\theta},\boldsymbol{y}) &\propto p(\boldsymbol{y}|\boldsymbol{X}_{M-1},\ldots,\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\ldots,\boldsymbol{X}_1,\boldsymbol{\theta}) \\ &\quad\times p(\boldsymbol{X}_{M-1},\ldots,\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\ldots,\boldsymbol{X}_1|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &\propto p(\boldsymbol{\theta})\left(\prod_{t=0}^{M-1} g(\boldsymbol{y}_t|\boldsymbol{X}_t,\boldsymbol{\theta}_g)\right)\left(\prod_{t=1}^{M-1} f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{\theta}_f)\right) \\ &\propto f(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t,\boldsymbol{\theta}_f)f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{\theta}_f)g(\boldsymbol{y}_t|\boldsymbol{X}_t,\boldsymbol{\theta}_g)p(\boldsymbol{\theta}), \end{aligned} \tag{8}$$

which is simply the density $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{X}_{t+1},\boldsymbol{y}_t,\boldsymbol{\theta})$.

Using Eq. (7) and Eq. (8), we can now factorise and re-write Eq. (6) as given by

$$p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{y}) = p(\boldsymbol{X}_{M-1}|\boldsymbol{X}_{M-2},\boldsymbol{\theta},\boldsymbol{y})\prod_{t=1}^{M-2} p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta}), \tag{9}$$

which completes the proof.

A.3 Pseudocode for the IDE training.

In Algorithm 1 we provide the pseudocode describing the process of creating a training dataset and then subsequently training the two MAF density estimators emulating the true factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t+1},\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})$, and the approximate factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})$.

B Nonlinear Gaussian state-space model

B.1 Model details

Here we want to evaluate how well the IDE can perform in comparison to an optimal SMC algorithm which uses the approximate factor $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1},\boldsymbol{y}_t,\boldsymbol{\theta})$ as the importance proposal. This density is tractable for Gaussian state-space models. Thus, for this evaluation we have chosen the following state-space model:

$$\begin{aligned} \boldsymbol{X}_t &\sim \mathcal{N}(\boldsymbol{A}\gamma(\boldsymbol{X}_{t-1}),\sigma_x^2\mathbb{I}) \quad t \geq 1 \\ \boldsymbol{y}_t &\sim \mathcal{N}(\boldsymbol{B}\boldsymbol{X}_t,\sigma_y^2\mathbb{I}), \end{aligned} \tag{12}$$

where $\gamma(\boldsymbol{X}) = \sin(\exp(\boldsymbol{X}_{t-1}))$, applied elementwise, $\boldsymbol{A} = \mathbb{I}_{K\times K}$, $B = 2\boldsymbol{A}$ and $\boldsymbol{X}_0 = \boldsymbol{0}$.

We considered the dimensionality of the state-space, $\dim(\boldsymbol{X}_t)$ and $\dim(\boldsymbol{y}_t)$ to be the same, $K = L = 10$. We also considered the parameters $\boldsymbol{\theta} = (\sigma_x, \sigma_y)$ to be fixed and known. Thus, we can drop $\boldsymbol{\theta}$ from the conditioning

Algorithm 1 Simulation and IDE training

Input: Training dataset size N, time series length M.

1. Simulate from HMM:

for $n = 1$ **to** N **do**
  for $t = 1$ **to** $M - 1$ **do**
    $(\boldsymbol{\theta}_f^n, \boldsymbol{\theta}_g^n, \boldsymbol{X}_0^n) \sim p(\boldsymbol{\theta}), \quad \boldsymbol{X}_t^n \sim f(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{\theta}_f), \quad \boldsymbol{y}_t^n \sim g(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{\theta}_g)$.
  end for
end for

2. Generate training examples for the density estimators

for $n = 1$ **to** N **do**
  for $i = 0$ **to** $M - 3$ **do**
    for $j = 1$ **to** $M - 2$ **do**
      for $k = 2$ **to** $M - 1$ **do**
        $q_{\boldsymbol{\phi}_2}(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ emulating the true factor: target $\boldsymbol{X}_j^n$, inputs $(\boldsymbol{X}_k^n, \boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n)$.
        $q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ emulating the approximate factor: target $\boldsymbol{X}_j^n$, inputs $(\boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n)$.
      end for
    end for
  end for
end for

3. Train the density estimators, using gradient ascent:

$$\begin{aligned} \boldsymbol{\phi}_1^* &= \underset{\boldsymbol{\phi}_1}{\operatorname{argmax}} \mathcal{L}(\boldsymbol{\phi}_1) \\ \boldsymbol{\phi}_2^* &= \underset{\boldsymbol{\phi}_2}{\operatorname{argmax}} \mathcal{L}(\boldsymbol{\phi}_2), \end{aligned} \tag{10}$$

where the loss functions $\mathcal{L}(\boldsymbol{\phi}_1)$ and $\mathcal{L}(\boldsymbol{\phi}_2)$ are given by the total likelihood of the MAF density estimators:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}_1) &= \sum_{n=1}^{N} \sum_{i=0,j=1}^{M-2,M-1} \log q_{\boldsymbol{\phi}_1}(\boldsymbol{X}_j^n|\boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n) \\ \mathcal{L}(\boldsymbol{\phi}_2) &= \sum_{n=1}^{N} \sum_{i=0,j=1,k=2}^{M-3,M-2,M-1} \log q_{\boldsymbol{\phi}_2}(\boldsymbol{X}_j^n|\boldsymbol{X}_k^n, \boldsymbol{X}_i^n, \boldsymbol{y}_j^n, \boldsymbol{\theta}^n). \end{aligned} \tag{11}$$

.

Output: Optimised parameters $\boldsymbol{\phi}_1^*, \boldsymbol{\phi}_2^*$.

variables for the true and approximate factors $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t+1}, \boldsymbol{y}_t)$ and $p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$ respectively. And we do the same for the corresponding density estimates: $q_{\boldsymbol{\phi}}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t)$ and $q_{\boldsymbol{\phi}}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t+1}, \boldsymbol{y}_t)$. For the model above, the approximate factor is known analytically and happens to be a Gaussian:

$$p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t) = \mathcal{N}(\boldsymbol{X}_t; \boldsymbol{m}, \boldsymbol{\Sigma}), \tag{13}$$

whose mean and the covariance are given by

$$\begin{aligned} \boldsymbol{\Sigma}^{-1} &= \Sigma_x^{-1} + B\Sigma_y^{-1}B \\ \boldsymbol{m} &= \boldsymbol{\Sigma}(\Sigma_x^{-1}\gamma(\boldsymbol{X}_{t-1}) + B\Sigma_y^{-1}\boldsymbol{y}_t), \end{aligned} \tag{14}$$

where $\Sigma_x = \sigma_x^2\mathbb{I}$ and $\Sigma_y = \sigma_y^2\mathbb{I}$.

We used $\sigma_x = \sigma_y = 0.5$ to generate the simulated data. We considered a long time series with $M = 500$ time points. We created the IDE training set as was described in section 4.2 (main text).

For the IDE's MAF we have used $J = 3$ transformations, each of which has two hidden layers of 50 units and ReLU nonlinearities. We found that chaining a few transformations was enough to learn a Gaussian density. Increasing the number of transformations did not improve the performance noticeably. For training the MAF

we used ADAM (Kingma and Ba, 2015) with a minibatch size of 256, and a learning rate of 0.0005. Following, Papamakarios et al. (2019) we used 10% of the training data as a validation set, and stopped training if validation log likelihood did not improve after 20 epochs.

B.2 Additional experiments with state-space model

In the main text we have furnished results for using parameters $\sigma_x = \sigma_y = 0.5$. However, we have carried out additional experiments firstly with noise $\sigma_x = \sigma_y = 1$ and then probing the performances for even more higher-dimensional states space, $K = 30$, along with this higher noise setting. See Figure 1 for the results of these additional experiments. Note that we consistently found the Bootstrap SMC to give extremely poor performance, and thus not shown in the plots.

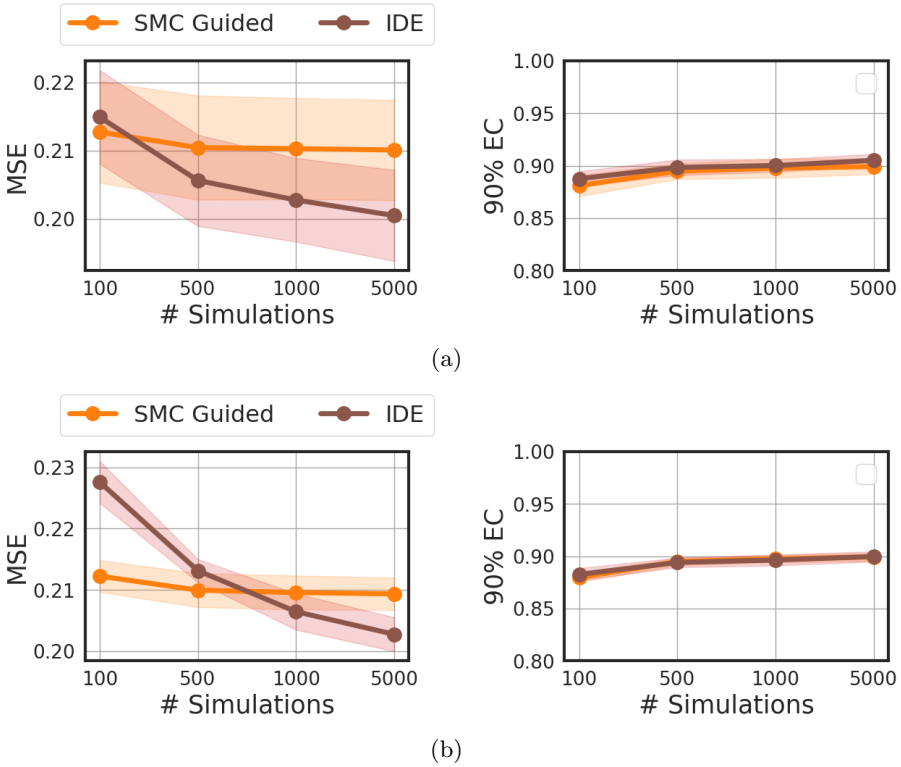


Figure 1: Estimation of the **hidden states** of a **nonlinear state-space** model, for two different experiments: (a) $\sigma_x = \sigma_y = 1$ and $K = 10$, (b) $\sigma_x = \sigma_y = 1$ and $K = 30$. The quality of approximations was quantified using the MSE and 90% EC, summarised using the mean (solid line) and 95% confidence intervals (shaded area), across 10 datasets.

Finally, we also compared the IDE's performance for longer times series, $M = 1000, 5000$. For these experiments, we set the number of particles for SMC algorithms and training set size for IDE to 500. We used $\sigma_x = \sigma_y = 0.5$ to generate the simulated data. In Table 1 we furnished the results.

Table 1: Metrics for longer time series.

Metrics for $M = 1000$		
Metrics	**SMC Guided**	**IDE**
MSE	0.0113	0.0139
Coverage	0.9073	0.8954

Metrics for $M = 5000$		
Metrics	**SMC Guided**	**IDE**
MSE	0.0112	0.0139
Coverage	0.9096	0.8960

C Model details

C.1 Stochastic Lotka-Volterra model

The stochastic Lotka-Volterra model, a stochastic kinetic system, can be defined through the following list of reactions:

$$\begin{aligned} \mathcal{R}_1: &\quad X^{prey} \xrightarrow{c_1} 2X^{prey} \\ \mathcal{R}_2: &\quad X^{prey} + X^{pred} \xrightarrow{c_2} 2X^{pred} \\ \mathcal{R}_3: &\quad X^{pred} \xrightarrow{c_2} \emptyset, \end{aligned} \tag{15}$$

where we denote by X^{prey}, X^{pred} the prey and predator species respectively. We further denote the corresponding numbers of the species as the system state $\boldsymbol{X}_t = (X_t^{prey}, X_t^{pred})$. The hazard vector for this system is $h(\boldsymbol{X}_t, \boldsymbol{c}) = \left(c_1 X_t^{prey}, c_2 X_t^{prey} X_t^{pred}, c_3 X_t^{pred}\right)$. The stoichiometry matrix for this system is given by

$$S = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}. \tag{16}$$

We set the initial values as $\mathcal{X}_0 = (100, 100)$ and consider them known.

A MJP describing a stochastic kinetic system, like the one above or the PKY model, is characterised by the transition probability $p(t_0, \boldsymbol{X}_0, t, \boldsymbol{X}_t) := p(\boldsymbol{X}, t)$ for the process arriving at state $\boldsymbol{X}_t$ at time t conditioned on an initial state $\boldsymbol{X}_0$ at time t_0. This is basically the transition density $f(\cdot)$ appearing in the definition of a HMM (see main text) defined here in continuous time. Now this transition probability is given by the solution of the following differential equation:

$$\frac{\partial p(\boldsymbol{X}, t)}{\partial t} = \sum_{i=1}^{v} = \{h_i(\boldsymbol{X} - S^i, c_i) p(\boldsymbol{X} - S^i, t) - h_i(\boldsymbol{X}, c_i) p(\boldsymbol{X}, t)\}, \tag{17}$$

known as the chemical master equation (Golightly and Gillespie, 2013, and the references therein). The chemical master equation (CME) only admits an analytical solution for a handful of simple models (not for the ones we have used: LV and PKY). Thus, the density $f(\cdot)$ cannot be evaluated. However, the seminal work in Gillespie (1977) developed an algorithm, commonly referred to as the *stochastic simulation algorithm*, that can simulate $\boldsymbol{X}$ exactly.

We generated simulated trajectories from this model using the stochastic simulation algorithm and added Gaussian noise corruption, with variance 100, at 50 time points. We used the following generative values of the parameters $\boldsymbol{\theta} = (0.3, 0.0025, 0.5)$ to ensure that the model follows an oscillatory regime. Moreover, following previous studies we considered the initial values to be known and set at $\boldsymbol{X}_{t_0} = (100, 100)$.

We used the following set of **prior distributions**: $c_1 \sim \text{Beta}(1, 2)$, $c_2 \times 10^3 \sim \mathcal{U}(15, 50)$ and $c_3 \sim \text{Beta}(2, 1)$.

For running ABC-SMC and all the NLFI methods we downsampled the generated time series by a factor of 5 to create a **summary statistic** $s(\boldsymbol{y}) \in \mathbb{R}^{20}$ which is used in place of the full data $\boldsymbol{y}$.

C.2 Prokaryotic autoregulatory gene network

We considered the autoregulatory model used to benchmark the particle MCMC method in Golightly and Wilkinson (2011). This is a simplified model that describes a mechanism for autoregulation in prokaryotes based on a negative feedback mechanism of dimers of a protein coded by a gene repressing its own transcription.

Essentially this is a stochastic kinetic model described by the following set of reactions:

$$\begin{aligned}
&\mathcal{R}_1 : DNA + P2 \to DNA \cdot P2 \\
&\mathcal{R}_2 : DNA \cdot P2 \to DNA + P2 \\
&\mathcal{R}_3 : DNA \to DNA + RNA \\
&\mathcal{R}_4 : RNA \to RNA + P \\
&\mathcal{R}_5 : 2P \to P2 \\
&\mathcal{R}_6 : P2 \to 2P \\
&\mathcal{R}_7 : RNA \to \emptyset \\
&\mathcal{R}_8 : P \to \emptyset.
\end{aligned} \tag{18}$$

We order the variables as $\boldsymbol{X} = (RNA, P, P2, DNA, DNA \cdot P2)$ leading to a stoichiometry matrix for the system:

$$S = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -2 & 2 & 0 & -1 & \\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{19}$$

and the associated hazard function is given by

$$h(\boldsymbol{X}, \boldsymbol{c}) = (c_1 DNA \times P2, c_2 DNA \cdot P2, c_3 DNA, c_4 RNA, c_5 P(P-1)/2, c_6 P2, c_7 RNA, c_8 P). \tag{20}$$

This model has one conservation law (Golightly and Wilkinson, 2011)

$$DNA \cdot P2 + DNA = k, \tag{21}$$

where k is the number of copies of this gene in the genome. Following Golightly and Wilkinson (2011) we use this relation to to remove $DNA \cdot P2$ from the model, replacing any occurrences of $DNA \cdot P2$ in rate laws with $k - DNA$. This leads to a reduced full-rank model with species $\boldsymbol{X} = (RNA, P, P2, DNA)$, stoichiometry matrix:

$$S = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -2 & 2 & & -1 & \\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{22}$$

and associated hazard function

$$h(\boldsymbol{X}, \boldsymbol{c}) = (c_1 DNA \times P2, c_2(k - DNA), c_3 DNA, c_4 RNA, c_5 P(P-1)/2, c_6 P2, c_7 RNA, c_8 P). \tag{23}$$

We consider k to be known and set to 10. Again we generated simulated trajectories from this model using the stochastic simulation algorithm.

Following Golightly and Wilkinson (2011), we considered the observations as a linear combination of the proteins $P, P2$ as follows:

$$y_t = P_t + 2P2_t + \epsilon_t, \tag{24}$$

where ϵ is assumed to be iid Gaussian noise. We generated 100 simulated observations from this model at times $t = [0 : .5 : 50]$ with generative rate constants $\boldsymbol{\theta} = (0.1, 0.7, 0.35, 0.2, 0.1, 0.9, 0.3, 0.1)$ and $\epsilon \sim \mathcal{N}(0, 4)$. In this case also we considered the initial values $\boldsymbol{X}_{t_0}$ to be known and set to $(8, 8, 8, 5)$.

We placed a Gamma$(2, 3)$ **prior** on all the rate constants.

We downsampled the simulated data by a factor of five to obtain the **summary statistic** $s(\boldsymbol{y}) \in \mathbb{R}^{20}$.

D NLFI, IDE and ABC-SMC implementation details for biological HMMs

For SNLE we used a MAF as the likelihood density estimator $q_\psi(s(\boldsymbol{y})|\boldsymbol{\theta})$ and for SRE we used a MLP classifier. For both uses of the MAFs, $q_{\boldsymbol{\phi}}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t-1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ for the IDE and $q_\psi(\boldsymbol{\theta}|s(\boldsymbol{y}))$ for

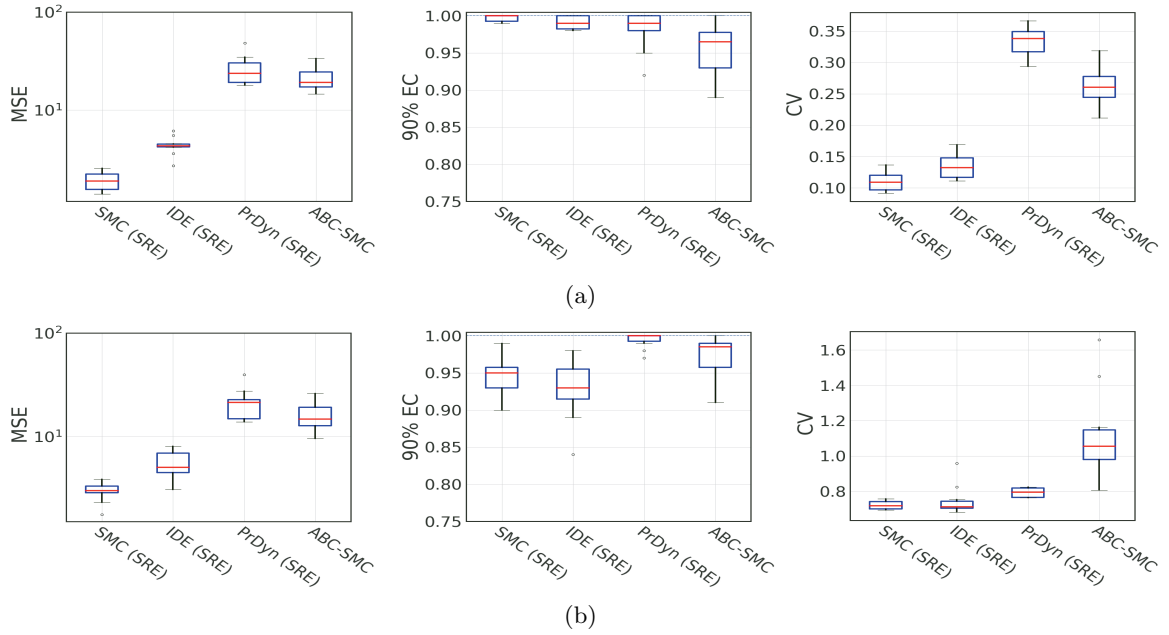


Figure 2: Comparison of the estimates of the (a) **posterior predictive distribution** and (b) **hidden states** of the **Prokaryotic autoregulator** models. We summarised the chosen metrics across 10 simulated datasets. The baseline is **SMC**. Here we are using the full data rather than the summaries.

SNLE, we used the same architecture. That is $J = 5$ transformations, each of which has two hidden layers of 50 units each and ReLU nonlinearities. For SRE we used a residual network based classifier with two residual layers of 50 units each and ReLU nonlinearities.

For training all the neural networks we used ADAM (Kingma and Ba, 2015) with the same minibatch size, learning rate and validation split as was used for the experiment with the state-space model. Following Papamakarios et al. (2019), we used the Slice Sampling algorithm (Neal, 2003) to draw samples from the posterior while using SNLE and SRE.

We applied the particular version of ABC-SMC algorithm, that was proposed in Toni et al. (2009), using 1000 particles. Furthermore, we used an adaptive tolerance sequence where the tolerance ϵ_τ at the τ-th step of the algorithm is selected as the 0.1-quantile of the distances of the accepted particles in the $\tau - 1$-th step. Moreover, we chose the perturbation kernel of ABC-SMC (see Toni et al. (2009)) as a multivariate Gaussian whose covariance is based on a *k-nearest neighbours* strategy, with $k = 15$, proposed in Filippi et al. (2013). We terminated the ABC-SMC algorithm when a predetermined number of simulations has been carried out. If that number is exceeded within the τ-th step, we then considered the weighted particle system at the $\tau - 1$-th step as the desired ABC posterior.

E Evaluations without using summary statistics

All our evaluations on the two biological HMMs were based on the use of hand-crafted summary statistics. Here we repeat the analysis for the PKY model without using summary statistics. For ABC-SMC this means calculating a distance between the full observed data (considering all the time points) and the simulated one. Note that the particular ABC-SMC algorithm that we have used (Toni et al., 2009) was originally designed to work with full data. For obtaining the hidden states and subsequently the posterior predictive distribution using SMC, IDE and PrDyn we have used an estimate of $\boldsymbol{\theta}$ obtained using SRE trained on the full dataset. For this we extended the classifier neural network with a 2-layer LSTM, trained simultaneously with the classifier, to embed the data into a smaller dimensional summary statistics. We used a LSTM with a 10-dimensional hidden state and fed the hidden state, corresponding to the last time-step, into a fully connected layer consisting 8 hidden units and a ReLU activation function. Thus, we have a 8-dimensional summary statistics that is learnt on the fly.

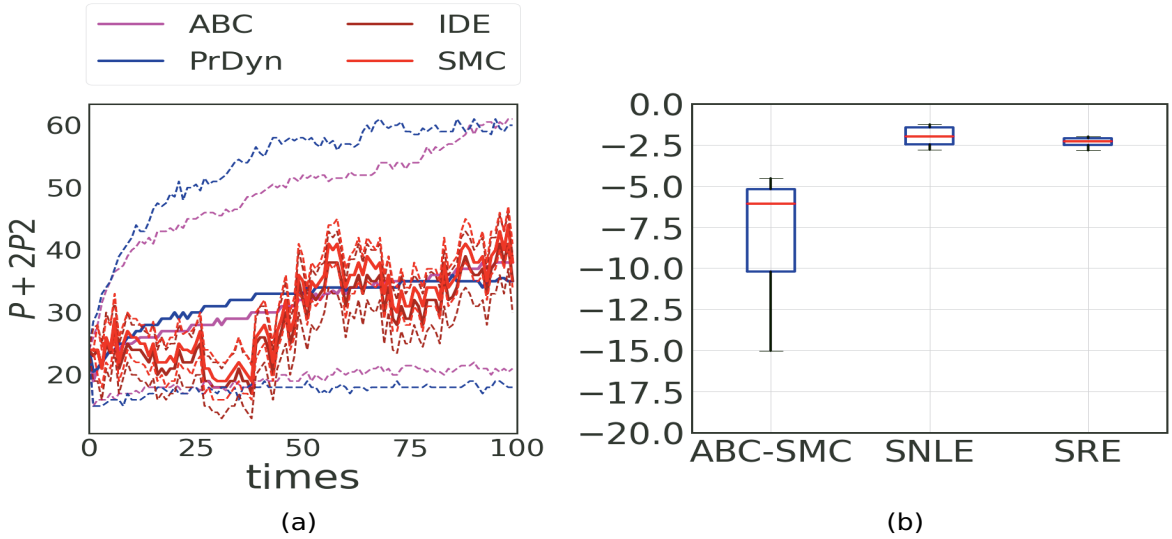


Figure 3: (a) Posterior distributions of the latent sample path $\boldsymbol{x}$ summarised by the mean (solid lines) and 95% credible intervals (broken lines), for the **Prokaryotic autoregulator**. The **ABC-SMC** is using the full dataset. (b) Accuracy of parameter estimates for the **Prokaryotic autoregulator** model, evaluated using the log probability of the true generative parameter vector, summarised across the 10 datasets. **SRE** and **ABC-SMC** is using the full dataset.

In Figure 2 we compare the estimates of the posterior predictive and the hidden states using the same metrics that we have used previously. We noticed that the IDE produced estimates of these quantities closer to the baseline (SMC's estimate) than ABC-SMC and PrDyn. Additionally, we noticed a slight improvement of ABC-SMC's performance in estimating the hidden states (see also Figure 3 (a) where we have plotted the estimated hidden states for one dataset), however the accuracy of the parameters estimates (summarised in Figure 3 (b)) did not change significantly from what was observed while using summary statistics. Note that the accuracy of the parameter estimates did not change significantly for the SRE as well. Despite having access to the full data the ABC-SMC's proposal mechanism for the hidden states is still too inefficient to significantly improve the accuracy of reconstructing the hidden states within a practically feasible simulation budget.

F Joint inference of the sample path and parameters using a MAF

We have argued before (see the last paragraph of section 3 in main text) that NLFI methods cannot be used directly for inferring the joint posterior $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$. Next, we have shown results for an experiment, using the LV model, that supports our argument. Note that since we cannot evaluate the joint density $p(\boldsymbol{x}, \boldsymbol{\theta})$, the only strategy that can be applied is of using a normalizing-flow to directly emulate the joint posterior $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y}) \approx q_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$. We denote this approach as neural posterior estimation (NPE). We used 10^6 simulations from the model to train a MAF representing $q_{\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$. Note that for the proposed IDE approach we have used much fewer simulations. We retained the same architecture and optimisation settings that we used in other experiments. Once trained, we used one of the simulated dataset for the LV model to carry out inference. This is the same dataset corresponding to the plot shown in Figure 6.

In Figure 4 we plot components of the hidden state estimated by SMC, IDE, ABC-SMC and NPE. Note that SMC, IDE are using same samples of $\boldsymbol{\theta}$ estimated using SNLE. All methods use 500 samples from the posteriors of $\boldsymbol{\theta}, \boldsymbol{x}$. In Figure 5 we show the corresponding parameter estimates. Although NPE estimates the hidden state better than ABC-SMC, its estimation quality drops at those time points where the concentration reaches a peak before decreasing again. This drop is much more pronounced near the last peak. The parameter estimates are however significantly different than all the other methods. From which it can be concluded that NPE performs worse than even ABC-SMC to produce the posterior of the parameters when targeting $\boldsymbol{x}, \boldsymbol{\theta}$ jointly.

Additionally, as further pilot experiments, we have also repeated this experiment without using summary statistics for NPE and rather (i) learning the summaries using a LSTM and (ii) feeding in the full data as the input to the

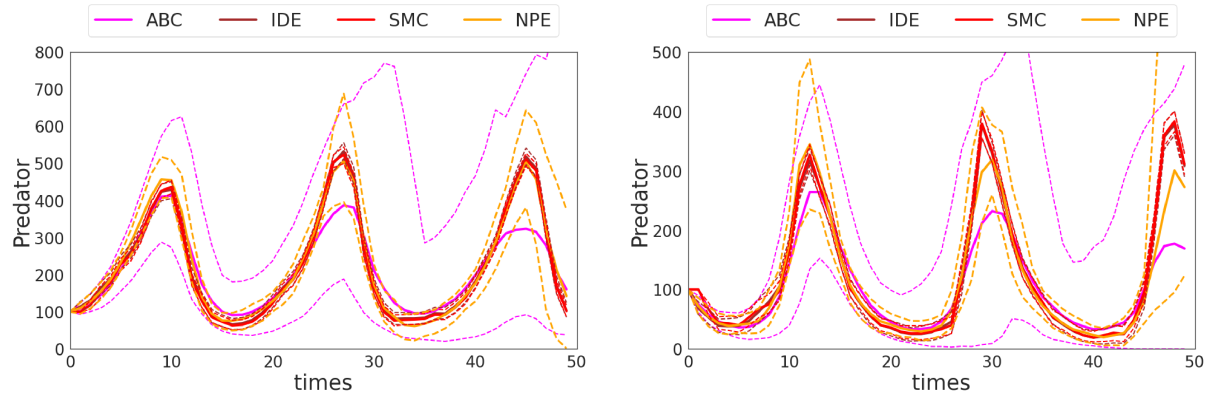


Figure 4: Comparison between methods that estimate jointly the parameters and hidden states of a HMM (in this case the **Lotka-Volterra** model), such as **ABC-SMC** & **NPE**, with those that estimate these quantities separately, such as **SMC** & **IDE**. The plot above shows the posteriors of the hidden states summarised by the mean (solid lines) and 95% credible intervals (broken lines). The proposed method **IDE** reduces the simulation burden by a large factor in comparison to **NPE**. Note that even with a much larger simulation budget **NPE** fails to correctly estimate the hidden states as well as the parameters (see Figure 5).

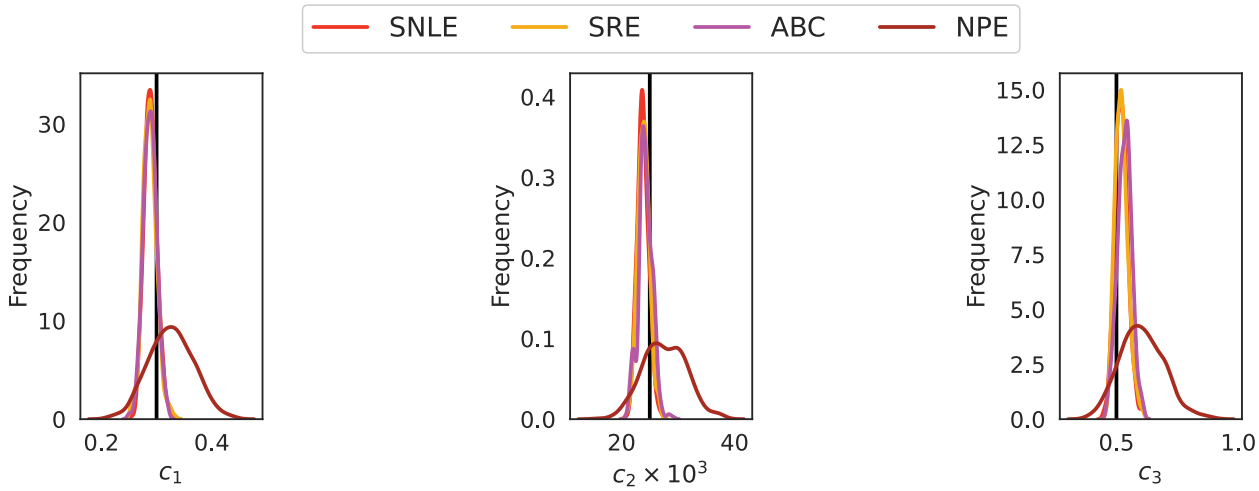


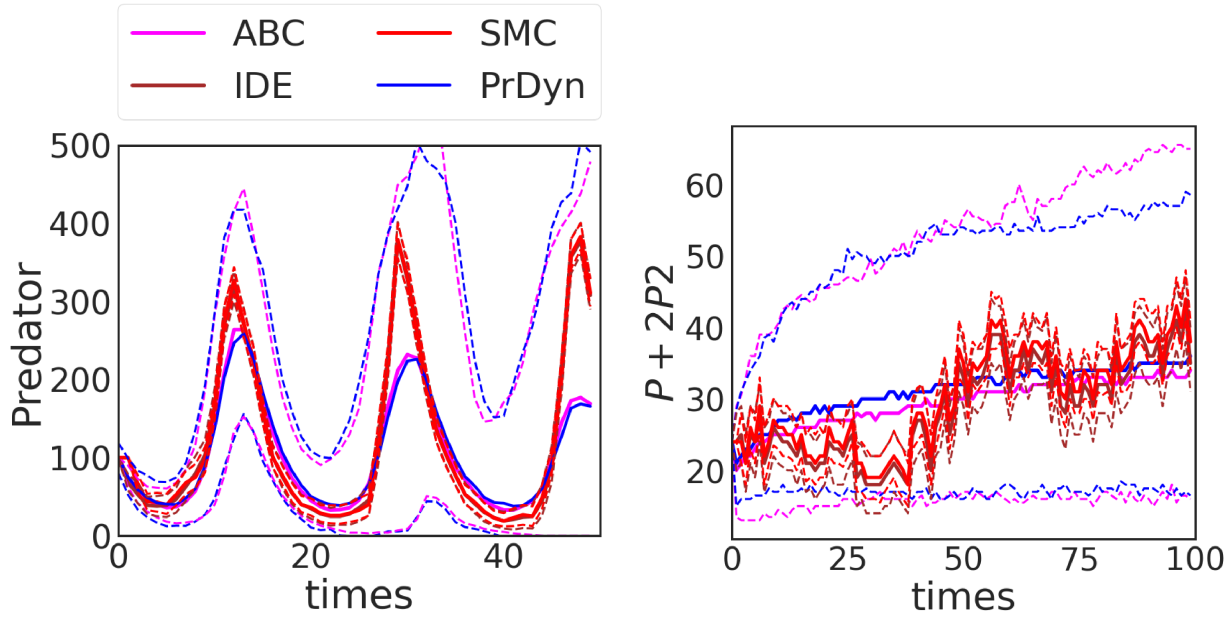
Figure 5: Posterior marginal densities of the parameters of the **Lotka-Volterra** model obtained using SNLE, SRE (both targeting the marginal $p(\boldsymbol{\theta}|\boldsymbol{y})$) with NPE, ABC-SMC (both targeting the joint $p(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y})$). NPE failed to estimate $\boldsymbol{\theta}$ correctly.

normalising-flow. However, the results were even worse and thus we have not shown them here.

G Plots of hidden states and parameter posteriors

G.1 Plots of hidden states

The following plots of the posterior sample paths (posterior of the hidden states) for one dataset (Figure 6), clearly show the overestimation of uncertainty in case of PrDyn and ABC-SMC, for all models.



(a) (b)

Figure 6: Posterior distributions of one component of the latent sample path (the hidden states) $\boldsymbol{x}$ summarised by the mean (solid lines) and 95% credible intervals (broken lines), for the **Lotka-Volterra** (a), **Prokaryotic autoregulator** (b) model. Here SMC, IDE and PrDyn estimates of $\boldsymbol{x}$ corresponds to an SNLE estimate of $\boldsymbol{\theta}$.

G.2 Plots of marginal posteriors of the parameters

In the subsequent plots Figure 7 and 8 we compare the parameter estimates of the models between NLFI based methods, SNLE/SRE, and ABC-SMC. Here we have shown the estimates for one of the 10 different simulated datasets. This is the same dataset corresponding to the plot shown in Figure 6. Note that the parameter estimates are reasonably close to each other and thus the estimate of the posterior predictive distribution is largely influenced by the estimates of the hidden states.

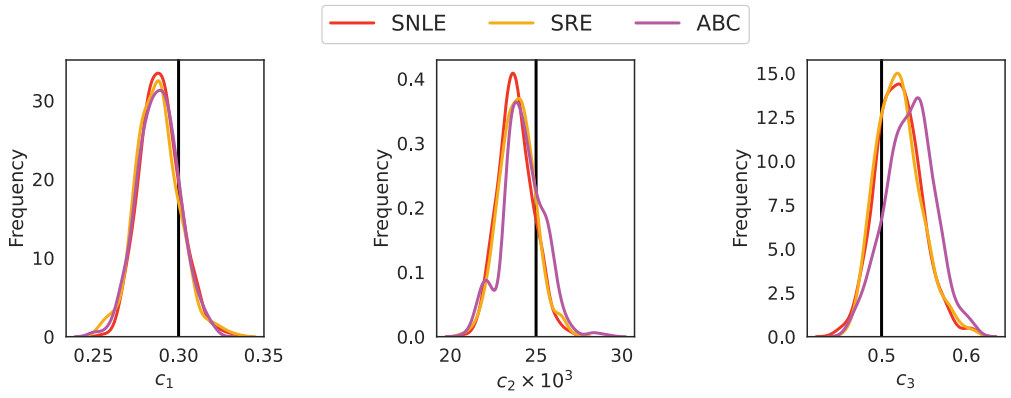


Figure 7: Posterior marginal densities of the parameters of the **Lotka-Volterra** model, inferred from one of the 10 datasets.

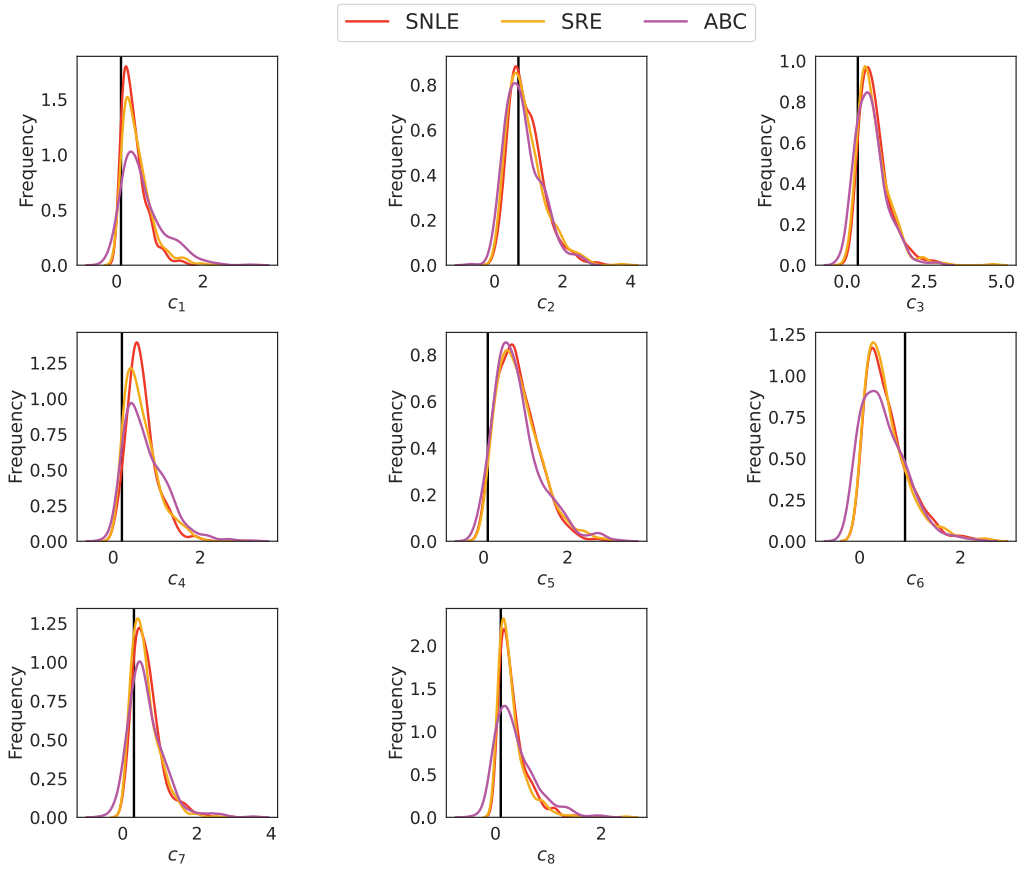


Figure 8: Posterior marginal densities of the parameters of the **Prokaryotic autoregulatory** model, inferred from one of the 10 datasets.

H Related work in inference of implicit HMMs

The most common approaches to tackle the inference of an implicit HMM consist largely of ABC methods (Dean et al., 2014; Martin et al., 2019; Picchini, 2014). Note that when the observational density is known analytically then the particle-MCMC (Andrieu et al., 2010) method can be used to carry out exact inference. However, the computational cost of this method is prohibitive, as in each step of MCMC a particle filter with a large number of particles is run to calculate an unbiased estimate of the marginal likelihood. Interestingly, a new avenue of research can be of combining our proposed IDE as an importance density within a particle-MCMC scheme. An alternative approach which combines SMC with ABC was proposed in (Drovandi et al., 2016). However, this approach requires the problematic choices of ABC tunning parameters.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Thomas A Dean, Sumeetpal S Singh, Ajay Jasra, and Gareth W Peters. Parameter estimation for hidden markov models with intractable likelihoods. *Scandinavian Journal of Statistics*, 41(4):970–987, 2014.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential monte carlo method for approximate bayesian computation. *Statistics and computing*, 22(5):1009–1020, 2012.

Christopher C Drovandi, Anthony N Pettitt, and Roy A McCutchan. Exact and approximate bayesian inference for low integer-valued time series models with intractable likelihoods. *Bayesian Analysis*, 11(2):325–352, 2016.

Sarah Filippi, Chris P Barnes, Julien Cornebise, and Michael PH Stumpf. On optimality of kernels for approximate bayesian computation using sequential monte carlo. *Statistical applications in genetics and molecular biology*, 12(1):87–107, 2013.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

Andrew Golightly and Colin S Gillespie. Simulation of stochastic kinetic models. In *In Silico Systems Biology*, pages 169–187. Springer, 2013.

Andrew Golightly and Darren J Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle markov chain monte carlo. *Interface focus*, 1(6):807–820, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

Gael M Martin, Brendan PM McCabe, David T Frazier, Worapree Maneesoonthorn, and Christian P Robert. Auxiliary likelihood-based approximate bayesian computation in state space models. *Journal of Computational and Graphical Statistics*, 28(3):508–522, 2019.

Radford M Neal. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019.

Umberto Picchini. Inference for sde models via approximate bayesian computation. *Journal of Computational and Graphical Statistics*, 23(4):1080–1100, 2014.

Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, 16(12): 1791–1798, 1999.

T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P.H Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31): 187–202, February 2009.